# Reducing Power Dissipation in FIR Filters using the Residue Number System

Gian Carlo Cardarilli, Alberto Nannarelli and Marco Re
Department of Electrical Engineering
University of Rome "Tor Vergata"
Rome, 00133 ITALY

*Abstract*—**The aim of this work is to reduce the power dissipated in high order Finite Impulse Response (FIR) filters, while maintaining the delay unchanged. We compare in terms of performance, area, and power dissipation the implementation of a traditional FIR filter with a Residue Number System (RNS) based one. The resulting implementations, designed to work at the same clock rate, show that the RNS filter is smaller and consumes less power than the traditional one for a number of taps larger than eight.**

## I. Introduction

The new generation of telecommunication equipment often require the use of high order FIR filters for the implementation of the new modulation schemes. Moreover, the telecommunication market demands for speed and low power consumption for the new portable multimedia terminals. In this context, computational intensive signal processing blocks can be effectively implemented by using Residue Number System (RNS) arithmetic.

The use of the RNS allows the decomposition of a given dynamic range in slices of smaller range on which the computation can be efficiently implemented in parallel [1], [2], [3]. The typical drawback presented by the RNS is related to the input-output conversion from binary to RNS and vice versa. This problem is solved by using new efficient conversion techniques [4], [5] or by converting directly the analog signal in the residue representation and vice versa [6]. Recently, a number of works on low power and RNS have been presented. In [7] and [8] the power dissipation is reduced by taking advantage of the speed-up due to the parallelism of the RNS structure. The supply voltage is reduced, resulting in a quadratic reduction of power, until the speed-up = 1 [7] or until the desired value of delay [8]. In [9] some encoding optimization techniques for small moduli (3 and 7) are presented.

In our work, we compare the performance, area and power of a FIR filter realized with the traditional binary arithmetic, with a RNS based one. Although the RNS filter has the same performance of the traditional one, its area and power dissipation are smaller for filters with more than eight taps. Furthermore, we reduce power dissipation, without sacrificing performance, by equalizing the parallel paths of the RNS filter with a dual voltage approach [10].

This work was partially supported by MURST National Project: Codesign Methods for Low Power Integrated Circuits.

## II. Traditional 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 (Figure 1) with input and coefficients size of 10 bits. The product is realized with a Booth multiplier [11] and the five resulting partial products are accumulated in a Wallace's tree structure which produces a carry-save (CS) representation of the product. In order to keep the cycle time as short as possible, the sum at the $k$-th tap is stored in carry-save representation as well. Therefore, an array of 4:2 compressors [12] is required to reduce the CS representation of $a_k x(n-k)$ and the CS representation of $\sum_{i=0}^{k-1} a_i x(n-i)$ to the CS representation of $\sum_{i=0}^{k} a_i x(n-i)$ in the $k$-th tap (Figure 2).

To have an error-free filter we must keep a number of bits sufficient to hold the carry-save representation of the sum at the $k$-th tap. In the worst case, we need to store $2 \cdot (10 \cdot 10 + \log_2 N)$ bits per tap. The CS representation is finally converted into two's complement representation by a carry-propagate adder (realized with a carry-look-ahead scheme) in the last stage of the filter.

## III. RNS FIR Filter

The RNS implementation of the FIR filter is shown in Figure 3. The FIR filter is decomposed into $P$ filters working in parallel, where $P$ is the number of moduli used in the RNS representation. In addition, the RNS filter requires both binary

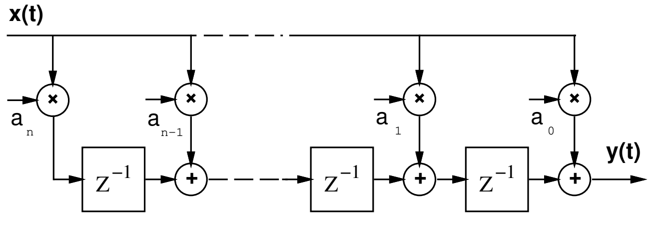


Fig. 1. FIR filter in transposed form.

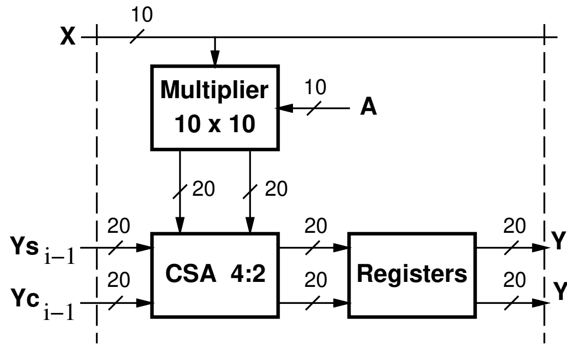


Fig. 2. Tap structure for the filter implementation.

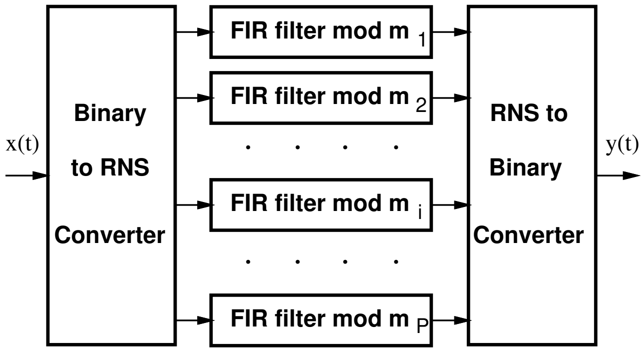


Fig. 3. RNS FIR filter.

to RNS and RNS to binary converters.

In order to have a dynamic range of 20 bits, as in the case of the traditional implementation, we chose the following set of moduli:

$$m_i = \{3, 5, 7, 11, 17, 64\}$$

such that

$$\log_2(3 \cdot 5 \cdot 7 \cdot 11 \cdot 17 \cdot 64) \geq 20.$$

### *A. Implementation of modular multiplication*

In each tap, a modular multiplier is needed to compute the term $\langle a_k x(n-k)\rangle_{m_i}$. Because of the complexity of modular multiplication, we used the isomorphism technique [13] to implement the product of residues in all the moduli but 3 (multiplication can be easily computed in tabular way) and 64 (modular product corresponds to the 6 least-significant bits of the conventional product). By using isomorphism, the product of the two residues is transformed into the sum of their indices which are obtained by an isomorphic transformation. According to [13], if $m$ is prime there exists a primitive radix $q$ such that its powers modulo $m$ cover the set $[1, m-1]$:

$$n_i = \langle q^{w_i}\rangle_m \qquad \text{with} \quad \begin{array}{l} n_i \in [1, m-1] \\ w_i \in [0, m-2]. \end{array}$$

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

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

where

$$w = \langle w_1 + w_2\rangle_{m-1} \qquad \text{with} \quad \begin{array}{l} a_1 = \langle q^{w_1}\rangle_m \\ a_2 = \langle q^{w_2}\rangle_m \end{array}$$

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

i) Two isomorphic transformations to obtain $w_1$ and $w_2$;

ii) One modulo $m-1$ addition $\langle w_1 + w_2\rangle_{m-1}$;

iii) One inverse isomorphic transformations to obtain the product.

For example, for the modular multiplication

$$\langle 3 \cdot 4\rangle_5 = 2$$

we have ($q = 2$):

$$\begin{array}{rl} i) & 3 = \langle 2^3\rangle_5 \to w_1 = 3 \\ & 4 = \langle 2^2\rangle_5 \to w_2 = 2 \\ ii) & \langle 2+3\rangle_4 = 1 \\ iii) & \langle 2^1\rangle_5 = 2 \end{array}$$

Because of the transposed form of the FIR filter, the input $x$ is the multiplicand of all the multiplications (see Figure 1). For this reason only one 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 (multiplicators) are constant terms loaded once at start-up, it is convenient to load directly the isomorphic representation modulo $m_i - 1$. 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 Figure 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.

### *B. 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 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 $\lceil \log_2 m \rceil$-bit adders as shown in Figure 5.

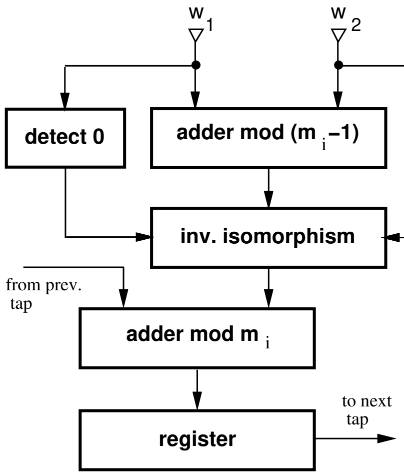


Fig. 4. Structure of one modulo RNS tap.

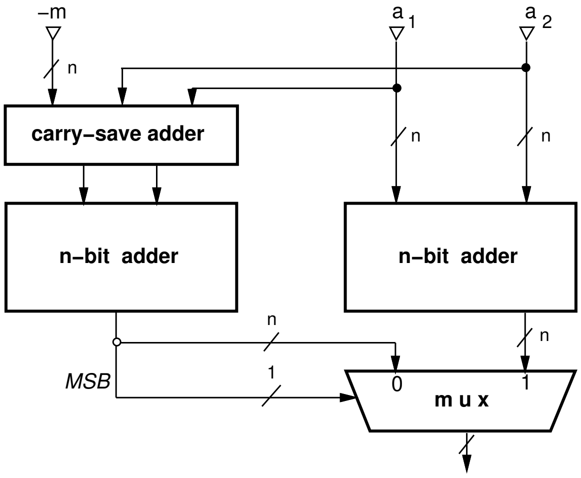


Fig. 5. Structure of modular adder.

### C. Implementation of input/output conversions

As already mentioned, the input conversion block includes the isomorphic transformation. If $x$ is zero, there is no exponent $w$ such that $\langle q^w \rangle_{m_i} = 0$. As a consequence, zero is encoded with a special pattern that is then detected in the block which computes the product using the isomorphism.

The output conversion is implemented by using the Chinese Remainder Theorem, as described in [4].

## IV. Results and Comparisons

Both the traditional and the RNS filters were implemented in a $0.35\mu m$ library of Standard Cells. Delay, area and power dissipation have been determined with Synopsys tools.

Table I summarizes the results. In the table, area is reported as number of NAND2 equivalent gates and power is computed at 100 MHz. However, both area and power dissipation do not take into account the contribution of interconnections.

TABLE I
Summary of Results

| Filter | Cycle $[ns]$ | Latency (cycles) | Area (gate equiv.) | Power $[mW]$ |
|---|---|---|---|---|
| Trad. | 5.0 | $N+1$ | $230+1250N$ | $14.9+13.5N$ |
| RNS | 5.0 | $N+3$ | $2910+745N$ | $51.0+6.9N$ |

Table I shows that the RNS filter has a higher latency, due to the conversions, but it can be clocked at the same rate of the traditional filter, and consequently, it can sustain the same throughput. The table also shows that the conversion overhead of the RNS filter is soon offset by the smaller area/power per tap, as the number of the taps increases.

More details on the implementation of the different paths corresponding to the moduli $m_i$ for the RNS filter are presented in Table II.

Because the single tap delay of moduli 3, 5 and 64 is less than $3.5\,ns$, and in order to obtain a larger reduction in power dissipation, we can use the available time slack ($5.0 - 3.5 = 1.5ns$) to lower the supply voltage for these moduli. For this specific case, a time slack of $1.5\,ns$, which corresponds to 30% of the cycle time, allows a reduction of the supply voltage from $3.3\,V$ to $2.7\,V$ without affecting the overall performance. We estimated the new power expression of the RNS filter to be

$$P_{(dual-volt.)} = 51.0 + 6.0N \qquad (1)$$

The use of a dual voltage requires level-shifting circuitry when going from the lower voltage to the higher one [14]. However in our case, voltage level shifters are only required for a few bits in the output conversion stage.

The expressions for area and power of Table I and Expr. (1) are plotted in Figure 6 and Figure 7. From the figures it is clear that for a filter with a number of taps larger than eight the RNS implementation of the filter outperforms the traditional one.

## V. Conclusions

We have implemented a RNS FIR filter and compared its delay, area and power with those of a traditional FIR filter in transposed form. The results obtained show that the RNS filter can sustain the same clock rate, although it has a slightly

TABLE II
Delay, area and power dissipation per tap in RNS FIR

| Modulus | Delay $[ns]$ | Area (gate equiv.) | Power $[mW]$ |
|---|---|---|---|
| 3 | 1.7 | 37 | 0.25 |
| 5 | 3.4 | 77 | 0.62 |
| 7 | 3.8 | 93 | 0.84 |
| 11 | 5.0 | 175 | 1.75 |
| 17 | 5.0 | 175 | 1.76 |
| 64 | 3.2 | 188 | 1.66 |

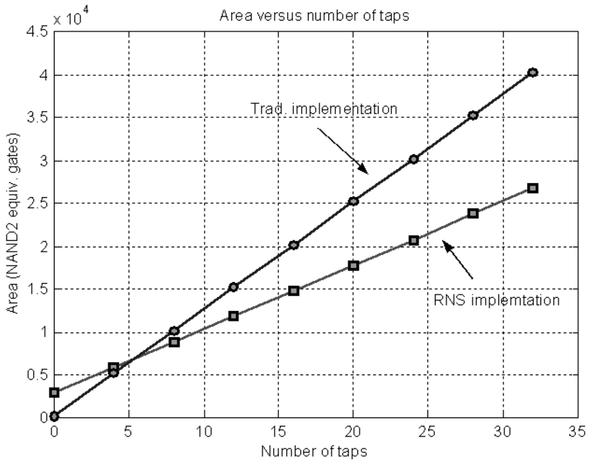


Fig. 6. Area versus number of taps.

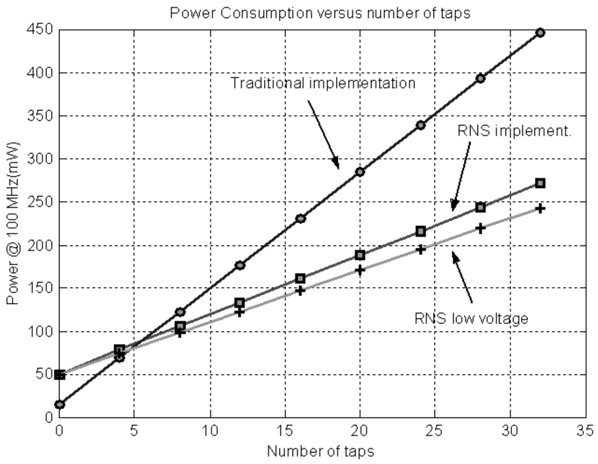


Fig. 7. Power dissipation versus number of taps.

longer latency. However, in terms of area and power the RNS version is more convenient for filters with more than eight taps. An additional reduction of about 15% is possible by using dual voltage.

## 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] M.A. Soderstrand and K.Al Marayati, "Vlsi implementation of very high-order fir filters," *IEEE International Symposium on Circuits and Systems (ISCAS'95)*, vol. Vol. 2, pp. 1436–1439, 1995.

[4] 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. Vol. 3, pp. 1456–1459, 1997.

[5] G. Cardarilli, M. Re, R. Lojacono, and G. Ferri, "A new efficient architecture for binary to rns conversion," *Proc. of European Conference on Circuit Theory and Design (ECCTD'99)*, vol. Vol. 2, pp. 1151–1154, 1999.

[6] A.P. Preethy and D. Radhakrishnan, "A vlsi architecture for analog-to-residue conversion," *Third International Conference on Advanced A/D and D/A Conversion Techniques and Their Applications*, pp. 83–85, 1999.

[7] 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. Vol. 5, pp. 3017–3020, 1998.

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

[9] M.N. Mahesh and M. Mehndale, "Low power realization of residue number system based fir filters," *Thirteenth International Conference on VLSI Design*, pp. 30–33, 2000.

[10] A. Nannarelli and T. Lang, "Low-power division: Comparison among implementations of radix 4, 8 and 16," *Proc. of 14th Symposium on Computer Arithmetic*, pp. 60–67, 1999.

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

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

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

[14] K. Usami and M. Horowitz, "Clustered voltage scaling technique for low-power design," *Proc. of International Symposium on Low Power Design*, pp. 3–8, Apr. 1995.