HIGH SPEED RNS A/D FRONT END

A Nannarelli, M. Re, A. Del Re, G. C. Cardarilli, R. Lojacono

Electronics Engineering Department – University of Rome "Tor Vergata", Rome, 00133 Italy
Phone +39 06 72597318 Fax +39 06 2020519 e-mail: lojacono@ing.uniroma2.it

Abstract - *The analog to digital front-end of high speed acquisition systems requires fast A/D conversion and fast filtering. These performances are difficult to obtain if the acquisition calls for large wordlength. Great design and implementation efforts are needed in order to match the required throughput. The Residue Number System (RNS) appears to be very attractive, but it requires an expensive conversion from binary to RNS representation. A possible solution is the direct conversion from analog to RNS. In this paper a Residue Number System approach to the realization of a high speed front-end for large wordlength is proposed. We study a possible RNS realization of the filter following the analog to RNS conversion stage. The filter is designed to have high-throughput and low power dissipation, and its timing parameters should set the design constraints in terms of architecture, technology and speed for the converter.*

Keywords – RNS, Analog to RNS Conversion.

1. INTRODUCTION

There are many applications requiring high-speed analog front-end processing (instrumentation input subsystems, telecom receivers, ...). These applications usually require an analog to digital conversion and a preprocessing based on digital filtering. The high processing speed can be obtained using the RNS representation for the signals. The advantages of the RNS representation of integer numbers are well known [1, 2]: this system is carry-free and allows the possibility to decompose the calculations in parallel flows of lower complexity. However, these advantages are significantly reduced by the need of a conversion from the classical binary representation to the RNS and by the inverse conversion from RNS to binary [3]. The complexity of binary to RNS conversion suggests a direct conversion from analog to RNS representation (ARNS in the following). Due to the non positional representation, the conversion is very difficult to implement and the limited conversion precision can produce large errors. To avoid these errors, specific procedures for the conversion have been proposed.

These methods are essentially based on a particular type of folding and on a redundancy of moduli in order to control the correctness of the results. The authors in [4] proposed an alternative method for implementing mixed analog/digital architecture.

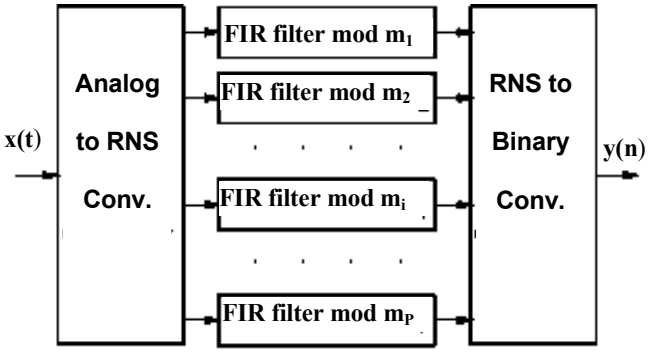

Fig. 1 – Architecture of A/D front end.

This new method realizes a self-correcting ARNS converter and does not require moduli redundancy. In this paper we study a possible RNS realization of the filter following the conversion stage. The filter is designed to have high-throughput and low power dissipation, and its timing parameters should set the design constraints in terms of architecture, technology used and speed of the ARNS converter.

2. THE ANALOG TO RNS CONVERTER

A RNS is based on a set of N relative prime moduli m_i, i=1,…,N and can represent univocally $M = \prod m_i$ different numbers. The representation is obtained by N residual quantities r_i, i= 1,…,N ; i.e. an integer X < M is represented by

$$r_i = \langle X \rangle_{mi} = \langle \alpha_i m_i + r_i \rangle_{mi} , \; i = 1,\ldots,N.$$

We suppose that all the moduli m_i are odd. In this case, from the right side of (1), we can see that:

i – $\alpha_i m_i$, being m_i an odd number, has the parity of α_i;

ii – if we consider the quantities X, α_i, r_i in binary form $LSB[LSB(\alpha_i) + LSB(r_i)] = p_i$, i = 1,…,N, all p_i , in a correct conversion, must be equal to LSB (X).

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 parities p_j , j =1,…,L, for which we have that either α_j or r_j are wrong (if α_j and r_j are together wrong, the parity p_j is

ISBN 972-98115-5-5

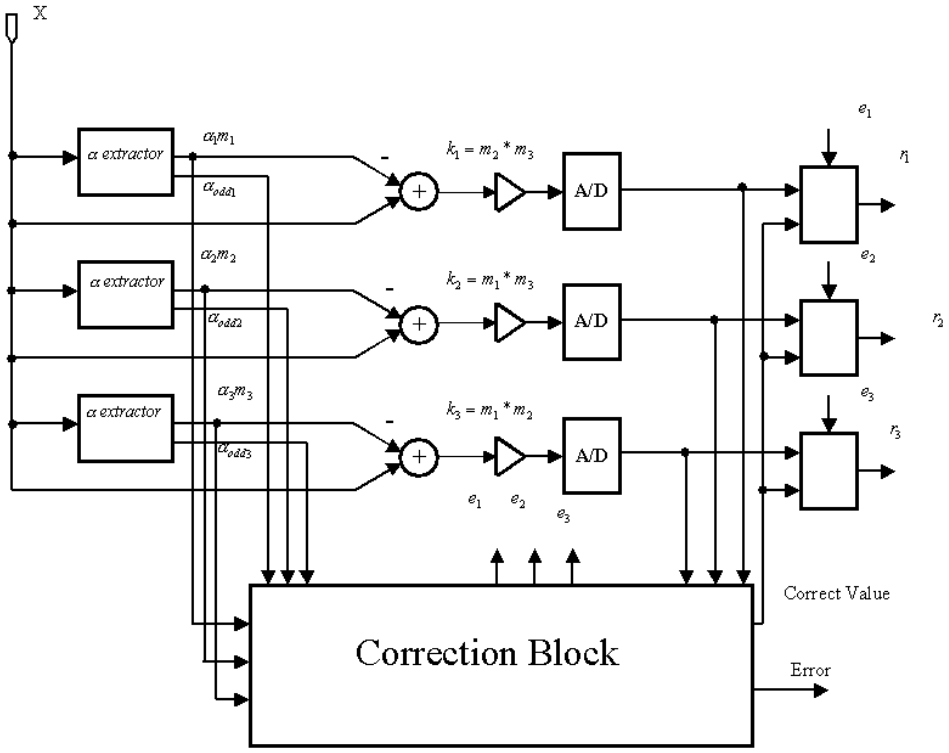

Fig. 2 – Analog to RNS converter architecture

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 result changed in p^*.

If X is not near to a multiple of m_j, α_j is right, but r_j is wrong and must be changed in either r_j+1 or r_j-1. Unfortunately, we do not have sufficient information to resolve this ambiguity, but, by introducing some additional hardware, we can perform a binary conversion of the difference $\alpha_i m_i - \alpha_j m_j = r_j - r_i = r_{ij}$ (which is positive if $m_i<m_j$) and compute the right residue as $r_j = r_{ij} + r_i$.

The proposed architecture for the analog to RNS converter is shown in Fig. 2. The system is divided into N parallel units, one for each modulo, plus a correction block. Each unit is based on the cascading of following blocks:

1. The α extractor for the determination of the level that must be subtracted.
2. A differential amplifiers, for subtracting the input voltage from the level obtained from the α extractor.
3. An ADC with reduced dynamic range, m_i–bit converter.
4. A final blocks for the correction of the obtained residue. The correction is done by adding the values coming from the correction block.

3. RNS FIR FILTER

The starting point of our design is a programmable N taps FIR filter

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

realized in transposed form (Fig. 3).

The RNS implementation of the FIR filter is shown in Fig. 1. The FIR filter is decomposed into P filters working in parallel. In each tap, a modular multiplier is needed to compute the term $\langle a_k\, x(n-k)\rangle_{mi.}$. Because of the complexity of modular multiplication, we used the isomorphism technique [5] to implement the product of residues. 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 [5], if m is prime there exists a primitive radix q such that its powers modulo m cover the set [1, m-1]:

$$n_i = \langle q^{w}{}_i \rangle_m \qquad \text{with } n_i \in [1, m-1]$$
$$w_i \in [0, m-2].$$

Both transformations $n \rightarrow w$ and $w \rightarrow n$ can be implemented with $m \rightarrow 1$ entries tables. Therefore, the product of a_1 and a_2 modulo m can be obtained as:

$$\langle a_1 \cdot a_2 \rangle_m = \langle q^{w} \rangle_m$$

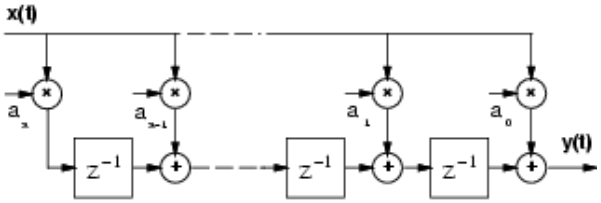

Fig. 3 – FIR filter in transposed form

where

$$w = \langle w_1 + w_2 \rangle_{m-1} \qquad \text{with } a_1 = \langle q^{w_1} \rangle_m$$

$$a_2 = \langle q^{w_2} \rangle_m$$

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

1. Two direct isomorphic transformations (DIT) to obtain w_1 and w_2;
2. One modulo m-1 addition $\langle w_1 + w_2 \rangle_{m-1}$;
3. One inverse isomorphic transformations (IIT) to obtain the product.

Because of the transposed form of the FIR filter, the input x is the multiplicand of all the multipliers (see Fig. 3). For this reason only one direct isomorphic transformation, incorporated in the binary to RNS conversion, is necessary for all the taps.

On the other hand, because the coefficients of the filter are constant terms loaded once at start-up, it is convenient to load directly the isomorphic representation modulo m_i-1.

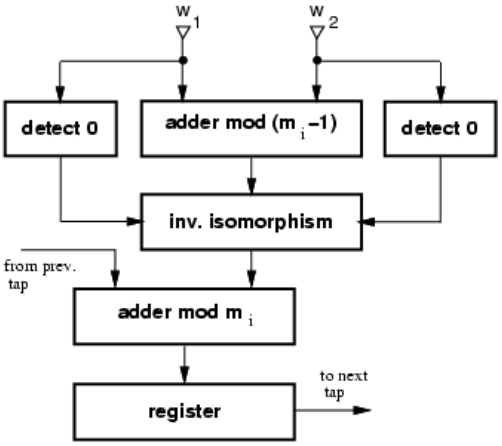

Fig. 4 - Structure of one modulo RNS tap.

As a result, in each tap, we reduce the modular multiplication to a modular addition followed by an access to table (inverse isomorphism) as depicted in Fig. 4. The table is implemented as synthesized 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 modular addition $\langle a_1 + a_2 \rangle_m$, consists of two binary additions. If the result of $a_1 + a_2$ exceeds the modulo (it is larger than m–1), we have to subtract the modulo m. In order to speed-up the operation we can execute in parallel the two operations:

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

If the sign of the three-term addition is negative it means than the sum $(a_1 + a_2) < m$ and the modular sum is $a_1 + a_2$, otherwise the modular addition is the result of the three-term addition. The above algorithm can be implemented with two $(\lceil \log_2 m \rceil + 1)$-bit adders as shown in Fig. 5.

In order to evaluate the performance of filters realized with the RNS, we have implemented a FIR filter in the traditional two's complement system.

As implementation benchmark, we chose a programmable 64 taps error-free FIR filter realized in transposed form (Fig. 3) with input and coefficients word-length of 10 bits, and an internal dynamic range of 20 bits.

The filters were implemented in a 0.35 μm library of Standard Cells. Delay, area and power dissipation have been determined with Synopsys tools. Although interconnections are not taken into account, local and global routing for the RNS is expected to be no worse than for the traditional filter.

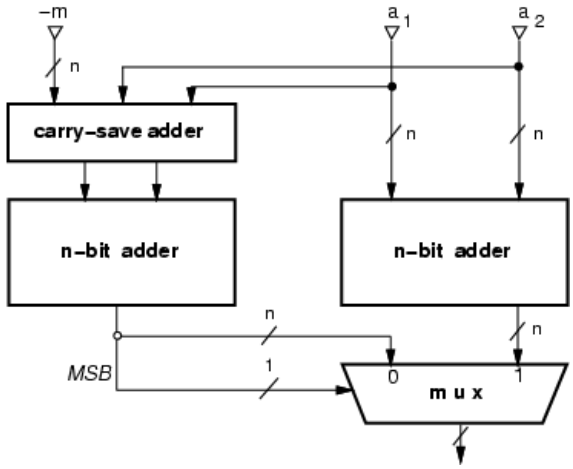

Fig. 5 – Structure of modular adder.

The results are shown in Table I. In the table, area is reported as number of NAND2 equivalent gates, and power is computed at 100 MHz.

The table shows that the RNS filter is faster than the traditional one, dissipates less power and is significantly smaller.

Table I – Summary of results for implemented filters.

Filter	Cycle (ns)	Area (NAND2)	Power (mW)
Traditional	5.0	90,000	1,000
RNS	3.5	62,000	520

4. CONCLUSIONS

In this work, we studied a possible RNS realization of an A/D front end. The front end consists of an analog to RNS converter followed by a FIR filter. The filter has been designed to achieve high throughput and low power dissipation. We showed that the RNS filter is about 30% faster than the filter realized in the conventional two's complement representation. Therefore, the RNS approach seems to be convenient to realize a high speed A/D front end. The timing parameters found for the filters should be used as design constraints to choose the architecture and the technology of the analog to RNS converter.

REFERENCES

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

[2] N. Szabo and R. I. Tanaka, *Residue Arithmetic and its application in computer technology,* McGraw-Hill, 1967.

[3] G.C. Cardarilli, M. Re, 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, 31 August - 3 September, Budapest, Hungary.

[4] R. Lojacono, G. C. Cardarilli and M. Re, *A self correcting analog to RNS converter*, 6th EuroWorkshop on ADC modelling and testing, Wien, Austria, September 2000.

[5] I. N. Vinogradov, *An introduction to the theory of numbers,* Pergamon Press, 1995.