RNS Implementation of High Performance Filters for Satellite Demultiplexing

G.C. Cardarilli, A. Del Re, R. Lojacono, A. Nannarelli, M. Re
Department of Electronics
University of Rome "Tor Vergata"
Via del Politecnico, 1
Rome, Italy
+39 06 72597370
http://www.dspvlsi.uniroma2.it

Abstract—In this paper we present a Residue Number System (RNS) implementation of digital filters to be used for space applications. The RNS is particularly attractive because of the reduced power dissipation with respect to filters realized in the traditional binary representation. Furthermore, the reduced circuit complexity allows the implementation of programmable structures, in terms of filter coefficients and dynamic range, a feature appealing for remote operations on systems onboard satellites.

TABLE OF CONTENTS

1. INTRODUCTION

Residue Number System (RNS) arithmetic has been considered for long time as an interesting theoretical topic due to the complexity of the architectures deriving by the use of this number representation. However, the rapid growth of Integrated Circuits (IC) technology makes the use of RNS suitable for many DSP applications. A number of papers have been presented in the past years showing the advantages of the implementation of entire DSP subsystems in RNS arithmetic [1], [2].

The main advantages of the RNS representation are:

- the decomposition of a given dynamic range into parallel paths of smaller dynamic ranges, defined by the moduli set, which leads to carry-free operations among paths in different moduli and reduced delay;
- reduced complexity of the arithmetical units when large word lengths are needed;
- reduced power consumption.

The drawbacks of the RNS are illustrated in the following points:

- an input converter is required to translate numbers from the standard format (two's complement) into the residual one;
- an output converter is needed to implement the translation from the RNS to the standard representation.

The converters have a significant impact on complexity, latency and power consumption. However, in the case of data intensive computations (e.g. filters or filter banks) these disadvantages can be easily compensated by the earnings in the internal RNS computations. For these reasons, recently, the RNS arithmetic has been used in public-key cryptography, which requires multiplications of very large numbers, and in transmission systems based on the Code Division Multiple Access (CDMA).

A number of papers have been presented on the direct conversion from analog to RNS [3],[4]. The inverse conversion, from RNS to analog, which is not very appealing in actual applications, has been presented in [2]. However, the most popular conversion schemes are from binary (two's complement) to RNS and vice-versa [5].

The payload of a satellite system represents a typical system which can take advantage from the use of RNS arithmetic. In this application two main aspects are crucial:

- the power consumption;
- the access to the onboard resources.

Low power consumption is an intrinsic feature of the RNS architectures, due to the lower complexity and, consequently, to a reduced switching capacitance. As already mentioned, the parallelization usually implies a shortening of the critical path of a system, because narrower datapaths have shorter carry chains and consequently smaller delays. In [6] and [7] the power dissipation in RNS structures is reduced by taking

0-7803-7651-X/03/$17.00 @ 2003 IEEE
IEEEAC paper # 1059

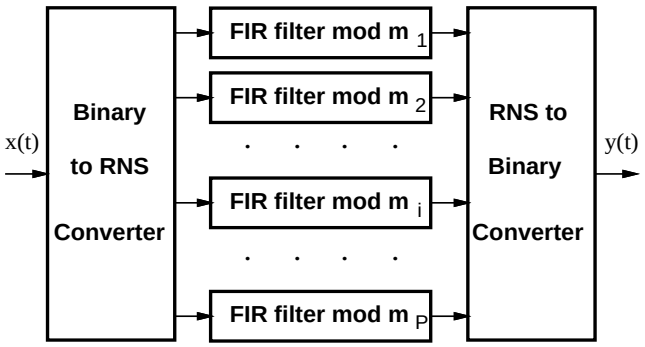

Figure 1. RNS FIR filter.

advantage of this parallelism.

In [6], the supply voltage is reduced, resulting in a quadratic reduction of power, until the speed-up over the traditional implementation is one. In [7], the supply voltage is reduced until a desired value of delay. In [8], we presented an approach to reduce the power dissipation in a RNS filter without penalizing its speed. We used the time slack available in the non critical parallel paths and reduced the power supply for these moduli until delays in all paths are equalized to the critical one. In this way, we reduce the power dissipation without affecting the overall performance.

In addition, because these structures are parallel, we can, for some applications, make partial use of the computing resources, by disabling the circuitry in some moduli path and reducing the power dissipation.

Another consequence of the lower complexity of the RNS datapaths, is that they can be made programmable or configurable at a smaller cost than corresponding binary datapaths. For example, in the implementation of FIR filters in the two's complement system (TCS), usually the multipliers are realized by hard-wiring the filter coefficient to simplify the partial products array (when a bit is zero the corresponding partial product is not generated) and obtain a smaller area. In RNS, multipliers have lower complexity and therefore, the impact on the area, and on the power, of implementing full multipliers (with programmable coefficients) is not as expensive as in TCS.

This work proposes a complete RNS architecture for an onboard satellite front-end. A RNS low power FIR filter architecture is connected to a direct converter from analog to RNS [4]. The performances of the proposed architecture are compared to the existing traditional realizations. Results show that RNS filter offer lower power dissipation than corresponding TCS filters, and also more flexibility in terms of programmability.

2. BACKGROUND ON RNS

In a Residue Number System, defined by a set of relatively prime integers $\{m_1, m_2, \ldots, m_P\}$ with dynamic range $M = \prod m_i$, any integer $X \in \{0, 1, \ldots M-1\}$ has a unique representation given by:

$$X \overset{RNS}{\rightarrow} (\langle X \rangle_{m_1}, \langle X \rangle_{m_2}, \ldots, \langle X \rangle_{m_P})$$

where $\langle X \rangle_{m_i} = X \ mod \ m_i = r_i$, i.e.

$$X = \alpha_i \cdot m_i + r_i \ ; \qquad i = 1, \ldots, P$$

Operations on single moduli are done in parallel

$$Z = X \ op \ Y \quad \overset{RNS}{\rightarrow} \quad \begin{cases} Z_{m_1} = \langle X_{m_1} \ op \ Y_{m_1} \rangle_{m_1} \\ Z_{m_2} = \langle X_{m_2} \ op \ Y_{m_2} \rangle_{m_2} \\ \ldots \qquad\qquad \ldots \qquad\qquad \ldots \\ Z_{m_P} = \langle X_{m_P} \ op \ Y_{m_P} \rangle_{m_P} \end{cases}$$

Therefore, the RNS allows the decomposition of a given dynamic range in slices of narrower bit-width on which the computation can be implemented in parallel [1], [2].

The conversion of the RNS representation of Z can be accomplished by the Chinese Remainder Theorem (CRT) [2]:

$$Z = \left\langle \sum_{i=1}^{P} \overline{m_i} \cdot \langle \overline{m_i}^{-1} \rangle_{m_i} \cdot Z_{m_i} \right\rangle_M \qquad \text{with } \overline{m_i} = \frac{M}{m_i}$$

and $\overline{m_i}^{-1}$ obtained by $\langle \overline{m_i} \cdot \overline{m_i}^{-1} \rangle_{m_i} = 1$.

3. RNS FIR FILTERS

The starting point of our design is a programmable N taps error-free FIR filter which, as shown in Figure 1, is decomposed into P filters working in parallel

$$y(n) = \sum_{k=1}^{N} a_k x(n-k) \quad \overset{RNS}{\rightarrow}$$

$$\begin{cases} Y_{m_1}(n) = \left\langle \sum_{k=1}^{N} \ \langle A_{m_1}(k) \cdot X_{m_1}(n-k) \rangle_{m_1} \right\rangle_{m_1} \\ Y_{m_2}(n) = \left\langle \sum_{k=1}^{N} \ \langle A_{m_2}(k) \cdot X_{m_2}(n-k) \rangle_{m_2} \right\rangle_{m_2} \\ \ldots \qquad\qquad \ldots \qquad\qquad \ldots \\ Y_{m_P}(n) = \left\langle \sum_{k=1}^{N} \ \langle A_{m_P}(k) \cdot X_{m_P}(n-k) \rangle_{m_P} \right\rangle_{m_P} \end{cases}$$

In the paper, we indicate the $i-$th modulus with m_i in expressions where other moduli appear, and just with m when we refer to operation done in a RNS path (only one modulus involved).

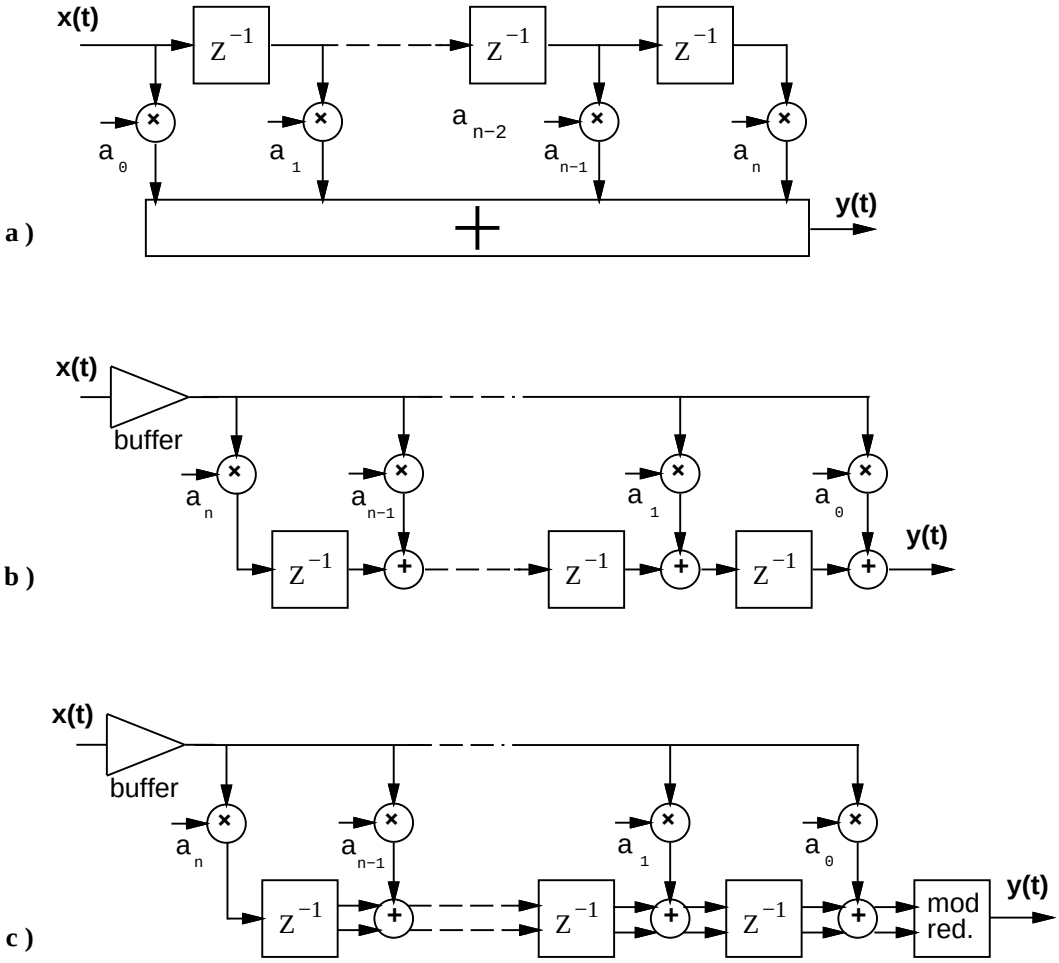

Figure 2. FIR filter in a) direct form, b) transposed form and c) carry-save transposed form.

FIR filters can be either realized in direct (Figure 2.a) or transposed form (Figure 2.b). For filters in direct form, the accumulation of the contributes of all the taps can be done with a tree of adders. In general, filters in different moduli paths can be implemented in different forms, as long as the timing is consistent.

Considering the delay of FIR filters, both the direct and the transposed form show a delay which depends on the number of taps (N). For filters in direct form the increased number of taps, implies a delay increase in the tree of adders. Furthermore, for direct form, we need to reduce the result of the addition in a number modulus m_i. We call this operation modulus reduction. For filters in transposed form, as N grows, we have to add (or increase) the buffering for x. The expressions for the delays of the filters in the two forms, except conversions, are:

$$t_{DIR} = t_{REG} + t_{modMULT} + t_{tree}(N) + t_{red}(N)$$
$$t_{TR} = t_{xbuf}(N) + t_{modMULT} + t_{modADD} + t_{REG}$$

where:

t_{REG} is the delay due to both propagation delay and set-up time in registers.

$t_{xbuf}(N)$ is the delay due to buffering of x: input to all taps.

$t_{modMULT}$ is the delay of the modular multiplication which does not depends on N, but is proportional to the size of the moduli.

t_{modADD} is the delay of the modular addition, which depends on the size of the moduli.

$t_{tree}(N)$ is the delay introduced by the tree of adders, which has a logarithmic dependency of N.

$t_{red}(N)$ is the time needed to reduce the output of the tree to a modulus m_i value.

In [8], we introduced the carry-save (CS) representation of the residues which allows a reduction on the addition time in the transposed form (Figure 2.c). With the CS representation we avoid to compute the modular addition in every tap and reduce the addition time to the delay of a XOR gate, as later explained in detail. The drawback of the CS representation is that registers are doubled and that, at some point, we need to perform the two term addition and extract its modulus m. However, the critical path for a carry-save transposed form RNS-path (CS-RNS in the following) is

$$t_{CS-RNS} = t_{xbuf}(N) + t_{modMULT} + t_{XOR} + t_{REG}$$

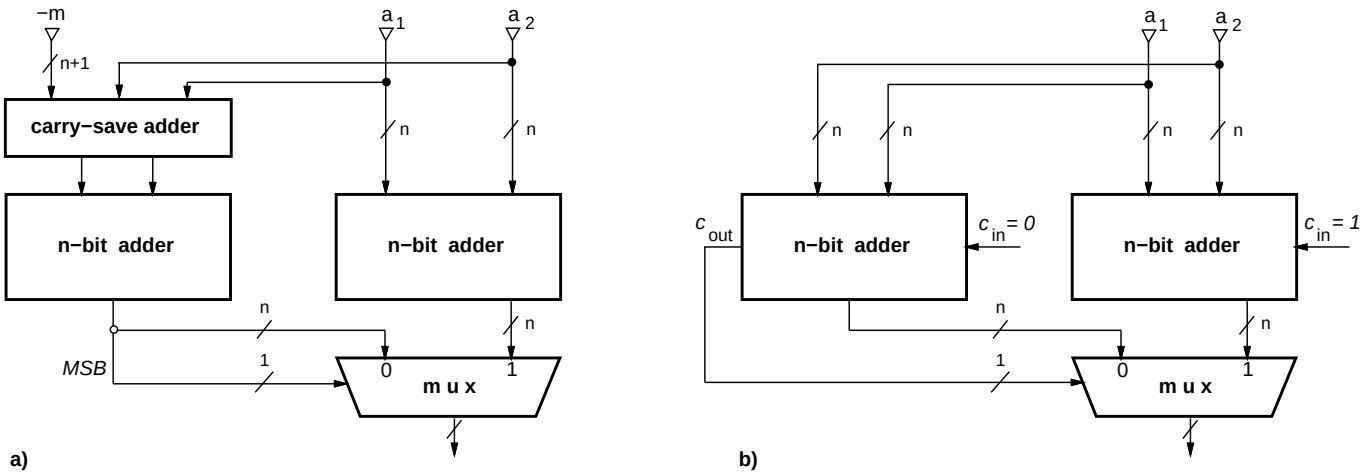

Figure 3. a) Structure of modular adder. b) Modular adder when $m = 2^n - 1$.

In the rest of this section, we describe the implementation detail of filter taps, limiting the discussion to filters in transposed and carry-save form.

Implementation of modular addition

The modular addition $\langle a_1 + a_2 \rangle_m$, consists of two binary additions. If the result of $a_1 + a_2$ exceeds the modulus (it is larger than $m - 1$), we have to subtract the modulus m. In order to speed-up the operation two additions are executed in parallel:

$$(a_1 + a_2) \quad \text{and} \quad (a_1 + a_2 - m).$$

If the sign of the three-term addition is negative it means that $(a_1 + a_2) < m$ and the modular sum is $(a_1 + a_2)$, otherwise the modular addition is the result of $(a_1 + a_2 - m)$. The above algorithm can be implemented with a carry-save adder (CSA), two n-bit ($n = \lceil \log_2 m \rceil$) adders[1] (nCPA), and a multiplexer as shown in Figure 3.a.

The modular adder can be simplified when:

$m = 2^n$. In this case a modulus 2^n addition only requires the sum of the n least-significant bits.

$m = 2^n - 1$. In this case, indicating with c_{out} the carry-out of the most-significant bit (MSB), the modulus $2^n - 1$ addition is computed as

$$a = \langle a_1 + a_2 \rangle_{2^n - 1} = a_1 + a_2 + c_{out}$$

and it can implemented, eliminating the carry-save adder of Figure 3.a, with two n-bit adders in parallel, one with carry-in $c_{in} = 0$, the other with $c_{in} = 1$ and then selecting the correct result according to the c_{out} computed in the adder with $c_{in} = 0$ (Figure 3.b).

In the general case of Figure 3.a, the delay of the modular adder is

$$t_{modADD} = t_{CSA} + t_{nCPA} + t_{MUX} \ .$$

The delay t_{nCPA} depends on the bit-width of the operands and, therefore, on the modulus m.

[1] Adders are implemented in a carry-look-ahead scheme. An extra inverter is required in the $(a_1 + a_2 - m)$ adder to obtain the sign.

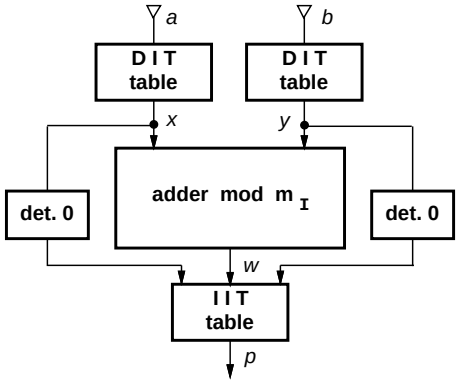

Figure 4. Structure of isomorphic multiplier.

Modular Multiplication

To reduce the complexity of modular multiplication, we use the isomorphism technique [9] to implement the product of residues. This method works quite well for small prime moduli ($m < 64$). By using isomorphisms, the product of the two residues is transformed into the sum of their indices which are obtained by an isomorphic transformation. According to [9], if m is prime there exists a primitive radix q such that its powers modulus m cover the set $[1, m-1]$:

$$p = \langle q^w \rangle_m \quad \text{with } p \in [1, m-1] \ , w \in [0, m-2] \ .$$

Both transformations $p \rightarrow w$ and $w \rightarrow p$ can be implemented with *m-1*-entry tables. Therefore, the product of a and b modulus m can be obtained as:

$$\langle a \cdot b \rangle_m = \langle q^w \rangle_m \ , \quad w = \langle x + y \rangle_{m-1} \ , \qquad \begin{matrix} a = \langle q^x \rangle_m \\ b = \langle q^y \rangle_m \end{matrix}$$

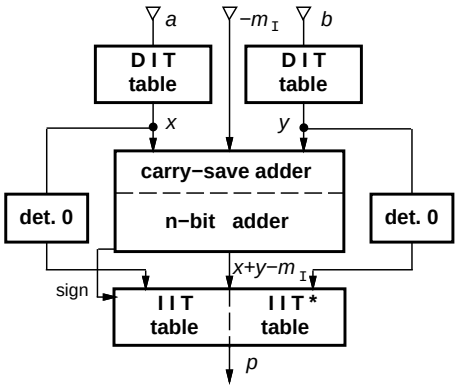

Figure 5. Isomorphic multiplier after first modification.

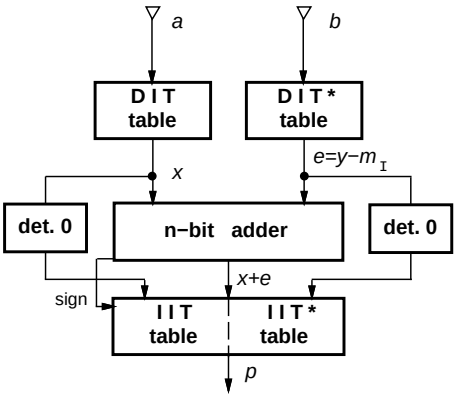

Figure 6. Isomorphic multiplier after second modification.

In order to implement the modular multiplication the following operations are performed:

1. Two direct isomorphic transformations (DIT) to obtain x and y;
2. One modulus $m_I = m - 1$ addition $\langle x + y \rangle_{m_I}$;
3. One inverse isomorphic transformations (IIT) to obtain the product.

The tables are implemented as synthesized multi-level logic and special attention has to be paid when one of the two operands is zero. In this case there exists no isomorphic correspondence and the modular adder has to be bypassed. The delay of the resulting scheme, shown in Figure 4, is

$$t_{isomult} = t_{DIT} + t_{CSA} + t_{nCPA} + t_{MUX} + t_{IIT} \ .$$

As a first step to simplify the structure, we eliminate one of the adders in parallel and we incorporate the modulus reduction in IIT table, as follows:

1. we compute: $x + y - m_I$ ($m_I = m - 1$, x and y are the indices of the isomorphism);
2. if the result is positive (e.g. $x + y \geq m_I$) we access the normal IIT table;
otherwise we access a modified table (IIT*) in which the inverse isomorphism is addressed by $\langle x + y - m_I \rangle_k$, where $k = \lceil \log_2 m_I \rceil$.

In this new scheme, depicted in Figure 5, the IIT tables are more complex (entries are doubled) than in the original scheme, but the delay is not increased because the multiplexer in the last stage of the modular adder is eliminated.

As a second step, we can incorporate the addition of the constant $-m_I$ in one of the DIT tables. The scheme of Figure 5 is modified as follows:

1. In one of the DIT tables instead of addressing the index of y we address $e = y - m_I$;
2. using a n-bit adder, we compute
$w = x + e = x + y - m_I$;
3. if w is positive (e.g. $x + y \geq m_I$) we access the normal IIT table;
otherwise we access the IIT* table, as previously shown.

In this new scheme (Figure 6) we have eliminated the carry-save adder which was in the critical path, while the modifications in DIT* do not affect the table delay.

In conclusion, with these two modifications we reduced the critical path of the isomorph multiplier to:

$$t_{modMULT} = t'_{isomult} = t_{DIT} + t_{nCPA} + t_{IIT^*} \ .$$

Because the input x (or its delayed value for direct form) is the multiplicand of all the multiplications (see Figure 2), DIT tables can be incorporated in the binary to RNS conversion, while the coefficients of the filter (multiplicators) can be directly loaded at start-up as isomorphism indexes.

Carry-Save RNS path

In order to speed-up the operations by making the clock period shorter, we can resort to a carry-save (CS) representation for the binary representation of residues (Ys_k, Yc_k) and avoid to compute the modular addition in every tap. The operands to be added in a tap are three: the product $p_k = \langle A(k)X(n-k) \rangle_m$ and the carry-save representation of Y_{k-1} (Figure 2.c)

$$Ys_k = sum(Ys_{k-1}, Yc_{k-1}, p_k),$$
$$Yc_k = carry(Ys_{k-1}, Yc_{k-1}, p_k).$$

The structure of the carry-save RNS tap is shown in Figure 7. The critical path is:

$$t_{CS-RNS} = t_{xbuf}(N) + t_{modMULT} + t_{XOR} + t_{REG}$$

The critical path is essentially determined by the multiplier latency, being t_{xbuf} and t_{REG} unavoidable, and being t_{XOR} the minimum delay for a half-adder.

However, the CS-representation implies the doubling of the registers, and, as the number of taps increases, a logarithmic

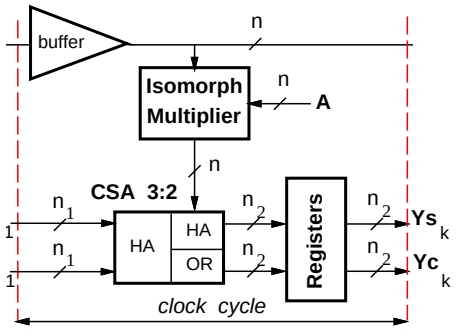

Figure 7. Tap structure for RNS carry-save.

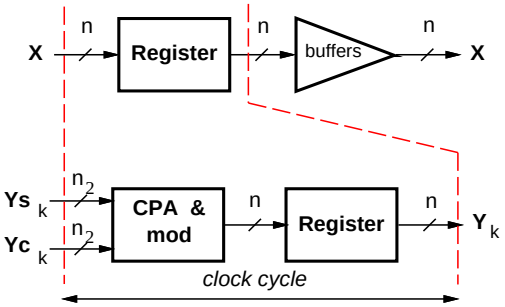

Figure 8. Relay station.

increase in the bit-width of CSAs and registers. For this reason, it might be convenient to insert every n taps some *relay stations* (RS), which assimilate the CS representation of Y_k and extract its modulus m, to prevent the bit-width from growing too much. Relay stations are depicted in Figure 8 and their delay is

$$t_{RS} = t_{cpa\&mod}(n) + t_{REG}$$

They introduce an extra cycle of latency, but they can also be used to better dimension the buffering of X (i.e. reduce t_{xbuf}).

The spacing of relay stations (i.e. the number n of taps between two relay stations) can be determined by finding the n which satisfies

$$\begin{cases} t_{buf_i} + t_{buf_L} \cdot n + t_{modMULT} + t_{XOR} + t_{REG} < T_{clock} \\ t_{cpa\&mod}(n) + t_{REG} < T_{clock} \end{cases}$$

4. Analog to RNS converter.

Recently, in order to take advantage from the RNS representation, a new architecture for a RNS flash Analog to Digital Converter (ADC) was proposed by the authors [4]. This architecture, which exhibits a complexity comparable with that of a traditional ADC, provides a suitable algorithm for the detection and the correction of the RNS residues, avoiding the errors that can be induced in such a non-positional representation.

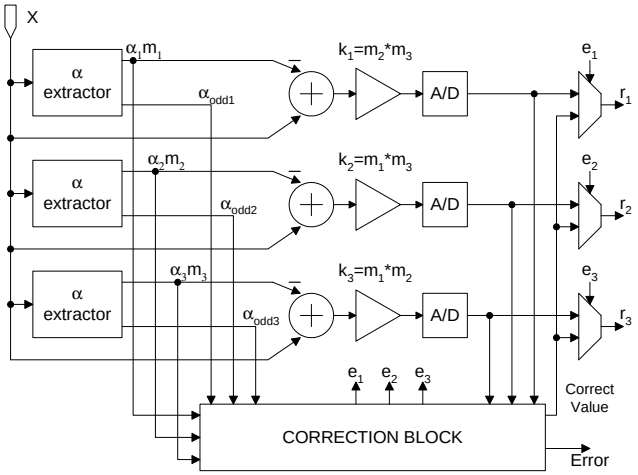

Figure 9. ADC architecture.

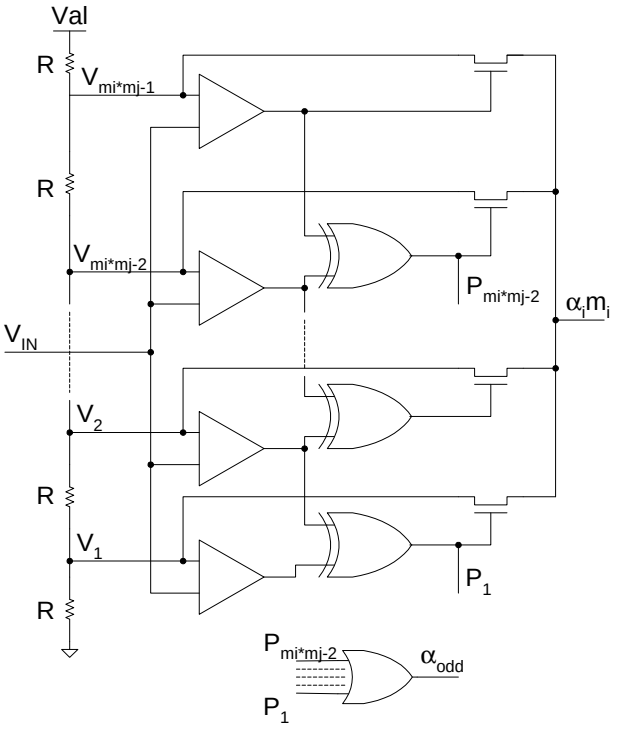

Figure 10. α-extractor architecture.

The proposed architecture is derived from the two stage flash converter. This last kind of converter was introduced because the flash converter becomes too complex when the the word-length is greater than eight bits and the resulting circuits are too slow. The analog to RNS converter (ARNS) proposed is shown in Fig. 9. The figure refers to a three moduli representation. In this figure, the determination of each residue is implemented by a structure similar to the scheme of the two stage flash converter. A first stage determines the integer quotient (α-extractor) and a second stage determines the residue. However, since the integer quotient is not of interest, its determination can remain analog, thus avoiding the A/D and

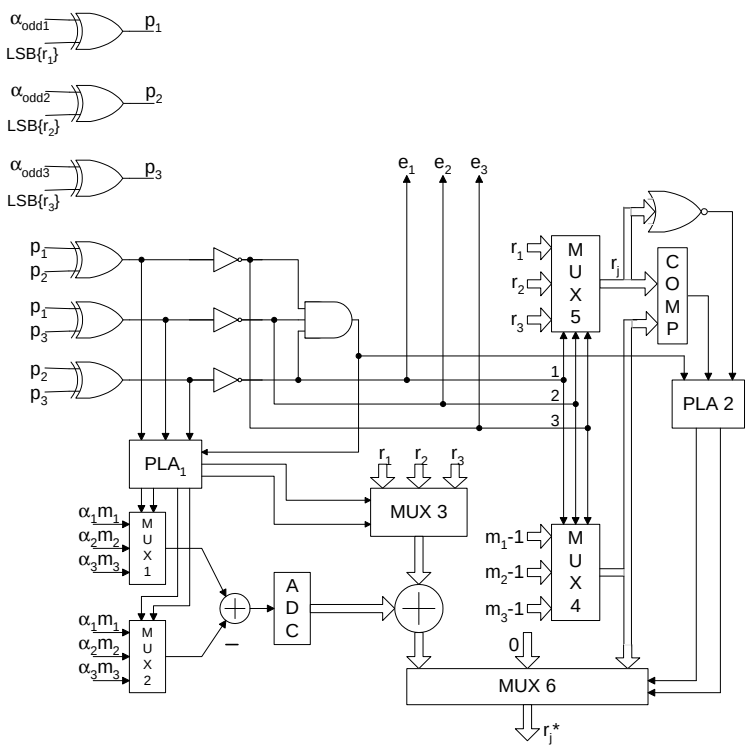

Figure 11. Correction block architecture.

D/A converters. The α-extractor is realized by the circuitry of Fig. 10, very similar to the input part of the two stage flash converter. A correction block is also introduced to avoid the problems of wrong conversion.

If some errors occur in the ARNS conversion, some of the p_i are not equal to $LSB(X)$. We can assume that the correct value of the parity (p^*) can be chosen on the basis of the majority voting rule. Therefore, we can select the wrong parity p_j $j = 1...L$ for which we have that either α_i or r_i are wrong (if α_i and r_i are together wrong, the parity p_j is equal to p^* and the presence of these errors cannot be detected.

Assuming that we have an error in the least significant bit, α_j can be wrong if X is very close to a multiple of m_j: in this case, the corresponding r_j will be either 0 or $m_j - 1$. To determine the correct α_j we must check r_j.

If $r_j = m_j - 1$, α_j must be changed in $\alpha_j + 1$ and r_j in 0.

If $r_j = 0$, α_j must be changed in $\alpha_j - 1$ and r_j in $m_j - 1$.

Therefore, the parity p_j is changed in p^*.

If X is not close to a multiple of m_j, i.e. $r_j \neq 0$ and $r_j \neq m_j - 1$, it is not reasonable to suppose that α_j is wrong; then we assume that r_j is wrong and must be changed in $r_j + 1$ or $r_j - 1$. Unfortunately, we do not have sufficient information to resolve this ambiguity. This problem can be overcome by introducing some additional hardware. In fact, we can perform a binary conversion of the difference

$$\alpha_i m_i - \alpha_j m_j = r_j - r_i = r_{ij}$$

Using this difference, it results that the correct value of r_j is given by

$$r_j = r_{ij} + r_i$$

Fig. 11 shows the structure of the correction block.

The structure of the proposed ARNS converter, shown in Fig. 9, is composed by N ideal converter chains (one for each modulus) compose that scheme. Each chain is considered ideal; i.e. noise and non-linearity effects are absent. Moreover, the overall actual internal noise is taken into account by noise sources placed at the input of each modular converter. In the following analysis, we consider truncation quantization but similar methods can be developed for analyzing ARNS converters based on other quantization laws.

Let us consider an input noiseless voltage V and the chain conversion related to the i-th modulus. The sum of the input and the noise voltages gives the final value $V_i = V + n_i$, where n_i represents the introduced noise.

It is worth noting that, in the analog to RNS conversion,

all the modular converters use the same quantization step q, therefore the distance Δ of the voltage V from the quantized quantity V' (defined by the expression $V' \leq V < V' + q$) is the same for all the modular chains. This concept is sketched in Fig. 12 where the input voltage V and its quantized representation V' are shown, together with noiseless quantization error Δ. For each modular chain is also reported the corresponding residue value.

In that figure, beside the quantization grid, for each modulus, is also represented the Probability Distribution Function (pdf) of the actual input voltage V_i of each channel. For the sake of simplicity, our analysis supposes that all the channel noise sources have the same variance σ^2. For the i-th modulus we have the error probability expression

$$p_e^{(i)} = \frac{1}{\sigma\sqrt{2\pi}} \left(\int_{|\Delta|}^{\infty} e^{-\frac{\varepsilon}{2\sigma}} d\varepsilon + \int_{q-|\Delta|}^{\infty} e^{-\frac{\varepsilon}{2\sigma}} d\varepsilon \right)$$

Above expression computes the probability of obtaining a voltage value V_i lying outside the correct quantization interval. For an N-moduli converter, according to the majority voting rule, we are able to correct the final values of the residues r_i if and only if the number of wrong residues is less than $\lfloor \frac{N}{2} \rfloor$ (here we assume that the probability of uncorrectable errors that preserve the parity is negligible).

Considering all the possible correctable error combinations and their own probabilities, the probability p_c to obtain a correct conversion result is given by the following expression:

$$p_c = \Sigma_{j=0}^{\lfloor \frac{N}{2} \rfloor} \frac{N(N-1)...(N-J+1)}{j!} p_e^j (1-p_e)^{N-j}$$

The probability values of correct conversion result computed for various values of N (the number of the moduli used in the RNS representation) are compared with the probability of correct result for the traditional ADC having the same quantization step (see fig.13).

5. The Output Conversion

In this section a new efficient architecture for the implementation of the output conversion is shown. This architecture can be configured for a set of N moduli and for the conversion of unsigned and signed integers. The classical formulation for the Chinese Remainder Theorem based on a N moduli set is

$$\langle X \rangle_M = \left\langle \sum_{i=1}^{N} \hat{m}_i \langle \hat{m}_i^{-1} \cdot r_i \rangle_{m_i} \right\rangle_M = \langle H \rangle_M \qquad (1)$$

where $\langle\ \rangle_T$ is the $mod\ T$ operator, $M = \prod_{i=1}^{N} m_i$, $r_i = \langle X \rangle_{m_i}\ i \in [1,\ N]$,

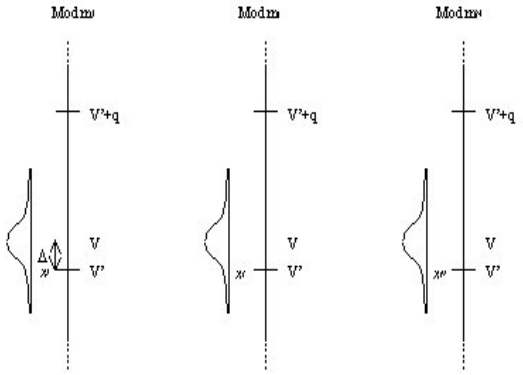

Figure 12. Quantization voltage levels grid for the different modular chains.

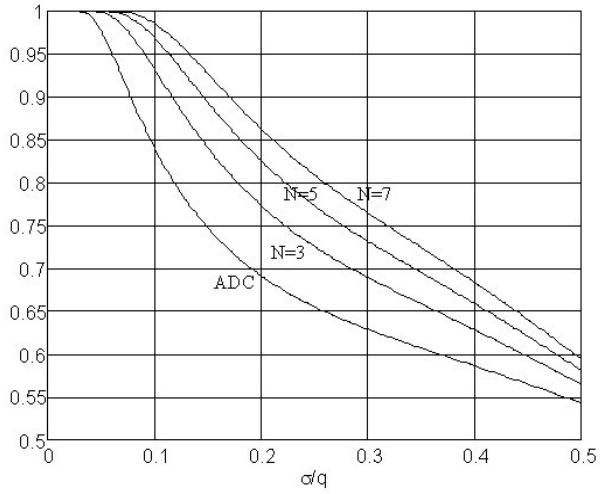

Figure 13. Error vs. noise variance

$$\hat{m}_i = \frac{M}{m_i}$$

and the quantities $\hat{m}_i^{-1}$ represent the multiplicative inverse of $\hat{m}_i$, i.e.

$$\langle \hat{m}_i \hat{m}_i^{-1} \rangle_{m_i} = 1 \qquad (2)$$

When (1) is implemented by a digital architecture two problems arise. The first one concerns the complexity of the involved arithmetic operations (a set of modulo additions and modulo multiplications are required). There are a number of methods to efficiently implement the computation of the term H. In [10] look-up tables (LUT) are used to compute the terms and a tree of carry save adders implements the summation.

The second problem is related to the computation of the external $mod\ M$ operation. This operation is very complex [11] due to the large value of M in the final $mod\ M$ operator and

to the dynamic range of the term H. In fact, from (1) we obtain the following bounds

$$0 \leq H = \sum_{i=1}^{N} \hat{m}_i \langle \hat{m}_i^{-1} \cdot r_i \rangle_{m_i} \leq \sum_{i=1}^{N} \frac{M}{m_i} \cdot (m_i - 1) < N \cdot M \tag{3}$$

Equation (3) shows the relation between the range of H and N. Moreover, the methodologies used for the modulo computation of specific modulus set (as those based on moduli close to powers of two) do not appear to be useful for this modulo operation. Indeed, if we maintain the generality of the procedure, the final modulo cannot be constrained. To obtain a more suitable form for the $mod\ M$ operation, let us consider the number $X \cdot 2^k$ being k a suitable integer quantity. Multiplying both the members of (1) by 2^k we obtain

$$\langle X \cdot 2^k \rangle_M = \left\langle \sum_{i=1}^{N} \hat{m}_i \langle \hat{m}_i^{-1} \cdot r_i \cdot 2^k \rangle_{m_i} \right\rangle_M \tag{4}$$

The terms of the summation in (4) have the same dynamic range as given by (3) since the factor 2^k appears inside a $mod\ m_i$ operation. Equation (4) can be rewritten as

$$X \cdot 2^k = \sum_{i=1}^{N} \hat{m}_i \langle \hat{m}_i^{-1} \cdot r_i \cdot 2^k \rangle_{m_i} - \alpha \cdot M \tag{5}$$

where α comes from the external modulo operation. From (5) we get

$$X = \frac{\sum_{i=1}^{N} \hat{m}_i \langle \hat{m}_i^{-1} \cdot r_i \cdot 2^k \rangle_{m_i} - \alpha \cdot M}{2^k} = \frac{H - \alpha \cdot M}{2^k} \tag{6}$$

Properties of (6) has been exploited in [12]. Due to the presence of a power of two modulus, this expression cannot be directly used for the computation of the output conversion. In the present case, (5) must be modified by taking into account that one of the residues, is a power of two (we suppose $m_N = 2^h$). In this case, we have

$$\langle X \rangle_{2^h} = r_N \tag{7}$$

From (7) it derives that the h least significant bits of X correspond to the h bits of r_N. This means that the reconstruction of these bits does not require any operation in the residue to binary conversion process. In this case, the main task of the converter is the reconstruction of the remaining most significant bits of X. These bits correspond to the number ε defined as

$$\varepsilon = \frac{X - \langle X \rangle_{2^h}}{2^h} = \frac{X - r_N}{2^h} \tag{8}$$

Starting from this value the converted value X can be obtained by

$$X = \varepsilon \cdot 2^h + r_N. \tag{9}$$

The ε value can be computed by introducing (6) in (8)

$$\varepsilon = \frac{\frac{H - 2^k r_N}{2^h} - \alpha \widetilde{M}}{2^k} \tag{10}$$

where $\widetilde{M} = M/2^h$. Since the definition of the term H implies that

$$\langle H \rangle_{2^h} = \left\langle 2^k r_N \right\rangle_{2^h} \tag{11}$$

the first term of the numerator of (10) is an integer quantity $\widetilde{H}$ given by

$$\widetilde{H} = \frac{H - 2^k r_N}{2^h} \tag{12}$$

Using (12), (10) can be rewritten as

$$\varepsilon = \frac{\widetilde{H} - \alpha \cdot \widetilde{M}}{2^k} \tag{13}$$

Due to the scaling by the factor 2^h, this expression requires for its computation a reduced dynamic range. Eq.(13) is similar to (6) and, as we show later, a simplified method can be used to select the value $\alpha\widetilde{M}$. In the following, all the expressions are defined in terms of ε, $\widetilde{H}$, $\widetilde{M}$.

The most difficult task, in the evaluation of (13), is the computation of the term $\alpha\widetilde{M}$. To solve this problem, we first evaluate the dynamic range of the term $\widetilde{H}$. Starting from (12) we obtain

$$-2^k < \widetilde{H} < N \cdot \widetilde{M} \tag{14}$$

consequently, the factor α belongs to the interval $-2^k < \alpha < N$.

Starting from this result, (13) suggests an efficient method to find the right value $\alpha \cdot \widetilde{M}$ to be subtracted to $\widetilde{H}$. In fact, in order to obtain integer values of ε (the reconstructed value), the quantity $\widetilde{H} - \alpha \cdot \widetilde{M}$ must be a multiple of 2^k. This means

that the k least significant bits of $\widetilde{H} - \alpha \cdot \widetilde{M}$ must be equal to zero. Starting from this observation, we can derive that the correct value of the term α belongs to the subset

$$\Upsilon = \{\alpha \in I : \langle \alpha \cdot \widetilde{M} \rangle_{2^k} = \langle \widetilde{H} \rangle_{2^k}\} \tag{15}$$

Where I is the set of integer numbers. This subset only depends on the k least significant bits of $\widetilde{H}$. Unfortunately, using these bits we are able to select only 2^k values of $\alpha \cdot \widetilde{M}$, out of the $N + 2^k + 1$ possible values, according with (14). If k is chosen such that

$$2^k \geq N - 1 \tag{16}$$

the values of $\alpha \cdot \widetilde{M}$ can be computed starting from the 2^k positive values stored in a very small LUT. In fact, since ε must be a positive number, the quantity $\widetilde{H} - \alpha \cdot \widetilde{M}$ must be positive. If this does not happen, the obtained value of $\alpha \in \Upsilon$ is incorrect. From (14) and (15) the correct value is obtained by subtracting 2^k from the incorrect one. So, if α' is the incorrect value addressed by the LUT and α is the correct one, ε is obtained by

$$\varepsilon = \frac{\widetilde{H} - \alpha \cdot \widetilde{M}}{2^k} = \frac{\widetilde{H} - \alpha' \cdot \widetilde{M}}{2^k} + \widetilde{M} \tag{17}$$

The procedure deriving from (17) can be summarized by the following steps.

1. The term $\alpha' \cdot \widetilde{M}$ is read from the LUT addressed by the k least significant bits of $\widetilde{H}$.
2. The sum $\widetilde{H} - \alpha \cdot \widetilde{M}$ is computed and the k least significant bits are discarded.
3. If the obtained result is negative the quantity $\widetilde{M}$ is added.

A numerical example

In the following, a numerical example is given. Let us consider the case of a RNS representation based on the moduli set,

$$m_i = \{3,\ 5,\ 7,\ 8\}$$

where $r_4 = 2^3$ (i.e. $h = 3$). The number of moduli is four therefore, from (16), $k = 2$. For this set we have

$$\hat{m}_i = \{280,\ 168,\ 120,\ 105\}, \quad \hat{m}_i^{-1} = \{1,\ 2,\ 1,\ 1\}, \quad M = 840$$
$$\widetilde{M} = 105, \qquad P = 420$$

and

$$H = 280\langle 1 \cdot 2^k \cdot (r_1 + P)\rangle_3 - 168\langle 2 \cdot 2^k \cdot (r_2 + P)\rangle_5 + 120\langle 1 \cdot 2^k \cdot (r_3 + P)\rangle_7 + 105\langle 1 \cdot 2^k \cdot (r_4 + P)\rangle_8$$

Consider the value $X = -209 \xrightarrow{RNS} \{1,\ 1,\ 1,\ 7\}$.

$$H = 1684, \quad \widetilde{H} = \frac{1684 - 4 \cdot \langle 7 + 420\rangle_8}{8} = 209$$

The correct α value is 1. Consequently we have $\varepsilon' = 26$ and for X' we obtain $X' = 26 \cdot 8 + \langle 7 + 420\rangle_8 = 211 > 0$.

In this case we have to subtract the term $P = 420$ obtaining $X = 211 - 420 = -209$.

The VLSI Architecture

The converter architecture for a generic set of moduli is sketched in Fig. 1. The N LUTs are addressed by the residues r_i and store the terms

$$\hat{m}_i \langle \hat{m}_i^{-1} \cdot 2^k \cdot (r_i + P)\rangle_{m_i}$$

The LUT-N stores the term $\hat{m}_N \langle \hat{m}_N^{-1} 2^k (r_N + P)\rangle_{m_N} - 2^k \langle r_N + P\rangle_{2^h}$. A Carry-Save Adder tree is used to compute $\widetilde{H}$. The k least significant bits of $\widetilde{H}$ are used to address the LUT $\alpha\widetilde{M}$ that stores the multiples $\alpha'\widetilde{M}$. The selected multiple is added to $\widetilde{H}$ in order to obtain the value ε'. The h least significant bits of the value $\langle r_n + P\rangle_{2^h}$ are directly juxtaposed with ε' to obtain the value X'. The correct signed value X is obtained by a final summation. Depending on the sign of X' the value $-P$ or $M - P$ is conditionally added to X'.

A VLSI implementation based on the moduli set $\{3,\ 5,\ 7,\ 11,\ 17,\ 64\}$ for a 20 bit converter has been implemented (Fig. 2). The architecture requires six LUTs that are normally very small. In fact the input LUTs are related to the moduli wordlength that can be chosen sufficiently small for the most common dynamic ranges. The computation of the term $\widetilde{H}$ has been obtained by using a Carry-Save Adder (CSA), and a carry-save representation has been maintained where possible. A Carry-Propagate Adder (CPA) has been used to obtain the address to the LUT-$\alpha\widetilde{M}$. In the architecture, two different results are computed in parallel and the correct one is selected by using $Sgn(\epsilon')$. The architecture has been mapped on a XILINX-V1000-6 FPGA. The number of used Configurable Logic Blocks (CLB) is 80 and the maximum delay is 14 ns (taking into account the routing delays).

6. FILTERS IMPLEMENTATION ON FPGAS

In order to evaluate the performance of RNS and CS-RNS filters, we have implemented error-free FIR filters (20 bits dynamic range) in the traditional two's complement system (TCS), in RNS and in RNS using the carry-save scheme (CS-RNS).

For the traditional TCS filter we opted for a carry-save representation in the taps to keep the cycle time as short as possible. The product $p_k = a_k x(n-k)$ is realized with a Booth multiplier [13] and the resulting partial products are accumulated in a Wallace tree structure which produces a carry-save representation of the product. Because the sum at the $(k-1)$-th tap $Y_{k-1} = \sum_{i=0}^{k-1} a_i x(n-i)$ is stored in carry-save representation, an array of 4:2 compressors

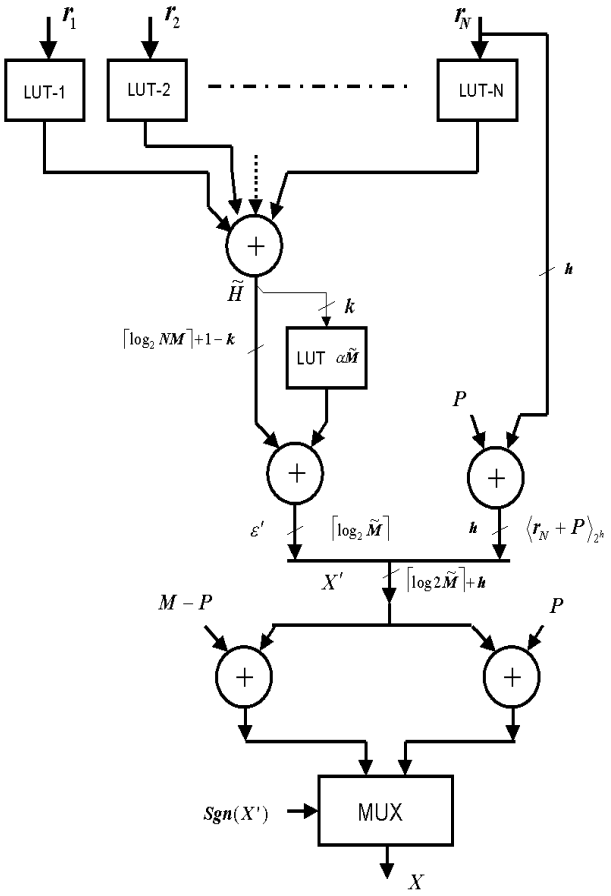

Figure 14. The converter architecture

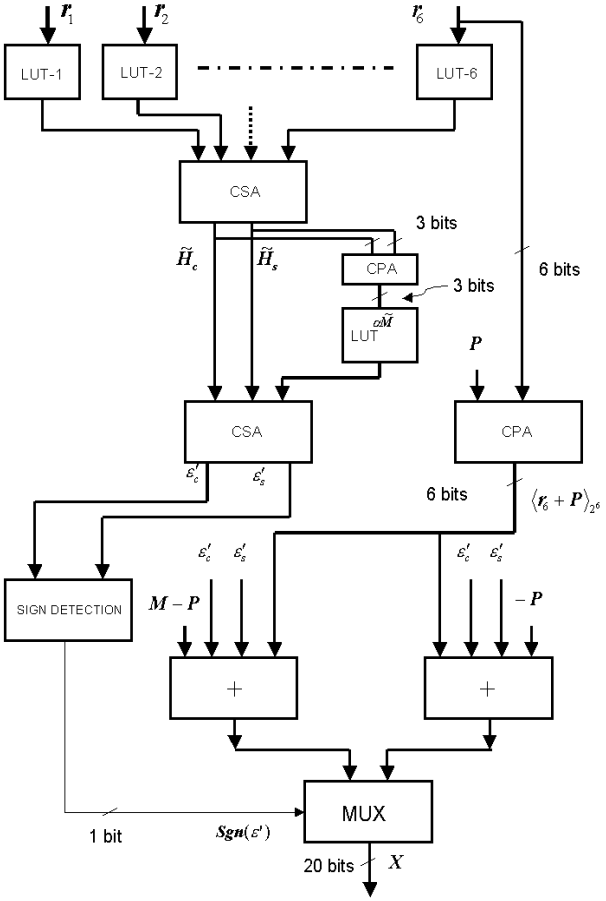

Figure 15. The implemented architecture

[14] is required to reduce the CS representation of p_k and the CS representation of Y_{k-1} to the CS representation of $Y_k = Y_{k-1} + p_k$ in the k-th tap.

The carry-save representation is finally converted into the two's complement representation by a carry-propagate adder (realized with a carry-look-ahead scheme) in the last stage of the filter.

The critical path is:

$$t_{TCS} = t_{xbuf} + t_{MULT} + t_{CSA-4:2} + t_{REG} \ .$$

We implemented six different filters: 8-tap and 16-tap TCS, 8-tap and 16-tap RNS and 8-tap and 16-tap CS-RNS, all with dynamic range of 20 bits. The VHDL RT-level description of the filters was synthesized and mapped on a FPGA by using the Xilinx Foundation suite of tools. By measuring the average current consumption $\overline{I}$, we computed the average power dissipation from

$$\overline{P} = V_{DD} \cdot \overline{I}$$

in which V_{DD} is the FPGA core voltage supply.

Table 1 shows the values of average power dissipation and area occupation of the different circuits implemented. To improve the accuracy of the measurement, we averaged the values obtained for different clock frequencies ($f = 1/T_c$) by converting average power dissipation into energy consumed in a cycle:

$$E_c = \overline{P} \cdot T_c \qquad [nJ] \ .$$

Values of E_c, and their average are also reported in Table 1. Then, by fitting the experimental points in a curve (Figure 16)

$$E_c(N) = E_1 \cdot N + E_0 \ , \tag{18}$$

we find an expression of the energy dissipated in filters of a given structure with any number of taps (Table 2). To better evaluate the term E_0, we implemented a circuit with just the two converters (binary-RNS and RNS-binary), connected in cascade, for the RNS and CS-RNS filter. We obtained an average value of $E_c(0) = 5.8\ nJ$

Table 2. Expressions of E_c for the filters.

	$E_c \quad [nJ]$	N^*
TCS	$2.5 \cdot N + 0.2$	-
RNS	$0.9 \cdot N + 5.6$	4
CS-RNS	$0.7 \cdot N + 5.7$	3

Table 1. Measurements of average power and E_c.

	TCS				RNS				CS-RNS			
	8-tap		16-tap		8-tap		16-tap		8-tap		16-tap	
T_c	$\overline{P}$	E_c	$\overline{P}$	E_c	$\overline{P}$	E_c	$\overline{P}$	E_c	$\overline{P}$	E_c	$\overline{P}$	E_c
$[ns]$	$[mW]$	$[nJ]$	$[mW]$	$[nJ]$	$[mW]$	$[nJ]$	$[mW]$	$[nJ]$	$[mW]$	$[nJ]$	$[mW]$	$[nJ]$
1,000	20.5	20.5	41.2	41.2	12.4	12.4	20.2	20.2	10.6	10.6	16.7	16.7
500	40.7	20.3	81.2	40.6	24.3	12.2	40.3	20.2	21.6	10.8	33.5	16.7
250	80.3	20.0	160.0	40.0	49.1	12.3	81.0	20.3	42.8	10.7	66.6	16.7
200	99.9	19.9	198.5	39.7	60.8	12.2	101.0	20.2	53.3	10.6	83.0	16.6
100	197.6	19.7	387.9	38.8	121.5	12.1	198.7	19.9	105.8	10.6	164.7	16.5
average		20.1		40.1		12.2		20.1		10.7		16.6
# slices	1240		2440		1364		2310		1358		2274	
(% area)	17%		35%		19%		33%		19%		32%	

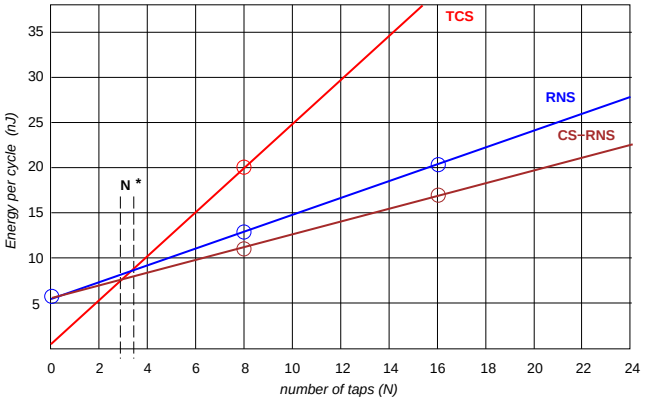

Figure 16. Curves of E_{TCS}, E_{RNS} and E_{CSRNS}.

From Figure 16 we can see that for filters with more than 4 taps ($N^* = 4$) the RNS filter consumes less power. The result obtained for CS-RNS filters is even more interesting: in carry-save RNS filters, the modular sum

$$s_k = \langle s_{k-1} + a_k x(n-k) \rangle_{m_i}$$

is not done in each tap, but s_k (kept in carry-save format) is reduced to modulo m_i every 8 taps. Because additional registers are required to keep a carry-save representation of s_k-s, there is a tradeoff between combinational logic (adders) and flip-flops. By eliminating modular adders we speed-up the operations and reduce the power dissipation. The power consumption in the extra flip-flops does not offset this reduction. Therefore, CS-RNS filters not only are faster than plain RNS (and TCS) filters, but also occupy less area and consume less power in a FPGA implementation.

7. Configurable Dynamic Range Filter

The RNS reconfigurable filter is depicted in Figure 17. The figure shows a control Unit which is used to configure the filter and the converters, according to a command string, to select the dynamic range and to load the filter coefficients. The filter includes the conversions binary/RNS and RNS/binary.

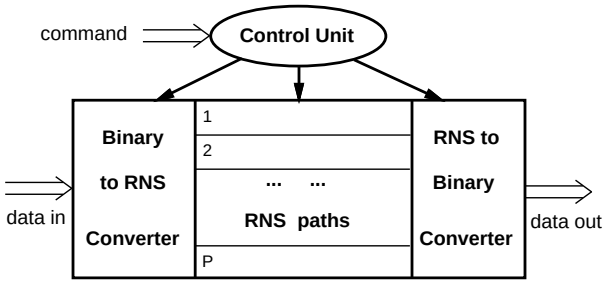

Figure 17. RNS reconfigurable filter.

Once a dynamic range $M_1 < M$ is chosen, and consequently, the number of bits required is smaller that the bit-width of the filter ($\log_2 M_1 < \log_2 M$) we can turn-off in the RNS filter the paths modulo m_i which are not needed to cover the dynamic M_1.

For example, by choosing the following set of moduli: $\{3, 5, 7, 11, 13, 17, 19\}$ a 22-bit dynamic range can be covered. If our system requires a dynamic range of 16 bits we can use the set of moduli: $\{3, 7, 11, 17, 19\}$ turning off the paths through the moduli 5 and 13. As a consequence, for dynamic ranges smaller than the maximum range the power dissipation is reduced.

We designed the configurable RNS filter to have a maximum dynamic range of 32 bits by choosing the following set of moduli:

$$\{13, 17, 19, 23, 29, 31, 64\}.$$

With these moduli, we have a granularity of four bits in scaling the dynamic range.

Each RNS path (Figure 18) is composed of 64 modular mul-

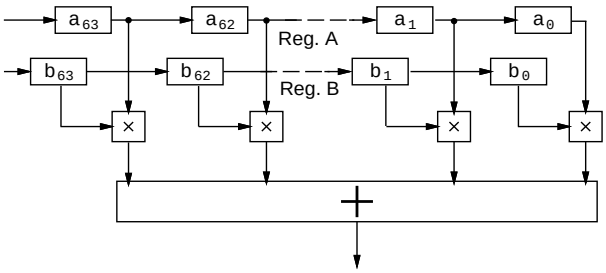

Figure 18. Architecture of a RNS path.

tipliers (isomorph multipliers except for $m_7 = 64$), a tree of adders to add 64 operands, and two $\lceil \log_2 m_i \rceil$-bit shift registers (Reg. A and B).

With this unit, we implement a programmable 64-tap FIR filter

$$y(n) = \sum_{k=0}^{63} a_k x(n-k)$$

realized in direct form. The coefficients a_k are loaded in register A in the first 64 cycles after reset, and the samples $x(n-k)$ are fed into register B one per clock cycle.

The dynamic range is adjusted to have an error free filter. The main feature of the filter is the reconfigurability in terms of dynamic range. The filter dynamic range is adapted to the processing specifications. The RNS paths not used to cover a given dynamic range are deactivated by disabling the clock of registers A and B in the path.

A command string, a sort of micro-instruction, is given to the controller to perform the following tasks:

- to set the dynamic range by selecting the required moduli paths in filter;
- to load the coefficients in shift-register A;
- to stop/start the data acquisition/output, if requested.

Because register A holds values which are constant for the whole processing, we can use one binary to RNS converter to load both register A (at start-up) and register B (runtime). The RNS to binary converter is programmable in terms of dynamic range as well: when some moduli are not used in the filter, the corresponding parts in the converter are disabled by switching off the clock in the registers at the interface between the RNS paths and the converter.

The implementation of the reconfigurable RNS filter has been realized with the AMS $0.35\mu m$ library of standard cells. In order to compare the performance of the reconfigurable RNS filter in terms of throughput, area and power dissipation, we implemented a programmable 64-tap FIR filter realized in the traditional two's complement system (TCS) with 32-bit dynamic range. Table 3 reports the characteristics of the TCS filter and the reconfigurable RNS filter. The table shows that

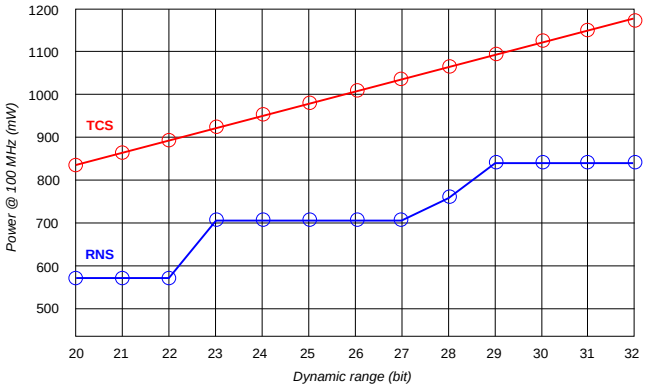

Figure 19. Power dissipation of TCS and RNS for different dynamic ranges.

the RNS processor is about 25% faster than the traditional FIR filter and consume less power (at the same frequency).

By reducing the dynamic range, we obtain for the power dissipation, computed at 100 MHz, the results shown in Figure 19. For the TCS filter the range is reduced simply by feeding data with shorter wordlength, while for the RNS filter, we also disabled, by turning off the clock, the paths through the moduli not necessary to cover that dynamic range. The figure shows that the TCS filter working at 20 bit dynamic range consumes almost the same energy of the RNS working at full dynamics. ¿From the RNS set of points (staircase shape), we can see that the power reductions are due to the different configurations of active moduli, and, once a moduli set is selected, the power dissipation is constant. For the TCS, the set of points is on a straight line because by reducing the input wordlength, the most-significant portion of the datapath get filled with sign-extension bits[2], which significantly reduce switching activity.

Table 3. Comparison of TCS and RNS.

	max. freq.	area (ND2 equiv.)	power @ 100 MHz
TCS	100 MHz	201,000	1,170 mW
RNS	125 MHz	177,000	840 mW

8. Conclusions

In this paper a RNS implementation of digital filters for satellite demultiplexing applications has been presented.

The RNS filter shows a lower power dissipation with respect to filters realized in the traditional binary representation. Furthermore, the reduced circuit complexity allows the implementation of coefficient programmable structures, a very appealing feature for satellite applications.

[2]Radix-4 Booth recoding in multipliers transforms a sequence of 1s (negative numbers) in a sequence of 0s.

9. ACKNOWLEDGMENTS

The research was supported in part by the PRIN grant of Italian Education Ministry.

REFERENCES

[1] N.S. Szabo and R.I. Tanaka, *Residue Arithmetic and its Applications in Computer Technology*, New York: McGraw-Hill, 1967.

[2] M.A. Sodestrand, W.K. Jenkins, G. A. Jullien, and F. J. Taylor, *Residue Number System Arithmetic: Modern Applications in Digital Signal Processing*, New York: IEEE Press, 1986.

[3] P.E. Pace, P. A. Ramamoorthy, and D. Styer, "A Preprocessing Architecture for Resolution Enhancement in High-Speed Analog-to-Digital Converters," *IEEE Trans. On Circuits and Systems*, vol. II, pp. 373–379, 1994.

[4] A. Nannarelli, M. Re, A. Del Re, GC. Cardarilli, and R. Lojacono, "High Speed RNS AD Front End," *Proceedings of 6th European Workshop on ADC Modelling and Testing*, pp. 19–22, September 2001.

[5] G. C. Cardarilli, M. Re, R. Lojacono, and G. Ferri, "A New Efficient Architecture for Binary to RNS Conversion," *European Conference on Circuit Theory and Design ECCTD'99*, vol. II, pp. 1151–1154, September 1999.

[6] M. Bhardwaj and A. Balaram, "Low power signal processing architectures using residue arithmetic," *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ASSP'98)*, vol. 5, pp. 3017–3020, 1998.

[7] W.L. Freking and K.K. Parhi, "Low-power digital filters using residue arithmetic," *Thirty-First Asilomar Conference on Signals, Systems and Computers*, vol. 1, pp. 739–743, 1998.

[8] A. Del Re, A. Nannarelli, and M. Re, "Implementation of Digital Filters in Carry-Save Residue Number System," *Proc. of 35th Asilomar Conference on Signals, Systems, and Computers*, pp. 1309–1313, Nov. 2001.

[9] I.M. Vinogradov, *An Introduction to the Theory of Numbers*, New York: Pergamon Press, 1955.

[10] K.M. Elleyth and M.A. Bayoumi, "Fast and Flexible Architectures for RNS Arithmetic Decoding," *IEEE Trans.Circuits Systems.-II Analog and Digital Signal Processing*, vol. 39, pp. 226–235, April 1992.

[11] F. Barsi, "Mod m Arithmetic in Binary Systems," *Information Processing Letters*, vol. 40, pp. 303–309, December 1991.

[12] G. Cardarilli, M. Re, and R. Lojacono, "A residue to binary conversion algorithm for signed numbers," *European Conference on Circuit Theory and Design (ECCTD'97)*, vol. 3, pp. 1456–1459, 1997.

[13] Israel Koren, *Computer Arithmetic Algorithms*, Prentice-Hall, Inc. , 1993.

[14] M.D. Ercegovac and T. Lang, *Division and Square Root: Digit-Recurrence Algorithms and Implementations*, Kluwer Academic Publisher, 1994.

Gian Carlo Cardarilli *was born in Rome, Italy, in 1956 and received the laurea (summa cum laude) in 1981 from the University of Rome "La Sapienza". He works for the University of Rome "Tor Vergata" since 1984 where he is currently professor of Digital Electronics. From 1992 to 1994 he worked for the University of L'Aquila. During the years 1987/1988 he worked for the Circuits and Systems team at EPFL of Lausanne (Switzerland). He is a regular consultant for Alenia Aerospazio, Rome, Italy. Scientific interest of Prof. Cardarilli concerns the design of special architectures for signal processing with emphasis in residue arithmetic. His interests also include mixed-signal neural network architectures and high-speed fuzzy-logic processors, oriented to the signal and image processing. In these fields he published more than 80 papers in international journals and conferences.*

Andrea Del Re *received the Laurea degree in Electronic Engineering from the University of Rome "Tor Vergata" in May 1999. Since Nov. 1999 he has been a PhD student at the Department of Electronic Engineering, University of Rome "Tor Vergata". In July 1999 he was awarded one year fellowship with the Coritel consortium. His main interests and activities are in the area of Fast Prototyping, hardware for DSP, Filter Banks and Residue Number System.*

Roberto Lojacono *was born in Palermo (Italy) in 1941. He received the Degree in Physics at University of Catania (Italy) in 1966. Until 1983 he had been a teacher in Italian High School and, from 1976 also a Lecturer in the Engineering Faculty of the University of Rome "La Sapienza" (Italy). He became associate professor and full professor in 1983 and 1990 respectively. He is presently with the Engineering Faculty of the University of Rome "Tor Vergata". He was invited professor at EPFL - Lausanne - Switzerland in 1984 and 1988. He is author or coauthor of about one hundred technical papers and several books. He has developed professional activities with Alenia Spazio, Foreign Office of Italy and some minor institutions and companies. His research activity presently concerns nonlinear circuit analysis, fuzzy logic, finite arithmetic, signal measurements, signal processing and related VLSI design.*

Alberto Nannarelli *graduated in electrical engineering from the University of Roma, Italy, in 1988 and received the M.S. and the Ph.D. in electrical and computer engineering from the University of California at Irvine in 1995 and 1999, respectively. He is currently with the Department of Electrical Engineering at University of Rome, Italy. He previously worked for SGS-Thomson Microelectronics, Ericsson Telecom, and for Rockwell Semiconductor Systems as a summer intern. His research interests include computer arithmetic,*

computer architecture, and VLSI design.

Marco Re *received the Laurea degree in Electronic Engineering from the University of Rome "La Sapienza" in 1991 and the Ph.D. in Microelectronics and Telecommunications Engineering from the University of Rome "Tor Vergata" in 1996. In 1998 he joined the Department of Electronic Engineering of the University of Rome "Tor Vergata" as Researcher. He was awarded two one-year NATO fellowships with the University of California at Berkeley in 1997 and 1998. At present, he is Visiting Researcher at the Cadence Berkeley Laboratories and at the University of California at Berkeley. His main interests and activities are in the area of DSP algorithms, fast DSP architectures, Fuzzy Logic hardware architectures, Hardware-Software Codesign, number theory with particular emphasis on Residue Number System, computer arithmetic and Cad tools for DSP.*