

Totally Fault Tolerant RNS based FIR Filters

S. Pontarelli^{†‡}, G.C. Cardarilli[†], M. Re[†], A. Salsano[†]

{pontarelli, salsano}@ing.uniroma2.it, {marco.re, g.cardarilli}@ieee.org

[†]University of Rome "Tor Vergata", Via del Politecnico 1, 00191, Rome, ITALY

[‡](ASI) Italian Space Agency, Viale Liegi, 26, 00198 Rome, ITALY

Abstract—In this paper, the design of a Finite Impulse Response (FIR) filter with fault tolerant capabilities based on the residue number system is analyzed. Differently from other approaches that use RNS, the filter implementation is fault tolerant not only with respect to a fault inside the RNS moduli, but also in the reverse converter. An architecture allowing fault masking in the overall RNS FIR filter is presented. It avoids the use of a trivial Triple Modular Redundancy (TMR) to protect the blocks that performs the final stages of the RNS based FIR computation.

I. INTRODUCTION AND CONCLUSIONS

Fault detection and fault masking in the RNS representation have been studied, such as, for example in [1], [2], [3], [4]. These works are based on the so called Redundant Residue Number System (RRNS) representation in which some additional moduli are used to detect and correct errors in one of the elements of the RNS processor. The techniques proposed to correct errors in the RRNS representation do not guarantees that a fault inside the reverse conversion block or inside the error detection and correction block will be detected and/or corrected. In this paper, to mask the effect of a fault inside these blocks a method based on a particular voter is proposed. This voter performs both the correction of an error inside a modulo of the RRNS and of an error inside the reverse converter. The use of this voter allows the implementation of a totally fault tolerant RNS based FIR filter saving more than 33% of resource with respect to a TMR implementation of the blocks performing RNS based error correction. The paper is organized as follows: in Section II a background on RNS and RRNS arithmetic is given. In Section III, the method to obtain fault masking in the blocks performing reverse conversion and error detection and correction is presented, while in Section IV, the area overhead of the proposed technique is presented and compared with a trivial TMR approach.

II. RNS BACKGROUND

A Residue Number System (RNS) is defined by a set of relatively prime integers $\{m_1, m_2, \dots, m_P\}$.

The dynamic range of the system M is given by the product of the moduli m_i , i.e. $M = \prod_{i=1}^P m_i$.

Any integer $X \in [0, M - 1]$ is represented in RNS as:

$$X \xrightarrow{RNS} (\langle X \rangle_{m_1}, \langle X \rangle_{m_2}, \dots, \langle X \rangle_{m_P}) \quad (1)$$

where $\langle X \rangle_{m_i} = X \bmod m_i$

In RNS, operations such as addition and multiplication, are executed in parallel as shown in the following formula:

$$Z = X \text{ op } Y \xrightarrow{RNS} \begin{cases} \langle Z \rangle_{m_1} = \langle X_{m_1} \text{ op } Y_{m_1} \rangle_{m_1} \\ \dots \\ \langle Z \rangle_{m_P} = \langle X_{m_P} \text{ op } Y_{m_P} \rangle_{m_P} \end{cases} \quad (2)$$

Therefore, operations on large word lengths are split into several modular operations with reduced word-length.

The conversion of the RNS representation of Z is accomplished by using the Chinese Remainder Theorem (CRT)

$$Z = CRT(Z_{m_1}, \dots, Z_{m_P}) = \left\langle \sum_{i=1}^P Z_{m_i} \cdot k_i \cdot M_i \right\rangle_M \quad (3)$$

where $M_i = \frac{M}{m_i}$ and k_i are obtained by $\langle M_i \cdot k_i \rangle_{m_i} = 1$.

From the above consideration the N taps FIR filter equation can be expressed by

$$y(n) = \sum_{k=0}^{N-1} a_k x(n-k) \xrightarrow{RNS} \begin{cases} Y_{m_1}(n) = \left\langle \sum_{k=0}^{N-1} \langle a_k \rangle_{m_1} \cdot x_{m_1}(n-k) \right\rangle_{m_1} \\ \dots \\ Y_{m_P}(n) = \left\langle \sum_{k=0}^{N-1} \langle a_k \rangle_{m_P} \cdot x_{m_P}(n-k) \right\rangle_{m_P} \end{cases} \quad (4)$$

The binary to RNS conversion for the FIR filter is performed by reducing modulo m_i the input sequence $x(n)$, providing the residue digits x_{m_i} . The parallel filters mod m_i compute the residues Y_{m_i} (eq. (4)), while result $y(n)$ is obtained by the RNS to the binary conversion block. Clearly, the input and output conversions, constitute a significant overhead in systems implemented in RNS. However, efficient methods to perform those conversions are presented in [5], [6], and [7].

A Redundant Residue Number System (RRNS) is defined as a residue number system with r additional moduli. The first k moduli form a set of non redundant moduli, and their product represents the legitimate range, M that is, $M = \prod_{i=1}^k m_i$.

The remaining $P - k = r$ moduli form the set of redundant moduli that allows error detection and correction.

The corresponding integer X of a residue vector $(x_{m_1}, \dots, x_{m_P})$ belongs to the interval $[0, M_T - 1]$, where $M_T = \prod_{i=1}^P m_i$.

This interval, usually called total range, can be divided into two adjacent intervals: the interval $[0, M - 1]$ is called the

legitimate range and the interval $[M, M_T - 1]$ is the *illegitimate range*. The m_i -projection of X , denoted X_i , is defined as the residue vector $(x_{m_1}, \dots, x_{m_{i-1}}, x_{m_{i+1}}, \dots, x_{m_P})$, i.e. the representation of X with the i -th residue digit deleted. In [4] has been proved that in a redundant RNS with $r = 2$ for any error affecting an element i of the residue vector the projections X_j , for $j \neq i, i = 1, \dots, P$ are all illegitimate, while the X_i projection is in the legitimate range.

From this consideration is straightforward to detect and correct an error in the RRNS representation. The erroneous module is that characterized by his i -projection belonging to the legitimate range, while the correct value of the integer can be obtained performing the reverse conversion of the X_i projection. The architecture of the RNS FIR filter with the error correction capability is shown in Fig. 1.

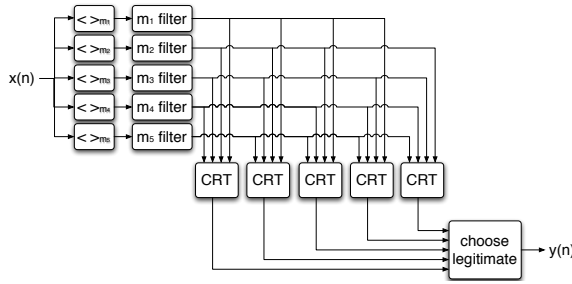


Fig. 1. RNS implementation of a FIR filter with error correction (P=5)

The blocks called CRT (Chinese Remainder Theorem) performs the reverse conversion for the m_i projections, while the block called "choose legitimate" select which input is in the legitimate range. The schema in Fig. 1 allows the correction of fault inside the m_i filters and the forward converters. The correction of the errors caused by a fault in a CRT block is not guaranteed by the schema in Fig. 1. The trivial way to supersede this problem is represented in Fig. 2, in which the TMR approach is used in the blocks realizing the reverse conversion and the error correction. Here, the final output is tripled and can re-converge outside the FPGA using the schema of minority voter presented in Fig. 3. The minority voter allows detecting and disabling the incorrect output.

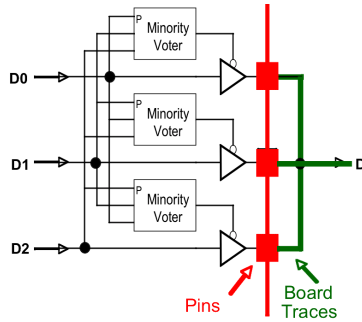


Fig. 3. minority voter and re-convergence outside the FPGA

III. FAULT MASKING IN THE REVERSE CONVERTERS

The complete fault coverage for the RNS based FIR filter requires the detection and correction of the faults that can occur in the following blocks:

- 1) the forward converter,
- 2) the modulo m_i filters,
- 3) the reverse converter.

The forward converter is realized implementing P independent m_i modulo reduction blocks. A fault in the forward converter produces an incorrect input only to the corresponding m_i filter and therefore has the same behavior of a fault inside one of the modulo m_i filters. The RRNS guarantees the detection and correction of a fault for these blocks.

Instead, if a fault occurs in one of the reverse converters (Fig. 1) the fault masking property is not guaranteed. All the input values for the block called "choose legitimate" belong to the legitimate range and the block cannot detect an error in one of its inputs. However, if the fault occurs in one of the reverse converters, the others reverse converters produce as output the correct value. Substituting the block called "choose legitimate" with a block performing a majority voting of the inputs a fault inside one of the CRT block can be masked. The block that substitutes the "choose legitimate" performs two operations:

- 1) Selection of a legitimate value between all the input values if a fault affects only a residue digit
- 2) Selection of the majority of its inputs if the fault affects one of the reverse converters.

To this purpose, a block called "legitimate voter" that selects the majority of the inputs belonging to the legitimate range to perform together these two operations has been introduced. The legitimate voter is therefore able to provide the correct output for all the faults occurring inside the RRNS based FIR filter. The only single point of failure of this architecture is represented by the legitimate voter itself. To obtain a total fault tolerance architecture for the filter, we can triplicate this block. It must be noticed that this choice is more convenient than a trivial TMR implementation of the reverse conversion requiring three of the m_i projection reverse converters and three "choose legitimate" blocks. In Fig. 4 the implementation of the totally fault tolerant RNS based FIR filter is presented. Also in this case the minority voter presented in Fig. 3 can be used to obtain the outputs re-convergence outside the FPGA.

IV. AREA OCCUPATION OF PROPOSED TECHNIQUE

In this section the technique previously presented is used to implement some typical FIR filter, to carry out the area occupation of the different filters and to compare these results with the trivial TMR technique. The FIR filters have been described in VHDL, and the implementation has been done using as target device a Xilinx Virtex V FPGA. The dynamic range of a filter depends on the dynamic range of coefficient, input data and number of taps, therefore the choice of moduli sets is dependent by these parameters. We choose two cases, one with 8 bits input data and coefficients and another with

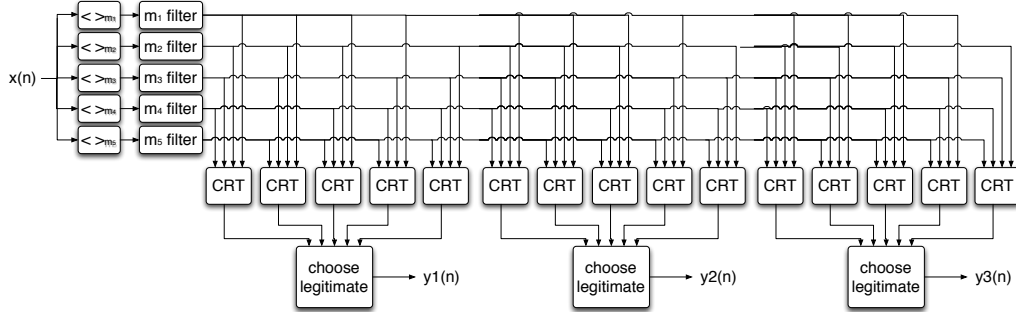


Fig. 2. RNS implementation of a FIR filter with error correction and TMR reverse converter (P=5)

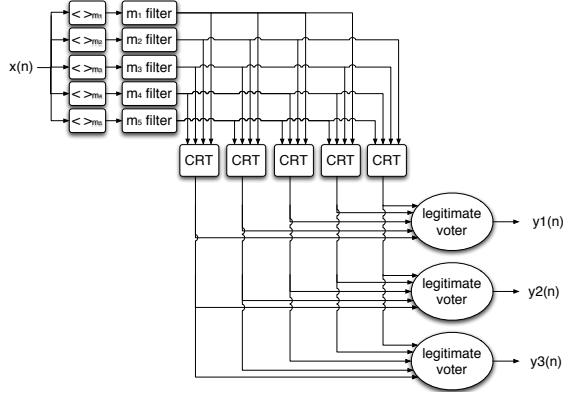


Fig. 4. Totally fault tolerant RNS based FIR filter

12 bits input data and coefficients. The number of taps of the filters vary from 16 to 256 taps. The set of non redundant moduli (the first k moduli) depends on the dynamic range of the filter, and the performance of RNS filter is strictly related to the chosen set of moduli. An RNS filter without error detection and correction capability requires the use of only a CRT module, while the RRNS presented in Fig. 1 requires a CRT block for each module of the moduli set and the TMR version of the RRNS filter (see Fig. 2) requires three CRT for each moduli. Instead, our proposed technique use a CRT for each moduli, but maintain the error correction capabilities of the reverse conversion procedure. The area occupation of a CRT is significant and is one of the problems that must be faced by a designer to optimize the implementation of the filter. For filter with error detection and correction capabilities this issue is yet more significant, due to the number of CRT that must be realized. From these consideration is clear that the number of moduli of the RNS system must be taken as small as possible and therefore in our implementation we use a minimal set of moduli ($k=3$ or 4) to realize the filters.

In Table I the different parameters, the dynamic range of the filter, and the chosen moduli set are presented for the filter taken into account for the evaluation of area occupation.

A fair comparison of our technique with the TMR approach must be done using as comparison the schema of Fig. 2. In fact, both these structures can correct an error in all the blocks composing the filter. For the TMR-RRNS filter we define the overhead as the number of LUTs to implement all the

Name	input data length	coefficients length	number of taps	dynamic range	moduli set
FIR1	8	8	16	20	129,128,127,125,121
FIR2	8	8	64	22	257,256,255,253,251
FIR3	8	8	256	24	257,256,255,253,251,241
FIR4	12	12	16	28	257,256,255,253,251,241
FIR5	12	12	64	30	257,256,255,253,251,241
FIR6	12	12	256	32	513,512,511,253,251,241

TABLE I

PARAMETERS AND DYNAMIC RANGE OF THE EXAMPLES FILTERS

CRT blocks and three choose legitimate blocks, while for our technique we use the number of LUTs for the CRT blocks and three legitimate voter block. Table II shows how the use of the legitimate voter blocks allows saving up than 33% of resource with respect to the TMR implementation of the blocks performing the error correction.

Name	TMR-RRNS overhead (# of LUTs)	overhead for our technique (# of LUTs)	%
FIR1	7407	2931	40%
FIR2	9774	3763	39%
FIR3	17037	5780	34%
FIR4	17127	5927	35%
FIR5	17196	5951	35%
FIR6	19242	7044	37%

TABLE II

COMPARISON OF OVERHEAD FOR TMR-RRNS AND OUR TECHNIQUE

REFERENCES

- [1] S. Bandyopadhyay, G.A. Jullien, A. Sengupta, "A Systolic Array for Fault Tolerant Digital Signal Processing Using a Residue Number System Approach", Proceedings of the International Conference on Systolic Arrays, Page(s):577 - 586, 25-27 May 1988
- [2] Mark H. Etzel and W. K. Jenkins "Redundant Residue Number Systems for Error Detection and Correction in Digital Filters", IEEE Transactions on Acoustics, Speech and Signal Processing, vol. ASS-28, No 5, pp. 538-544, October 1980.
- [3] W. K. Jenkins, "The Design of Error Checkers for Self-Checking Residue Number Arithmetic", IEEE Trans. on Computers, Volume C-32, Issue 4, Apr 1983 pp. 388 - 396
- [4] F. Barsi and P. Maestrini, "Error Correcting Properties of Redundant Residue Number Systems", IEEE Transactions on Computers, vol. C-22, pp. 307-315, Mar. 1973.
- [5] T. V. Vu, "Efficient Implementation of the Chinese Remainder Theorem for Sign Detection and Residue Decoding", IEEE Transactions on Circuits Systems-I, vol. 45, pp. 667-669, June 1985.
- [6] S. Piestrak, "A High-Speed Realization of a Residue to Binary Number System Converter", IEEE Transactions Circuits Systems-II Analog and Digital Signal Processing, vol. 42, pp. 661-663, Oct. 1995.
- [7] G. Cardarilli, M. Re, and R. Lojaco, "A Residue to Binary Conversion Algorithm for Signed Numbers", European Conference on Circuit Theory and Design (ECCTD97), vol. 3, pp. 1456-1459, 1997.