Improving residue number system multiplication with more balanced moduli sets and enhanced modular arithmetic structures

R. Chaves and L. Sousa

Abstract: Residue number systems (RNS) are non-weighted systems that allow to perform addition, subtraction and multiplication operations concurrently and independently on each residue. The triple moduli set $\{2^n - 1, 2^n, 2^n + 1\}$ and its respective extensions have gained unprecedent importance in RNS, mainly because of the simplicity of the arithmetic units for the individual channels and also of the converters to and from RNS. However, there is neither a perfect balance between the various elements of this moduli set nor an exact equivalence in the complexity of the individual arithmetic units for each individual residue. Two complementary approaches have been proposed to improve the efficiency of RNS based on this type of moduli sets: enhancing multipliers modulo $2^n + 1$, which perform the most complex arithmetic operation, and overloading the binary channel in order to obtain a more balanced moduli set. Experimental results show that, when applied together, these techniques can improve the efficiency of the multipliers up to 32%.

1 Introduction

In the last years, the residue number system (RNS) has became an important research field in the area of efficient digital computing [1]. A great amount of research has already been done to exploit the RNS properties for performing non-weighted, carry-free arithmetic. This characteristic can be greatly advantageous when repetitive arithmetic operations have to performed, especially when large operators are involved. A known example of such operations is the multiply-accumulate operation for linear filtering and in number theoretic transforms, typically found in digital signal processing (DSP) [2, 3]. Cryptography is another growing field where efficient multipliers are needed. This includes the asymmetrical encryption algorithm RSA [4, 5], that intensively uses modular multiplications with operands up to 2048 bits, and the international data encryption algorithm (IDEA) symmetrical algorithm [6], which applies the $2^{16} + 1$ modular multiplication.

By splitting the computation of a large operand into several smaller non-dependent channels the performance can be improved. This approach also reduces the required area, especially in the multiplication units where the area grows with the square of the operand's length [7]. Nevertheless, in RNS the need to perform forward and reverse conversions also has been taken into account, which represent an unavoidable overhead, because the majority of the standards use the binary representation.

With the increase of the modulus size, combinatorial RNS architectures become more efficient than look-up table architectures [8]. Therefore modulus using a nearby value to a power of 2 is preferable, because only modified functional blocks of binary architectures have to be applied for the same technology. This is the case not only of the commonly adopted 3-moduli set $\{2^n - 1, 2^n, 2^n + 1\}$, but also of the 4-moduli supersets that introduce a fourth element of the type $\{2^{n+1} \pm 1\}$ to increase the dynamic range [9]. Mathematical properties of these moduli sets are also important to develop efficient converters between the RNS and the binary number system, using either the Chinese Remainder Theorem (CRT) or the mixed-radix conversion technique [10].

This paper addresses the problem of improving the speed-up achieved with the RNS-based systems. Having the multiply operation as the main focus of this paper, the methods and techniques herein proposed benefit the RNS arithmetic in general. From a detailed analysis of the RNS characteristics, the two main critical aspects that prevent from obtaining a higher speed-up are identified as follows.

1. The differences in the complexity of the arithmetic units for the various moduli of the set are not negligible, in particular, the longer critical path of the $2^n + 1$ moduli set.
2. In spite of just exceeding by one the power of 2, the additional bit of the moduli $2^n + 1$ multiplication originates a more complex computational structure and consequently the processing time increases, when compared with the other moduli multiplications.

These two aspects stress the imbalance between the elements of not only the traditional moduli set, but also the extended moduli sets and degrade the performance of the RNS. In this paper, two independent but complementary techniques are proposed to tackle this problem: the enhancement of the arithmetic unit for the modulo $2^n + 1$ multiplication [11] and a modification of the moduli sets to obtain more balanced structures. This last proposed

doi:10.1049/iet-cdt:20060059

Paper first received 12th April 2006 and in revised form 14th March 2007
The authors are with the Department of Electrical Engineering, IST/INESC-ID, TU Lisbon, R. Alves Redol, 9, 1º, Lisboa 1000, Portugal

E-mail: las@inesc.pt

modification corresponds to the overloading of the binary base element (channel), which allows us to alleviate the other non-binary channels for a given dynamic range.

To evaluate the improvements obtained from the proposed modulo $2^n + 1$ multiplication structure and the overloading of the binary channel, the arithmetic units for the modified and more balanced moduli set were implemented. They are described using both behavioural and fully structural parameterisable IEEE VHDL and implemented using Synopsys synthesis tools with a high-density StdCell library from UMC, which is based on a 0.13-μm CMOS process from Virtual Silicon Technology, Inc. [12]. Experimental results suggest the advantages of applying the proposed $2^n + 1$ multiplier and the balanced moduli set, evidence the gains, not only in the multiplication, with an improvement up to 32%, but also in the RNS arithmetic in general. Moreover, no additional overhead is imposed for conversion. The proposed conversion units are, on average, 4 to 10% more efficient than the conversion units for the original moduli sets.

This paper is organised as follows. In Section 2, enhanced $2^n + 1$ multipliers are proposed and evaluated. The extension of the binary channel is proposed in Section 3 and new converters are presented for the derived moduli sets. Experimental results are presented in Section 4, and Section 5 concludes the paper.

2 Enhanced modulo $2^n + 1$ multipliers

Unlike multiplication modulo $2^n - 1$ or modulo 2^n, the multiplication modulo $2^n + 1$ has to accommodate the cases where one or both operands are equal to 2^n. This is the main reason why the modulo $2^n + 1$ multipliers are the slowest, in both the 3- and the 4-moduli sets. We have recently shown that although the modified Booth recoding can be applied for designing modulo $2^n + 1$ multipliers, it usually does not lead to faster multipliers nor significantly reduces the required hardware [13]. This paper proposes to enhance one of the most efficient modulo $2^n + 1$ multiplication architectures, proposed by Zimmermann [14], that does not use Booth recoding. By manipulating the values for the particular case when one or both operands are equal to zero, we seek to perform this calculation in a more efficient way by speeding-up the computation without significantly increasing the required circuit area.

Let us consider an integer number X represented with $n + 1$ bits

$$X = \sum_{i=0}^{n} 2^i x_i = 2^n x_n + \sum_{i=0}^{n-1} 2^i x_i = 2^n x_n + X' \tag{1}$$

and the multiplication of numbers X and Y, which can be computed as

$$\begin{aligned}\langle X \times Y \rangle_{2^n+1} = \langle & 2^n x_n \cdot 2^n y_n + 2^n x_n \cdot Y' \\ & + 2^n y_n \cdot X' + X' \cdot Y' \rangle_{2^n+1}.\end{aligned} \tag{2}$$

By exploiting the following properties

$$\langle 2^n \rangle_{2^n+1} = \langle -1 \rangle_{2^n+1} \Rightarrow \langle 2^n X \rangle_{2^n+1} = \langle \overline{X} + 2 \rangle_{2^n+1} \tag{3}$$

(2) can be rewritten as

$$\langle X \times Y \rangle_{2^n+1} = \left\langle \underbrace{x_n \cdot y_n}_{P_3} \underbrace{-x_n \cdot Y' - y_n \cdot X'}_{P_2} \underbrace{+ X' \cdot Y'}_{P_1} \right\rangle_{2^n+1} \tag{4}$$

In the modulo $2^n + 1$ representation, when $a_n = 1$ the other bits are zero ($A' = 0$); consequently the multiplication can be divided into three distinct operations:

1. for $x_n = 0$ and $y_n = 0$:

$$\langle X \times Y \rangle_{2^n+1} = \langle P_1 \rangle_{2^n+1} = \langle X' \cdot Y' \rangle_{2^n+1} \tag{5}$$

2. When one and only one of the nth bits is equal to 1:

$$\begin{aligned}\langle X \times Y \rangle_{2^n+1} = \langle P_2 \rangle_{2^n+1} &= \langle 2^n y_n \cdot X' + 2^n x_n \cdot Y' \rangle_{2^n+1} \\ &= \langle y_n \cdot (-X') + x_n \cdot (-Y') \rangle_{2^n+1} \\ &= \langle y_n \cdot \overline{X'} + 2 + x_n \cdot \overline{Y'} + 2 \rangle_{2^n+1} \\ &= \langle y_n \cdot \overline{X'} + x_n \cdot \overline{Y'} + 4 \rangle_{2^n+1}\end{aligned} \tag{6}$$

3. Finally when the two nth bits of X and Y are equal to 1:

$$\langle X \times Y \rangle_{2^n+1} = \langle P_3 \rangle_{2^n+1} = \langle 2^n \cdot 2^n \rangle_{2^n+1} = \langle 1 \rangle_{2^n+1} \tag{7}$$

2.1 Computation of $\langle P_1 \rangle_{2^n+1}$

To compute (5), the approach presented in [14] can be applied:

$$\begin{aligned}&\langle P_1 \rangle_{2^n+1} \\ &= \langle X' \cdot Y' \rangle_{2^n+1} \\ &= \left\langle \sum_{i=0}^{n-1} 2^i x_i \cdot Y' \right\rangle_{2^n+1} \\ &= \left\langle \sum_{i=0}^{n-1} x_i \cdot \langle 2^i Y' \rangle_{2^n+1} \right\rangle_{2^n+1} \\ &= \left\langle \sum_{i=0}^{n-1} (x_i \cdot y_{n-i-1} \cdots y_0 \, \overline{y}_{n-1} \cdots \overline{y}_{n-i} + 1) + 2 \right\rangle_{2^n+1} \\ &= \left\langle \sum_{i=0}^{n-1} (PP_i + 1) + 2 \right\rangle_{2^n+1}\end{aligned} \tag{8}$$

In the above equation, it can be considered that modulo $2^n + 1$ adders intrinsically add an extra unit. Instead of using a multiplication matrix with $n \times n$ bits inputs for adding the partial products (PP_i), followed by a carry save adder (CSA) to add the constant 2 in (8) [14], the multiplication matrix can be redesigned to encompass an extra input, where such a constant is directly introduced. This is advantageous when multipliers are implemented using Wallace tree (WT) adders, because for most values (n) the delay is approximately the same as for $n + 1$ inputs: the number of full adders (FAs) in the critical path of an n-bit WT is given by $\mathrm{WT}(n + 1) = \lfloor 3/2\mathrm{WT}(n) \rfloor$ [15], as presented in Table 1. However, this is only a rough approximation of the computation delay not taking into account the wire propagation delay, which is becoming increasingly more significant in nowadays' technologies.

2.2 Computation of $\langle P_3 \rangle_{2^n+1}$

The computation of (7) is rather simple by taking into consideration that when $x_n = y_n = 1$, the partial results P_1 and P_2 are null and thus the final product is equal to 1. This particular case can be accommodated simply by setting the least significant bit of the multiplication to 1 whenever $x_n = y_n = 1$ (this operation can be efficiently performed using a simple OR gate). As most efficient modulo adders

Table 1: Number of FA stages in a WT structure

j	3	4	5–6	7–9	10–13	14–19	20–28	29–42	43–63	–
WT(n, j)	1	2	3	4	5	6	7	8	9	–

have the result of the least significant bit calculated before the most significant bit [14, 16], the OR gate should not have any implication in the computation time.

2.3 Computation of $\langle P_2 \rangle_{2^n+1}$

The most significant improvement proposed for the multiplication unit concerns the introduction of the partial product described in (6) into the final result. Zimmermann [14] adopted a multiplexer to select the result based on the fact that when the result of (6) is not null, then (5) is zero and vice versa. Such multiplexer is located at the output of the parallel multiplier, thus contributing to the critical path of the circuit. Our proposal introduces this partial product directly in the multiplication matrix, by adding extra inputs in the WT. However, each extra input line added to the matrix of adders corresponds to the introduction of an extra modulo $2^n + 1$ CSA, which intrinsically adds an extra unit [16]. Consequently, the constant initially introduced in the multiplication matrix has to be corrected in order to take into consideration both the extra inputs and the constant required by the partial product P_2 in (6). The two elements of P_2 are introduced in distinct inputs of the adder matrix, which requires the addition of two extra units and results in

$$\begin{aligned}
&\langle P_1 + P_2 \rangle_{2^n+1} \\
&= \left\langle \sum_{i=0}^{n-1} (PP_i + 1) + 2 + \underbrace{y_n \cdot \overline{X}'}_{PP_n} + \underbrace{x_n \cdot \overline{Y}'}_{PP_{n+1}} + 4 \right\rangle_{2^n+1} \\
&= \left\langle \sum_{i=0}^{n-1} (PP_i + 1) + (PP_n + 1) + (PP_{n+1} + 1) + 4 \right\rangle_{2^n+1} \\
&= \left\langle \sum_{i=0}^{n+1} (PP_i + 1) + 4 \right\rangle_{2^n+1} \qquad (9)
\end{aligned}$$

where the first constant term 2 in the above equation is intrinsically added.

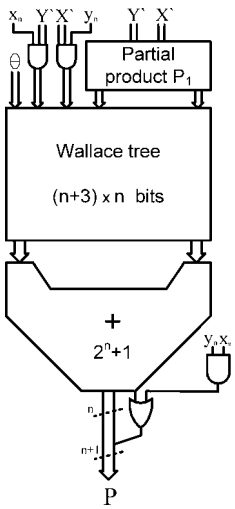

Fig. 1 *Modulo $2^n + 1$ multiplier*

The resulting architecture is depicted in Fig. 1, where θ represents the added constant.

To assess the practical efficiency of the new proposed architecture, modulo $2^n + 1$ multipliers were synthesised based on the proposed and on the original Zimmermann architectures. The results for the 0.13-μm CMOS technology are presented in Section 4, and, in particular, in Fig. 12; it can be observed the relative efficiency of the multipliers by considering the product of the circuit area (A) by the square of the time (T), that is, AT^2. Given the quadratic variation of the area and the linear variation of the delay, the AT^2 metric is typically used as an efficiency metric [13, 17]. These results show a significant improvement, that is, 20% in average and up to 50% for large ranges.

However, the relative performance of the proposed multipliers is still lower than that of the binary multipliers. Fig. 2 depicts the delay difference between the binary multiplier and the already optimised modulo $2^n + 1$ multiplier. This delay difference can rise upto 70% (for $n = 4$), being, on average, about 20%.

Also in the alternative diminished-1 number representation, proposed by Leibowitz [18], for the residues modulo $2^n + 1$, the difference between the binary multipliers and the $2^n + 1$ multipliers is even higher, as depicted in Fig. 3. The recently published [13] optimised diminished-1 modulo $2^n + 1$ multiplier was used. In this analysis, the difference is on average about 30%.

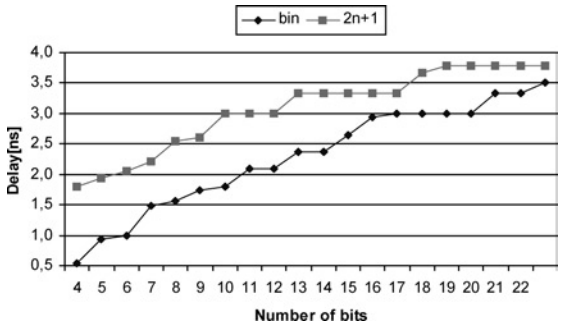

Fig. 2 *Delay imposed by the binary and the proposed modulo $2^n + 1$ multipliers*

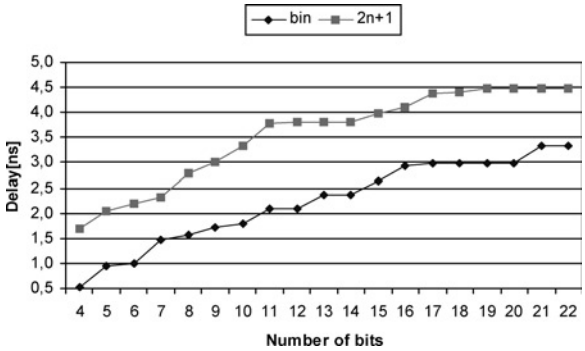

Fig. 3 *Delay imposed by the binary and the proposed modulo $2^n + 1$ diminished-1 multipliers*

These results underline the need for the other technique proposed in this paper in order to balance the moduli sets, and thus to compensate these significant differences in performance.

3 Balanced RNS moduli sets

As it was stated before, the fact that non-binary channels, mainly of the type 2^n+1, exhibit lower performance than the corresponding binary ones has a negative impact on the efficiency of RNS. In this section, the overloading of the binary channel is proposed in order to balance the computation time for the various channels, especially the 2^n+1 multiplication which is where the critical path is commonly located. The encoder and the decoder are the only new arithmetic units required by these new moduli sets, because the addition and the multiplication units for the traditional moduli set can be directly used.

This section is organised as follows: first (i) the correctness of the proposed binary channel overloading technique is proved and then (ii) the encoder and the decoder units for the proposed and modified 3- and 4-moduli sets are presented.

3.1 Binary channel extension

Recently, moduli sets with larger binary channels have been proposed, mainly to increase the dynamic range [19]. The width of the binary channel was doubled in order to increase the dynamic range to approximately the value achieved with 4-moduli supersets. In this paper, the width of the binary channel is increased by a variable factor 2^k, in order to overcome the relative difference in the complexity of the arithmetic units. The new class of moduli sets is comprehensive, in the sense that it also integrates the traditional 3-moduli sets and 4-moduli supersets. Although k can take any general value, that is, $k > -n$, we consider only values with practical interest: $k \geq 0$. Therefore the following general new class of moduli sets is defined

$$\{2^n-1,\ 2^{n+k},\ 2^n+1[,\ 2^{n+\gamma}+\alpha]\} \quad \text{for } k \in \mathbb{N}_0;\ \gamma,\ \alpha = \pm 1 \tag{10}$$

which includes the traditional 3-moduli and 4-moduli sets, for $k=0$. It is worth noting that the binary channel is the one that requires the simplest arithmetic units. For example, for fast adders based on binary tree structures the increase of, at most, n bits leads to the introduction of just one further level in the tree, which corresponds to the additional stage usually required by the corresponding modulo 2^n+1 adders. However, in our proposal k is a variable and can be adjusted according to the technology and the required dynamic range.

Let us prove the validity of the new class of moduli sets, which corresponds to proving that all the elements of the moduli set are relative primes.

3.1.1 Validity proof of the moduli sets: For proving that 2^n-1, 2^n+1, $2^{n+\gamma}+\alpha$ ($\gamma,\ \alpha=\pm 1$) and 2^{n+k} ($k \in \mathbb{N}$) are relative primes, it is only necessary to prove that 2^{n+k} is a relative prime to each of the other elements, because it has already been proved that all the other elements are relative primes (e.g. see [9, 20]).

Lemma 1: $(2^{n+k}, 2^n-1)$, $(2^{n+k}, 2^n+1)$, $(2^{n+k}, 2^{n+1}+1)$, $(2^{n+k}, 2^{n+1}-1)$, $(2^{n+k}, 2^{n-1}+1)$ and $(2^{n+k}, 2^{n-1}-1)$ are pairwise relative prime numbers for $k \in \mathbb{N}$.

Proof: The Euclidian algorithm for computing the greatest common divisor of two numbers ($\gcd(a, b)$) can be used to prove that a pair of numbers (a, b) are relative primes.

$$r_1 = \langle a \rangle_b,\ r_2 = \langle b \rangle_{r_1},\ \ldots,\ r_{k+1} = \langle r_{k-1} \rangle_{r_k}$$
$$r_{k+1} = \langle r_{k-1} \rangle_{r_k} = 0 \Rightarrow \gcd(a, b) = r_k \tag{11}$$

The six pairs of numbers in Lemma 1 can be reduced to only two different types, simply by replacing the variables. Therefore to prove Lemma 1 one only needs to compute the gcd for the following two pairs of numbers: $(2^{n+k}, 2^n-1)$ and $(2^{n+k}, 2^n+1)$, for $k \in \mathbb{N}_0$.

For 2^n-1 ($\varphi < n$, $\phi \in \mathbb{N}$):

$$\begin{aligned} \langle 2^{n+k} \rangle_{2^n-1} &= \langle 2^{\phi n+\varphi} \rangle_{2^n-1} = 2^\varphi, \\ \langle 2^n-1 \rangle_{2^\varphi} &= \langle -1 \rangle_{2^\varphi} = 2^\varphi - 1, \\ \langle 2^\varphi \rangle_{2^\varphi-1} &= 1, \\ \gcd(2^{n+k}, 2^n-1) &= 1, \end{aligned} \tag{12}$$

and for 2^n+1 ($\varphi < n$; $\phi \in \mathbb{N}$):

$$\begin{aligned} \langle 2^{n+k} \rangle_{2^n+1} &= \langle 2^{\phi n+\varphi} \rangle_{2^n+1} \\ &= \langle (2^n)^\phi \times 2^\varphi \rangle_{2^n+1} \\ &= \langle (-1)^\phi \times 2^\varphi \rangle_{2^n+1} \end{aligned} \tag{13}$$

For ϕ odd:

$$\begin{aligned} \langle -2^\varphi \rangle_{2^n+1} &= 2^n + 1 - 2^\varphi, \\ \langle 2^n+1 \rangle_{2^n+1-2^\varphi} &= 2^\varphi, \\ \langle 2^n+1-2^\varphi \rangle_{2^\varphi} &= 1 \end{aligned} \tag{14}$$

and for ϕ even:

$$\begin{aligned} \langle 2^\varphi \rangle_{2^n+1} &= 2^\varphi, \\ \langle 2^n+1 \rangle_{2^\varphi} &= 1 \end{aligned} \tag{15}$$

Thus

$$\gcd(2^{n+k}, 2^n+1) = 1 \tag{16}$$

From (12) and (16), it can be concluded that $(2^{n+k}, 2^n-1)$, $(2^{n+k}, 2^n+1)$, $(2^{n+k}, 2^{n+1}-1)$, $(2^{n+k}, 2^{n+1}+1)$, $(2^{n+k}, 2^{n-1}-1)$ and $(2^{n+k}, 2^{n-1}+1)$ are pairwise relative prime numbers for $n, k \in \mathbb{N}$. □

3.2 New conversion units

In order to implement the binary channel overloading technique, the encoder and the decoder memoryless units for the new sub class of 3-moduli sets $\{2^n-1, 2^{n+k}, 2^n+1\}$ are proposed. Values of k such as $0 < k \leq n$, where $n \in \mathbb{N}$, are considered, as well as the standard and the diminished-1 number representations.

3.2.1 Binary to RNS converters: An integer X in the range $[0, M-1]$, represented as

$$X = \sum_{i=0}^{4n-1} X_i 2^i = N_3 2^{3n} + N_2 2^{2n} + N_1 2^n + N_0 \tag{17}$$

where N_3 is a variable-length value, with k bits, represented as

$$N_3 = \{X_{3n+k-1} \cdots X_{3n}\} \quad \text{for } 1 \leq k \leq n \tag{18}$$

can be uniquely represented in RNS by the 3-tuple $\{x_1, x_2, x_3\}$ for the moduli set $\{2^n-1, 2^{n+k}, 2^n+1\}$ ($\{m_1, m_2, m_3\}$).

The dynamic range of this moduli takes the value M which is given by

$$M = \prod_{i=1}^{3} m_i = (2^n + 1) \times (2^{n+k}) \times (2^n - 1) = 2^{3n+k} - 2^{n+k} \quad (19)$$

In order to obtain the RNS representation of the integer X, three independent converters, one for each channel, are required. The simplest converter is the one for the m_2 channel, because for the $2^n - 1$ and $2^n + 1$ channels the calculation of the corresponding residues depends on the value of all the bits of X.

Converter for the 2^{n+k} channel. The value of x_2 can be obtained by the remainder of the division of X by 2^{n+k}, which can be accomplished by truncating the $(n + k)$ less significant bits of X

$$x_2 = \langle X \rangle_{2^{n+k}} = X_{n+k-1} \cdots X_0 \quad (20)$$

Converter for the 2^{n-1} channel. Instead of using a division operation to calculate the $2^n - 1$ residue, which is a complex and expensive operation, both in terms of area and speed, this calculation can be performed as a sequence of additions

$$x_1 = \langle X \rangle_{2^n-1} = \langle N_3 2^{3n} + N_2 2^{2n} + N_1 2^n + N_0 \rangle_{2^n-1} \quad (21)$$

By taking into account that

$$\langle 2^{k \times n} \rangle_{2^n-1} = \langle 1 \rangle_{2^n-1} \quad (22)$$

(21) can be rewritten as

$$x_1 = \langle N_3 + N_2 + N_1 + N_0 \rangle_{m_1} \quad (23)$$

Hence, the conversion to the $2^n - 1$ channel can be performed simply by adding modulo $2^n - 1$ the several segments of X. However, in order to perform these four additions one does not need to use four modulo $2^n - 1$ carry propagate adders (CPA), which would be slow and inefficient. As represented in Fig. 4, it is possible to group the values to be added as

$$x_1 = \langle \langle N_3 + \langle N_2 + N_1 + N_0 \rangle_{m_1} \rangle_{m_1} \rangle_{m_1} \quad (24)$$

with the first and second additions ($\langle N_2 + N_1 + N_0 \rangle_{2^n-1}$ and $\langle N_3 + S_1 + C_1 \rangle_{2^n-1}$, respectively), being performed by a 3–2 compressor (CSA), without a significant increase in the delay or in the area. Nevertheless, the third and last modulo $2^n - 1$ addition ($\langle S_2 + C_2 \rangle_{2^n-1}$) requires a modulo $(2^n + 1)$ CPA.

Converter for the $2^n + 1$ channel. Similarly, the $2^n + 1$ residue can be calculated as

$$x_3 = \langle X \rangle_{2^n+1} = \langle N_3 2^{3n} + N_2 2^{2n} + N_1 2^n + N_0 \rangle_{2^n+1} \quad (25)$$

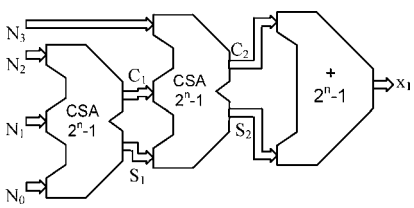

Fig. 4 *Modulo $(2^n - 1)$ binary to RNS converter*

Since

$$\langle 2^n \rangle_{2^n+1} = \langle -1 \rangle_{2^n+1}, \quad (26)$$

(25) can be simplified to

$$x_3 = \langle -N_3 + N_2 - N_1 + N_0 \rangle_{2^n+1} \quad (27)$$

The computation of (27) can be further simplified using only additions, given that a modulo $2^n + 1$ subtraction can be expressed as

$$\langle -N_i \rangle_{2^n+1} = \langle (2^n + 1) - N_i \rangle_{2^n+1} = \langle \overline{N}_i + 2 \rangle_{2^n+1} \quad (28)$$

and therefore (27) can be rewritten as

$$x_3 = \langle \overline{N}_3 + 2 + N_2 + \overline{N}_1 + 2 + N_0 \rangle_{2^n+1} \quad (29)$$

In the standard representation, the value of x_3 represents the $2^n + 1$ residue of value X, instead of value $X - 1$. It will be seen that the complexity of the conversion unit is greater in the former case than in the latter situation.

Like in the $2^n - 1$ modulo conversion, the additions can be grouped and partially added by 3–2 compressors. Nevertheless, the constant value 4 has to be added to compute (29). However, as referred before, modulo $2^n + 1$ compressors not only add the input values, but also intrinsically add an extra unit [16]. Therefore using this characteristic of the $2^n + 1$ modulo adders, the value of x_3 can be computed as

$$x_3 = \langle 1 + \langle 1 + \langle 1 + \overline{N}_3 + \langle 1 + N_2 + \overline{N}_1 + N_0 \rangle_{2^n+1} \rangle_{2^n+1} \rangle_{2^n+1} \rangle_{2^n+1} \quad (30)$$

The binary to modulo $2^n + 1$ converter is depicted in Fig. 5, where it can be noticed that the last CSA is only used to add one unit to $S_2 + C_2 + 0$, resulting in a simplified 2–2 compressor.

In the diminished-1 representation, the remainder of the division is represented as $x_3 = \langle X - 1 \rangle_{2^n+1}$, where the obtained value is always the correct value minus one. The computation of x_3 for this representation, using a similar approach to the one used in (29), results in the following expression:

$$x_3 = \langle X - 1 \rangle_{2^n+1} = \langle \overline{N}_3 + 2 + N_2 + \overline{N}_1 + 1 + N_0 \rangle_{2^n+1} \quad (31)$$

Decomposing the addition, (31) can be rewritten as

$$x_3 = \langle 1 + \langle 1 + \overline{N}_3 + \langle 1 + N_2 + \overline{N}_1 + N_0 \rangle_{2^n+1} \rangle_{2^n+1} \rangle_{2^n+1} \quad (32)$$

which corresponds to the hardware structure presented in Fig. 6, which is even simpler than the one depicted in Fig. 5 for the standard representation.

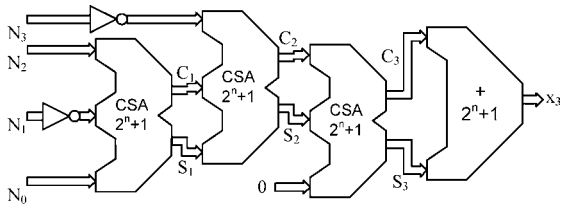

Fig. 5 *Modulo $(2^n + 1)$ binary to RNS converter*

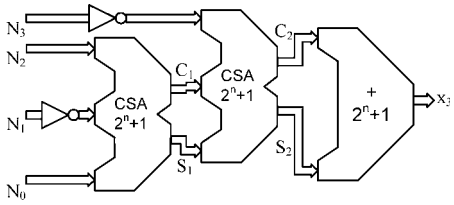

Fig. 6 *Modulo $(2^n + 1)$ binary to diminished-1 RNS converter*

3.2.2 RNS to binary converters: The proposed RNS to binary memoryless converter is based on the CRT [16, 20, 21]

$$X = \left\langle \sum_{i=1}^{3} \hat{m}_i \left\langle \frac{x_i}{\hat{m}_i} \right\rangle_{m_i} \right\rangle_M \tag{33}$$

where $\hat{m}_i = M/m_i$, $\langle 1/\hat{m}_i \rangle_{m_i}$ is the multiplicative inverse of $\hat{m}_i$ and M is given by (19). Equation (33) can be written as [16]

$$X + MZ(X) = \sum_{i=1}^{3} \hat{m}_i \left\langle \frac{1}{\hat{m}_i} \right\rangle_{m_i} x_i \tag{34}$$

where $Z(X)$ is a non-negative integer which is a function of X. For the proposed moduli set $\{m_1 = 2^n - 1, m_2 = 2^{n+k}, m_3 = 2^n + 1\}$, the multiplicative inverse of $\hat{m}_i$ $(i = 1, 2, 3)$ is [21]

$$\left\langle \frac{1}{\hat{m}_1} \right\rangle_{m_1} = 2^{n-1} \tag{35a}$$

$$\left\langle \frac{1}{\hat{m}_2} \right\rangle_{m_2} = 2^{n+k} - 1 \tag{35b}$$

$$\left\langle \frac{1}{\hat{m}_3} \right\rangle_{m_3} = 2^{n-1} \tag{35c}$$

By replacing these values in (34) we obtain

$$\begin{aligned} X + MZ(X) = {} & 2^{n+k}(2^n - 1)(2^{n-1})x_3 \\ & + (2^{2n} - 1)(2^{n+k} - 1)x_2 \\ & + 2^{n+k}(2^n + 1)(2^{n-1})x_1 \end{aligned} \tag{36}$$

and dividing X by 2^{n+k}:

$$X = 2^{n+k} \left\lfloor \frac{X}{2^{n+k}} \right\rfloor + x_2 \tag{37}$$

where

$$\begin{aligned} \left\lfloor \frac{X}{2^{n+k}} \right\rfloor = \Big\langle & \underbrace{\langle (2^{2n-1} + 2^{n-1})x_1 \rangle_{2^{2n}-1}}_{F} - 2^n x_3 \\ & + \underbrace{\langle (2^{2n} - 2^{n-k} - 1)x_2 \rangle_{2^{2n}-1}}_{G} \\ & + \underbrace{\langle (2^{2n-1} + 2^{n-1})x_3 \rangle_{2^{2n}-1}}_{H} \Big\rangle_{2^{2n}-1} \end{aligned} \tag{38}$$

Finally, with the simplification of the partial expressions F, G, H, $-2^n x_3$

$$\begin{aligned} F &= x_{1,0}x_{1,n-1} \cdots x_{1,1}x_{1,0}x_{1,n-1} \cdots x_{1,1} \\ G &= \bar{x}_{2,n+k-1} \cdots \bar{x}_{2,0} 1 \cdots 1 \\ H &= x_{3,x}x_{3,n-1} \cdots x_{3,1}x_{3,x}x_{3,n-1} \cdots x_{3,1} \\ -2^n x_3 &= -r_3 = \bar{x}_{3,n-1} \cdots \bar{x}_{3,0} 1 \cdots 1 \bar{x}_{3,n} \end{aligned} \tag{39}$$

the overlapping bits $x_{3,n}$ and $x_{3,0}$ are merged together and represented by $x_{3,x} = (x_{3,n}$ or $x_{3,0})$. This is only possible because $x_{3,n}$ and $x_{3,0}$ are mutually exclusive and can not be simultaneously equal to 1.

The upper 2^{n+k} bits of X can be calculated using two 3–2 compressors and one modulo $2^{2n} - 1$ CPA

$$\begin{aligned} \left\lfloor \frac{X}{2^{n+k}} \right\rfloor &= \langle H + G + F + (-2^n x_3) \rangle_{2^{2n}-1} \\ &= \langle \langle H + G + F \rangle_{2^{2n}-1} + (-2^n x_3) \rangle_{2^{2n}-1} \end{aligned} \tag{40}$$

As it is observed in the above equation, the conversion from RNS to binary using the new extended moduli set can be performed by adding modulo $2^{2n} - 1$ the four partial expressions in (39), directly obtained from the three RNS channels. In order to optimise the converter, only one modulo $2^{2n} - 1$ CPA is used in the final stage of the computation. The other additions are performed by two modulo $2^{2n} - 1$ CSAs, as depicted in Fig. 7, resulting in an RNS decoder with exactly the same structure as the RNS decoder for the traditional moduli set [20].

To decode an RNS value encoded in the diminished-1 representation, the value from the $2^n + 1$ channel (x_3) has to be incremented by one, which is equivalent to adding $2^{2n-1} - 2^{n-1}$ modulo $2^{2n} - 1$ in (38). This extra value can be added by an extra simplified CSA, in which one of the inputs is a constant value. As a result, only an extra HA is introduced in the critical path of the decoder, as depicted in Fig. 8.

Note that the complexity of the RNS to binary conversion for the new moduli sets is exactly the same as for the traditional moduli set. The only difference lays in the partial expression F, in that this new moduli set has no constant terms [20, 22].

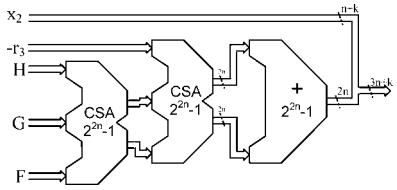

Fig. 7 *RNS decoder for the standard representation*

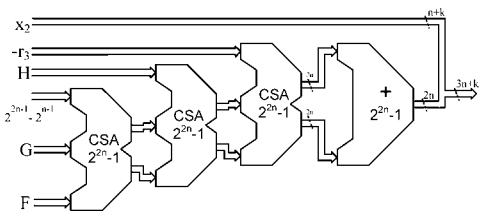

Fig. 8 *RNS decoder for the diminished-1 representation*

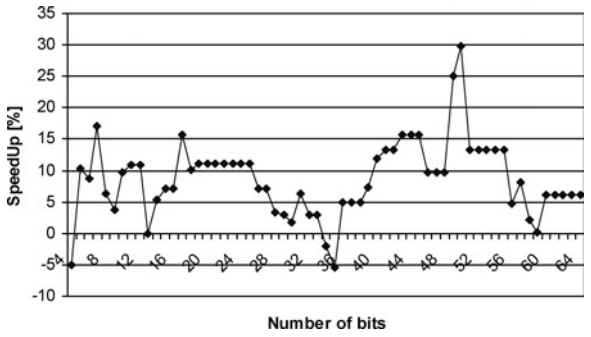

Fig. 9 *Relative delay of the new multiplier regarding the multiplier proposed by Zimmermann*

4 Experimental results

We have synthesised circuits to implement the enhanced modulo $2^n + 1$ multipliers and the backward and forward converters proposed for the new moduli sets, using the Synopsys synthesis tools and the 0.13-μm standard cell technology mentioned in Section 1. The obtained experimental results allow to evaluate the real impact of the proposed techniques individually, as well as the efficiency of the enhanced multipliers applied to the modified RNS moduli set.

4.1 Enhanced modulo $2^n + 1$ multipliers

Fig. 9 depicts the ratio of the processing time between the proposed modulo $2^n + 1$ multiplier and the one proposed by Zimmermann [14]. From this figure, it can be observed that the proposed multiplication architecture significantly improves the processing time. However, there are exceptions, as is the case for 37 bits. In this particular situation, the Zimmermann architecture has a better performance, since the advantages provided by the WT characteristics are not so significant for such width. The new multiplication architecture is, on average, 9% better than the one proposed by Zimmermann [14]. In fact, for bigger word lengths the improvement obtained by this multiplication architecture is, on average, above 10% (see Table 2).

The approach taken to introduce the partial product P_2 in the new multiplication architecture leads to an increase of the circuit area. This difference becomes less significant as the number of bits increases (see Fig. 10). This is only a significant disadvantage, in comparison to the architecture proposed by Zimmermann, when the number of bits is less than 8 (the circuit area increases by more than 20%). When the number of bits becomes greater than 16, the difference in the circuit area decreases to less than 10%. For operands with more than 40 bits, this difference is approximately 2%, as depicted in Fig. 11, while improving the delay by more than 10%.

To evaluate the efficiency of the proposed multipliers, the product of the circuit area (A) by the square of the time (T), AT^2, was adopted as figure of merit. Using such metric, and according to the experimental results depicted in Fig. 12, one can conclude that a significant improvement is achieved using the new multipliers, the efficiency being superior in more than 20%.

Table 2: Average improvement for the standard multiplication

Number of bits	Speed-Up	Improvement, [%]
4–64	1.1	9
40–64	1.12	11

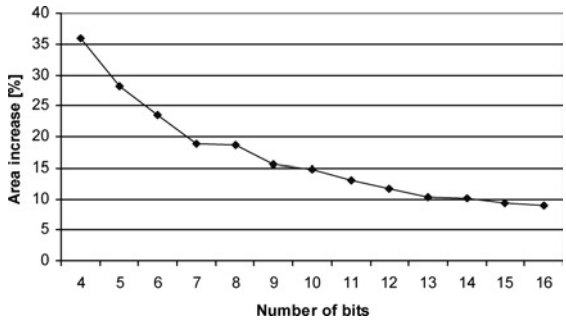

Fig. 10 *Relative circuit area of the new multiplier: word length below 16 bits*

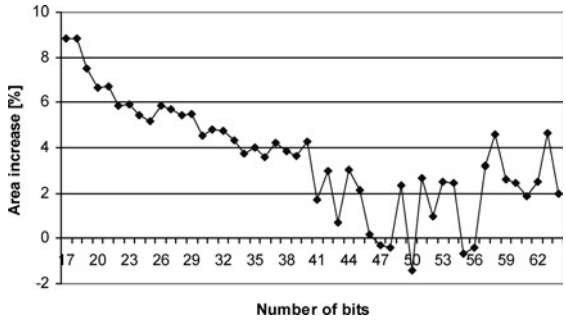

Fig. 11 *Relative circuit area of the new multiplier: word length above 16 bits*

4.2 New moduli set

The values presented for the new moduli set $\{2^n - 1, 2^{n+k}, 2^n + 1\}$ are for $0 \leq k \leq n$, whereas the values for the traditional moduli set are for $k = 0$. From Fig. 13, depicting the delay results for the binary to RNS

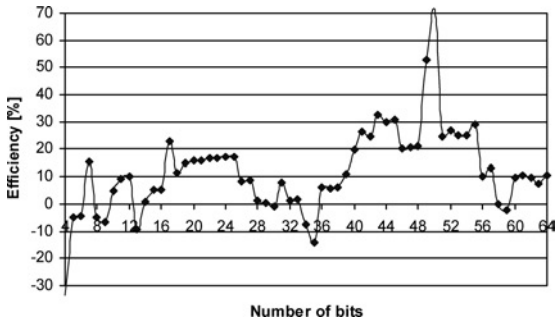

Fig. 12 *Relative efficiency (AT^2) of the new multiplier regarding the multiplier proposed by Zimmermann*

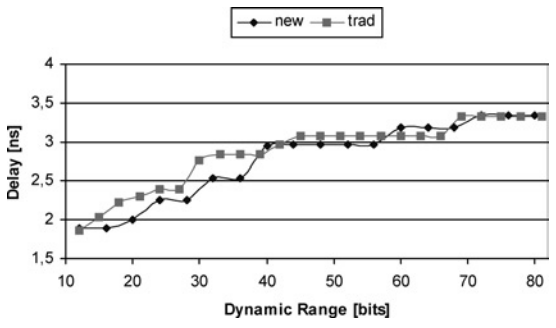

Fig. 13 *Delay of the new and original bin to RNS converters as a function of the dynamic range*

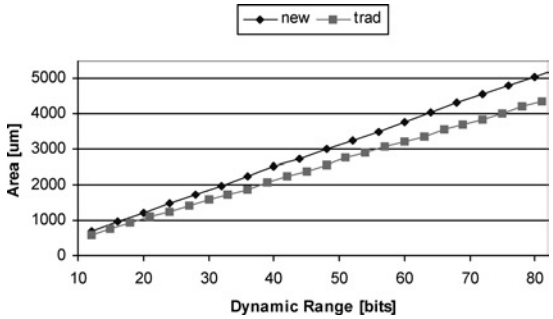

Fig. 14 *Area of the new and original bin to RNS converters as a function of the dynamic range*

converters, it can be seen that the converter for the new moduli set exhibits a slight performance improvement, about 4% on average. This is because of the shorter length of the modulo $(2^n \pm 1)$ adders used in the converters, because for the same dynamic range the value of n is smaller. However, these conversion units require about 16% more circuit area (Fig. 14) because of the usage of an extra CSA adder in the $2^n + 1$ channel to add the extra constant 1 (see Fig. 5 and (30)).

The proposed conversion units from binary to diminished-1 RNSs are extremely compact, even more than the conversion units for the traditional moduli set. These conversion units are, on average, 10% faster (Fig. 15) and, unlike for the standard representation, require slightly less circuit area than the traditional moduli set (Fig. 16).

The RNS to binary converter for the new moduli set uses exactly the same structure as the converter for the original moduli set, with the advantage of having a smaller value of n, thus using smaller adders. This advantage not only causes an average reduction of about 10% in the conversion time (Fig. 17), but also a decrease of about 15% in the required implementation area (Fig. 18). Such improvement is mostly due to the longer binary channel (2^{n+k}) that is directly used to obtain the lower $n + k$ bits of the conversion result.

Taking into account the total cost of the two conversion units, the advantage of using the new moduli set becomes clear. Both the forward and the backward conversion units have approximately the same delay, which does not happen for the traditional moduli set, where the RNS to binary conversion is slower than the converter in the opposite direction. Concerning the required circuit area, the proposed new moduli set allows for a reduction of, on average, 2%.

4.2.1 RNS multiplication in the new moduli set: The proposed moduli set achieves a more balanced processing delay between the arithmetic units, due to the overloading of the binary channel; thus leading to an improvement of

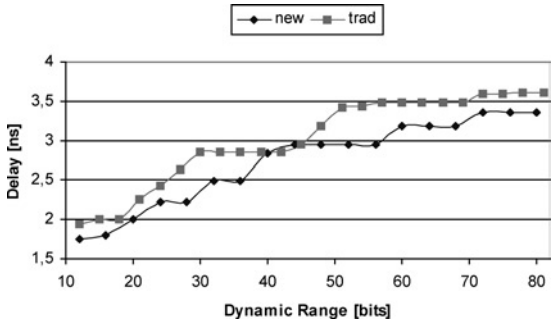

Fig. 15 *Delay of the new and original bin to diminished-1 RNS converters as a function of the dynamic range*

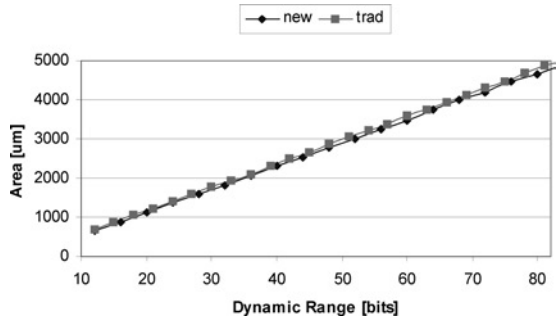

Fig. 16 *Area of the new and original bin to diminished-1 RNS converters as a function of the dynamic range*

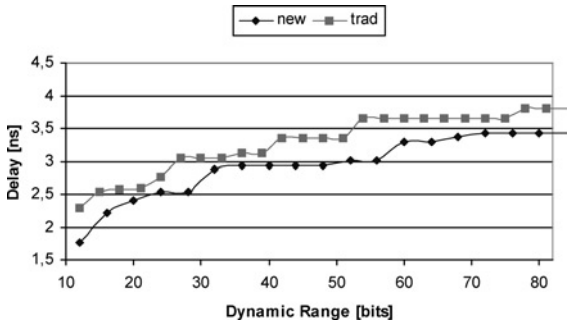

Fig. 17 *Delay of the new and original bin to RNS converters as a function of the dynamic range*

the overall computation efficiency. Fig. 19 depicts the delay difference for the multiplication operation, considering the new and the original moduli sets.

The use of the already optimised multiplier for the standard representation in the proposed moduli set allows an

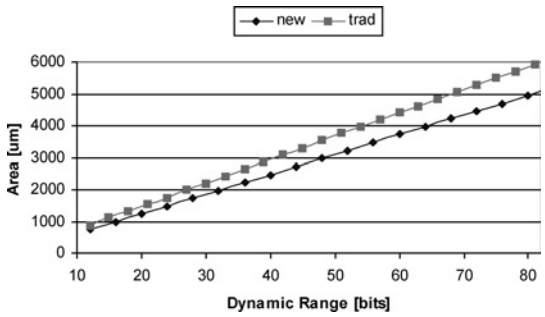

Fig. 18 *Area of the new and original bin to RNS converters as a function of the dynamic range*

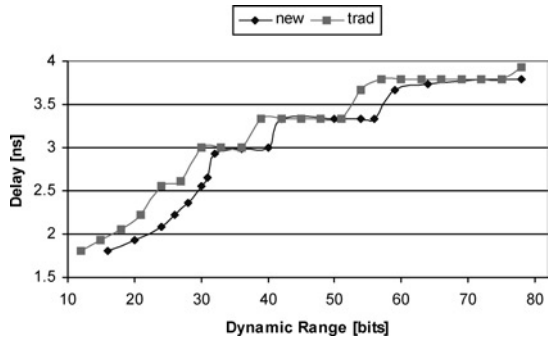

Fig. 19 *Delay of the proposed multipliers for the new and original moduli sets*

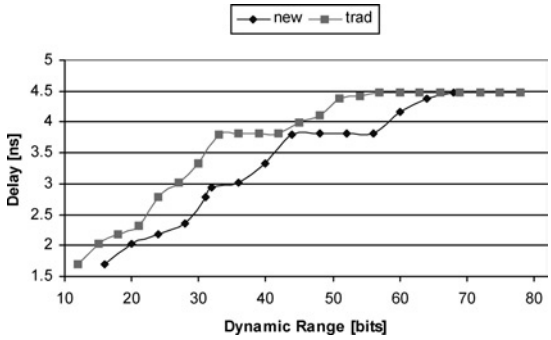

Fig. 20 *Delay of the proposed diminished-1 multipliers for the new and original moduli sets*

improvement in the multiplication delay up to 18%. For example, for $n = 17$ the proposed modulo $2^n + 1$ multiplier is 16% faster and 23% more efficient than the related art. Combining it with the proposed moduli set, for a dynamic range of 56 bits ($2^{17} - 1$, 2^{22}, $2^{17} + 1$), the multiplication becomes even 12% faster. This results in an overall improvement in the multiplication time of 32%, regarding the case where neither the improved multiplier nor the proposed moduli set are used.

For the diminished-1 representation, the improvement to the multiplication time is even higher. Fig. 20 depicts the difference between the multiplication delay in the original and in the proposed moduli set, using the already improved diminished-1 multiplier proposed by the authors of this paper [13]. For example, for a dynamic range of 56 bits, the proposed moduli set allows for a 15% faster multiplication.

5 Conclusions

This paper proposes an improvement to RNS that use the $2^n + 1$ modulo multiplication. The first step to improve the multiplication in RNS concerns the optimisation of the multiplication unit itself. However, because of the improved unit being even slower than the binary multipliers, we propose a new class of modified moduli sets that is capable of adapting the length of the binary channel ($\{2^n - 1, 2^{n+k}, 2^n + 1\}$) to the employed technology.

The proposed multiplier is an improved architecture based on the modulo $2^n + 1$ multiplier proposed by Zimmermann. Experimental results suggest that multipliers 10% faster can be achieved at the expense of some additional area. Nevertheless, for large dynamic ranges the area increase is not significant, requiring, on average, an additional 2% of silicon area. Furthermore, it can be concluded that for units using more than 9 bits, the proposed architecture is, on average, 20% more efficient than the original architecture.

The proposed extension of the binary channel allows the creation of more balanced and adaptable moduli sets. Conversion units were proposed for the particular case of the new 3-moduli set ($\{2^n - 1, 2^{n+k}, 2^n + 1\}$). Experimental results evidence more balanced forward and backward conversion times. Moreover, regarding the original conversion units, the proposed ones are faster and require less circuit area.

When both balanced moduli sets are adopted and enhanced multipliers are used, the overall multiplication can be further improved up to 32%, with a delay reduction between 7 and 15%.

6 References

1 Soderstrand, M., Jenkins, W., Jullien, G., and Taylor, F. (Eds.): 'Residue number system arithmetic: modem applications in digital signal processing' (IEEE, New York, 1986)

2 Chaves, R., and Sousa, L.: 'RDSP: a RISC DSP based on residue number system'. Symp. on Digital System Design: Architectures, Methods and Tools, September 2003, pp. 128–135

3 Toivonen, T., and Heikkïla, J.: 'Video filtering with Fermat number theoretic transforms using residue number system', *IEEE Trans. Circuits Syst. Video Technol.*, 2006, **16**, (1), pp. 92–101

4 Bajard, J.-C., and Imbert, L.: 'A full RNS implementation of RSA', *IEEE Trans. Comput.*, 2004, **53**, (6), pp. 769–774

5 Nozaki, H., Motoyama, M., Shimbo, A., and Kawamura, S.-I.: 'Implementation of RSA algorithm based on RNS montgomery multiplication' in Çetin Kaya Koç, Naccache, D., and Paar, C. (Eds.): 'CHES' vol. 2162 of Lecture Notes in Computer Science, (Springer, 2001), pp. 364–376

6 Lai, X.J., and Massey, L.: 'A proposal for a new block encryption standard'. EUROCRYPT, 1990, pp. 389–404

7 Nakamura, S., and Chu, K.-Y.: 'A single chip parallel multiplier by MOS technology', *IEEE Trans. Comput.*, 1988, **37**, pp. 274–282

8 Claudio, E.D., Piazza, F., and Orlandi, G.: 'Fast combinatorial RNS processors for DSP applications', *IEEE Trans. Comput.*, 1995, **44**, pp. 624–633

9 Bhardwaj, M., Srikanthann, T., and Clarke, C.: 'A reverse converter for the 4-moduli superset $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} + 1\}$'. Proc. 14th IEEE Symp. on Computer Arithmetic, April 1999, pp. 168–175

10 Conway, R., and Nelson, J.: 'Fast converter for 3 moduli RNS using new property of CRT', *IEEE Trans. Comput.*, 1999, **48**, pp. 852–859

11 Chaves, R., and Sousa, L.: 'Faster modulo $2^n + 1$ multipliers without Booth recoding', In XX Conf. on Design of Cricuit and Integrated Systems (DCIS), November 2005, ISBN 972-99387-2-5 (full paper in CD-Rom format).

12 Virtual Silicon Technology, Inc.: UMC high density standard cells library – 0.13 μm CMOS process v2.3 ed December 1999

13 Sousa, L., and Chaves, R.: 'A universal architecture for designing efficient modulo $2^n + 1$ multipliers', *IEEE Trans. Circuits Syst. I*, 2005, **52**, pp. 1166–1178

14 Zimmermann, R.: 'Efficient VLSI implementation of modulo $(2^n \pm 1)$ addition and multiplication'. Proc. IEEE Symp. on Computer Arithmetic, April 1999, pp. 158–167

15 Koren, I.: 'Computer arithmetic algorithms' (Prentice-Hall, 1993)

16 Ashur, A.S., Ibrahim, M.K., and Aggoun, A.: 'Novel RNS structures for the moduli set $(2^n - 1, 2^n, 2^n + 1)$ and their application to digital filter implementation', *Signal Process.*, 1995, **46**, (3), pp. 331–343

17 Claudio, E.D.D., Piazza, F., and Orlandi, G.: 'Fast combinatorial RNS processors for DSP applications', *IEEE Trans. Comput.*, 1995, **44**, (5), pp. 624–633

18 Leibowitz, L.: 'A simplified binary arithmetic for the Fermat number transform', *IEEE Trans. Acoust. Speech Signal Process.*, 1976, **24**, pp. 356–359

19 Hiasat, A., and Sweidan, A.: 'Residue-to-binary decoder for an enhanced moduli set', *IEE Proc., Comput. Digit. Technol*, 2004, **151**, pp. 127–130

20 Andraos, S., and Ahmad, H.: 'A new efficient memoryless residue to binary converter', *IEEE Trans. Circuits and Syst.*, 1988, **35**, pp. 1441–1444

21 Chaves, R., and Sousa, L.: '$\{2^n + 1, 2^{n+k}, 2^n - 1\}$: a new RNS moduli set extension'. Symp. on Digital System Design: Architectures, Methods and Tools, September 2004, pp. 210–217

22 Piestrak, S.J.: 'A high-speed realization of a residue to binary number system converter', *IEEE Trans. Circuits Syst.*, 1995, **42**, pp. 661–663