RESIDUE-TO-BINARY ARITHMETIC CONVERTER FOR MODULI SET $\{2^n-1, 2^n, 2^n+1, 2^{n+1}-1\}$

Jimson Mathew, D. Radhakrishnan, T. Srikanthan

Division of Computer Engineering, School of Applied Science
Nanyang Technological University
Singapore 639798

ABSTRACT

This paper presents a new reverse converter architecture for the moduli set $\{2^n-1, 2^n, 2^n+1, 2^{n+1}-1\}$. It exploits the special properties of the numbers of the form $2^x \pm 1$, and extends the dynamic range of the present triple moduli based systems. Here we use a combination of CRT and MRC for reverse conversion. With a pipelined system the throughput rate is that of a single 3n+1 bit binary adder delay.

1. INTRODUCTION AND BACKGROUND

Residue number system (RNS) is popular in high performance arithmetic applications like Digital Signal Processing (DSP) systems because of its carry free nature, modular structure and error correcting properties [1,2]. It simplifies large number computation by decomposing them into a set of parallel independent computations on residues generated with respect to a convenient moduli set. Even though RNS is capable of performing high speed arithmetic, it was not popular in general computing because of the difficulties involved in division, magnitude comparisons, forward and reverse conversions between binary and residue numbers etc.

In RNS, an integer in the interval [0,M) can be represented as a set of n residues $(x_1, x_2, \ldots, x_n)$, with respect to a moduli set of n relatively prime integers $(m_1, m_2, \ldots, m_n)$ where $x_i = X \bmod m_i$ and $M = \prod_{i=1}^{n} m_i$. Arithmetic operations are done using the formula:

$(x_1, x_2, \ldots, x_n) \otimes (y_1, y_2, \ldots, y_n) = (z_0, z_2, \ldots, z_n)$ where $z_i = |x_i \otimes y_i|_{m_i}$ and $\otimes$ denotes one of the operations of addition, subtraction or multiplication. Thus arithmetic operations on residues can be performed in parallel without any carry propagation among the residue digits. However, before performing any operations on residues, first the number has to be converted from binary to residue. The problem of translating a binary integer X to the residue representation is called forward conversion. Getting back to the weighted representation of X from its residue representation is referred to as reverse conversion. The reverse conversion is still considered a complex problem and hence we attempt to provide some simplification in this direction. The reverse conversion is generally based on Chinese Remainder Theorem (CRT) and is given by: $X = \left|\sum_{i=1}^{n} a_i x_i \hat{M}_i\right|_M$, where $\hat{M}_i = \frac{M}{m_i}$ and $a_i = \left|\frac{1}{\hat{M}_i}\right|_{m_i}$. A second approach uses Mixed Radix Conversion (MRC) and is given by the following equation

$X = A_n \prod_{i=1}^{n-1} m_i + \ldots + A_2 m_1 + A_1$ where $A_i \in [0, m_i)$ and

$$A_i = \left| \left| \frac{1}{m_1 m_2 \cdots m_{i-1}} \right| (x_i - (A_{i-1} m_{i-2} \ldots m_1 + \ldots + A_2 m_1 + A_1)) \right|_{m_i}$$

. Majority of the high speed reverse converter architectures available are based on the triple moduli set $\{2^n-1, 2^n, 2^n+1\}$ [3-7]. However, this popular set is inefficient if increased parallelism is required to benefit from the residue arithmetic. In other words, it is inefficient in dealing with a large dynamic range. One way to overcome this inadequacy is to add extra moduli to this set from the same family $(2^n \pm 1)$. But it can be shown that if we add one more modulus to the triple moduli set, then the closed form multiplicative inverses become complex and cannot be implemented with simple

hardware as in the case of triple moduli set. In Section 2 we derive the muliplicative inverses for the four moduli. Section 3 describes the proposed algorithm for conversion. The hardware implementations are shown in Section 4 illustrated with an example.

2. EXTENDED MODULI SET $\{2^n-1, 2^n, 2^n+1, 2^{n+1}-1\}$

A valid moduli set exists only for even values of n. In our notation (m_1, m_2, m_3, m_4) and ($\hat{M}_1^{-1}$, $\hat{M}_2^{-1}$, $\hat{M}_3^{-1}$, $\hat{M}_4^{-1}$) are the moduli set and multiplicative inverse set respectively, where $M = (2^n-1)(2^n)(2^n+1)(2^{n+1}-1)$, $\hat{M}_1 = \frac{M}{m_1}$, and $\hat{M}_i^{-1} = \left|\frac{1}{\hat{M}_i}\right|_{m_i}$. From basic definitions we have $\left|\hat{M}_1^{-1}\hat{M}_1\right|_{m_1} = 1$, i.e, $\left|\hat{M}_1^{-1}(2^n)(2^n+1)(2^{n+1}-1)\right|_{2^n-1} = 1$.

But $\left|2^n\right|_{2^n-1} = 1$ and $\left|2^{n+1}\right|_{2^n-1} = 2$, therefore, $\left|\hat{M}_1^{-1}(1)(1+1)(2-1)\right|_{2^n-1} = 1$; and hence $\hat{M}_1^{-1} = 2^{n-1}$.

Similarly, $\hat{M}_2^{-1} = \left|\frac{1}{(2^n-1)(2^n+1)(2^{n+1}-1)}\right|_{2^n}$. So

$\left|\hat{M}_2^{-1}(2^n-1)(2^n+1)(2^{n+1}-1)\right|_{2^n} = 1$. But $\left|2^n\right|_{2^n} = 0$

Hence $\left|\hat{M}_2^{-1}(-1)(+1)(-1)\right|_{2^n} = 1$; i.e., $\hat{M}_2^{-1} = 1$.

Also $\left|\hat{M}_3^{-1}(2^n-1)(2^n)(2^{n+1}-1)\right|_{2^n+1} = 1$.

But $\left|2^n\right|_{2^n+1} = -1$ and $\left|2^{n+1}\right|_{2^n+1} = -2$. So

$\left|\hat{M}_3^{-1}(-6)\right|_{2^n+1} = 1$; i.e., $\hat{M}_3^{-1} = 2^{n-1} + \frac{2}{3}(2^{n-1}+1)$

Finally, $\left|\hat{M}_4^{-1}(2^n-1)(2^n)(2^n+1)\right|_{2^{n+1}-1} = 1$.

But $\left|2^n\right|_{2^{n+1}-1} = \frac{1}{2}$, therefore; $\left|\hat{M}_4^{-1}(\frac{-3}{8})\right|_{2^{n+1}-1} = 1$.

It can be written as $\hat{M}_4^{-1} = \frac{-8}{3}(1-(2^{n+1}-1)) + 2^{n+1} - 1$. Simplifying,

$\hat{M}_4^{-1} = \frac{16}{3}(2^n-1)$

The above multiplicative inverses are essential for CRT and MRC implementations. From the above derivations it is very clear that when we add a fourth modulus to the triple moduli set the "nice" closed form inverse property is ruined. Some of multiplicative inverses are no longer a power of 2 or a diminished power of 2, which leads to complex hardware compared to much popular triple moduli implementations. Hence we propose the following algorithm for reverse conversion.

3. REVERSE CONVERTER FOR THE FOUR MODULI SET

It is well known that the popular triple moduli converter needs only very simple hardware and takes only minimum conversion time compared to different reverse converter architectures. Basically superset is an extension of the triple moduli set and hence we use the same concept for reverse conversion as well. That is, first we use a triple moduli converter, then we extend it to the fourth modulus by MRC. The procedure is illustrated as follows:

Consider the superset $\{2^n-1, 2^n, 2^n+1, 2^{n+1}-1\}$, where n is any even integer. For the moduli set (m_1, m_2, m_3, m_4) the residue representation of the number X, in the interval [0, M) is given by (x_1, x_2, x_3, x_4) and the associated mixed radix representation will be then (A_1, A_2, A_3, A_4). The number X can be written as:

$$X = A_1 + m_1A_2 + m_1m_2A_3 + m_1m_2m_3A_4 \qquad (3.1)$$

Now let $X_1 = A_1 + m_1A_2 + m_1m_2A_3$, and $X_2 = m_1m_2m_3A_4$, so that $X = X_1 + X_2$,

where $A_4 = \left|(x_4 - \left|X_1\right|_{m_4})k\right|_{m_4}$ (3.2)

and $k = \left|\frac{1}{m_1m_2m_3}\right|_{m_4}$.

4. HARDWARE IMPLEMENTATION

First X_1 can be evaluated from the residue set $\{x_1, x_2, x_3\}$ using a triple moduli converter. The architecture proposed in [7] is assumed here. As far as the implementation is concerned, we can have two architectures depending on the way in which we do the operation $\left|X_1\right|_{m_4}$. We can compute $\left|X_1\right|_{m_4}$ directly from X_1 using a forward converter.

Since the fourth modulus is 2^{n+1}-1, the forward converter can be implemented with simple adder circuits. The procedure is based on the partitioning of the binary number X_1 into a number of (n+1) bit fields and then performing modulo addition. That is, the number can be expressed as $X_1 = k_2(2^{n+1})^2 + k_1 2^{n+1} + k_o$. Then, $|X_1|_{2^{n+1}-1} = |k_2 + k_1 + k_o|_{2^{n+1}-1}$. Further, we can rewrite Equation 3.2 as: $A_4 = |(x_4 - \hat{x})k|_{m_4}$, where $\hat{x} = |X_1|_{m_4}$. We propose the following technique for implementing the above equation. Here we have subtraction of two variables and multiplication by a constant followed by modulo correction with m_4. For simplicity let us denote the operation as:

$$A_4 = |(X - Y)k|_m \quad (4.1)$$

X and Y can be expressed as $X = 2^{n-1}x_{n-1} + 2^{n-2}x_{n-2} + \cdots + x_0$, and $Y = 2^{n-1}y_{n-1} + 2^{n-2}y_{n-2} + \cdots + y_0$. It can be shown that

$$A_4 = ((v_{n-1}(x_{n-1}\text{-}y_{n-1}) + v_{n-2}(x_{n-2} - y_{n-2}) + \ldots + v_0(x_0 - y_0)) \bmod m \quad (4.2)$$

where $v_i \in [0, m)$. The term x_i - y_i may take values 1, 0 or –1. Hence the modulo subtraction and modulo multiplication by a constant in Equation 4.1 have been reduced to the addition/ subtraction of the numbers v_i. This method is used for generating A_4. The next step is to find the product $A_4m_1m_2m_3$ and adding it with X_1, but $A_4m_1m_2m_3$ is $A_4(2^{3n} - 2^n)$. Hence X can generated by adding X_1, $A_4 2^{3n}$ and $-A_4 2^n$. A block schematic of the implementation is shown in Figure 1.

The second method is shown in Figure 2. The only difference from the above is that we extract $|X_1|_{m_4}$ in a different fashion. That is, directly from the residues x_1, x_2 and x_3 using a base extension circuit. The base extension circuit and triple moduli converter are working in parallel as shown in Figure 2. In other words, while performing the triple moduli conversion we can do $|X_1|_{m_4}$ operation using a base extension circuit. The method described in [1] can be used for this. Since here we considered the superset as triple moduli with an extended modulus, the proposed architecture stands good for any extended modulus of the form $2^k \pm 1$, where k is an integer. Of course the set should satisfy the basic relative prime requirement.

An example to illustrate the validity of our proposed technique is given below.

Example 1: Consider the moduli set {15, 16, 17, 31} for n = 4. Here M = 126480. Let the residue representation of number X=5083 be {13, 11, 0, 30}. Decoding the first three residues using a high speed reverse converter we get X_1 = 1003. It has a binary representation of 1111101011. Now $|X_1|_{31} = |0 + 11111 + 01011|_{31} = 11$ and $k = \left|\frac{1}{15.16.17}\right|_{31} = 18$. From Equation 3.2, $A_4 = |(30 - 11)18|_{31} = 1$ and X_2 = 4080. Then, X = 1003 + 4080 = 5083, thus verifying the result.

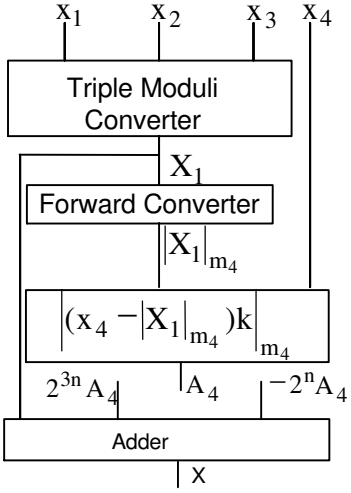

Fig. 1. Memoryless Reverse Converter

5. CONCLUSIONS

In this paper, we proposed an extension to the popular $\{2^n\text{-}1, 2^n, 2^n+1\}$ moduli set by adding a fourth modulus 2^{n+1}-1. The extension leads to higher parallelism while retaining the forward conversion and residue arithmetic units simple. The first method is totally memoryless and fully arithmetic based. The pipelined throughput rate is 3n+1 adder delays which is same for both the methods described.

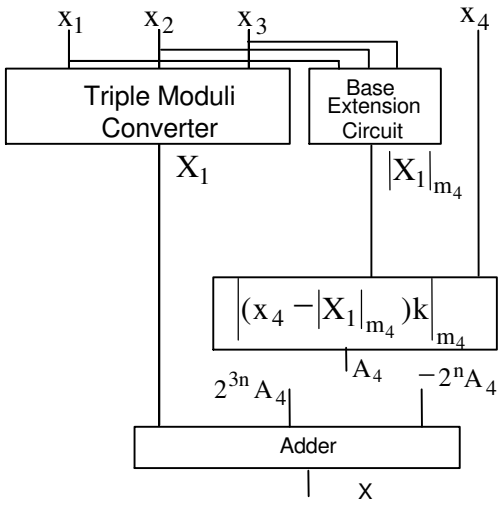

Fig. 2. Reverse Converter using Base Extension

6. REFERENCES

[1] Szabo N.W. and Tanaka R.I., *Residue Number Arithmetic and its Application to Computer Technology.* New York: McGraw Hill, 1967.

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

[3] Piestrak S.J., "A High Speed Realization of Residue to Binary Number System Conversion," IEEE Trans. Circuits Syst.-II, vol. 41, no. 12, pp. 661-663, Dec. 1994.

[4] Bi G. and Jones E.V., "Fast conversion between binary and residue numbers," Electronic Letters, vol. 24, no. 19, pp. 195-1197, Sept. 1988.

[5] Ibrahim K.M. and Saloum, "An Efficient Residue-to-binary Converter Design," IEEE Trans. Circuit and Syst. AS, vol. 35, no. 9, pp. 1156-1158, Sept. 1988.

[6] Piestrak S.J., "Design of Residue Generators and Multioperand Modular Adders Using Carry Save Adders," IEEE Trans. Comput., vol. 423, no. 1, pp. 68-77, Jan. 1994.

[7] Bhardwaj M., Premkumar A.B. and Srikanthan T., "Breaking the 2n-bit Carry Propagation Barrier in Reverse Conversion for the $\{2^n-1, 2^n, 2^n+1\}$ Moduli Set," IEEE Trans. Circuits and Syst.-I, vol. 45, no. 9, Sept. 1998.