2012 15th Euromicro Conference on Digital System Design

RNS Arithmetic Units for *Modulo* $\{2^n \pm k\}$

Pedro Miguens Matutino
ISEL/INESC-ID/IST
Lisbon, Portugal
pmiguens@deetc.isel.pt

Hector Pettenghi, Ricardo Chaves, and Leonel Sousa
IST/INESC-ID
Lisbon, Portugal
(hector,rjfc,las)@sips.inesc-id.pt

***Abstract*—Recently new Residue Number Systems (RNS) *moduli* sets have been proposed in order to increase the dynamic range and reduce the width of channels, therefore, reducing the processing time and further exploiting the carry-free characteristic of the modular arithmetic. In this paper we propose improved units for addition, subtraction, and multiplication in RNS for *modulo* $\{2^n \pm k\}$. With this work, the somewhat disregarded field of RNS unit design is covered, encouraging the development of *moduli* sets with channels other than the traditional $\{2^n\}$, $\{2^n \pm 1\}$ *modulo*. In order to evaluate the performance of the proposed structures, they are compared with the well known units for *modulo* $\{2^n\}$, $\{2^n \pm 1\}$, and $\{2^n \pm 3\}$. These structures allow to implement generic units for *modulo* $\{2^n \pm k\}$, in the case of modular multiplication it is achieved the same critical path delay and merely 4% of increase on area resources, when compared with the dedicated structure presented in the state-of-art for *modulo* $\{2^n \pm 3\}$.**

***Keywords*-Residue Number System; Modular addition; Modular multiplication; Compressor units; Arithmetic;**

I. Introduction

Residue Number Systems (RNS) are a good alternative to the conventional arithmetic, based on a weighted number system. The main cause for performance degradation in arithmetic circuits is the carry propagation scheme, which is avoided using RNS. In applications requiring intensive computation, such as digital signal processing [1], [2], this carry free characteristics allow for concurrent computation in each of the RNS moduli channels.

Over the last years many authors have proposed several *moduli* sets and arithmetic structures, in order to achieve a better performance. The main focus of the RNS community is being given to the development of novel conversion units, somewhat disregarding the arithmetic units. Furthermore, another focus should also be considered, in order to take into account the complexity of the basic operations on each channel, in particular the multiplication units, usually the most critical units of the RNS implementations. Depending on the channel *modulo* considered, different computational delays can be observed, for the same input bit length. In order to achieve more efficient and balanced systems the length of each channel should be adjusted, to accomplish identical delays on each *moduli* set channels.

The most common *moduli* set used in RNS applications is the traditional $\{2^n-1, 2^n, 2^n+1\}$ set [3]. Also, the novel $\{2^{2n}-1, 2^n, 2^{2n}+1\}$ [4] *moduli* set, with a dynamic range of $5n$ bit. Recently, a 4-*modulus* base set $\{2^n-1, 2^n+3, 2^n+1, 2^n-3\}$ has been proposed [5] as a hierarchical base composed by two pairs of *moduli* [6], [7]. Even though several *moduli* sets have been proposed using $\{2^n \pm k\}$ *modulo* channels [8], [9], however the development of arithmetic units for these *moduli* sets have been disregarded. Consequently, RNS systems have been implemented with the traditional $\{2^n\}$ and $\{2^n \pm 1\}$.

In this paper a new generic structure to perform the modular addition for two, three and four input operand are proposed, for the $\{2^n-k\}$ and $\{2^n+k\}$ *modulo*. A relative evaluation, regarding area and delay, for the generic adder proposed in [10] and the proposed units is performed. Results suggest that the proposed adders improves the current state-of-the-art reducing the required area up to 36% and achieving a speed up up to 16%, considering the addition operation for more than two operands. In this work the design of modular subtraction operations are also proposed. It is important to emphasize that this specific arithmetic unit is not covered in the state-of-the-art. However, the use of *modulo* subtracters have been recently demanded [11], [12], and [13]. Due to the fact that the modular subtraction is performed in the same way than the modular addition, efficient *modulo* adder/subtracter architectures by sharing the common circuitry are proposed. Additionally, generic structures for multiplication *modulo* $\{2^n \pm k\}$, for which is performed a similar evaluation are proposed. Experimental results on Application Specific Integrated Circuits (ASIC), in particular for the UMC 0.13 μm CMOS technology from UMC [14], suggest that the proposed generic structures implemented for *modulo* $\{2^n \pm 3\}$ have the same critical path delay, with a cost of merely 4% more of area resources, when compared with the specific structure defined in [15].

The rest of the paper is organized as follows. The next section formulates the problem of computing the addition, subtraction, and multiplication for the *modulo* $\{2^n \pm k\}$. Section III describe the proposed modular arithmetic units (addition, subtraction, and multiplication). Section IV describes the implementations of the proposed structures in ASIC, and evaluates their relative performance against existing units. Section V, concludes this paper with some final remarks.

II. Formulation

Addition, subtraction, and multiplication are the three basic operations intensively required for digital signal processing [2] and [16]. This section analyses the modular arithmetic operations required for addition and multiplication for *modulo* $\{m\}$, where $m = 2^n \pm k$.

978-0-7695-4798-5/12 $26.00 © 2012 IEEE
DOI 10.1109/DSD.2012.114

A. Addition/Subtraction

The addition *modulo* $\{m\}$ of the values X and Y can be implemented by performing a binary addition of the two operands and executing the reduction when the result is larger or equal to m. The reduction can be performed by subtracting the value m. In a similar fashion to the $\{2^n \pm 1\}$ modular addition [17], the following presents the approach for *modulo* $\{2^n - k\}$ and $\{2^n + k\}$.

Modulo $\{2^n - k\}$**:** Applying the generic approach to this *modulo*, it is necessary to detect when the addition of X and Y is greater or equal to $2^n - k$. This is equivalent to detecting if $X + Y + k$ is greater or equal than 2^n. Furthermore, when the result is greater or equal than 2^n the modular result is given by $X + Y - (2^n - k) = X + Y + k - 2^n$. Thus, considering two input values, as integers X and Y in the range $[0, 2^n - k - 1]$, the addition *modulo* $\{2^n - k\}$ can be computed by:

$$\langle X+Y\rangle_{2^n-k} = \begin{cases} X+Y & , \ X+Y < 2^n-k \\ X+Y-(2^n-k), & X+Y \geq 2^n-k \end{cases} \quad (1)$$

However, the subtraction operation is computed as an addition, including a correction factor, $COR_{2^n-k} = -k + 1$, given by:

$$\begin{aligned} \langle X-Y\rangle_{2^n-k} &= \langle X+\overline{Y}-k+1\rangle_{2^n-k} \\ &= \langle X+\overline{Y}+COR_{2^n-k}\rangle_{2^n-k} \ . \end{aligned} \quad (2)$$

Considering,

$$\begin{aligned} \langle -Y\rangle_{2^n-k} &= \langle 2^n-1-Y-k+1\rangle_{2^n-k} \\ &= \langle \overline{Y}-k+1\rangle_{2^n-k} \ . \end{aligned} \quad (3)$$

Modulo $\{2^n + k\}$**:** Identically to *modulo* $\{2^n - k\}$, one needs to detect when the addition of X and Y is greater or equal than $2^n + k$. This is the same as detecting if $X + Y + (2^{n+1} - (2^n + k))$ is greater or equal than 2^{n+1}. Furthermore, when the result is greater or equal to 2^{n+1} the modular result is given by $X + Y - (2^n + k)$. These formulations are valid for an integer X and Y in the range $[0, 2^n + k - 1]$ and is represented by equation (4).

$$\langle X+Y\rangle_{2^n+k} = \begin{cases} X+Y & , \ X+Y < 2^n+k \\ X+Y-(2^n+k), & X+Y \geq 2^n+k \end{cases} \quad (4)$$

However, the subtraction operation is computed as an addition, including a correction factor, $COR_{2^n+k} = k + 1 + kX_n - kY_n$, given by:

$$\begin{aligned} \langle X-Y\rangle_{2^n+k} &= \langle X_{[n:0]} - Y_{[n:0]}\rangle_{2^n+k} \\ &= \langle X_{[n-1:0]} + \overline{Y_{[n-1:0]}} - k \cdot X_{[n]} + \\ &+ \ k \cdot Y_{[n]} + k + 1\rangle_{2^n+k} \\ &= \langle X_{[n-1:0]} + \overline{Y_{[n-1:0]}} + \\ &+ \ COR_{2^n+k}\rangle_{2^n+k} \ , \end{aligned} \quad (5)$$

where

$$\begin{aligned} \langle -Y_{[n-1:0]}\rangle_{2^n+k} &= \langle 2^n-1-Y_{[n-1:0]}+k+1\rangle_{2^n+k} \\ &= \langle \overline{Y_{[n-1:0]}}+k+1\rangle_{2^n+k} \ . \end{aligned} \quad (6)$$

B. Multiplication

The multiplication *modulo* $\{m\}$ of the values A and B can be performed as a binary product P, followed by the modular reduction steps. Furthermore, on each reduction step it is computed a constant multiplication factor by the value that exceeds 2^n and added with remain value.

Modulo $\{2^n - k\}$**:** Considering the binary product of A by B, the reduction steps can be described as (7).

$$\begin{aligned} \langle A \times B\rangle_{2^n-k} &= \langle 2^n P_{[2n-1:n]} + 2^0 P_{[n-1:0]}\rangle_{2^n-k} \\ &= \langle 2^n P_1 + P_0\rangle_{2^n-k} \\ &= \langle k \cdot P_1 + P_0\rangle_{2^n-k} \end{aligned} \quad (7)$$

Modulo $\{2^n + k\}$**:** Similar to modulo $\{2^n - k\}$, the reduction steps needed on the multiplication modulo $\{2^n + k\}$ are computed by:

$$\begin{aligned} \langle A \times B\rangle_{2^n+k} &= \langle 2^{2n} P_{[2n+1:2n]} + 2^n P_{[2n-1:n]} + \\ &\quad + 2^0 P_{[n-1:0]}\rangle_{2^n+k} \\ &= \langle 2^{2n} P_2 + 2^n P_1 + P_0\rangle_{2^n+k} \\ &= \langle k^2 \cdot P_2 - k \cdot P_1 + P_0\rangle_{2^n+k} \\ &= \langle k^2 \cdot P_2 + k \cdot (\overline{P_1} + k + 1) + P_0\rangle_{2^n+k} \\ &= \langle k^2 \cdot P_2 + k \cdot \overline{P_1} + P_0 + k^2 + k\rangle_{2^n+k} \ . \end{aligned} \quad (8)$$

III. Architecture

In this section adders, subtracters and multipliers structures are presented, based on the equations presented in the previous section.

A. Addition

Herein the addition units structures for the *modulo* $\{2^n \pm k\}$ are proposed and described. In [10] two addition units are presented for a generic *modulo* $\{m\}$, a more straightforward structure with smaller area requirements, designated as *basic*, and the other one with the delay improved, that requires additional area resources, designated as *speed*.

However, these structures assume that the input values are in the range of $[0, m-1]$. When considering the subtraction operation in (2) and (6), the value of $\overline{Y}$ can be in the range of $\left[0, 2^\delta - 1\right]$ for the particularly case of $\{2^n - k\}$ *modulo*, δ is equal to n. Although, the two proposed units in [10] are not able to perform the reduction needed in these cases, they are only able to reduce $2^n - k$ and not $2 \cdot (2^n - k)$ as required.

Modulo $\{2^n - k\}$**:** In order to implement a structure to compute the addition of X and Y for the entire range it is necessary to perform some modifications in (1), namely:

$$\begin{aligned} &\langle X_{[n-1:0]} + Y_{[n-1:0]}\rangle_{2^n-k} = \\ &= \begin{cases} X+Y & , X+Y < 2^n-k \\ X+Y-(2^n-k) & , 2^n-k \leq X+Y < 2(2^n-k) \\ X+Y-2\cdot(2^n-k), & X+Y \geq 2(2^n-k) \end{cases} . \end{aligned} \quad (9)$$

This structure can be implemented by three Carry-Save-Adder (CSA) structures, three full additions, and a result selection as final step. However, this implementation can be improved if the compensation value needed is selected from the most significant bit of X and Y, selecting the two possible reduction values. This new structure uses two CSAs, two full additions, two constant multiplexers (grey multiplexers), and one multiplexer, as depicted in Figure 1. The output value is represented with δ bits (where $\delta = n$). The values k_0, k_1 and k_2 are 0, k and $2k$, respectively. Furthermore, k_3 and k_4 are not used, simplifying the 4:1 multiplexer to a 2:1, since only two reduction are needed in this case. The logic block A is implemented by an AND gate between X_{n-1} and Y_{n-1}. Also, an OR gate is used to implement the logic block B.

The new proposed adder computes the addition between X and Y, considering an input in the range $[0, 2^n - 1]$, denoting that the inputs are not necessary reduced to the respectively *modulo*.

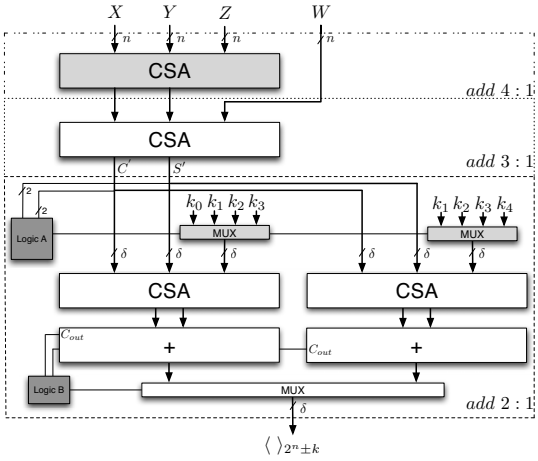


Figure 1. Modular adder structure

Modulo $\{2^n+k\}$: For this *modulo*, and considering the input range of X and Y as maximum value of $2^{n+1}-1$, the residue value is given by:

$$\begin{aligned}\langle X_{[n:0]} + Y_{[n:0]}\rangle_{2^n+k} &= \langle 2^{n+1} - 1 + 2^{n+1} - 1\rangle_{2^n+k} \\ &= \langle 2\cdot 2^n - 1 + 2\cdot 2^n - 1\rangle_{2^n+k} \\ &= \langle 3\cdot(2^n + k - k) + 2^n - 2\rangle_{2^n+k} \\ &= \langle 3\cdot(2^n + k) + 2^n - 3k - 2\rangle_{2^n+k} \\ &= \langle 2^n - 3k - 2\rangle_{2^n+k}\ , \end{aligned} \quad (10)$$

in this particularly case, it is necessary to subtract the constant $3\cdot(2^n + k)$ to compute the reduction value. Considering the three reduction factors, addition of X and Y can be computed as:

$$\langle X_{[n:0]} + Y_{[n:0]}\rangle_{2^n+k} = \quad (11)$$
$$= \begin{cases} X+Y & , A+B < 2^n + k \\ X+Y-(2^n+k) & , 2^n + k \le X+Y < 2(2^n+k) \\ X+Y-2(2^n+k), & 2(2^n+k) \le X+Y < 3(2^n+k) \\ X+Y-3(2^n+k), & X+Y \ge 3(2^n-k) \end{cases}.$$

The output value width (δ), *modulo* $\{2^n + k\}$ requires $n+1$ bits, also the k_i values are 0, $2^n - k$, $2(2^n - k)$, and $3(2^n - k)$, respectively for k_0, k_1, k_2, and k_3. The logic block A is implemented by the function given in (12). However, the logic block B is only a wire, since the second C_{out} is directly connected to the multiplexer.

$$Logic\ A \begin{cases} sel\_mux_0 & = \overline{a_n}\cdot\overline{b_n} + \\ & \quad + \overline{a_{n-1}}\cdot\overline{b_{n-1}}\cdot(\overline{a_n} + \overline{b_n}) \\ sel\_mux_1 & = a_n \cdot b_n \cdot (a_{n-1} + b_{n-1}) \end{cases} \quad (12)$$

Furthermore, in order to compute (2), (6), (7) and (8) described on section II it is required to implement structures to compute the addition 3 and 4 n bit operands into one δ bit value.

From the proposed adder structures, two similar structures to compute the addition of three and four operands for *modulo* $\{2^n \pm k\}$ are also proposed. The addition of four n bit operands into one is compute by:

$$\begin{aligned} &\langle X_{[n-1:0]} + Y_{[n-1:0]} + Z_{[n-1:0]} + W_{[n-1:0]}\rangle_{2^n\pm k} = \\ &= \langle C_{[n:1]} + S_{[n-1:0]} + W\rangle_{2^n\pm k} \\ &= \langle 2^n C_{[n]} + C_{[n-1:1]} + S_{[n-1:0]} + W\rangle_{2^n\pm k} \\ &= \langle 2^n C_{[n]} + C'_{[n:1]} + S'_{[n-1:0]}\rangle_{2^n\pm k} \\ &= \langle 2^n C_{[n]} + 2^n C'_{[n]} + C'_{[n-1:1]} + S'_{[n-1:0]}\rangle_{2^n\pm k} \\ &= \langle \mp k\cdot C'_{[n]} \mp k\cdot C_{[n]} + C'_{[n-1:1]} + S'_{[n-1:0]}\rangle_{2^n\pm k}\ . \end{aligned} \quad (13)$$

The two C_{out} bits are used to select the compensation value. The values of k_i are the same respectively to *modulo* adders.

Equation (13) can be rewritten for a three operand implementation, and is computed as:

$$\begin{aligned} &\langle X_{[n-1:0]} + Y_{[n-1:0]} + Z_{[n-1:0]}\rangle_{2^n\pm k} = \\ &= \langle C_{[n:1]} + S_{[n-1:0]}\rangle_{2^n\pm k} \\ &= \langle 2^n C_{[n]} + C_{[n-1:1]} + S_0\rangle_{2^n\pm k} \\ &= \langle \mp k\cdot C_{[n]} + C_{[n-1:1]} + S_0\rangle_{2^n\pm k}\ . \end{aligned} \quad (14)$$

This structure is similar to the 4:1 adder, but uses less area resources and reduces the critical path. These improvements are achieved reducing in one CSA and simplifying the constant multiplexers, as depicted in Figure 1.

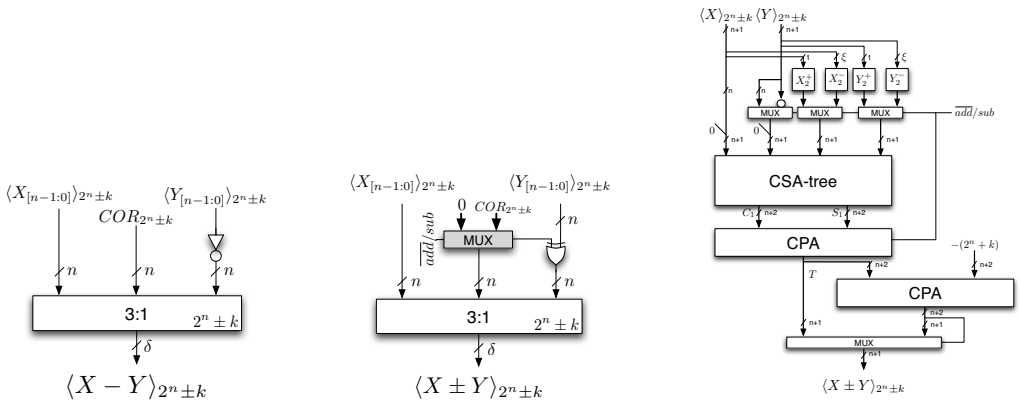
B. Subtraction

From (2) and (6) it is shown that the subtraction operation can be computed as an addition of X, the one's complement of Y, and a correction factor (COR). This correction factor depends on the *modulo* and k values, $-k+1$ and $k+1+kX_n+kY_n$, respectively for *modulo* $\{2^n - k\}$ and $\{2^n + k\}$.

The proposed subtracter depicted in Figure 2(a), is implemented with a 3:1 *modulo* adder. Furthermore, comparing the 2:1 modular addition operation with the subtraction, the last one uses one more n-bit CSA and the delay is consequently degraded in the time of a Full-Adder (FA).

C. Addition/Subtraction

Here in, it is also proposed a new adder/subtracter structure performing both operations using the same hardware resources using two approaches, the first one



(a) subtracter (b) adder/subtracter I (c) adder/subtracter II

Figure 2. Modular adder/subtracter structures

uses arithmetic computation and the second uses the weight selection to pre-compute the values to be feed to the final adder.

Arithmetic approach: The *adder/subtracter I* it is derived from the proposed subtracter, uses a 3:1 modular adder, n XOR gates to implement the controlled negation operation of Y, and one multiplexer of constants to select between 0 and the correction factor used in the subtraction operation, depicted in Figure 2(b).

This structure can be optimized by using a 2:1 modular adder (full-range adder, the same as proposed in this section), and a modified CSA. This CSA is implemented by:

$$if\ COR_i = 0 \quad \begin{cases} sum_i &= a_i \oplus b_i \oplus \overline{add}/sub \\ carry_i &= a_i \cdot (b_i \oplus \overline{add}/sub) \end{cases}$$

$$if\ COR_i = 1 \quad \begin{cases} sum_i &= a_i \oplus b_i \\ carry_i &= a_i \cdot b_i + \overline{add}/sub \cdot \overline{b_i}\ , \end{cases}$$

where COR_i is the $i-th$ bit of the correction factor, and $\overline{add}/sub$ is the control operation bit.

Weight-selection approach: Another adder/subtracter *modulo* $\{2^n + k\}$ implementation is described, for k *odd* $\in [1, 2^n - 1]$, which is herein denoted as *adder/subtracter II*.

In the case of the addition $\langle X + Y\rangle_{2^n+k}$, with $X = \{x_n, ..., x_1, x_0\}$ and $Y = \{y_n, ..., y_1, y_0\}$ is:

$$\langle X + Y\rangle_{2^n+k} = \sum_{j=0}^{n} \langle 2^j\rangle_{2^n+k} \cdot x_j + \sum_{j=0}^{n} \langle 2^j\rangle_{2^n+k} \cdot y_j\ . \quad (15)$$

For the subtraction:

$$\langle X - Y\rangle_{2^n+k} = \sum_{j=0}^{n} \langle 2^j\rangle_{2^n+k} \cdot x_j - \sum_{j=0}^{n} \langle 2^j\rangle_{2^n+k} \cdot y_j\ . \quad (16)$$

The subtracted terms in (16) can be adjusted by adding a correction term, COR^+ and COR^- for modulo addition and subtraction, respectively. The value of $COR^\pm$ depends on the weight selection used. It is important to emphasize that all the weights associated to the inputs, $\langle 2^j\rangle_{2^n+k}$, were assumed positive up to now. In this work, negative weights associated to $\langle 2^n\rangle_{2^n+k} = -k$ are also proposed in order to simplify this *adder/subtracter II* structure as is explained next. Let us denote the **hybrid weight selection** as the one that $\sum_{j=0}^{n-1}\langle 2^n\rangle_{2^n+k} = 2^j$ and $\langle 2^n\rangle_{2^n+k} = -k$, whereas **positive weight selection** when $\sum_{j=0}^{n}\langle 2^n\rangle_{2^n+k} = 2^j$. Due to the fact that $-k$ is not in binary form when the **hybrid weight selection** is applied, the split into powers of two is also proposed without any hardware cost. Therefore, $\langle 2^n\rangle_{2^n+k} = -\sum_{l=0}^{\xi-1} 2^l$, with $\xi = \lceil log_2(k-1)\rceil$, before the application of COR.

Once that the COR value is obtained, the *modulo* computation of the addition and the subtraction can be derived by:

$$\langle X \pm Y\rangle_{2^n+k} = \begin{cases} T^\pm - (2^n + k), & \text{if } T^\pm \geq 2^n + k \\ T^\pm, & \text{otherwise,} \end{cases} \quad (17)$$

where $T^\pm = X \pm Y + COR^\pm$. The condition of one single comparison to derive the *modulo* computation in Equation (17) is set only when the maximum value of $T^\pm$, herein denoted as $T^\pm_{max}$ is:

$$T^\pm_{max} < 2 \times (2^n + k)\ . \quad (18)$$

This condition (18) depends on the COR value. Let us to apply both weight selection in order to set the Equation (17).

- For the **positive weight selection** applied to the *modulo* addition, $COR^+ = 0$, and substituting in Equation (18):

$$T^+_{max} = 2 \times (2^n + k) - 2 \leq 2 \times (2^n + k). \quad (19)$$

Thus, with only positive weights the *modulo* addition can be derived with one single comparison.

- For the **hybrid weight selection** applied to the *modulo* addition, $COR^{+} = 2^n - k$, and substituting in Equation (18):

$$T^{+}_{max} = 3 \times (2^n + k) - 2 \geq 2 \times (2^n + k). \quad (20)$$

Thus, by using this weight selection the *modulo* addition can not be derived with one single comparison.

- For the **positive weight selection** applied to the *modulo* subtraction, $COR^{-} = 2k+1$, and substituting in Equation (18):

$$T^{-}_{max} = 2 \times (2^n + k) + 2 \times k - 1 \geq 2 \times (2^n + k). \quad (21)$$

Thus, by using this weight selection the *modulo* subtraction can not also be derived with one single comparison.

- For the **hybrid weight selection** applied to the *modulo* subtraction, $COR^{-} = 1$, and substituting in Equation (18):

$$T^{-}_{max} = 2 \times (2^n + k) - 1 < 2 \times (2^n + k). \quad (22)$$

Thus, with the **hybrid weight selection** the *modulo* computation can be derived with one single comparison.

Therefore, (18) is set when the **positive weight selection** and **hybrid weight selection** is used for the addition and subtraction, respectively.

Let us to denote from now on, the inputs associated to the addition as (X^{+}, Y^{+}) and the associated to the subtraction as (X^{-}, Y^{-}). In this case:

$$\begin{aligned} X^{+} &= \sum_{j=0}^{n} \langle 2^j \rangle_{2^n+k} \cdot x_j = \sum_{j=0}^{n} 2^j \cdot x_j \\ &= \sum_{j=0}^{n-1} 2^j \cdot x_j + 2^n \cdot x_n = X_1^{+} + X_2^{+} , \end{aligned} \quad (23)$$

$$\begin{aligned} X^{-} &= \sum_{j=0}^{n} \langle 2^j \rangle_{2^n+k} \cdot x_j = \sum_{j=0}^{n-1} 2^j \cdot x_j + \langle 2^n \rangle_{2^n+k} \cdot x_n \\ &= \sum_{j=0}^{n-1} 2^j \cdot x_j - (-k) \times \overline{x}_n \\ &= \sum_{j=0}^{n-1} 2^j \cdot x_j + \sum_{l=0}^{\xi-1} 2^l \cdot \overline{x}_n = X_1^{-} + X_2^{-} , \end{aligned} \quad (24)$$

$$\begin{aligned} Y^{+} &= \sum_{j=0}^{n} \langle 2^j \rangle_{2^n+k} \cdot y_j = \sum_{j=0}^{n} 2^j \cdot y_j \\ &= \sum_{j=0}^{n-1} 2^j \cdot y_j + 2^n \cdot y_n = Y_1^{+} + Y_2^{+} , \end{aligned} \quad (25)$$

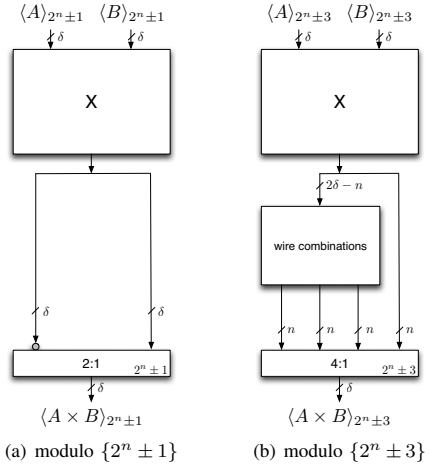
$$\begin{aligned} Y^{-} &= \sum_{j=0}^{n} \langle 2^j \rangle_{2^n+k} \cdot \overline{y}_j = \sum_{j=0}^{n-1} 2^j \cdot \overline{y}_j + \langle 2^n \rangle_{2^n+k} \cdot y_n \\ &= \sum_{j=0}^{n-1} 2^j \cdot \overline{y}_j - (-k) \cdot y_n \\ &= \sum_{j=0}^{n-1} 2^j \cdot \overline{y}_j + \sum_{l=0}^{\xi-1} 2^l \cdot y_n = Y_1^{-} + Y_2^{-} . \end{aligned} \quad (26)$$

A control must be included in order to select the desirable operation due to we have different arrays for the same input, X or Y. This can be derived by including (2:1) MUXs in the first level of the architecture to multiplex the required terms $X_i^{\pm}$ and $Y_i^{\pm}$, with $i = 1, 2$. However, the MUX is not needed in the particular case, $X_1^{\pm}$, due to the arrays are equal. The structure of the proposed *adder/subtracter II* is presented in Figure 2(c).

The array of the multiplexed inputs associated to the selected operation are added in the next stage by means of a CSA and a Carry-Propagate-Adder (CPA) due to to the fact that the constrain in (18) is set for both operation by using the chosen weight selection, it is only needed one comparison in the last stage to derive the *modulo* addition/subtraction. The final $\langle X \pm Y \rangle_{2^n+k}$ computation is carried out by one CPA, and the final result is selected by the $S_{[n+1]}$ bit of the last addition.

D. Multiplication

In this subsection, the existing multiplication structures for *modulo* $\{2^n \pm 1\}$ and $\{2^n \pm 3\}$, proposed by [17] and [15], depicted in Figures 3(a) and 3(b), are compared with the novel *modulo* $\{2^n \pm k\}$ structures herein proposed, depicted in Figures 4(a) and 4(b). The generic multipliers proposed by [18] and [19] are not considered in this paper given: *i)* their restrictions to small values of n and, *ii)* the restriction for the value k due to the periodicity properties [18].



(a) modulo $\{2^n \pm 1\}$ (b) modulo $\{2^n \pm 3\}$

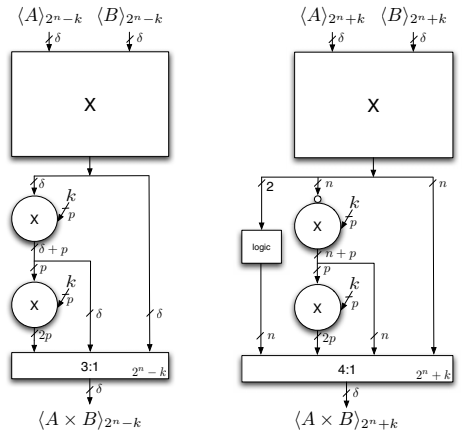
Figure 3. Modular multipliers structures

Modulo $\{2^n - k\}$: A modification to the calculation described in (7) must be performed, in order to efficiently compute the $k \cdot P_1$ value. This can be performed by multiplying P_1 by the constant k. Considering in this case the restriction of k as:

$$p = log_2(k) \leq \frac{n}{2}\,, \tag{27}$$

$$\begin{aligned}\langle A \times B\rangle_{2^n-k} &= \langle k \cdot P_1 + P_0\rangle_{2^n-k} \\ &= \left\langle P_0 + m^{1,1}_{[n+p-1:0]}\right\rangle_{2^n-k} \\ &= \left\langle P_0 + m^{1,1}_{[n-1:0]} + k \cdot m^{1,1}_{[n+p-1:n]}\right\rangle_{2^n-k} \\ &= \left\langle P_0 + m^{1,1}_{[n-1:0]} + m^{1,2}_{[2p-1:0]}\right\rangle_{2^n-k}\,, \end{aligned} \tag{28}$$

considering $m^{i,r}$ as the result of multiplying the constant k^i by the addition vector P_r. To compute the final addition a modular adder 3:1 *modulo* $\{2^n - k\}$ is required. The proposed structure to compute the modular multiplication *modulo* $\{2^n - k\}$ is depicted in Figure 4(a).



(a) modulo $\{2^n - k\}$ (b) modulo $\{2^n + k\}$

Figure 4. Proposed modular multipliers structures

Modulo $\{2^n + k\}$: In order to compute (8), it is need to calculate the constant multiplication $\langle k \cdot P_{\#}\rangle_{2^n+k}$. Considering the same method as in (28) to calculate the constant multiplication, the computation of (8) can be performed as:

$$\begin{aligned}&\langle X \times Y\rangle_{2^n+k} = \\ &= \langle k^2 \cdot P_{[2n+1:2n]} + k \cdot \overline{P_{[2n-1:n]}} + P_0 + k^2 + k\rangle_{2^n+k} \\ &= \langle k^2 \cdot P_{2_{[2n+1:2n]}} + k \cdot \overline{P_{1_{[2n-1:n]}}} + P_0 + k^2 + k\rangle_{2^n+k} \\ &= \langle m^{2,2}_{[2p+1:0]} + m^{1,1}_{[n+p-1:0]} + P_0 + k^2 + k\rangle_{2^n+k} \\ &= \langle m^{2,2}_{[2p+1:0]} + k \cdot (\overline{m^{1,1}_{[n+p-1:n]}} + k + 1) + m^{1,1}_{[n-1:0]} + P_0 + \\ &\quad + k^2 + k\rangle_{2^n+k} \\ &= \langle m^{2,2}_{[2p+1:0]} + m^{1,2}_{[n+p-1:n]} + m^{1,1}_{[n-1:0]} + P_0 + \\ &\quad + 2 \cdot (k^2 + k)\rangle_{2^n+k}\,. \end{aligned} \tag{29}$$

Taking into account that $m^{2,2}$ is a $2 - bit$ vector, it can be included in the correction factor. Consequently, the correction value is the addition of $m^{2,2}$ and $2 \cdot (k^2 + k)$. In this case a 4:1 adder *modulo* $\{2^n + k\}$ is used to perform the final addition, as depicted in Figure 4(b).

IV. Experimental Results

In order to evaluate the proposed adders, subtracters and multipliers *modulo* $\{2^n \pm k\}$ and *modulo* $\{2^n \pm 1\}$ and $\{2^n \pm 3\}$ units described in the state-of-art were described in VHSIC Hardware Description Language (VHDL) and mapped to the UMC 0.13 μm CMOS technology from UMC [14]. Both synthesis and mapping were performed using Design Vision Version A-2007.12-SP5 from Synopsys.

The experimental results presented in this paper were obtained for variation of $n \in [6, 32]$. For the k it was synthesized for the lowest possible value $k = 3$, and the maximum value considered $k = 2^{n/2} - 1$, and it is presented the average value of these results.

A. Adders/Subtracters structures

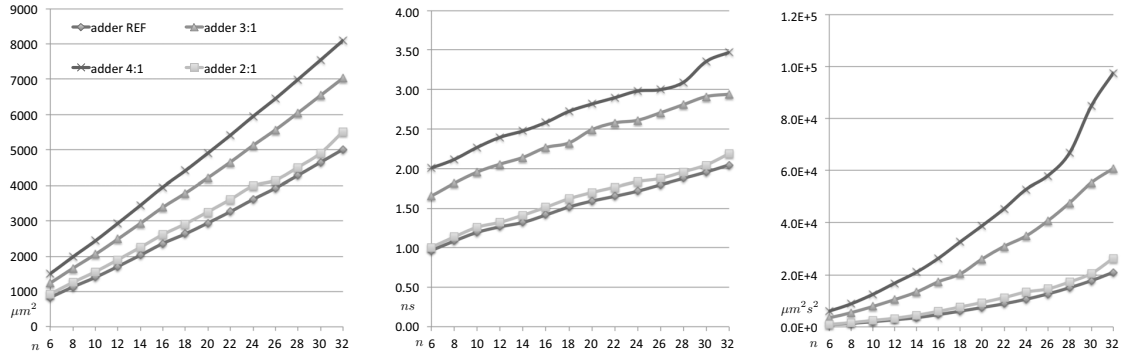
The first step taken was to evaluate the impact of implementing the proposed 2:1 adder for inputs in the range of $[0, 2^n - 1]$ comparative to the existing ones. The obtained experimental results for area and delay, are depicted in Figure 5. From the obtained results our proposal only needs 10% more area resources and is merely 6% slower than the $speed$ adder proposed by [10], but performing a full range operation, considering input values greater than $2^n \pm k$.

Furthermore, when comparing the 3:1 adder with the reference adder, the proposed one uses 44% more area resources and is 57% slower. However, if it is compared with the implementation of one modular compressor [20] and one adder, our proposal is 22% faster using 24% less area. Performing the same analysis for the 4:1 adder, our proposal is 10% faster, using 44% less resources than the implementation with two modular compressors and one modular adder.

A comparative analysis where made for the subtracter and adder/subtracter, using a binary Carry-Propagate-Adder (CPA) and a binary prefix-adder (PA) [17] to compute the binary addition of two operand. The obtained experimental results for area and delay, are depicted in Figure 6. From the obtained results it is show that the generic *adder/subtracter I* has identically delay (only 3% slower) when compared to the *subtracter* unit, with similar uses of area resources (uses merely 6% more), in both implementations (CPA and PA). The *adder/subtracter II* has been only implemented for CPA configuration since our intention was to reduce the area resources in this structure, which does not occur for the average analysis presented. However, this structure achieves better results than the others for *modulo* $\{2^n + k\}$ with n greater than 12, achieving in these cases 2% less area and 3% faster than the *adder/subtracter I*.

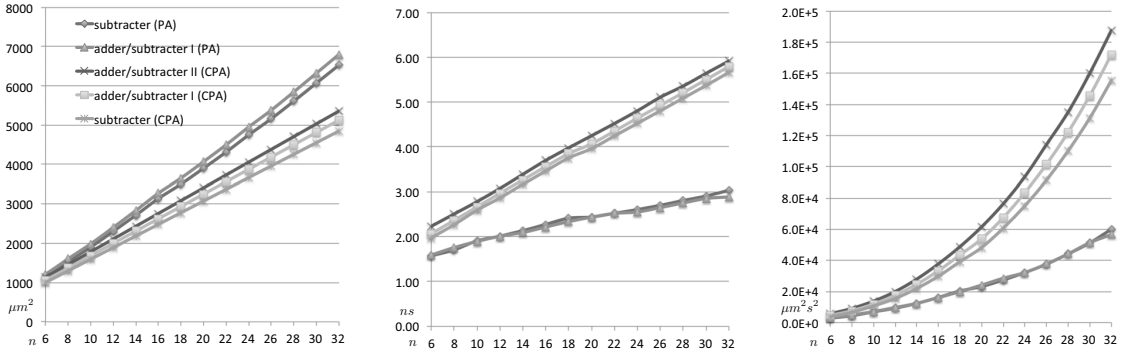
B. Multiplier structures

Identically to the addition/subtraction units, the relative performance of the several multipliers *modulo* structures



(a) Area (b) Delay (c) AT^2

Figure 5. Experimental results for adder (2:1, 3:1 and 4:1) structure modulo $\{2^n \pm k\}$



(a) Area (b) Delay (c) AT^2

Figure 6. Experimental results for subtracter/adder structure modulo $\{2^n \pm k\}$

have also been analysed. The analysis evaluates the behaviour of multiplier structures, the ones proposed by [17] *modulo* $\{2^n \pm 1\}$, depicted in Figure 3(a), [15] *modulo* $\{2^n \pm 3\}$ described in Figure 3(b) and the ones herein proposed for *modulo* $\{2^n \pm k\}$.

The experimental results obtained for circuit area and delay are depicted in Figure 7. Also, these were obtained for variation of $n \in [6, 32]$, and for $k = 3$ and $k = 2^{n/2} - 1$.

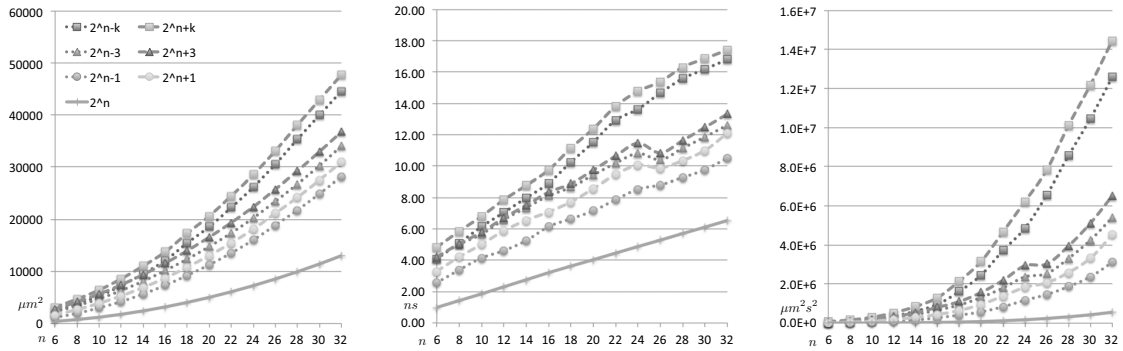
In Figure 7(b) it can be observed that the binary channel 2^n has less time delay, followed by the channels *modulo* $\{2^n \pm 1\}$, $\{2^n \pm 3\}$ and the $\{2^n \pm k\}$ have the larger critical path delay. This differences are concern in the *modulo* reduction for each channel, on the $\{2^n\}$ the final result is the least significant bits of the product of A and B no modular reduction is necessary in this particular case, in others cases is necessary to implemented a modular reduction, as proposed in this paper.

The most important result is obtained when comparing the $\{2^n \pm 3\}$ channels with the $\{2^n \pm k\}$ channels. This analysis suggests that the $\{2^n \pm 3\}$ modular operation is 23% faster than the $\{2^n \pm k\}$ channels. Higher area requirements are also imposed for the $\{2^n \pm k\}$ channels for the same bit length, 22% and 66% more than $\{2^n \pm 3\}$ and $\{2^n \pm 1\}$ channels, respectively. Note that the implemented $\{2^n \pm 1\}$ *modulo* multiplication structures are the most straightforward ones, as depicted in Figure 3(a).

It can be concluded from the experimental results that even though the generic $\{2^n \pm k\}$ *modulo* structures channels are slower than the $\{2^n \pm 1\}$ and $\{2^n \pm 3\}$, the full RNS computation is improved. In fact, the number of channels can be increased and the bit length for each RNS channel can be reduced for the same Dynamic Range.

V. Conclusions

In this paper, adders subtracters and multipliers units for Residue Number Systems (RNS) using modulo $\{2^n \pm k\}$ channels are proposed. Experimental results for the proposed units for modulo $\{2^n \pm k\}$ were obtained, for 0.13 μm ASIC technology. The proposed 2:1, 3:1 and 4:1 adders units, are respectively 6%, 57% and 81% slower than the 2:1 reference adder, at a cost of area resource of 10%, 44% and 68%. However, our 3:1 modular adder is 22% faster using 24% less area than performing the computation by one modular CSA and one modular adder. The adder/subtracter proposed is only 3% slower and uses merely 6% more area resources than a single modular subtracter unit. Furthermore, the analysis for the proposed generic multiplier *modulo* $\{2^n \pm k\}$ unit suggests that requires 66% and 22% more area resources, whereas is 33% and 18% slower than the specific implementation, respectively for $\{2^n \pm 1\}$ and $\{2^n \pm 3\}$ *modulo* channels.



(a) Area (b) Delay (c) AT^2

Figure 7. Experimental results for multiplier structure modulo $\{2^n \pm k\}$

However, when comparing the proposed generic multiplier structure, implemented for the *modulo* $\{2^n \pm 3\}$, with the specific structure [15], the generic has the critical path delay, with a cost of merely 4% more of area resources. As final remark, this paper proposes generic and efficient structures for 2:1, 3:1 and 4:1 adders, used to implement the computation of modular addition, subtraction and multiplication *modulo* $\{2^n \pm k\}$.

ACKNOWLEDGMENT

This work was supported by the Portuguese Foundation for Science and for Technology (INESC-ID multi-annual funding) through the PIDDAC Program funds and by the PROTEC Program funds under the research grant SFRH / PROTEC / 49763 / 2009.

REFERENCES

[1] M. Soderstrand, W.Jenkins, G. Jullien, and F. Taylor, Eds., *Residue number system arithmetic: modern applications in digital signal processing*. Piscataway, NJ, USA: IEEE Press, 1986.

[2] P. M. Matutino and L. Sousa, "An RNS based specific processor for computing the minimum SAD," in *11th EUROMICRO Conference on Digital System Design: Architectures, Methods and Tools*, 2008.

[3] F. E. P. Dale Gallaher and P. Srinivasan, "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*, 1997.

[4] A. Hariri, K. Navi, and R. Rastegar, "A new high dynamic range moduli set with efficient reverse converter," *Computers and Mathematics with Applications*, vol. 55, no. 4, pp. 660 – 668, 2008.

[5] M.-H. Sheu, S.-H. Lin, C. Chen, and S.-W. Yang, "An efficient VLSI design for a residue to binary converter for general balance moduli $\{2^n - 3,2^n + 1,2^n - 1,2^n + 3\}$," *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 51, no. 3, pp. 152 – 155, march 2004.

[6] L.-S. Didier and P.-Y. Rivaille, "A generalization of a fast RNS conversion for a new 4-modulus base," *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 56, no. 1, pp. 46 –50, jan. 2009.

[7] P. Ananda Mohan, "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*, 11-14 2004, pp. 188 – 192.

[8] A. Skavantzos, "An efficient residue to weighted converter for a new residue number system," *Proceedings of the Great Lakes Symposium on VLSI*, pp. 185–191, 1998.

[9] A. Hiasat, "VLSI implementation of new arithmetic residue to binary decoders," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 13, no. 1, pp. 153 –158, 2005.

[10] A. Omondi and B. Premkumar, Eds., *Residue Number Systems: Theory and Implementation*. London, UK: Imperial College Press, 2007.

[11] S. Timarchi, K. Navi, and M. Hosseinzade, "New design of rns subtractor for modulo $\{2^n + 1\}$," *Information and Communication Technologies, ICTTA '06.*, pp. 2803–2808, 2006.

[12] *Wireless Security and Cryptography: Specifications and Implementations*. CRC-Press, A Taylor and Francis Group, 2007.

[13] P. M. Matutino, R. Chaves, H. C. Neto, and L. Sousa, "Decimal multiplication on fpga based on a rns system representation," in *8th Portuguese Meeting on Reconfigurable Systems*, 2012.

[14] Virtual Silicon Technology Inc., "v2.4 esimroute/11^{TM}," High Performance Standard Cell Library (UMC 0.13 μm), Tech. Rep., 2004.

[15] P. M. Matutino, R. Chaves, and L. Sousa, "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*, 2010.

[16] J. O. *et al.*, "Video Coding with H.264/AVC: Tools, Performance, and Complexity," *IEEE Circuits and Systems Magazine*, pp. 7–28, 2004.

[17] R. Zimmermann, "Efficient VLSI implementation of modulo $\{2^n \pm 1\}$ addition and multiplication," in *14th IEEE Symposium on Computer Arithmetic*, 1999.

[18] S. J. Piestrak and K. S. Berezowski, "Design of residue multipliers-accumulators using periodicity," in *Signals and Systems Conference, 208. (ISSC 2008). IET Irish*, june 2008, pp. 380 –385.

[19] V. Paliouras, K. Karagianni, and T. Stouraitis, "A low-complexity combinatorial rns multiplier," *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, vol. 48, no. 7, pp. 675 –683, july 2001.

[20] G. Alia and E. Martinelli, "Designing multioperand modular adders," *Electronics Letters*, vol. 32, no. 1, pp. 22 –23, jan 1996.