EFFICIENT MODULO EXTRACTION FOR CRT BASED RESIDUE TO BINARY CONVERTERS

G.C. Cardarilli, M. Re, R. Lojacono

Dept. of Electronic Engineering, University of Rome Tor Vergata
Via della Ricerca Scientifica 1, 00133 Rome - Italy
Phone:(+39-6-72594481)
Fax:(+39-6-2020519)
E-Mail: {Cardarilli, Re, Lojacono}@utovrm.it

ABSTRACT

RNS to binary conversion is a very critical task for an advantageous implementation of RNS based Digital Signal Processors.
This paper presents a new technique for the modulo extraction useful for the implementation of efficient Chinese Remainder Theorem (CRT) based converters. The method, based on the multiplication of the CRT terms by a power of two, greatly simplifies the extraction of the final modulo only requiring a look-up table access and two sums. The resulting structures appear very efficient in terms of speed and area occupancy.

1. INTRODUCTION

Residue number system (RNS) offers fast carry-free, addition and multiplication, as well as the possibility to easily implement error detection and correction procedures. The use of RNS arithmetic for designing special-purpose digital signal processing hardware has been greatly investigated [1], [2], [3].
The output conversion has been pointed out as a critical step for an advantageous implementation of RNS digital signal processors. This step is required to return the RNS processed data into the conventional binary representation.
Different techniques are available in literature to perform this task. The more used ones are those based on the Chinese Remainder Theorem (CRT). A number of hardware architectures have been proposed for the implementation of such converters. These proposals consider two main problems related to the CRT approach

- efficient computation of the arithmetic operations involved in CRT
- efficient computation of the final modulo operation.

Most of the papers face the first problem, presenting very efficient solutions either for general class of moduli [4], [5] or for specific sets (e.g. the set $\{2^n-1, 2^n, 2^n+1\}$) [6].
The second problem is related to the need of the final modulo operation. This operation can be performed by using a tree of modular adders [5]. The related structure is regular but requires a quite large number of binary adders. So it appears to be slow. On the other hand there exist a numbers of architectures that compute directly the final sum by using distributed structures [4]. These structures do not permit the use of modular adder trees at all.
Above considerations suggest to search an efficient method for applying the modulo operator only on the result of the summation. This means that we can obtain a final number that largely exceed the modulus value. Some architecture are presented in the literature for this kind of modulo operator [5], [7].
To reduce the complexity of the architectures some authors proposed solutions which avoid the modulo operation by introducing the scaling of the CRT, computing an approximated result [8], [9]. These methods require quite large tables that must be implemented by using ROM. Due to the table dimension, the efficiency of the structure generally decreases (in terms of speed, silicon area, and power consumption).
In this paper we present a new method to overcome the drawbacks related to the implementation of the final modulo operation. This method is based on the scaling of the CRT terms by a power of two factor. Its implementation only requires two adders and a very small look-up table. The small dimension of the look-up table allows the realization of a simple and fast hardware structure.

2. ARITHMETIC OPERATIONS IN CRT

It is well known that the CRT is able to compute the number X starting from its RNS representation in a set of pairwise prime moduli $\{m_i\}$, with $i = 1,2,.....N$. Considering the set of N residue numbers x_i, there exists only a number $X < m_1\, m_2\, \dots\, m_N = M$ such that $x_i = \langle X \rangle_{m_i}$ for $i = 1,2,.....N$. Using the CRT we are able to compute this number by means of one of the following expressions

$$X = \left\langle \sum_{i=1}^{N} \hat{m}_i \hat{m}_i^{-1} x_i \right\rangle_M = \langle H \rangle_M \tag{1.a}$$

$$X = \left\langle \sum_{i=1}^{N} \hat{m}_i \left\langle \hat{m}_i^{-1} x_i \right\rangle_{m_i} \right\rangle_M = \langle H \rangle_M \tag{1.b}$$

where $\hat{m}_i = \dfrac{M}{m_i}$ and $\hat{m}_i^{-1}$ stands for $\left\langle \dfrac{1}{\hat{m}_i} \right\rangle_{m_i}$.

Where the eq. (1.a) and (1.b) correspond to different CRT realizations.

When eq. (1) are implemented by a digital circuit two problems arise. The first one concerns the complexity of the arithmetic operations involved, since a set of addition and (modulo) multiplications are required.

There are a number of methods to efficiently implement the computation of the term H. In [5] look-up tables are used to compute the terms $\left\langle \hat{m}_i \left\langle x_i \hat{m}_i^{-1} \right\rangle \right\rangle_M$ and a tree of carry save adders implements the summation. In our work we suppose that a similar method is used to compute the term H.

The second problem regards the computation of the external $mod\, M$ operation. This operation is very complex [7] and its complexity grows with the number of the moduli.

Our approach simplifies the above modulo operation multiplying the CRT terms $\left\langle \hat{m}_i \left\langle x_i \hat{m}_i^{-1} \right\rangle \right\rangle_M$ by a proper power of two. This scaling greatly simplifies the modular operation. Moreover we suppose that all the moduli used are odd.

In the following section we will introduce this methodology and will show as it can be applied to the modulo operation independently from the moduli set. The use of the scaling to simplify the hardware computation will be discussed in the next section.

3. NEW METHOD FOR THE MODULO COMPUTATION

As pointed out above, the modulo computation is one of the major bottleneck in the hardware implementation of the CRT based converters. This is mainly due to the large value of the final M modulus and to the dynamic range of the term H. These difficulties grow with the number of moduli used in the RNS arithmetic. In fact, considering eq. (1.b), we obtain the following bounds for H

$$0 \le H = \sum_{i=1}^{N} \hat{m}_i \left\langle x_i \hat{m}_i^{-1} \right\rangle_{m_i} \le \sum_{i=1}^{N} \frac{M}{m_i}(m_i - 1) < NM \tag{2}$$

Equation (2) shows that the range of H is directly related to the number N. Moreover, the methodologies used for the modulo computation of specific modulus set (as those based on prime numbers close to power of two) appear not useful for this modulo operation. Indeed, if we will maintain the generality of the procedure, the final modulus M cannot be constrained. To obtain a more suitable form of modulo operation, let us consider the number $2^k X$ being k a suitable integer quantity. Applying (1.b) to the number $2^k X$ we obtain

$$\left\langle 2^k X \right\rangle_M = \left\langle \sum_{i=1}^{N} \hat{m}_i \left\langle 2^k x_i \hat{m}_i^{-1} \right\rangle_{m_i} \right\rangle_M \tag{3}$$

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

$$2^k X = \sum_{i=1}^{N} \hat{m}_i \left\langle 2^k x_i\; \hat{m}_i^{-1} \right\rangle_{m_i} - \alpha M \tag{4}$$

where α comes from the two modulo operations.

From (4) we get

$$X = \frac{\sum_{i=1}^{N} \hat{m}_i \left\langle 2^k x_i\; \hat{m}_i^{-1} \right\rangle_{m_i} - \alpha M}{2^k} = \frac{\widetilde{H} - \alpha M}{2^k} \tag{5}$$

The term αM of (5) can assume either positive or negative values.

From the last term of (5), it is possible to bound the value of α. In fact, if the term $\widetilde{H}$ is bounded in the range $0 \le \widetilde{H} \le \widetilde{H}^+$, α will lies in the interval

$$-2^k < \alpha \le \frac{\widetilde{H}^+}{M} \tag{6}$$

Again, introducing the bounds given by (2) in eq.(6) we obtain

$$-2^k < \alpha < N \tag{7}$$

Eq. (5) suggests a very efficient method to compute X starting from $\widetilde{H}$. In order to obtain integer values of X, the value $\widetilde{H}-\alpha M$ must be a multiple of 2^k, i.e. the least significant k bits of $\widetilde{H}-\alpha M$ are zero. This means that the correct value of α belong to the set

$$\Omega=\left\{\alpha \in I:\ \langle \alpha M\rangle_{2^k}=\langle \widetilde{H}\rangle_{2^k}\right\} \tag{8}$$

Where I is the set of integer numbers. This set only depends on the k least significant bits of $\widetilde{H}$.
Using these bits we are able to select only 2^k values of αM, of the $N+2^k-1$ possible values, positive and negative, according with (7).
If k is chosen such that

$$2^k \geq N-1 \tag{9}$$

the $N+2^k-1$ values of αM can be computed starting from the 2^k positive values stored in a proper look-up table. In fact, since X must be a positive number, the quantity $\widetilde{H}-\alpha M$ must be positive. If this does not happens, the obtained value of $\alpha \in \Omega$ is incorrect. From (7) and (8) the correct value is obtained by subtracting 2^k from the incorrect one.
So, if α' is the incorrect value obtained by the look-up table and α is the correct one, X is obtained by (see Fig.1)

$$X=\frac{\widetilde{H}-\alpha M}{2^k}=\frac{\widetilde{H}-(\alpha'-2^k)M}{2^k}=$$
$$=\frac{\widetilde{H}-\alpha' M}{2^k}+M \tag{10}$$

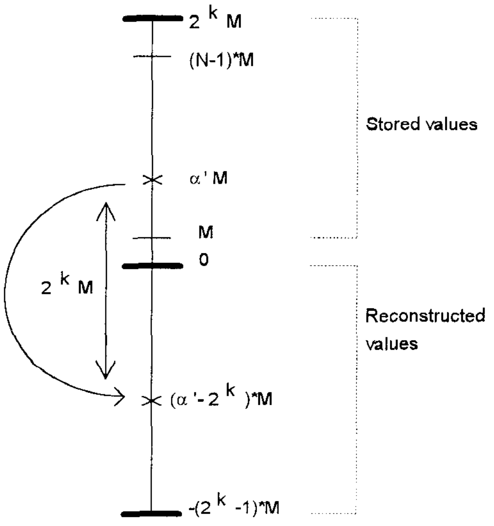

Fig. 1 Dynamic range of the term αM

The procedure deriving from eq. (10) can be summarized by the following steps.

- The term $\alpha' M$ is read from the look-up table addressed by the k least significant bits of $\widetilde{H}$.
- The sum $\widetilde{H}-\alpha M$ is computed and the k least significant bits are discarded.
- If the result of the above step is negative the modulus M is added.

3.1 Numerical example

Let consider the RNS composed by the six moduli *{3, 5, 7, 11, 13, 17}*, *M*=*255255*, *N*=6. For this set eq. (9) give *k*=*3*. Consequently, α results in the range $-8<\alpha<6$.
The look-up table stores all the multiples αM corresponding to positive value of α in the range $1 \leq \alpha \leq 8$.
Considering for example the value *X*=*1530*. It corresponds to the residue set *{0, 0, 4, 1, 9, 0}*. Applying CRT as described in (5) we obtain $\widetilde{H}$=*267495*. Addressing the look-up table by using the three least significant bits of $\widetilde{H}$ namely a bit string 111 the multiple of αM corresponding to $\alpha=1$ is selected. Consequently (5) becomes

$$X=\frac{\widetilde{H}-\alpha M}{2^k}=\frac{267495-255255}{8}=1530.$$

In this case the obtained value is the correct one.
On the other side, let us consider the value *X*=*120300*. Its RNS representation is given by *{ 0, 0, 5, 4, 11, 8}*. Using (5) we obtain $\widetilde{H}$=*707145* and $\alpha'=7$. From (5) we obtain

$$X=\frac{\widetilde{H}-\alpha' M}{2^k}=\frac{707145-1786785}{8}=-134955.$$

Since the result is negative the above procedure requires the sum of X with M. In this way we obtain the correct result

X=-134955+255255=120300.

4. HARDWARE ARCHITECTURE

Fig. 2 shows the block diagram of the converter. The structure is based on an array of small look-up tables that stores the terms $\langle 2^k x_i \hat{m}_i^{-1}\rangle_{m_i}$. The input look-up tables are very small and fast because we use a number of small moduli to implement a large dynamic range. This input stage is followed by a tree of adders [5] implementing the summation.

The final modulo operation is obtained implementing equation (10) by using two adders and a look-up table containing 2^k words, each of $n = \lceil log_2 M \rceil$ bits.

For example let us consider the implementation of a *32* bit arithmetic using the eigth moduli *{7, 11, 13, 17, 23, 25, 27, 29}*. With this moduli set the inputs of the look-up tables storing the terms $\left\langle 2^k x_i \hat{m}_i^{-1} \right\rangle_{m_i}$ require from *3* to *5* bits.

Moreover the look-up table of the final modulo operation requires only *8* words of *32* bits each. This table can be realized by sparse logic or by using a set of multiplexers.

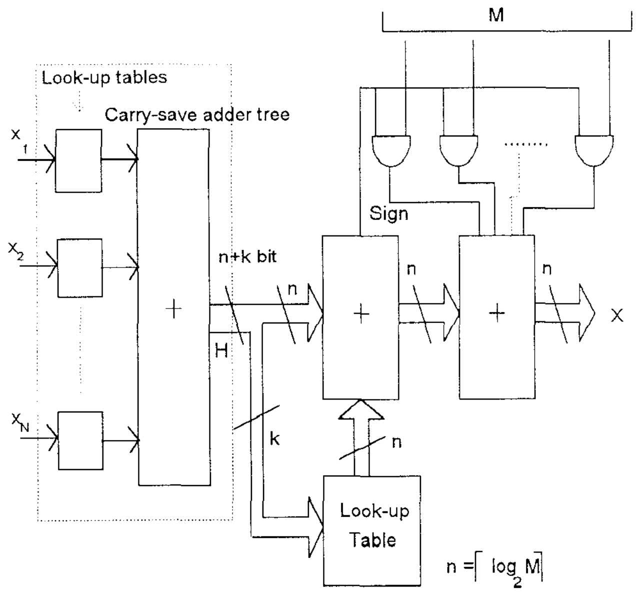

Fig. 2 Block diagram of the converter

5. CONCLUSIONS

In this paper a new method for the implementation of the modulo operation required by the CRT is presented. By introducing a multiplication by 2^k of the terms in the summation, we are able to determine the multiple of the modulus that must be subtracted only checking the k the least significant bits of $\widetilde{H}$. The method can be applied either to the CRT architectures based on adder trees that those which compute the summation result by using distributed structures. The resulting architecture appears very attractive in terms of speed and area occupancy. Work is in progress to develop a 32 bit CRT architecture for very high-speed DSP applications.

The converter is under development by using ES2 CMOS 0.7 μm technology.

REFERENCES

[1] 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.

[2] N.S. Svabo and R.I. Tanaka, "Residue Arithmetic and its Application to Computer Technology", New York: 1967, McGRAW-HILL BOOK COMPANY.

[3] G.C. Cardarilli, R. Lojacono, G. Martinelli, M. Salerno, "Structurally Passive Digital Filters in Residue Number Systems", *IEEE Trans. Circuits System*, vol. 35, no. 2, pp. 149-158, February 1988.

[4] F. Barsi, M.C. Pinotti, "A fully parallel algorithm for residue to binary conversion", *Information Processing Letters,* Vol.50, No.6, pp. 1-8, 1994.

[5] 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.

[6] S.Piestrak, " A High-Speed Realization of a Residue to Binary Number System Converter", *IEEE Trans. Circuits Systems-II Analog and Digital Signal Processing*, vol. 42, no. 10, pp. 661-663, October 1995.

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

[8] M.A. Sodestrand, C. Vernia, and J.-H. Chang, "An improved residue number system digital-to-analog converter", *IEEE Trans. Circuits and Systems,* Vol.30, pp. 903-907, December 1983.

[9] J.Y. Kim, K.H. Park, and H.S. Lee, "Efficient residue-to-binary conversion technique with rounding error compensation", *IEEE Trans. Circuits and Systems,* Vol.38, pp. 315-317, March 1991.