J Sign Process Syst (2014) 77:191–205
DOI 10.1007/s11265-014-0875-2

An Efficient Scalable RNS Architecture for Large Dynamic Ranges

Pedro Miguens Matutino · Ricardo Chaves · Leonel Sousa

Received: 9 September 2013 / Revised: 16 January 2014 / Accepted: 6 February 2014 / Published online: 13 March 2014
© Springer Science+Business Media New York 2014

Abstract This paper proposes an efficient scalable Residue Number System (RNS) architecture supporting moduli sets with an arbitrary number of channels, allowing to achieve larger dynamic range and a higher level of parallelism. The proposed architecture allows the forward and reverse RNS conversion, by reusing the arithmetic channel units. The arithmetic operations supported at the channel level include addition, subtraction, and multiplication with accumulation capability. For the reverse conversion two algorithms are considered, one based on the Chinese Remainder Theorem and the other one on Mixed-Radix-Conversion, leading to implementations optimized for delay and required circuit area. With the proposed architecture a complete and compact RNS platform is achieved . Experimental results suggest gains of 17 % in the delay in the arithmetic operations, with an area reduction of 23 % regarding the RNS state of the art. When compared with a binary system the proposed architecture allows to perform the same computation 20 times faster alongside with only 10 % of the circuit area resources.

Keywords Computer arithmetic · Residue number system · Large dynamic range · Chinese remainder theorem · Mixed radix conversion

This work was partially supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) under project PEst-OE/EEI/LA0021/2013, project "FARNuSyC - Framework for Automatic RNS-Based Computation" (reference number EXPL/EEI-ELC/1572/2013), and by the PROTEC Program funds under the research grant SFRH/PROTEC/49763/2009.

P. M. Matutino (✉)
ISEL / INESC-ID / IST, Universidade de Lisboa,
Lisbon, Portugal
e-mail: pmmm@sips.inesc-id.pt

R. Chaves · L. Sousa
INESC-ID / IST, Universidade de Lisboa,
Lisbon, Portugal

R. Chaves
e-mail: ricardo.chaves@inesc-id.pt

L. Sousa
e-mail: las@inesc-id.pt

1 Introduction

Residue Number System (RNS) is a non-weighted numbering system using remainders to represent numbers [28], being an alternative to the typical binary system. The basic arithmetic operations (add, subtract, and multiply) are performed over operands that are significantly shorter than the equivalent binary representation, offering the potential for high-speed, parallel arithmetic operations, given its carry-free arithmetic properties. Typical applications for RNS are in Digital Signal Processing (DSP), namely filtering, convolutions, correlations, FFT computations, and more recently cryptography [3, 4, 6, 9, 10, 12, 26].

The RNS moduli sets are composed by several m_i channels, where m_i represents positive relatively prime integers. In RNS, a number X is represented by its residues $x_i = \langle X \rangle_{m_i}$, where x_i is the remainder of the division of X by m_i. The Dynamic Range (M), determined by the Least Common Multiple (LCM), for this moduli set is given by:

$$M = LCM\left(m_0, m_1, \cdots, m_j\right) = \prod_{i=0}^{j} m_i \ . \tag{1}$$

In a typical RNS three different components have to be considered: *i*) conversion from binary-to-RNS; *ii*) arithmetic

calculations; and *iii*) conversion from RNS-to-binary, as depicted in Fig. 1.

In the first component, all the operands have to be converted to RNS, computing the remainder of the division of each operand by all m_i. Considering X and Y as two integers, represented in RNS by $\{x_0, x_1, \cdots, x_i\}$ and $\{y_0, y_1, \cdots, y_i\}$ respectively, the result of the operation $Z = X \circ Y$ is given by:

$$\{z_0, z_1, \cdots, z_i\} = \{x_0 \circ y_0, x_1 \circ y_1, \cdots, x_i \circ y_i\}\,, \tag{2}$$

where, $\circ$ denotes the arithmetic operation of addition, subtraction, or multiplication. These arithmetic calculations are performed in parallel and independently on each channel. The last component is the conversion from RNS to binary.

The conversion from the binary system to the RNS (direct conversion or binary-to-RNS), and vice-versa (reverse conversion or RNS-to-binary), introduces an overhead to the system, and is considered the main drawback of RNS. Despite this overhead, the faster arithmetic operations in each of the RNS channels can lead to better performances than the weighted binary system.

The choice of the moduli set is one of the most important aspects, in order to obtain an improved and balanced system able to exploit the parallelism for the required Dynamic Range (DR). The use of moduli sets with modulo $\{2^n \pm k\}$ channels, with unrestricted k values, is rather useful in the definition of larger RNS moduli sets, resulting in arithmetic systems with better performances, given that the operands in each channel require a smaller number of bits.

The state of the art on the design of direct converters can be divided into two main categories, namely for specific moduli sets and for more generic approaches. For specific moduli sets using the modulo $\{2^n \pm 1\}$ and $\{2^n \pm 3\}$ two structures are proposed in [33] and [13, 14], respectively. A generic conversion method is proposed in [19], for the direct conversion modulo $\{2^n \pm k\}$ using the periodic proprieties of modulo, and is implemented using a ROM-based topology. Nevertheless, the state of the art is mainly focused on RNS moduli sets based on co-prime numbers of the form $\{2^n - 1, 2^n, 2^n + 1\}$ [2, 5, 8, 20, 21, 23, 25, 31]. Different moduli sets for the RNS processors have been proposed in order to increase the Dynamic Range or to reduce the width of RNS channels, such as the moduli sets $\{2^n-3, 2^n-1, 2^n+1, 2^n+3\}$ [2, 22], $\{2^n-1, 2^{n+\beta}, 2^n+1, \}$ [5], and $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} - 1\}$ or $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} + 1\}$ [16]. A moduli set with a $8n$-bit DR has been proposed in [24], using only modulo $\{2^n \pm 1\}$ [33] channel operations.

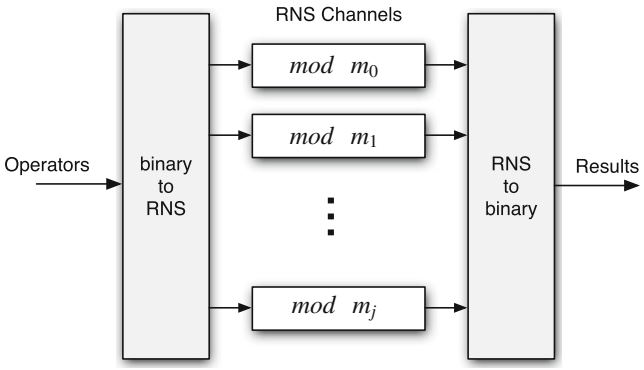


Figure 1 Residue number system architecture.

More recently, the moduli set with a DR up to $(8n+1)$-bit has been proposed. This set [18] is composed by the moduli $\{2^{n+\beta}, 2^n-1, 2^n+1, 2^n-2^{\frac{n+1}{2}}+1, 2^n+2^{\frac{n+1}{2}}+1, 2^{n+1}+1\}$, requiring arithmetic units modulo $\{2^n \pm 1\}$ [33], and generic arithmetic units modulo $\{2^n \pm k\}$ [11] for the modulo $\{2^n - 2^{\frac{n+1}{2}} + 1, 2^n + 2^{\frac{n+1}{2}} + 1\}$ channels. However, to the best of the authors knowledge, the existing moduli sets are restricted up to eight channels considering specific modulo values.

This paper improves and extends our previous work [15], proposing a unified structure for a scalable RNS computation based on $\{2^n \pm k_i\}$ moduli channels, allowing the design of RNS with any moduli set.

The considered moduli allows the arbitrary increase of the number of RNS channels and consequently, the increase of the DR, or the reduction of the width of the channels leading to the reduction in the delay, and area cost, and further exploiting the RNS parallelism.

The proposed full RNS architecture allows computing the conversion, from binary-to-RNS and RNS-to-binary, and the arithmetic operations of each channel. Herein, two RNS-to-binary conversion approaches are considered and studied, one parallel and one serial, namely one base on the Chinese Remainder Theorem (CRT) and the other on the Mixed-Radix-Converter (MRC) [17], respectively. In order to obtain a more compact RNS structure, the arithmetic units of each arithmetic channel are also used in the conversion computation. In order to evaluate the performance of the proposed architecture, theoretical and experimental results, for a 90 *nm* standard cell CMOS technology, were obtained. The obtained results suggest that the proposed RNS allows obtaining a system with significantly larger DR and allowing for a significant improvement of the performance. These results also suggest that the added conversion penalty of the more complex moduli set, for the proposed RNS, are mitigated after few arithmetic operations performed in parallel on the channels. The analysis of these results suggests that the RNS arithmetic achieves better efficiency than the binary, with gains of more than 20 times in delay alongside an area reduction of 90 %. Moreover, when comparing with the state of the art the proposed architecture achieves better efficiency for the same amount of moduli channels, and up to 50 % better metrics when considering 16 to 32 channels.

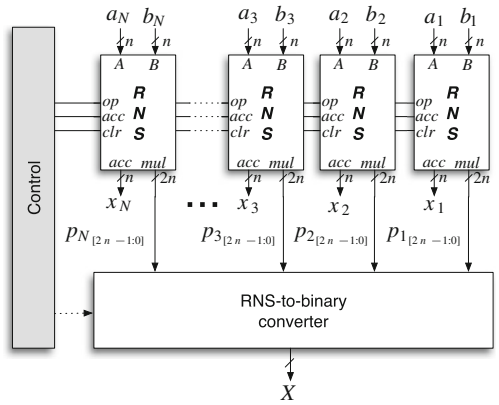


Figure 2 Proposed RNS architecture.

The results also suggest that the proposed architecture is the most compact solution for the majority of the case studies.

This paper is organized as follows. Section 2 describes the proposed RNS architecture, together with the formulation needed to design the channel's arithmetic modulo $\{2^n \pm k_i\}$ and the conversions between binary and RNS. Sections 3 and 4 present the hardware complexity analysis and the experimental results obtained with the proposed architecture, respectively. Conclusions for this work are presented in Section 5.

2 Proposed Architecture

The first step of an RNS calculation process is the conversion from binary-to-RNS, followed by the arithmetic operations performed in each channel, and finally the conversion from RNS-to-binary. The approach herein proposed considers that the number of required conversions is less than the number of arithmetic operations performed in the channels. Therefore, these two conversion steps are executed in a serial way using the hardware resources of the modular channels. The architecture herein proposed is organized in three main blocks, depicted in Fig. 2: *i*) channel arithmetic blocks; *ii*) RNS-to-binary converter; and *iii*) control block.

The arithmetic blocks perform the modular addition, subtraction and multiplication on each channel. They also, perform the binary-to-RNS conversion without imposing area overheads. The RNS-to-binary converter module contains the additional circuits required to compute the reverse conversion, which cannot be made with the channel's arithmetic blocks.

2.1 Binary-to-RNS Conversion

The conversion of a binary number X with jn-bits to RNS for modulo $\{2^n - k\}$ can be computed as follows, where $X_{[msb:lsb]}$ represents the bits msb to lsb of integer X, by considering lsb of X as the bit 0:

$$\begin{aligned}\langle X\rangle_{2^n-k} &= \left\langle \sum_{i=0}^{j-1} 2^{i\cdot n} X_{[(i+1)\cdot n-1:i\cdot n]} \right\rangle_{2^n-k} \\ &= \left\langle \sum_{i=0}^{j-1} \left(2^{(i-1)\cdot n}\cdot k\right) X_{[(i+1)\cdot n-1:i\cdot n]} \right\rangle_{2^n-k} \\ &= \left\langle \sum_{i=0}^{j-1} k^i X_{[(i+1)\cdot n-1:i\cdot n]} \right\rangle_{2^n-k} . \end{aligned} \tag{3}$$

Similarly to modulo $\{2^n - k\}$ and considering $X_i = X_{[(i+1)\cdot n-1:i\cdot n]}$, the binary-to-RNS conversion modulo $\{2^n + k\}$, can be computed as:

$$\begin{aligned}\langle X\rangle_{2^n+k} &= \left\langle \sum_{i=0}^{j-1} 2^{i\cdot n} X_{[(i+1)\cdot n-1:i\cdot n]} \right\rangle_{2^n+k} \\ &= \left\langle \sum_{i=0}^{j-1} (-k)^i \cdot X_i \right\rangle_{2^n+k} \\ &= \left\langle \sum_{i=0}^{\lfloor \frac{j-1}{2} \rfloor} k^{2i} X_{2i} - \sum_{i=0}^{\lfloor \frac{j-1}{2} \rfloor} k^{2i+1} X_{2i+1} \right\rangle_{2^n+k} . \end{aligned} \tag{4}$$

To compute the above reduction $\langle X\rangle_{2^n+k}$, modular subtracters are required. Moreover, (4) can be modified to use only addition operations, since:

$$\begin{aligned}\left\langle -X_{[n-1:0]}\right\rangle_{2^n+k} &= \left\langle 2^n - 1 - X_{[n-1:0]} + k + 1\right\rangle_{2^n+k} \\ &= \left\langle \overline{X_{[n-1:0]}} + k + 1\right\rangle_{2^n+k} . \end{aligned} \tag{5}$$

Thus, $\langle X\rangle_{2^n+k}$ can be computed as:

$$\begin{aligned}\langle X\rangle_{2^n+k} &= \left\langle \sum_{i=0}^{\lfloor \frac{j-1}{2} \rfloor} k^{2i} X_{2i} \right. \\ &\qquad \left. + \sum_{i=0}^{\lfloor \frac{j-1}{2} \rfloor} k^{2i+1} \left(\overline{X_i} + k + 1\right) \right\rangle_{2^n+k} \\ &= \left\langle \sum_{i=0}^{\lfloor \frac{j-1}{2} \rfloor} k^{2i} X_{2i} + \sum_{i=0}^{\lfloor \frac{j-1}{2} \rfloor} k^{2i+1} \overline{X_{2i}} \right. \\ &\qquad \left. + \sum_{i=0}^{\lfloor \frac{j-1}{2} \rfloor} k^{2i+1} (k+1) \right\rangle_{2^n+k} . \end{aligned} \tag{6}$$

The last term is a constant value that can be pre-calculated.

These operations can be serially performed using the channel's arithmetic blocks.

2.2 Arithmetic in the Channel

This section analyses the modular arithmetic operations required for addition, subtraction, and multiplication, with and without accumulation capability, for modulo $\{2^n \pm k\}$. The channel structure proposed to compute the needed arithmetic operations are herein described for modulo $\{2^n - k\}$ and $\{2^n + k\}$.

2.2.1 Modulo $\{2^n - k\}$

The addition modulo $\{2^n - k\}$ can be easily computed as described as in [11]. To derive a subtraction of two residue values, let us start by computing the symmetric of a residue as:

$$\begin{aligned}\left\langle -x_{[n-1:0]}\right\rangle_{2^n-k} &= \left\langle 2^n - 1 - x_{[n-1:0]} - k + 1\right\rangle_{2^n-k} \\ &= \left\langle \overline{x_{[n-1:0]}} - k + 1\right\rangle_{2^n-k} . \end{aligned} \tag{7}$$

The subtraction between the residue a and b with accumulation, can be described as:

$$\begin{aligned} acc_{q+1} &= \left\langle acc_q + a - b\right\rangle_{2^n-k} \\ &= \left\langle acc_{q_{[n-1:0]}} + a_{[n-1:0]} + \overline{b_{[n-1:0]}} - k + 1\right\rangle_{2^n-k} . \end{aligned} \tag{8}$$

Identically, the subtraction of $(a + b)$ from the accumulated value is given by:

$$\begin{aligned} acc_{q+1} &= \left\langle acc_q - (a + b)\right\rangle_{2^n-k} \\ &= \left\langle acc_{q_{[n-1:0]}} + \overline{a_{[n-1:0]}} + \overline{b_{[n-1:0]}} + 2 \cdot (1 - k)\right\rangle_{2^n-k} . \end{aligned} \tag{9}$$

The multiplication of the residue a by b, with positive accumulation, can be computed as in [11] (note: the width of k is represented by w_k, and $w_k = log_2(k) \leq n/2$, $m^1_{[n+w_k-1:0]}$ represents the result of the k constant multiplication by $p_{[2n-1:n]}$, and $m^2_{[2w_k-1:0]}$ represents the constant multiplication of k by $m^1_{[n+w_k-1:n]}$):

$$\begin{aligned} acc_{q+1} &= \left\langle acc_q + a \times b\right\rangle_{2^n-k} \\ &= \left\langle acc_{q_{[n-1:0]}} + p_{[2n-1:0]}\right\rangle_{2^n-k} \\ &= \left\langle acc_{q_{[n-1:0]}} + k \cdot p_{[2n-1:n]} + p_{[n-1:0]}\right\rangle_{2^n-k} \\ &= \left\langle acc_{q_{[n-1:0]}} + m^1_{[n+w_k-1:0]} + p_{[n-1:0]}\right\rangle_{2^n-k} \\ &= \left\langle acc_{q_{[n-1:0]}} + m^2_{[2w_k-1:0]} + m^1_{[n-1:0]} + p_{[n-1:0]}\right\rangle_{2^n-k} . \end{aligned} \tag{10}$$

The value $p_{[2n-1:0]}$ results from the binary multiplication of $a \times b$, and can be computed by a $n \times n$-bit binary multiplier.

The same multiplication but with negative accumulation, can be obtained by computing:

$$\begin{aligned} acc_{q+1} &= \left\langle acc_q - a \times b\right\rangle_{2^n-k} \\ &= \left\langle acc_{q_{[n-1:0]}} - \left(m^2_{[2w_k-1:0]} + m^1_{[n-1:0]} + p_{[n-1:0]}\right)\right\rangle_{2^n-k} \\ &= \left\langle acc_{q_{[n-1:0]}} + \overline{m^2_{[2w_k-1:0]}} + \overline{m^1_{[n-1:0]}} + \overline{p_{[n-1:0]}} \right. \\ &\quad \left. +3 \cdot (1 - k)\right\rangle_{2^n-k} . \end{aligned} \tag{11}$$

2.2.2 Modulo $\{2^n + k\}$

Similarly to modulo $\{2^n - k\}$, additions modulo $\{2^n + k\}$ can be easily computed. The subtraction operations, with accumulation, for channels modulo $\{2^n + k\}$, considering (5), can be formulated as:

$$\begin{aligned} acc_{q+1} &= \left\langle acc_q + a - b\right\rangle_{2^n+k} \\ &= \left\langle 2^n \cdot \left(acc_{q_{[n]}} + a_{[n]} - b_{[n]}\right) + acc_{q_{[n-1:0]}} \right. \\ &\quad \left. + a_{[n-1:0]} + \overline{b_{[n-1:0]}} + k + 1\right\rangle_{2^n+k} \\ &= \left\langle acc_{q_{[n-1:0]}} + a_{[n-1:0]} + \overline{b_{[n-1:0]}} \right. \\ &\quad \left. + \left(1 + k \cdot \left(1 + b_{[n]} - acc_{q[n]} - a_{[n]}\right)\right)\right\rangle_{2^n+k} . \end{aligned} \tag{12}$$

Using the same approach, the subtraction of $(a + b)$ with accumulation can be computed as:

$$\begin{aligned} acc_{q+1} &= \left\langle acc_q - (a + b)\right\rangle_{2^n+k} \\ &= \left\langle acc_{q_{[n-1:0]}} + \overline{a_{[n-1:0]}} + \overline{b_{[n-1:0]}} \right. \\ &\quad \left. + \left(2 + k \cdot \left(2 + b_{[n]} - acc_{q[n]} + a_{[n]}\right)\right)\right\rangle_{2^n+k} . \end{aligned} \tag{13}$$

Similarly to modulo $\{2^n - k\}$, the multiplication of a by b, with positive accumulation, modulo $\{2^n + k\}$, can be computed as:

$$\begin{aligned} acc_{q+1} &= \left\langle acc_q + a \times b\right\rangle_{2^n+k} \\ &= \left\langle acc_{q_{[n:0]}} + a_{[n:0]} \times b_{[n:0]}\right\rangle_{2^n+k} \\ &= \left\langle acc_{q_{[n:0]}} + p_{[2n+1:0]}\right\rangle_{2^n+k} \\ &= \left\langle acc_{q_{[n:0]}} + k^2 \cdot p_{[2n+1:2n]} - k \cdot p_{[2n-1:n]} \right. \\ &\quad \left. + p_{[n-1:0]}\right\rangle_{2^n+k} \\ &= \left\langle acc_{q_{[n:0]}} + k^2 \cdot p_{[2n+1:2n]} + k \cdot \overline{m^1_{[n+w_k-1:n]}} \right. \\ &\quad \left. + m^1_{[n-1:0]} + p_{[n-1:0]} + 2 \cdot (k + 1)\right\rangle_{2^n+k} \\ &= \left\langle acc_{q_{[n-1:0]}} + m^2_{[2w_k-1:0]} + m^1_{[n-1:0]} + p_{[n-1:0]} \right. \\ &\quad \left. + k^2 \cdot p_{[2n+1:2n]} + 2 \cdot (k + 1) - k \cdot acc_{q[n]}\right\rangle_{2^n+k} . \end{aligned} \tag{14}$$

Using (5) to rewrite (14) considering negative accumulation, results in:

$$acc_{q+1} \left\langle acc_q - a \times b \right\rangle_{2^n+k} = \Big\langle acc_{q_{[n-1:0]}} + \overline{m^2_{[2w_k-1:0]}} + \overline{m^1_{[n-1:0]}} + \overline{p_{[n-1:0]}} + (k+1) - k^2 \cdot p_{[2n+1:2n]} - k \cdot acc_{q_{[n]}} \Big\rangle_{2^n+k} . \quad (15)$$

The described operations, can be implemented in a single hardware structure, using a binary multiplier, two constant multipliers (for computing $(k \cdot p_{[2n-1:n]})$ and $(k \cdot m^1_{[n+w_k-1:n]})$) used to perform the modular reduction, and one 5:1 modular adder. This 5:1 modular adder may be implemented by using one modular carry-save-adder [1] and one 4:1 adder [11]. The arithmetic structure for channels modulo $\{2^n - k\}$ and $\{2^n + k\}$ are similar. However, for channels modulo $\{2^n + k\}$, a $((n+1) \times (n+1))$-bit binary multiplier is used, instead of a $(n \times n)$-bit multiplier. Two constant multipliers are also used, but more constants have to be selected at the input of the 5:1 modular adder, as depicted in Figs. 3 and 4.

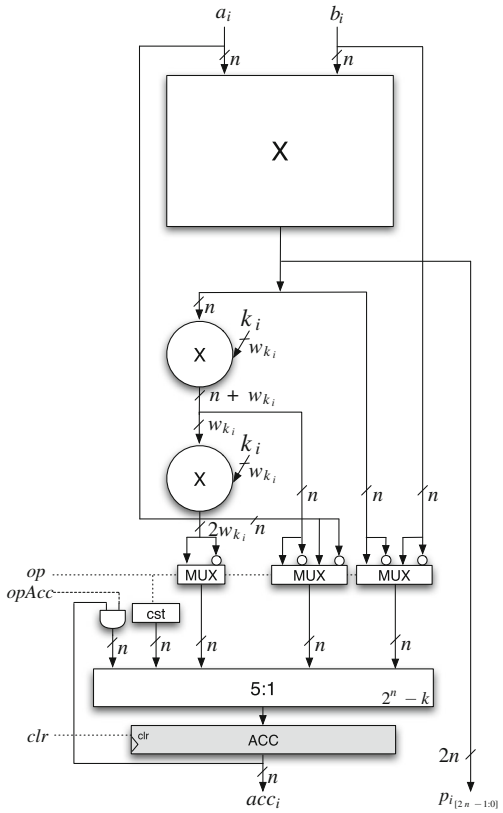


Figure 3 Channel structure *modulo* $\{2^n - k\}$.

2.3 RNS-to-Binary Conversion

The related reverse converters state of art is based on the Chinese Remainder Theorem (CRT) [17], on the Mixed-Radix-Converter (MRC) [17] or on the more recent New CRT [30]. Herein, both the CRT and MRC algorithms are considered. The first approach results in a more parallel computation while the second one results on a more serial approach but requiring simpler modular arithmetic operations. Both algorithms allow to reuse the hardware for modular arithmetic individually available in the channels, to perform the conversion from RNS-to-binary in order to reduce the system's overall area. Nevertheless, the CRT approach requires additional hardware to implement the modular adder modulo M, mandatory for the computation of the final binary value. The New CRT I [30] is herein not considered since modulo channels have to satisfy the condition $P_i > 2P_{i-1}$, imposing an unbalanced system. The New CRT II [30] requires intermediate modular operations that are not supported by the modular channels; given its added cost, it is not herein considered.

The computation of X using the CRT algorithm, for N residues, can be achieved by [17]:

$$\langle X \rangle_M = \left\langle \sum_{i=1}^{N} x_i \cdot \left\langle M_i^{-1} \right\rangle_{m_i} \cdot M_i \right\rangle_M = \left\langle \sum_{i=1}^{N} x_i \cdot W_i \right\rangle_M , \quad (16)$$

where W_i is the weight of the residue m_i, given by:

$$W_i = \left\langle M_i^{-1} \right\rangle_{m_i} \cdot M_i , \quad (17)$$

where M_i^{-1} is the multiplicative inverse of M_i, such that with $M_i = M/m_i$ and $\left\langle M_i^{-1} \cdot M_i \right\rangle_{m_i} = 1$.

Given x_i, the n-bit residue of the division of X by m_i, and W_i a constant multiplication factor, with a dynamic range of m bits:

$$x_i = \sum_{j=0}^{n-1} x_{i_j} 2^j , \quad W_i = \sum_{j=0}^{m-1} W_{i_j} 2^j , \quad (18)$$

the product $x_i \cdot W_i$, can be computed by decomposing the operand W_i into blocks of n bits, from the higher to the lower bits, where $c = N - 1$, and W_{i_l} represents the $[l \cdot (n+1) - 1 : l \cdot n]$ bits of W_i, given by:

$$W_i = (2^n)^{N-1} \cdot \sum_{j=m-n}^{m-1} W_{i_j} 2^{j-m+n} + \cdots + \sum_{j=0}^{n-1} W_{i_j} 2^j = 2^{c \cdot n} \cdot W_{i_c} + \cdots + 2^n \cdot W_{i_1} + W_{i_0} . \quad (19)$$

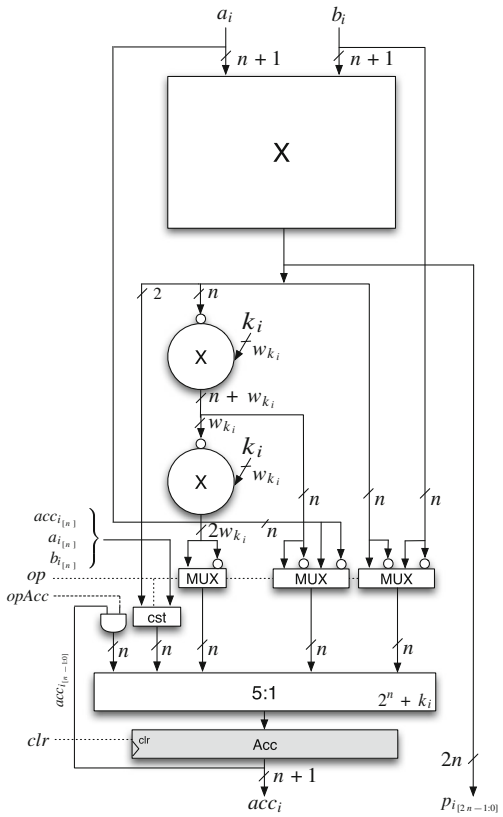


Figure 4 Channel structure *modulo* $\{2^n + k\}$.

Applying this decomposition to the modular multiplication in (16), the computation of X is given by:

$$\langle X\rangle_M = \left\langle \sum_{i=1}^{N} x_i \cdot \left(2^{cn}\cdot W_{i_c} + \cdots + 2^n \cdot W_{i_1} + W_{i_0}\right)\right\rangle_M$$
$$= \left\langle 2^{cn}\cdot \sum_{i=1}^{N} x_i \cdot W_{i_c} + \cdots + \sum_{i=1}^{N} x_i \cdot W_{i_0}\right\rangle_M . \quad (20)$$

In the proposed RNS architecture this constant multiplication is computed by the multiplication units of each RNS channel, requiring N steps to perform all the constant multiplications. On each iteration the several constant multiplication results are added in a binary adder-tree, compressing the N input values, with $2n$-bit length, into one vector of $2n + e$ bits (note that: $e = \lceil log_2(N)\rceil$ represents the extra bits required). This result is then shifted to compute the multiplication by 2^{ln}. The shifted value is feed into the m-bit modulo M adder to compute the binary value of X, as depicted in Fig. 5. This CRT base conversion structure requires $N + 1$ steps to compute the conversion of X, from RNS to binary.

The other reverse conversion approach considers the MRC algorithm. This approach associates a mixed-radix representation with a residue representation. Considering the moduli set $\{m_1, m_2, \cdots, m_N\}$, where z_i is the residue modulo m_i $(0 \le z_i < m_i)$, the value X can be described as:

$$X = z_N \cdot m_{N-1} \cdot m_{N-2} \cdots m_1 + \cdots + z_2 \cdot m_1 + z_1 . \quad (21)$$

Applying the modular reduction m_1 to X, equation (21) can be re-written as:

$$z_1 = \langle X\rangle_{m_1} = x_1 . \quad (22)$$

In order to calculate z_2 we can consider:

$$X - z_1 = z_N \cdot m_{N-1} \cdot m_{N-2} \cdots m_1 + \cdots + z_2 \cdot m_1 . \quad (23)$$

Applying the reduction modulo m_2 to (23) results:

$$\langle X - z_1\rangle_{m_2} = \langle z_2 \cdot m_1\rangle_{m_2} . \quad (24)$$

Multiplying by $\left\langle m_1^{-1}\right\rangle_{m_2}$ results:

$$\left\langle \left\langle m_1^{-1}\right\rangle_{m_2} (X - z_1)\right\rangle_{m_2} = \langle z_2\rangle_{m_2} . \quad (25)$$

Since z_2 is always a residue modulo m_2, and $\langle X\rangle_{m_2} = x_2$, equation (25) can be re-written as:

$$z_2 = \left\langle \left\langle m_1^{-1}\right\rangle_{m_2} \cdot \left(\langle X\rangle_{m_2} - \langle z_1\rangle_{m_2}\right)\right\rangle_{m_2}$$
$$= \left\langle \left\langle m_1^{-1}\right\rangle_{m_2} \cdot (x_2 - z_1)\right\rangle_{m_2} . \quad (26)$$

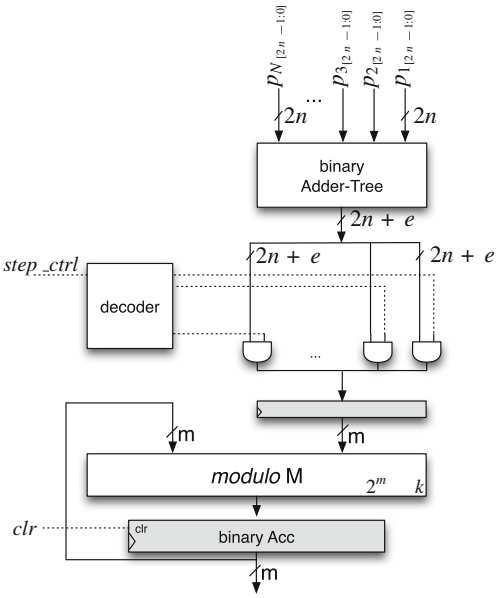


Figure 5 CRT based conversion block.

Continuing the process, the mixed-radix digits (z_i) can be iteratively calculated. The z_N value can thus be computed as:

$$z_N = \left\langle \left\langle (m_{N-1} \cdots m_1)^{-1} \right\rangle_{m_N} \times \left(x_N - \langle z_{N-1} \cdot m_{N-2} \cdots m_1 + \cdots + z_2 \cdot m_1 + z_1 \rangle_{m_N} \right) \right\rangle_{m_N} . \tag{27}$$

This iterative process requires $2N$ cycles for the reverse conversion computations, and can be performed by the arithmetic channels of the proposed RNS structure. Furthermore, all the multiplicative inverse values required in the computation can be pre-computed and stored in memory. The final value X can be computed by multiplying the mixed-radix digits (z_i) by a constant factor (W_i), as represented in (21).

Considering z_i and W_i as:

$$z_i = \sum_{j=0}^{n-1} x_{i_j} 2^j \ , \quad W_i = \prod_{j=1}^{i-1} m_j = \sum_{j=0}^{n\cdot(i-1)} W_{i_{[j]}} 2^j \ . \tag{28}$$

The multiplication of digit z_i by its weight W_i, can be computed identically to CRT (20) as:

$$X = 2^{cn} \cdot \sum_{i=1}^{N} z_i \cdot W_{i_c} + \cdots + \sum_{i=1}^{N} z_i \cdot W_{i_0} \ . \tag{29}$$

The computation of (29) requires N additional steps to compute the final value of X. The final value X is thus computed by a binary adder-tree, compressing the $N+1$ input values, with $2n$-bit length, into one vector of $2n+e$ bits (note that: $e = \lceil log_2(N+1) \rceil$ represents the extra bits required). The shifted value is stored into a register. The binary adder-tree is fed with the results given by the binary multiplier of each channel, output $p_{i_{[2n-1:0]}}$ of the arithmetic channel structure, depicted in Fig. 4. With this approach, the conversion from RNS-to-binary requires a total of $3N$ cycles to compute X. The hardware overhead for this solution is reduced when compared with the CRT algorithm, which requires a final modulo M adder, as depicted in Fig. 6.

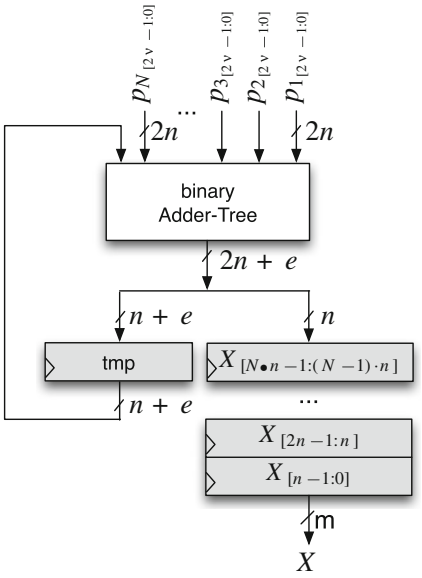


Figure 6 MRC based conversion block.

3 Hardware Requirements

To obtain a technology independent assessment of the resulting RNS architecture, an analysis has been carried out using a neutral Full Adder based model [18]. This model relates the area and propagation delay of the circuit, with the area and delay of a Full-Adder (FA), considering the FA as the finest grain component. The estimation model considers the area of 1-bit FA is represented by Δ_{FA}, and τ_{FA} represents its delay. Bitwise logic operations are not considered, since they do not impose a significant delay or area cost in our design. Furthermore, it considers a Carry-Propagate-Adder (CPA) as the simplest addition structure, with $n\Delta_{FA}$ area and $n\tau_{FA}$ delay cost. This analysis also considers that a modulo $\{2^n \pm 1\}$ adder is implemented by a CPA with EAC (End-Around-Carry), with an area cost similar to a binary CPA and twice the delay of a CPA [27].

The Multi-Operand Modular Adders for modulo $\{2^n \pm 1\}$ (MOMA [29]) and binary Adder-Trees are considered to be organized as Wallace Trees, requiring approximately $log_{1.5}(N) \simeq 2log_2(N)$ stages for N operands [32], with a 1-bit FA delay per stage, and a total of $n(N-2)\Delta_{FA}$ of area resources.

For the binary multipliers a delay of $2n\tau_{FA}$ and an area of n^2 FA is assumed [11]. For the multiplication of a n-bit operand by a w_k-bit constant, a simpler estimation is considered, with $nw_k\Delta_{FA}$ of area resources usage and $(n+w_k)\tau_{FA}$ of delay.

The modulo $\{2^n \pm k\}$ adders proposed in [11] are considered, with a 4:1 adder imposing a delay of $(n+3)\tau_{FA}$ and an area of $5n\Delta_{FA}$. For the modulo $\{2^n \pm k\}$ Carry-Save-Adder (CSA) structure, a delay of $2\tau_{FA}$ and an area requirement of $2n\Delta_{FA}$ is considered [1].

Given these metrics, the first step to assess the efficiency of the proposed compact RNS architecture is to analysis the channel's arithmetic structures, described in Section 2.2 and depicted in Fig. 4. The resulting area requirements for the modulo $\{2^n - k\}$ channel arithmetic block is imposed by the area of one binary multiplier ($n^2\Delta_{FA}$), two constant multipliers, contributing with $2nw_k\Delta_{FA}$, and one 5:1 modular adder. The 5:1 modulo adder is implemented with one 4:1 [11] adder and one modular CSA [1], with a total area cost of $7n\Delta_{FA}$. As illustrated in Fig. 4, the critical path of

the resulting channel arithmetic structure is imposed by the binary multiplier, with $2n\tau_{FA}$, the two constant multipliers, contributing with $2(n + w_k)\tau_{FA}$, and the 5:1 modular adder with $(n + 5)\tau_{FA}$ of delay. An identical analysis can be performed for the modulo $\{2^n + k\}$ channel's arithmetic, resulting in the values depicted in Table 1.

The estimation of the cost for moduli $\{2^n\}$ and $\{2^n \pm 1\}$ channels are also presented in Table 1, since the considered related art employs these moduli. For modulo 2^n, the area cost is imposed by the n-bit output, binary multiplier ($\frac{n^2}{2}\Delta_{FA}$), one CSA to compute the addition of the operands A and B with the accumulated value, contributing with $n\Delta_{FA}$, and one CPA as the final adder with an $n\Delta_{FA}$ area cost. The estimated delay for modulo 2^n channel structure is $(n + 1 + n)\tau_{FA}$, resulting from the contribution of the binary multiplier, the CSA, and the final CPA. Identically, for modulo $\{2^n \pm 1\}$ results an estimated delay of $(2n + 1 + 2n)\tau_{FA}$, given by the contribution of the modulo multiplier [32], the CSA, and the final CPA with EAC. The estimated area is $(n^2 + n + n)\Delta_{FA}$, as presented in Table 1.

To complete the evaluation of the proposed RNS architecture, the reverse conversion units have also to be characterized. From Fig. 5 and the presented equations, the delay and area cost for the CRT based conversion can be derived as:

$$\begin{aligned}\Delta_{CRT} &= \Delta_{bin\ Adder-Tree} + \Delta_{add\ mod\ M} \\ &= [(2n + log_2(N))\,(N-1)]\,\Delta_{FA} + 3n \cdot N\Delta_{FA}\ ,\end{aligned} \tag{30}$$

$$\begin{aligned}\tau_{CRT} &= \tau_{FA} \cdot max \begin{cases} \tau_{CH_{2^n \pm k}} \\ \tau_{bin\ multiplier} + \tau_{bin\ Adder-Tree} \\ \tau_{add\ mod\ M} \end{cases} \\ &= \tau_{FA} \cdot max \begin{cases} (5n + 2w_{k_{max}}) + 7) \\ 4n + 3log_2(N) \\ n \cdot N + 1\ . \end{cases}\end{aligned} \tag{31}$$

As discussed in the previous section, the CRT reverse conversion requires $N + 1$ iterations, which in this case means $N + 1$ clock cycles to compute the binary value X.

For the MRC conversion it can be derived an area cost and a delay, from Fig. 6 and the presented equations of:

$$\begin{aligned}\Delta_{MRC} &= \Delta_{bin\ Adder-Tree} \\ &= [(2n + log_2(N + 1)) \cdot (N)]\,\Delta_{FA}\ ,\end{aligned} \tag{32}$$

Table 1 Area and delay estimation of channel arithmetic structures.

Modulo	Δ (area) [Δ_{FA}]	τ (delay) [τ_{FA}]
$2^n - k$	$n^2 + 7n + 2nw_k$	$5n + 2w_k + 5$
$2^n + k$	$(n + 1)^2 + 2nw_k + 7n$	$5n + 2w_k + 7$
2^n	$n^2/2 + 2n$	$2n + 1$
$2^n \pm 1$	$n^2 + 2n$	$4n + 1$

$$\begin{aligned}\tau_{MRC} &= \tau_{FA} \cdot max \begin{cases} \tau_{CH_{2^n \pm k}} \\ \tau_{binmultiplier} + \tau_{bin\ Adder-Tree} \end{cases} \\ &= \tau_{FA} \cdot max \begin{cases} (5n + 2w_{k_{max}}) + 7) \\ 4n + 3log_2(N + 1)\ . \end{cases}\end{aligned} \tag{33}$$

As discussed in the previous section, the MRC reverse conversion requires $3N$ iterations, which in this case means $3N$ clock cycles to compute the binary value X.

4 Comparison with the Related Art

In order to properly evaluate the proposed RNS architecture it is herein compared with the binary implementation and the related state of the art, namely: *i*) for the Traditional moduli set $\{2^n \pm 1, 2^{2n}\}$ with a $4n$-bit DR [5]; *ii*) the Mohan [16] for the moduli set $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} - 1\}$, which corresponds to a DR with $4n$ bits; *iii*) a RNS architecture proposed by Skavantzos [24] for the moduli set $\{2^{n-5} - 1, 2^{n-3} - 1, 2^{n-3} + 1, 2^{n-2} + 1, 2^{n-1} - 1, 2^{n-1} + 1, 2^n, 2^n + 1\}$ supporting a DR up to $(8n - 15)$ bits; and *iv*) a RNS-to-binary converter proposed by Pettenghi [18] with a DR up to $(8n + 1)$-bit supported on the moduli set $\{2^{n+\beta}, 2^n - 1, 2^n + 1, 2^n - 2^{\frac{n+1}{2}} + 1, 2^n + 2^{\frac{n+1}{2}} + 1, 2^{n+1} + 1\}$.

Table 4 presents the estimated values for these RNS architectures considering the previously described performance model. The Traditional, Skavantzos, and Mohan reverse converters were implemented by using the arithmetic operators presented in [33]. For Pettenghi, a binary-to-RNS converter and arithmetic units were added in order to obtain a complete system, considering the same units used in the proposed architecture. For the $\{2^n \pm 2^{\frac{n+1}{2}} + 1\}$ modulo channels, generic modulo $\{2^n \pm k\}$ arithmetic units are considered. Thus, the main difference in the full RNS lies in the reverse conversion, which consider a dedicated binary-to-RNS converter.

The proposed architecture takes advantage when it is consider a larger number of RNS channels. In order to quantify the maximum number of channels allowed and the resulting moduli sets, the number of available moduli were counted for different values of n. Table 2 depicts the number of relative prime numbers available for moduli sets composed by $\{2^n \pm k_i\}$ moduli, and for moduli sets only composed by $\{2^n - k_i\}$ moduli.

Table 2 Number of relative prime numbers available for n-bit channel length.

Channel width n[bit]	8	12	16	20	···	32
$\{2^n - k\}$	4	11	41	127	···	4713
$\{2^n \pm k\}$	8	22	82	232	···	8675

Given the large amount of relative prime numbers, in particular for larger values of n, an adequate selection approach needs to be considered. This selection needs to take into account that the constant multipliers and the final modular compression depend on the value of k, and are in the critical path of the arithmetic channels. Smaller values of k, with lower Hamming weight, lead to faster and smaller implementations of these arithmetic units, as depicted in Fig. 7. Therefore, the selection of the moduli set should look for the lower values for k. Table 3, depicts the relative prime numbers for architectures with 8, 16, and 32 channels, considering arithmetic channels with 32 bits, and using the mentioned selection approach.

If other constant multiplication or modular compressor units are used, which characteristics vary differently with the values of k, another moduli selection approach should be considered.

In order to better evaluate the cost and the performance gains obtained with the proposed RNS architecture, experimental values were normalized to the values obtained for the Binary implementation, considering the area and delay estimations presented in Table 4. Given the scalability of the proposed RNS architecture, three implementations are considered, varying the number of moduli channels, namely using moduli sets with 8, 16, and 32 channels modulo $\{2^n \pm k\}$.

From the area results estimated, depicted in Fig. 8, it can be concluded that the Skavantzos is the most area demanding system, requiring up to 2.3 and 4.6 times more area resources than the Binary and Traditional system, respectively, given the high cost of the considered ROM blocks. The Mohan and Pettenghi systems requires almost the same area resources as the Traditional solution. The proposed 8 channel system reduces the area usage up to 50 %, in comparison to the Traditional and the Mohan solutions, and up to 74 %, when compared to the Binary system. The proposed architecture with 16 and 32 channels requires significantly less area resources. In comparison with the Binary solution, the 16 and 32 channel implementations allow to reduce the area up to 90 %. An area reduction estimation of up to 85 % can be achieved when compared to the Traditional RNS system. In what concerns to the remaining state of the art, the obtained results suggest that significant area reductions can also be achieved, up to 57 %. Note that the 16 and 32 channel implementations can only be considered for larger DR, as required by cryptographic applications, given the minimum required size of each modulo channel. This area reduction is achieved through the reduction of the area at the level of the RNS channels. With the increase of the number of RNS channels, the bit length of each channel is reduced. One of the biggest area gains is achieved in the multiplication units, which have a quadratic increase with n. The MRC based system are the less area demanding, achieving up to 9 % less area than the CRT based system. The difference between the proposed CRT and the MRC

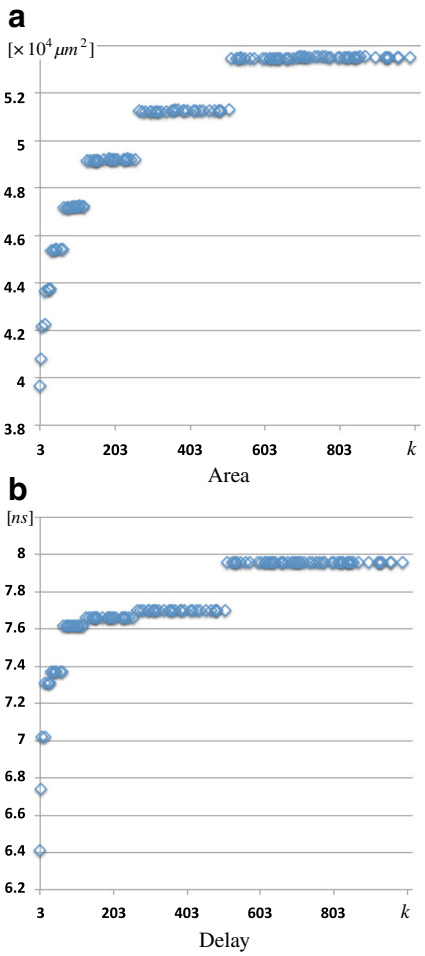


Figure 7 Experimental results for the arithmetic channels, with $n = 32$ bits, varying k.

Table 3 Moduli set adopted for the proposed architectures, ($n = 32$).

Architecture [N]	DR	moduli set composed by $\{2^{32} - k_i, \cdots, 2^{32} - k_{i+N-1}\}$, with k_i assuming the values	
8 channels	256		1, 3, 5, 9, 15, 17, 23, 27,
16 channels	512	all of the above and	29, 33, 39, 45, 47, 57, 63, 65,
32 channels	1024	all of the above and	75, 77, 83, 89, 93, 99, 105, 107, 117, 119, 129, 135, 143, 149, 153, 155

Table 4 Theoretical analysis of the full RNS.

Architecture	Block		Δ (area) [Δ_{FA}]	τ (delay) [τ_{FA}]
Binary	channel		$n^2/2 + 2n$	$2n + 1$
Traditional [5]	bin-to-RNS		$6n$	$2n + 2$
	channels		$4n^2 + 8n$	$4n + 1$
	RNS-to-bin		$6n$	$4n + 3$
Mohan [16]	bin-to-RNS		$9n$	$2n + 2$
	channels		$(7/2) \cdot n^2 + 10n + 2$	$4n + 5$
	RNS-to-bin		$n^2/2 + 15/2n + 3$	$(23/2) \cdot n + 6$
Skavantzos [24]	bin-to-RNS		$47n - 104$	$2n + 4$
	channels		$(15/2) \cdot n^2 - 14n - 75$	$4n + 1$
	RNS-to-bin		$66n^2 - 87n - 15$	$46n - 42 + 2log_2\left(16n^2 - 62n + 12\right) + log_2\left((2n - 6)n^4\right)$
Pettenghi [18]	bin-to-RNS		–	$8 \cdot (6n + 7)$
	channels		$27/9 \cdot n^2 + 30n + 4$	$6n + 7$
	RNS-to-bin		$n^2/2 + 81n/2 - 13$	$10n + 8 + log_2(n/2 + 4)$
Proposed	bin-to-RNS		–	$N \cdot (5n + 2w_{k_{max}} + 7)$
	channels		$n^2 + 9n + 2nw_{k_{max}} + 1$	$5n + 2w_{k_{max}} + 7$
	RNS-to-bin	CRT	$(2n + log_2(N + 1))(N - 1) + 3nN$	$(N + 1) \cdot (5n + 2w_{k_{max}} + 7)$
		MRC	$(2n + log_2(N + 1))(N - 1)$	$3N \cdot (5n + 2w_{k_{max}} + 7)$

based RNS is in the dedicated conversion blocks, depicted in Figs. 5 and 6; the CRT based conversion block required 2.54 more circuit area than the MRC one. It can be noticed that the CRT conversion block requires an additional final adder modulo M.

The delay analysis is difficult to realise, since it depends on the amount of arithmetic operations that needs to be performed. The overhead of the direct and reverse conversion

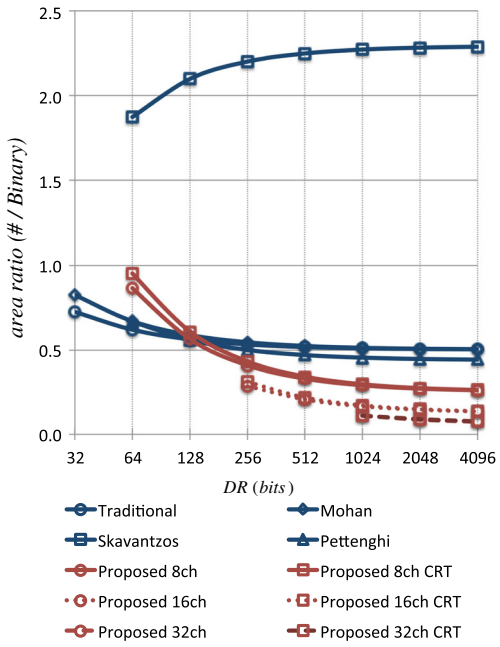


Figure 8 Total area resources of the full RNS.

becomes less significant with the increase of the number of arithmetic operations. To better evaluate this impact, the results herein presented consider 1, 10, and 100 arithmetic operations for each set of direct and reverse conversions. Figure 9 depicts the results, also normalized to the considered Binary implementation. As expected, a system using the Binary representation has better delay performances when a single arithmetic operation is considered. Given the simple moduli set it adopts, the Traditional architecture is the one suggesting less relative degradation, being around 1.3 times slower than the Binary. The architecture proposed by Mohan is 2.2 times slower than the binary, and around two times slower than the Traditional. The Skavantzos architecture also suggests a degradation in delay when compared with the Binary based system, being in the order of 3.3 times slower. The Skavantzos architecture also suggests a degradation in delay, when compared with the Traditional RNS system, being 2.6 times slower. The proposed MRC architectures, with 8, 16 and 32 modulo channels suggests the worst delay performance, when considering one arithmetic operation per conversion, to be 15 to 10 times slower than the Binary. The proposed CRT conversion imposes a lower delay degradation than the MRC conversion, being 7.5 to 5.2 times slower than the Binary. In this case, the delay performance degradation is imposed mostly by the conversions, which are heavier in RNS with more moduli channels.

Considering 10 arithmetic operations per conversion, almost all the RNS architectures suggest better delay performances than the Binary implementation. However, the

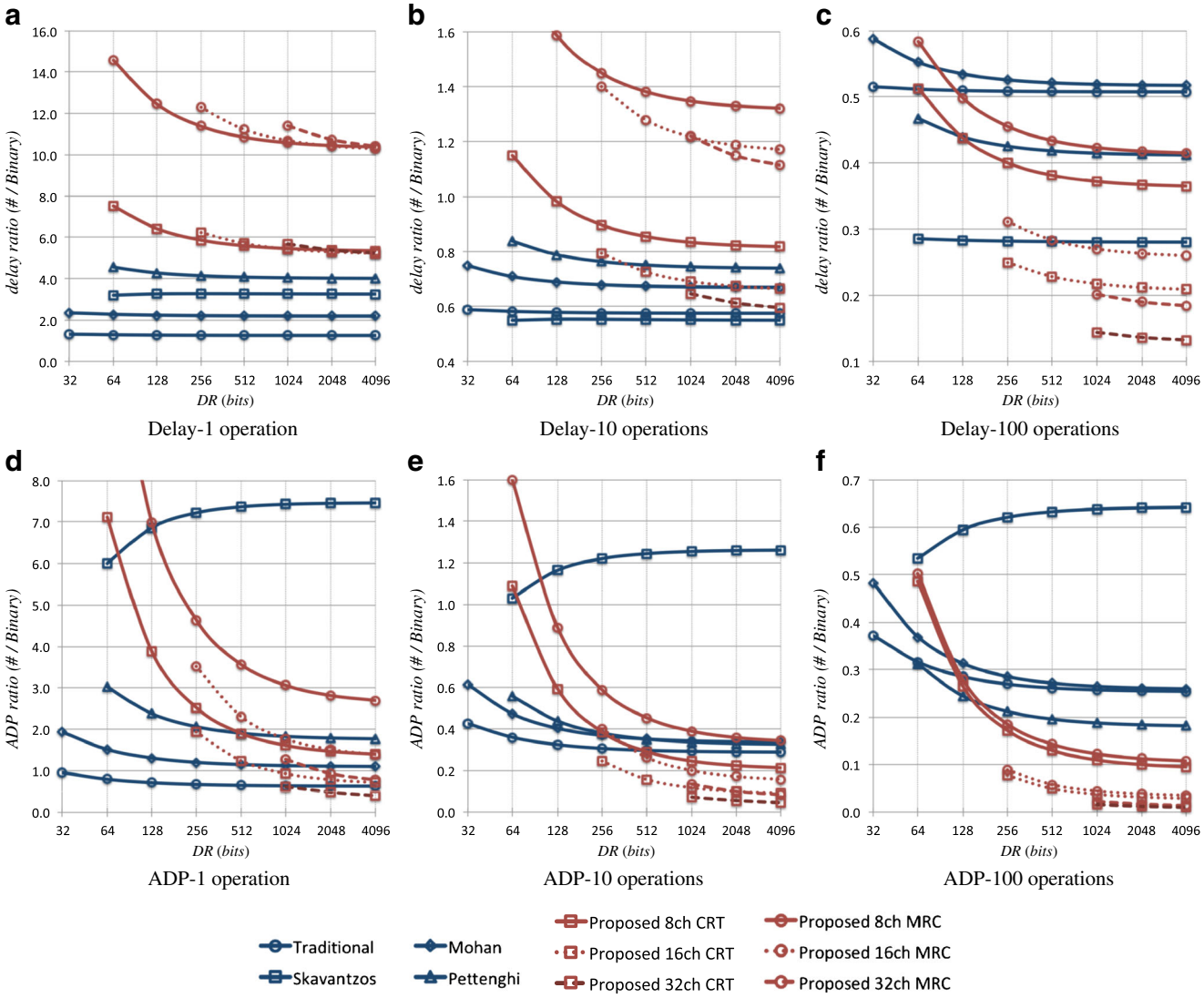


Figure 9 Delay and ADP of the full RNS for the various RNS architectures.

proposed MRC architectures are slower than the Binary for all the dynamic range considered. The proposed CRT system only has a worst delay performance than the Binary, for dynamic ranges below 128 bits. The system proposed by Skavantzos suggest the best delay performances.

When considering 100 arithmetic operations, the conversion cost becomes negligible. The Skavantzos 8 channel RNS is, as expected, the one suggesting better delay performances. This is due to the use of a moduli set composed by modulo channels in the form $\{2^n \pm 1\}$, thus allowing the use of optimized modulo arithmetic units [33]. Furthermore, it uses a dedicated structure for the conversion from RNS-to-binary. In this scenario, the proposed RNS with 16 and 32 channels suggests performance gains of up to 2.5 times in comparison with the Traditional RNS, and up to 7.5 times regarding the Binary system. The proposed RNS supported by the CRT conversion algorithm allows the lowest delay for dynamic ranges above 256 bit, being particular suited for cryptography applications.

To better evaluate the considered RNS the Area-Delay-Product (ADP) efficiency metric is also considered. Figure 9 presents the ADP efficiency metric considering 1, 10, and 100 arithmetic operations per conversion. From these results, it can be concluded that all RNS have worst ADP performance than Binary system, when considering a 32 bit dynamic range. However, the Traditional RNS solution is always the best RNS solution for 1 operation between a dynamic range of 64 and 1024 bit, as expected from the previous results. For 1 operation, with a DR greater than 2048, the proposed 32 channel system with CRT conversion has the best ADP metric.

For the 10 operations scenario, and considering the full RNS, the Traditional implementation has the best efficiency for DR up to 256. However, the proposed architecture has

better ADP values for DR greater than 256, achieving up to 50 % better RNS processors. Furthermore, the proposed architecture only has worst ADP values than Binary, for DR smaller than 64 bits, as expected since this scalable architecture is based on a serial conversion form binary-to-RNS and RNS-to-binary.

When considering 100 arithmetic operations, the proposed architecture is overall the best solution, achieving, on average, a gain of 70 % in comparison with the Traditional RNS, and gains up to 90 % when compared with the Binary system. Even thought being more efficient in the arithmetic operations, the Skavantzos RNS is always the worst implementation when considering the ADP metric, given its area requirements, in particular due to the ROM based reverse conversion, using up to 85 % of the resulting system area. The Mohan architecture always suggests worse or similar ADP metrics when compared to the Traditional based system. From the theoretical analysis the proposed CRT based architectures allows to have better ADP performance than the MRC based one.

In order to validate the theoretical analysis, the channel's arithmetic blocks were described in VHSIC Hardware Description Language (VHDL) and mapped to the UMC 90 *nm* CMOS technology [7]. Both synthesis and mapping were performed using Design Vision Version E-2010.12-SP4 from Synopsys. The experimental results were obtained in an AMD Phenom II X4 945 Quad Core at 3.0GHz, with 12GB of RAM at 1333MHz and 32GB of swap memory. The obtained experimental results for area and delay are presented in Fig. 10. Note that the results for the Binary system between 512 and 4096 bit (dashed line) where estimated based on the results obtained for the range of 32 to 256 bit, since for values greater than 256 bit was not able to synthesis the circuit. The results confirm the theoretical estimations, suggesting that, for an 8 channel RNS, the Skavantzos architecture has the best arithmetic performances to DR up to 512 bits, however at a high conversion cost [18]. The best arithmetic performances, both in area and delay, are obtained for the 16 and 32 channel RNS, using the architecture herein proposed. The results suggest arithmetic gains of more than 20 times in delay and an area reduction of 90 % regarding the Binary system. Furthermore, when comparing the proposed systems with the Traditional RNS the results suggest arithmetic gains of more than 5 times in delay and an area reduction of 77 %. Also, a delay improvement of 17 % and an area reduction of 23 % regarding the arithmetic operations performed by the Skavantzos RNS. When considering large dynamic ranges, the use of RNS with a larger number of channels, with the consequent lower channel width, leads to better delay performances in the arithmetic channels, and reduces the area requirements as depicted in Fig. 10. When considering the ADP efficiency metric, the proposed arithmetic channels have the best efficiency metric for DR greater than 512 bits. For lower DR the Skavantzos system is the more efficient implementation. However, when considering the full RNS system, the proposed architecture can lead to better efficiency metrics even for lower DR, as 256 bits.

In conclusion, the work herein proposed suggest that an efficient and scalable RNS architecture can be achieved when compared with the binary and the RNS state of the art. The results also suggest that, for the same number of channels, this architecture has similar ADP efficiency metrics than the existing state of the art, while conveying a high

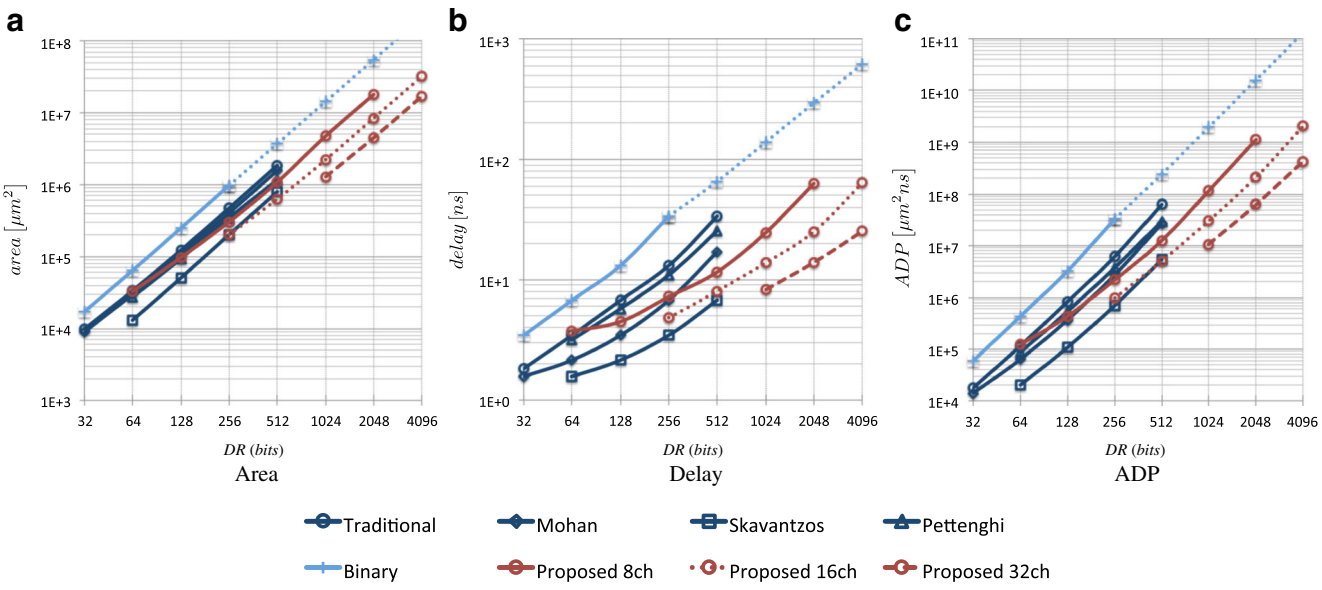


Figure 10 Experimental results of channel arithmetic blocks.

flexibility to adapt the architecture to the desired number of modulo channels and DR.

Furthermore, this work suggest that the Residue Number Systems allows better performances than the binary implementations, even for small dynamic ranges (as 32 bit) when considering at least 10 arithmetic operations.

The work herein presented can be considered as the first step towards the development of a flexible and adaptable comprehensive framework to automatically design RNS processors. This RNS framework can also be used to advance the development of novel moduli sets and optimized reverse converters.

5 Conclusions

In this paper a compact and scalable RNS architecture is proposed, allowing the design of RNS based processors for any $\{2^n \pm k_i\}$ moduli set. The herein proposed RNS architecture is composed of modulo $\{2^n \pm k_i\}$ channels, making use of a unified arithmetic structure. This structure allows modular additions, subtractions, and multiplication operations, with or without accumulation, while also supporting the needed forward and reverse conversions. The proposed architecture allows to use two reverse conversion algorithms, the Chinese Remainder Theorem and the Mixed Radix Converter. With the proposed architecture the delay and area cost of dedicated processors are reduced while further exploring the parallelism and carry-free characteristic of RNS. The presented performance analysis suggests that better performance metrics can be obtained with the proposed RNS architecture when a sufficient number of arithmetic operations are performed between conversions. When considering the complete RNS, with more modulo channels, area, and delay improvements up to 50 % can be achieved. Furthermore, when considering efficiency metrics such as ADP, the proposed architecture can be more efficient even when considering as few as 10 arithmetic operations per conversion for DR greater than 512 bits. In conclusion, the presented results suggest that the proposed RNS architecture allows a scalable and compact implementations with competitive performances when compared with the usually binary representation and the RNS related state of the art.

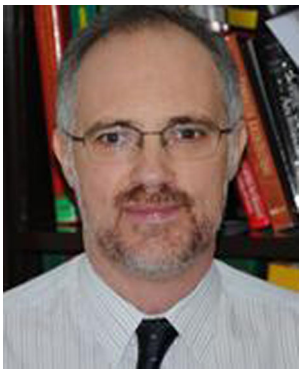
References

1. Alia, G., & Martinelli, E. (1996). Designing multi-operand modular adders. *Electronics Letters*, *32*(1), 22–23. doi:10.1049/el:19960026.
2. Ananda Mohan, P. (2004). Reverse converters for the moduli sets $\{2^{2N}-1, 2^N, 2^{2N}+1\}$ and $\{2^N-3, 2^N+1, 2^N-1, 2^N+3\}$. In *SPCOM '04* (pp. 188–192).
3. Barraclough, S., Sotheran, M., Burgin, K., Wise, A., Vadher, A., Robbins, W., Forsyth, R. (1989). *The design and implementation of the IMS A110 image and signal processor* (pp. 24.5/1 –24.5/4). doi:10.1109/CICC.1989.56826.
4. Wang, C.-L. (1994). New bit serial VLSI implementation of RNS FIR digital filters. *IEEE Transactions Circuits Systems II*, *41*(11), 768–772.
5. Chaves, R., & Sousa, L. (2004). $\{2^n+1, 2^{n+k}, 2^n-1\}$: a new RNS moduli set extension. In *EUROMICRO systems on digital system design* (pp. 210–217).
6. Chren, W. (1995). RNS-based enhancements for direct digital frequency synthesis. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, *42*(8), 516–524. doi:10.1109/82.404073.
7. Corporation, F.T. (2006). *90 nm technology standard cell library*. Tech. rep., Faraday Technology Corporation.
8. Dale Gallaher, F.E.P., & Srinivasan, P. (1997). The digit paralell method for fast RNS to weighted number system conversion for specific moduli $\{2^n-1, 2^n, 2^n+1\}$. *IEEE Transactions on Circuits and Systems - II: Analog and Digital Signal Processing*, *44*(1), 53–57.
9. Di Claudio, E.D., Piazza, F., Orlandi, G. (1995). Fast combinatorial RNS processor for DSP applications. *IEEE Transactions on Computers*, *44*(5), 624–633.
10. Ostermann, J., Bormans, J., List, P., Marpe, D., Narroschke, M., Pereira, F., Stockhammer, T., Wedi, T. (2004). Video coding with H.264/AVC: tools, performance, and complexity. *Circuits and Systems Magazine, IEEE 4*(1), 7–28.
11. Matutino, P., Pettenghi, H., Chaves, R., Sousa, L. (2012). RNS arithmetic units for modulo $\{2^n \pm k\}$. In *15th euromicro conference on digital system design (DSD), 2012* (pp. 795 –802). doi:10.1109/DSD.2012.114.
12. Matutino, P., & Sousa, L. (2008). An RNS based specific processor for computing the minimum sum-of-absolute-differences. In *11th EUROMICRO conference on digital system design architectures, methods and tools, 2008. DSD '08* (pp. 768–775). doi:10.1109/DSD.2008.107.
13. Matutino, P.M., Chaves, R., Sousa, L. (2010). Arithmetic units for RNS moduli $\{2^n-3\}$ and $\{2^n+3\}$ operations. In *13th EUROMICRO conference on digital system design: architectures, methods and tools* (pp. 243–246). doi:10.1109/DSD.2010.77.
14. Matutino, P.M., Chaves, R., Sousa, L. (2011). Binary-to-RNS conversion units for moduli $\{2^n \pm 3\}$. In *14th EUROMICRO conference on digital system design: architectures, methods and tools* (pp. 460–467).
15. Miguens Matutino, P., Chaves, R., Sousa, L. (2013). A compact and scalable rns architecture. In *IEEE 24th international conference on application-specific systems, architectures and processors (ASAP), 2013* (pp. 125–132). doi:10.1109/ASAP.2013.6567565.
16. Mohan, P., & Premkumar, A. (2007). RNS-to-binary converters for two four-moduli sets $2^n-1, 2^n, 2^n+1, 2^{n+1}-1$ and $2^n-1, 2^n, 2^n+1, 2^{n+1}+1$. *IEEE Transactions on Circuits and Systems I: Regular Papers*, *54*(6), 1245–1254. doi:10.1109/TCSI.2007.895515.
17. Omondi, A., & Premkumar, B. (Eds.) (2007). *Residue number systems: theory and implementation*. London: Imperial College Press.
18. Pettenghi, H., Chaves, R., Sousa, L. (2012). RNS reverse converters for moduli sets with dynamic ranges up to (8n+1)-bit. *IEEE Transactions on Circuits and Systems I*, *PP*(99), 1–14. doi:10.1109/TCSI.2012.2220460.
19. Piestrak, S. (1994). Design of residue generators and multi operand modular adders using carry-save adders. *IEEE Transactions on Computers*, *43*(1), 68–77. doi:10.1109/12.250610.

20. Patel, R.A., Benaissa, M., Boussakta, S. (2006). Efficient new approach for modulo $\{2^n-1\}$ addition in RNS. In *IEE proceedings on computers and digital techniques* (Vol. 153, pp. 399–405).
21. Patel, R.A., Benaissa, M., Boussakta, S., Powell, N. (2005). Power-delay-area efficient modulo $\{2^n+1\}$ adder architecture for RNS. *Electronics Letters*, *41*(5), 231–232.
22. Sheu, M.H., Lin, S.H., Chen, C., Yang, S.W. (2004). An efficient VLSI design for a residue to binary converter for general balance moduli $\{2^n-3,2^n+1,2^n-1,2^n+3\}$. *IEEE Transactions on Circuits and Systems II: Express Briefs*, *51*(3), 152–155. doi:10.1109/TCSII.2003.821516.
23. Skavantzos, A., & Abdallah, M. (1999). Implementation issues of the two-level residue number system with pairs of conjugate moduli. *IEEE Transactions on Signal Processing*, *47*(3), 826–838.
24. Skavantzos, A., Abdallah, M., Stouraitis, T., Schinianakis, D. (2009). Design of a balanced 8-modulus RNS. In *16th IEEE international conference on electronics, circuits, and systems, 2009. ICECS 2009* (pp. 61–64). doi:10.1109/ICECS.2009.5410923.
25. Skavantzos, A., & Wang, Y. (1999). Applications of new Chinese remainder theorems to RNS with two pairs of conjugate moduli. In *1999 IEEE Pacific Rim conference on communications, computers and signal processing*, 165–168. doi:10.1109/PACRIM.1999.799503.
26. Slegel, T., & Veracca, R. (1991). Design and performance of the IBM enterprise system / 9000 type 9121 vector facility. *IBM Journal of Research and Development*, *35*, 367–381.
27. Sousa, L., & Antao, S. (2012). MRC-based RNS reverse converters for the four-moduli sets $\{2^n+1,2^n-1,2^n,2^(2n+1)-1\}$ and $\{2^n+1,2^n-1,2^(2n),2^(2n+1)-1\}$. *IEEE Transactions on Circuits and Systems II: Express Briefs*, *59*(4), 244–248. doi:10.1109/TCSII.2012.2188456.
28. Szabo, N., & Tanaka, R. (1967). *Residue arithmetic and its applications to computer technology*. McGraw-Hill.
29. Vergos, H., Bakalis, D., Efstathiou, C. (2008). Efficient modulo 2^n+1 multi-operand adders. In *15th IEEE international conference on electronics, circuits and systems, 2008. ICECS 2008* (pp. 694–697). doi:10.1109/ICECS.2008.4674948.
30. Wang, Y. (2000). Residue-to-binary converters based on new chinese remainder theorems. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, *47*(3), 197–205. doi:10.1109/82.826745.
31. Wang, Y., Song, X., Aboulhamid, M., Shen, H. (2002). Adder based residue to binary number converters for $\{2^n-1,2^n,2^n+1\}$. *IEEE Transactions on Signal Processing*, *50*(7), 1772–1779. doi:10.1109/TSP.2002.1011216.
32. Wang, Z., Jullien, G.A., Miller, W.C. (1996). An efficient tree architecture for modulo 2^n+1 multiplication. *Journal VLSI Signal Processing System*, *14*(3), 241–248. doi:10.1007/BF00929618.
33. Zimmermann, R. (1999). Efficient VLSI implementation of modulo $\{2^n\pm1\}$ addition and multiplication. In *14th IEEE symposium on computer arithmetic* (pp. 158–167).

Pedro Miguens Matutino received the Graduate and Masters degree in electronics and computer science from the Instituto Superior Tcnico (IST), Portugal, in 1999 and 2002, respectively. Currently, he is pursuing the Ph.D. degree in electrical engineering from IST. He is currently a lecture with the Department of Telecommunications, Electronics and Computer at Instituto Superior de Engenharia de Lisboa (ISEL), and a researcher at Instituto de Engenharia de Sistemas e Computadores (INESC-ID). His research interests include reconfigurable computing, computer architecture, computer arithmetic, design of VLSI circuits, and embedded systems.

Ricardo Chaves received the Ph.D. degree in electrical and computer engineering from the Instituto Superior Tcnico (IST), Universidade de Lisboa, Portugal, and Delft University of Technology (TU Delft), Delft, The Netherlands, in 2007. He is currently an assistant professor with the Department of Computer Science at IST and a senior researcher at INESC-ID. His research interests include VLSI architectures, computer arithmetic, reconfigurable hardware, security systems, and embedded systems, in which he has published more than 30 papers in international journals and conferences. He is currently a member of the Management Committee of EU COST Action TRUDEVICE.



Leonel Sousa received his Ph.D. in electrical and computer engineering from the Instituto Superior Técnico (IST), Universidade de Lisboa, Portugal, in 1996. He is currently a full professor in the Electrical and Computer Engineering Department at IST and a senior researcher at INESC-ID. His research interests include VLSI architectures, computer architecture and parallel computing, and signal processing systems. He has contributed more than 200 papers to international journals and conferences. He has been associate editor of several journals, namely IEEE Transactions on Circuits and Systems for Video Technology, IEEE Access, Journal of Real-Time Image Processing from Springer, and Eurasip Journal on Embedded Systems, and has served in the program committee of more than 100 international conferences and symposia. He is currently a member of the HiPEAC and the NESUS European Actions and Networks. He is a Fellow of the IET and a senior member of both the IEEE and the ACM.