FPGA REALIZATION OF RNS TO BINARY SIGNED CONVERSION ARCHITECTURE

Marco Re, Alberto Nannarelli, Gian Carlo Cardarilli, Roberto Lojacono

Univ. of Rome ”Tor Vergata”, Dept. of Electronic Eng.
00133 Rome, Italy
Fax:(+39-06-2020519)
E-Mail: {marco.re, nannarelli, g.cardarilli}@ieee.org, lojacono@ing.uniroma2.it

ABSTRACT

The use of the Residue Number System (RNS) in modern telecommunication and multimedia applications is becoming more and more important because it allows interesting advantages in terms of precision, power consumption and speed. Generally, the output conversion from residue to binary is the crucial point in effective realizations of application specific architectures based on residual arithmetic. This paper presents a general conversion procedure based on a N moduli set. The algorithm can process both unsigned and signed numbers. Based on this algorithm an architecture which efficiently implements the output conversion is illustrated. The architecture has been mapped on a FPGA.

1. INTRODUCTION

The advantages of Residue Number System (RNS) processing are discussed in several publications and books [5], [13], [14]. Error free computation, simplified and fast addition and multiplication, possibility to obtain parallel architectures are among the more important advantages. New important RNS research topics, as for example those related to low power DSP implementations, are also emerging [1], [2], [11]. The practical use of RNS is however strongly limited by the input and output conversions required for the translation from the binary to the RNS representation and vice versa. In fact, the implementation of the converters constitutes a fixed overhead on the total area, delay and power dissipation. For these reasons the output conversion, which is generally performed using the Chinese Remainder Theorem (CRT in the following), still appears to be a crucial point in the realization of competitive RNS subsystems and, therefore, represents one of the main topics in the recent RNS research activities. Some authors have proposed the use of three moduli sets [3], [4], [6], [7], [8] in order to obtain simpler and more efficient output conversion architectures. For example papers [7] and [8] consider the set $(2n-1,\ 2n,\ 2n+1)$. Of course, this approach reduces the exploitation of the RNS properties (the maximum advantages are obtained by using a lot of small value moduli). In fact, n grows with the desidered wordlength and, correspondly, the resulting modular processor becomes slower. On the other hand, high speed and low power multimedia applications require DSP hardware with large dynamic range and fine granularity in the wordlength selection (this aspect is related to the wordlength optimization). Of course these requirements cannot be fully matched by using three moduli. To overcome these problems, in a lot of applications moduli sets with more than three moduli are required. In [12] an efficient method for the RNS-Binary conversion, based on a set of N moduli, has been proposed. Although this method does not limit the number of moduli, it imposes an important limitation because only odd moduli can be used. This reduces the RNS advantages because power of two modular arithmetic exhibits very efficient implementations (for this reason, normally the greatest modulo is chosen of the form 2^h). Our work is aimed to remove the above limitation preserving the properties of the method presented in [12]. The paper is organized as follows. Section 2 describes the algorithm while in Section 3 the extension for the signed conversion is given. Section 4 shows an actual implementation of the proposed algorithm. The VLSI architecture is presented and the mapping on FPGA XILINX V1000-6 is discussed. The conclusions are drawn in Section 5.

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

2. THE NEW CONVERSION ALGORITHM

The classical CRT formulation for 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, $M = \prod_{i=1}^{N} m_i$,
$r_i = \langle X\rangle_{m_i}\ i \in [1,\ N]$,

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

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 circuit two problems arise. The first one concerns the complexity of the arithmetic operations involved (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 [9] 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 [10] 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 firstly 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 (see Appendix A)

$$-2^k < \alpha < N \tag{15}$$

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 the factor 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

$$\Upsilon = \{\alpha \in I : \langle \alpha \cdot \widetilde{M} \rangle_{2^k} = \langle \widetilde{H} \rangle_{2^k}\} \tag{16}$$

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

$$2^k \geq N - 1 \tag{17}$$

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 (16) 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

$$\varepsilon = \frac{\widetilde{H} - \alpha \cdot \widetilde{M}}{2^k} = \frac{\widetilde{H} - \alpha' \cdot \widetilde{M}}{2^k} + \widetilde{M} \tag{18}$$

The procedure deriving from (18) 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.

3. SIGNED NUMBER CONVERSION

The conversion into a two's complement representation can be easily performed by using the following conventions for the RNS representation of signed numbers. Since M is even, positive numbers are into the range $[0, (M/2) - 1]$ and negative ones are in $[M/2,\ M-1]$. The signed conversion must translate these ranges into the ranges of the two's complement representation (for m bits, the positive numbers are in the range $[0,\ 2^{m-1}-1]$ while the negative ones are mapped in the interval $[2^{m-1},\ 2^m-1]$). This translation can be performed by considering the following procedure. As first step we add, $mod\ M$, the quantity $P = M/2$. This operation translates the positive numbers into the range $[M/2,\ M-1]$, while the negative ones are now in the interval $[0,\ (M/2)-1]$. As a final step, the two's complement value of the output can be reconstructed through the binary subtraction of the value $M/2$ from the final result.

This procedure has been embedded in our algorithm and in order to reduce the algorithm steps, the final subtraction has been merged with the conditional subtraction required for the α correction. Therefore if the reconstructed value $X' = \varepsilon' \cdot 2^h + \langle r_N + P\rangle_{2^h}$ is positive we only subtract the value P. Otherwise, for negative values, the quantity $M - P = M/2$ is added. The above algorithm can be summarized in the following steps

1. Compute the quantity $\widetilde{H}$ using the modified residue $r_i' = \langle r_i + P\rangle_{m_i}$.
2. Compute the quantities ε', $X' = \varepsilon' \cdot 2^h + \langle r_N + P\rangle_{2^h}$.
3. Compute the quantity X. If X' is negative the two's complement output result is obtained as $X = X' + M - P$ otherwise $X = X' - P$

3.1. 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 (17), $k = 2$. For this set we have

$$\begin{array}{ll} \hat{m}_i = \{280,\ 168,\ 120,\ 105\}, & \hat{m}_i^{-1} = \{1,\ 2,\ 1,\ 1\}, \quad M = 840 \\ \widetilde{M} = 105, & P = 420 \end{array}$$

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 $X = 211 - 420 = -209$.

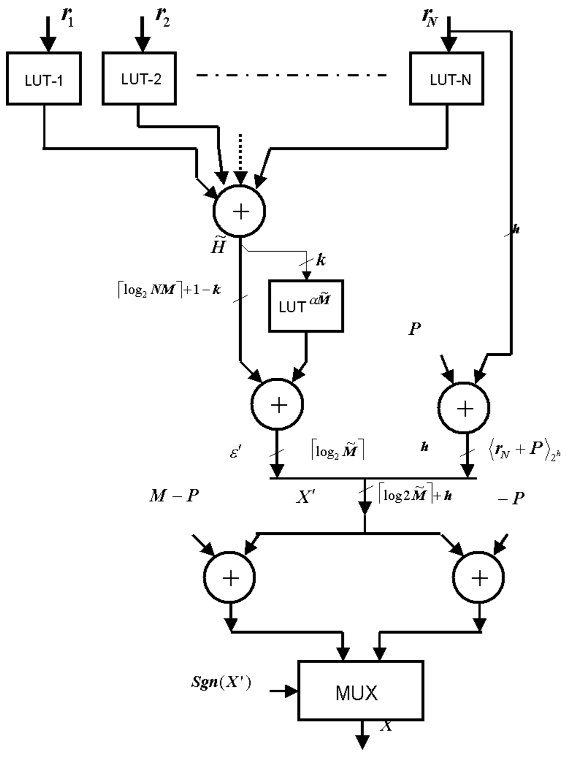
4. THE VLSI ARCHITECTURE

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 (CSA) 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 $Sgn(X')$ 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 fast 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).



r_1
r_2
r_N
LUT-1
LUT-2
LUT-N
$\widetilde{H}$
k
$\lceil \log_2 NM \rceil + 1 - k$
LUT $\alpha\widetilde{M}$
P
h
ε'
$\lceil \log_2 \widetilde{M} \rceil$
h
$\langle r_N + P\rangle_{2^h}$
$M - P$
X'
$\lceil \log 2\widetilde{M} \rceil + h$
$-P$
$Sgn(X')$
MUX
X

Figure 1: The converter architecture

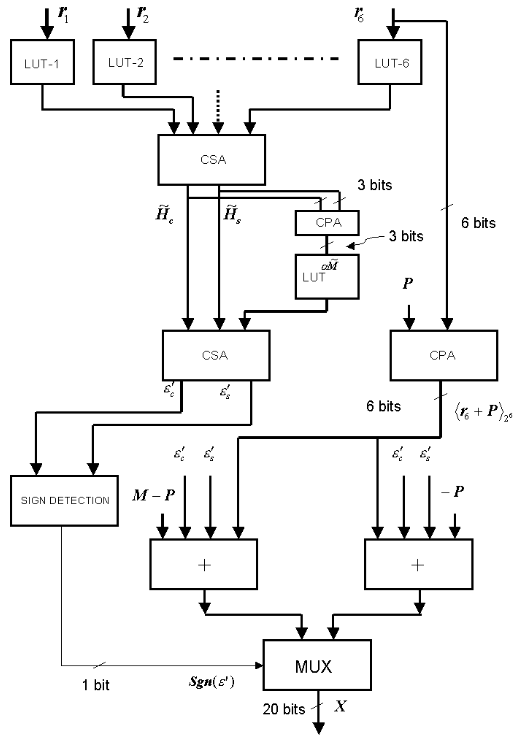


Figure 2: The implemented architecture

5. CONCLUSIONS

In this paper a general algorithm for N moduli CRT based conversion is presented. Starting from this formulation a fast architecture has been obtained. This architecture is able to perform the conversion by using general moduli sets including a power of two module. The architecture has been implemented by using carry save adders and has been mapped on a FPGA. The obtained results indicate that LUT based FPGAs can be an effectively used to map RNS converters architectures.

Appendix A

The range of ε is

$$0 \leq \varepsilon = \frac{\widetilde{H} - \alpha \cdot \widetilde{M}}{2^k} \leq \widetilde{M} - 1 \qquad (A.1)$$

Moreover, the term $\widetilde{H}$ is bounded by

$$\widetilde{H}^- = -2^k + 1 \leq \widetilde{H} < \widetilde{H}^+ = N \cdot \widetilde{M} \qquad (A.2)$$

From A.1 we have

$$\frac{\widetilde{H}}{\widetilde{M}} \geq \alpha \geq \frac{\widetilde{H}}{\widetilde{M}} - \frac{2^k(\widetilde{M}-1)}{\widetilde{M}} \qquad (A.3)$$

Substituting $\widetilde{H}^+$ respectively in the left side and $\widetilde{H}^-$ in the right side of A.3 we obtain

$$\frac{\widetilde{H}^+}{\widetilde{M}} = \frac{N\widetilde{M}}{\widetilde{M}} > \alpha \geq -2^k + \frac{1}{\widetilde{M}}$$

Consequently the range of α is $-2^k < \alpha < N$

6. REFERENCES

[1] M. Bhardwaj and A. Balaram, ”Low Power Signal Processing Architectures Using Residue Arithmetic,” Proceedings of IEEE International Conference on Acoustic, Speech and Signal Processing (ASSP’98), vol. 5, pp. 3017-3020, 1998.

[2] G. C. Cardarilli, A. Nannarelli and Marco Re, ”Reducing Power Dissipation in FIR Filters using the Residue Number System,” Midwest Symposium on Circuit and Systems 2000, Lansing, Michigan, Augut 8-11, 2000.

[3] R. Conway, J. Nelson, ”Fast Converter for 3 Moduli RNS Using New Property of CRT,” IEEE Trans. on Computers, Vol. 48, no. 8, pp. 852-860, August 1999.

[4] D. Gallaher, F. E. Petry, and P. Srinivasan, ” The Digit Parallel Method for Fast RNS to Weighted Number System Conversion for Specific Moduli ($2^k-1,\ 2^k,\ 2^k+1$,” IEEE Trans. Circuits Syst.-II, Vol. 44, no. 1, pp. 53-57, January 1997.

[5] S. K. Mitra, J. F. Kaiser,”Handbook for Digital Signal Processing,” Chapter 9, pp. 611-677, John Wiley & Sons.

[6] S. Piestrak, ”A High-Speed Realization of a Residue to Binary Number System Converter,” IEEE Trans. Circuits Syst.-II, Vol. 42, no. 10, pp. 661-663, October 1995.

[7] A. B. Premkumar, ”An RNS to Binary Converter in 2n-1, 2n, 2n+1 Moduli Set,” IEEE Trans. Circuits Syst.-II, Vol. 39, no. 7, pp. 480-482, July 1992.

[8] A. B. Premkumar, M. Bhardwaj, and T. Srikathan, ”High-Speed and Low-Cost Reverse Converters for the $2n-1,\ 2n,\ 2n+1$ Moduli Set,” IEEE Trans. Circuits Syst.-II, Vol. 45, no. 7, pp. 903-908, July 1998.

[9] K.M. Elleyth, M.A. Bayoumi, ”Fast and Flexible Architectures for RNS Arithmetic Decoding,” IEEE Trans.Circuits Systems.-II Analog and Digital Signal Processing, Vol.39, No.4, pp. 226-235, April 1992.

[10] F. Barsi, ”Mod m arithmetic in binary systems,” Information Processing Letters, Vol.40, No.6, pp. 303-309, 30 December 1991.

[11] W.L.Freking, K.K.Parhi, ”Low-Power Digital Filters Using Residue Arithmetic,” Thirthy-First Asilomar Conference on Signals, Systems and Computers 1998, Vol. 1, pp.739-743, 1998.

[12] G.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 1997, Budapest, Hungary.

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

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