Fault-Tolerant Linear Convolution Using Residue Number Systems

by

M.G.Parker,M.Benaissa

The authors are with:

School of Eng., Univ.of Huddersfield, Huddersfield HD1 3DH, UK.,

Phone - 0484 422288 Ext 2385, E-Mail M.Parker@eng.hud.ac.uk

Fault-Tolerant Linear Convolution Using Residue Number Systems

Abstract

This paper proposes a Fault-Tolerant Linear Convolution architecture using Residue Number Systems (RNS) and Polynomial Residue Number Systems (PRNS). The RNS and PRNS are both given error-detection capability by the addition of redundant residue channels, and the combined redundancy enables errors to be corrected without explicit error-decoding. The method is simple, fast, and amenable to VLSI implementation.

1 Introduction

Fault-tolerance (FT) within VLSI systems is to be encouraged as increasing VLSI design complexity heightens the likelihood of component or module failure. The simplest form of FT uses Modular Redundancy (MR) where multiple replication of a component or module can identify system error (by o/p comparison), and extract the most probable correct o/p (by rejecting the 'odd one out'). However, this technique is costly in terms of area, a protected system requiring at least three times the area of an unprotected system. More sophisticated error-correction procedures have been developed. The use of Reed-Solomon (RS) codes [2] to protect data at the digit (or bit) level, has gained popularity in recent years. Another means of FT uses Residue Number Systems (RNS) [4]. A RNS performs computation over a number of independent residue channels, where the computation within each channel is performed over a different modulus. As the channels are independent, any fault occuring within the system does not permeate throughout the whole system but affects only the channel in which it ocurred. This makes the fault(s) easier to detect and correct. A scheme similar to MR is therefore applicable using RNS by the addition of extra, redundant, residue channels, though the extra hardware required is a fraction of that required for MR.

Linear Convolution (LC) is of great importance in Signal Processing applications [1, 3], forming the hub of filtering and correlation operations. Hence, it's implementation within VLSI systems is a frequent requirement. It can be computed efficiently by using a Polynomial Residue Number System (PRNS) [5], (note, PRNS is a direct extension of RNS into polynomial arithmetic). In a recent paper, Beckmann and Musicus [6] have proposed the addition of extra, redundant, polynomial residue channels within a PRNS LC to detect and correct errors. Error-correction is achieved by an exhaustive search for a possible set of valid residues. This is certainly effective. However, it requires a large amount of post-processing. An alternative correction method, suggested in [6], is an estimation method for a special choice of channel moduli. This is faster though sensitive to noise.

In this paper we propose a single error-correcting LC system similar to that described in [6]. However, whereas they emphasise LC in the complex field, we perform LC in a finite integer ring, mod M. This allows us to decompose the PRNS LC using RNS. By adding one extra PRNS channel, a single PRNS channel error is detectable. If we also have one extra RNS channel, we can ignore the o/p from the errored PRNS channel and still have enough RNS residues to compute the required LC result. Thus the error-decoding task, tackled in [6], is averted, at the price, (for single error-correction), of one extra RNS channel, and one <u>less</u> PRNS channel. An example is given and it is shown how the method can be used to correct more than one error. The primary advantages of this system are it's immediate and simple error correction capability and the small size and independent nature of its computing elements.

2 RNS and PRNS

2.1 RNS

Let us perform the linear operation,

$$\langle f(a,b)\rangle_M \qquad 0 \le a,b < R \tag{1}$$

where $R \le M$ is the i/p dynamic range, and $\langle * \rangle_M$ means the residue of $*$, mod M.

We can write,

$$M = \prod_{i=0}^{n-1} m_i^{w_i} \tag{2}$$

where the $m_i^{w_i}$ are mutually prime, and n and w_i are positive integers.

A RNS solution to (1), using n independent channels, is as follows

$$f_i(a,b) = \left\langle f(\langle a\rangle_{m_i^{w_i}}, \langle b\rangle_{m_i^{w_i}})\right\rangle_{m_i^{w_i}} \qquad 0 \le i < n \tag{3}$$

We can reconstruct $f(a,b)$, using the Chinese Remainder Theorem (CRT),

$$f(a,b) = \left\langle \sum_{i=0}^{n-1} f_i(a,b).M_i.\left\langle M_i^{-1}\right\rangle_{m_i^{w_i}} \right\rangle_M \tag{4}$$

where,

$$M_i = M/m_i^{w_i} \tag{5}$$

We note that if $f(a,b) < M$ for all a and b,

$$f(a,b) \equiv \langle f(a,b)\rangle_M \tag{6}$$

We have embedded an integer function in a modular ring in order to use RNS without affecting the o/p.

Theorem 1 *$f(a,b)$ can be computed using n residue channels if the product of the residue moduli, M, is $> f_m$, (where $f_m = max(f(a,b))$).*

If, from (5), $M_i \le f_m$, for all i, then the RNS is non-redundant. That is, all n channels are vital to the computation of $f(a,b)$.

Adding one extra residue channel,

$$M^{+1} = \prod_{i=0}^{n} m_i^{w_i} \tag{7}$$

We can define the product of any n of these channel moduli as,

$$M_i^{+1} = M^{+1}/m_i^{w_i} \qquad \text{for } 0 \le i < n+1 \tag{8}$$

If $M_i^{+1} > f_m$, for all i, then the RNS contains one redundant residue channel.

Theorem 2 *$f(a,b)$ can be computed using $n+1$ residue channels, and if all possible combinations of n out of $n+1$ channels can also compute $f(a,b)$, the RNS can detect a single channel error. This error is detected when $f(a,b)$ is computed $> f_m$ using all $n+1$ channels.*

Adding further RNS channels enables more errors to be detected and some to be corrected.

2.2 PRNS

PRNS is a direct extension of RNS to the polynomial domain. For clarity, we re-state the arguments used for RNS in their polynomial form.

Consider computation over a polynomial modulus $M(x)$, of degree N, ($\deg(M(x)) = N$), where all polynomial coefficients are in a field, (or ring), F which can be finite or infinite. Let us perform the linear operation,

$$\langle g(a(x), b(x)) \rangle_{M(x)} \tag{9}$$

where $a(x)$ and $b(x)$ are polynomials of degree $R-1$ or less, with $R \leq N$.

We can write,

$$M(x) = \prod_{i=0}^{n-1} m_i(x)^{v_i} \tag{10}$$

where $m_i(x)^{v_i}$ are mutually prime, and n and v_i are positive integers.

A PRNS solution to (9). using n independent channels, is as follows,

$$g_i(a(x), b(x)) = \left\langle g(\langle a(x) \rangle_{m_i(x)^{v_i}}, \langle b(x) \rangle_{m_i(x)^{v_i}}) \right\rangle_{m_i(x)^{v_i}} \qquad 0 \leq i < n \tag{11}$$

We can reconstruct $g(a(x), b(x))$, using the Chinese Remainder Theorem for Polynomials (CRTP),

$$g(a(x), b(x)) = \left\langle \sum_{i=0}^{n-1} g_i(a(x), b(x)).M_i(x).\left\langle M_i(x)^{-1} \right\rangle_{m_i(x)^{v_i}} \right\rangle_{M(x)} \tag{12}$$

where,

$$M_i(x) = M(x)/m_i(x)^{v_i} \tag{13}$$

If $\deg(g(a(x), b(x))) < \deg(M(x))$ for all $a(x)$ and $b(x)$,

$$g(a(x), b(x)) \equiv \langle g(a(x), b(x)) \rangle_{M(x)} \tag{14}$$

We have embedded a polynomial function in a polynomial ring in order to use PRNS <u>without</u> affecting the o/p.

Theorem 3 *$g(a(x), b(x))$ can be computed using n polynomial residue channels if the product of the residue moduli, $M(x)$, has degree $> g_m$, (where g_m = max(deg($g(a(x), b(x))$))).*

If, from (13), $\deg(M_i(x)) \leq g_m$, for all i, then the PRNS is non-redundant.

Adding one extra residue channel,

$$M^{+1}(x) = \prod_{i=0}^{n} m_i(x)^{v_i} \tag{15}$$

with,

$$M_i(x)^{+1} = M^{+1}(x)/m_i(x)^{v_i} \qquad 0 \leq i < n+1 \tag{16}$$

If $\deg(M_i(x)^{+1}) > g_m$, for all i, then the PRNS contains <u>one</u> redundant residue channel.

Theorem 4 *$g(a(x), b(x))$ can be computed using $n+1$ residue channels, and if all possible combinations of n out of $n+1$ channels can also compute $g(a(x), b(x))$, the PRNS can detect a single channel error. This error is detected when deg($g(a(x), b(x))$) is computed $> g_m$ using all $n+1$ channels.*

Adding further PRNS channels enables more errors to be detected and some to be corrected.

The above analysis for RNS and PRNS is equally valid for functions of more or less than two variables.

In the next section we apply RNS and PRNS to the Linear Convolution (LC) operation and show how, by using the single error detection capability of both RNS and PRNS, a single channel error may be corrected.

3 Linear Convolution (LC) Using RNS and PRNS

Consider,

$$c(x) = g(a(x), b(x)) = a(x).b(x) \tag{17}$$

where we define $a(x)$ as follows,

$$a(x) = \sum_{k=0}^{R_p-1} a_k.x^k \qquad 0 \leq a_k < R \tag{18}$$

with $0 \leq a_k < R$. (Similarly for $b(x)$). $c(x)$ is,

$$c(x) = \sum_{j=0}^{2.R_p-2} c_j.x^j \tag{19}$$

This polynomial multiplication is equivalent to a LC of the coefficients of $a(x)$ and $b(x)$ of o/p blocklength $= 2.R_p - 1$. Hence we can define the LC as a function of $2 \times R_p$ integer variables of input dynamic range $< R$. Thus,

$$c_j = f_j(a_0, a_1, \ldots, a_{R_p-1}, b_0, b_1, \ldots, b_{R_p-1}) = \sum_{k=0}^{R_p-1} a_{j-k}.b_j \quad 0 \leq j < 2.R_p - 2 \tag{20}$$

where $a_j, b_j = 0$ for $0 > j \geq R_p - 1$.

We observe that we can embedd the polynomial multiplication of (17) in a PRNS whilst embedding the equivalent LC of the polynomial coefficients, (20), in a RNS. To determine the minimum number of residues required to fulfill the RNS and PRNS without reducing the o/p, we refer to Theorems (1) and (3) and determine the maximum possible integer o/p dynamic range, and polynomial o/p degree. Thus, to satisfy (6),

$$c_j \equiv \langle c_j \rangle_M \qquad \text{for } 0 \leq j < 2.R_p - 2 \tag{21}$$

where, from (20),

$$M > R_p.(R-1)^2 \qquad (\text{we assign } f_m = R_p.(R-1)^2) \tag{22}$$

and, to satisfy (14),

$$c(x) \equiv \langle c(x) \rangle_{M(x)} \tag{23}$$

where, from (17) and (19),

$$\deg(M(x)) > 2.R_p - 2 \quad \text{(we assign } g_m = 2.R_p - 2) \tag{24}$$

Given these lower limit restrictions on M and $M(x)$, we can always embedd an integer LC in a RNS and PRNS. We show the general RNS/PRNS LC in Fig 1. Note the array of relatively small, independent, processing elements (PEs). The granularity of the array depends on (2) and (10), i.e. the number of integer and polynomial residue channels. This highlights a particularly important criteria for the feasible implementation of LC using RNS/PRNS:

$M(x)$ will only be highly factorisable (hf) if its coefficients are defined over a large enough field (or ring). From (2), we require $M(x)$ hf over each of $m_i^{w_i}$ in turn. If $M(x)$ is to be fully factorised into degree-one factors, with all $v_i = 1$, we can state, without proof, the following,

Theorem 5 *For $M(x)$ to factorise into degree-one factors over $m_i^{w_i}$, deg($M(x)$) must be $< m_i$*

3.1 Example

We wish to implement the LC of two data blocks of length 29. We represent the LC as a polynomial multiplication where $a(x)$ and $b(x)$ are of degree 28. $R_p = 29$ and the maximum possible degree of the polynomial product is therefore $g_m = 56$. (i.e. a 57 blocklength LC). We can embedd this polynomial product in a PRNS, where $\deg(M(x)) > 56$. If the i/p dynamic range of the polynomial coefficients is given by $R = 92$, then the maximum possible o/p polynomial coefficient size is, $f_m = 240149$.

The LC can also be embedded in a RNS when M is chosen such that $M > f_m$. We note that choosing $m_0 = 59$, $m_1 = 61$ and $m_2 = 67$, (with all $w_i = 1$), specifies M as,

$$M = m_0.m_1.m_2 = 241133 > f_m$$

.

For no redundancy, $\deg(M(x))$ must be chosen $= 57$. m_0, m_1 and m_2 allow full factoristation of $M(x)$ into 57 degree-one factors, as 57 mutually prime elements exist in each of m_0, m_1 and m_2. Thus a 3×57 RNS/PRNS array of residue PEs performs the LC.

Other cases can be analysed in a similar fashion.

In the next section we show that the addition of single error detection capability for both RNS and PRNS allows a simple form of single error correction.

4 Fault Tolerant RNS/PRNS LC

4.1 Single Error-Correction

If, from (5) and (13), the RNS and PRNS satisfy the following, respectively,

$$M_i \leq f_m \qquad \deg(M_i(x) \leq g_m) \tag{25}$$

then the RNS/PRNS LC is non-redundant. Let us add one extra residue channel to each of the RNS and PRNS (RNS:1,PRNS:1), such that theorems 2 and 4 are satisfied. Thus both the PRNS and RNS can detect one channel error. Fig 2 shows this 'expanded' system. If we now take an example of a single PE failure. The following chain of events takes place,

1. One PE failure in Row 2, Column 2 of Fig 2.
2. CRTP_1 in Row 2 detects o/p of degree $> g_m$. (i.e. highest coefficient non-zero).
3. Row 2 scrapped.
4. CRT reconstructs correct LC o/p for each coefficient, independently, using the polynomial row residues, but ignores Row 2.

Hence, single error-correction is achieved Note, the CRTP and CRT modules are <u>not</u> included in the RNS/PRNS protection scheme and must be protected separately.

4.1.1 Example

To continue the example of the previous section. Let us add $m_3 = 71$ to give $m_0 = 59$, $m_1 = 61$, $m_2 = 67$ and $m_3 = 71$. We note that, using these four residue moduli, the RNS has single redundancy. We can add another degree-one polynomial to produce $M^{+1}(x)$, a fully factorisable degree 58 polynomial, giving single PRNS redundancy. (Note, each of the four m_i possesses at least 58 mutually prime elements, enabling $M^{+1}(x)$ to be fully factorisable). Thus a 4×58 RNS/PRNS array of residue PEs performs the LC.

4.2 Multiple Error-Correction

4.2.1 RNS:2,PRNS:1

Let us add a second RNS residue so that two redundant RNS residues exist. This scheme detects and corrects up to two PE failures <u>as long as</u> the PEs are not in the same row. Correction is achieved by ignoring the Row residues from the two Rows errored by the PRNS. The RNS can 'afford' to ignore up to two residues in each column.

4.2.2 RNS:2,PRNS:2

We now have two redundant residues for each of RNS and PRNS. The PRNS can detect up to two errors per row. Hence, the system can afford up to two PE failures.

4.2.3 RNS:t,PRNS:t

This system can correct up to t errors.

4.3 Area Assessment

If a non-redundant system uses n RNS channels and n_p PRNS channels, and up to t errors are to be corrected, then we can state the approximate area increase for protection as the ratio,

$$\mathrm{O}((n+t).(n_p+t)/n.n_p) \tag{26}$$

Clearly, the more residues there are, the larger n and n_p are. Large n and n_p implies a small percentage rise in area cost. This scheme is more effective for small t. For high t, explicit error-decoding searches may be more appropriate.

5 Conclusion

In this paper we have presented an architecture using simultaneous RNS and PRNS decomposition to compute a Linear Convolution. We have stated the conditions necessary for system redundancy and used this redundancy to detect single errors in either RNS or PRNS. Combining the detection capabilities of RNS and PRNS allows us to correct a single processing element failure. We then showed how increased fault tolerance can be achieved by adding even more RNS and PRNS residue channels. The scheme is regular and fast, requiring a relatively small area overhead for a given fault tolerance.

References

[1] J.H.McClellan,C.M.Rader, **Number Theory in Digital Signal Processing,** Prentice Hall, '79

[2] R.E.Blahut, **Theory and Practice of Error Control Codes,** Reading, MA: Addison Wesley, '84

[3] R.E.Blahut, **Fast Algorithms for Digital Signal Processing,** Reading, Addison-Wesley, '85

[4] M.A.Soderstrand,W.K.Jenkins,G.A.Jullien, F.J.Taylor, **Residue Number System Arithmetic: Modern Applications in Digital Signal Processing**, IEEE Press, New York, NY '86

[5] A.Skavantzos,N.Mitash, "Computing Large Polynomial Products using Modular Arithmetic," *IEEE Trans on Circuits and Systems - II*, Vol 39, No 4, pp 252 - 254, April '92

[6] P.E.Beckmann,Bruce.R.Musicus, "Fast Fault-Tolerant Digital Convolution Using a Polynomial Residue Number System," *IEEE Trans on Signal Processing*, Vol 41, No 7, pp 2300 - 2313, July '93

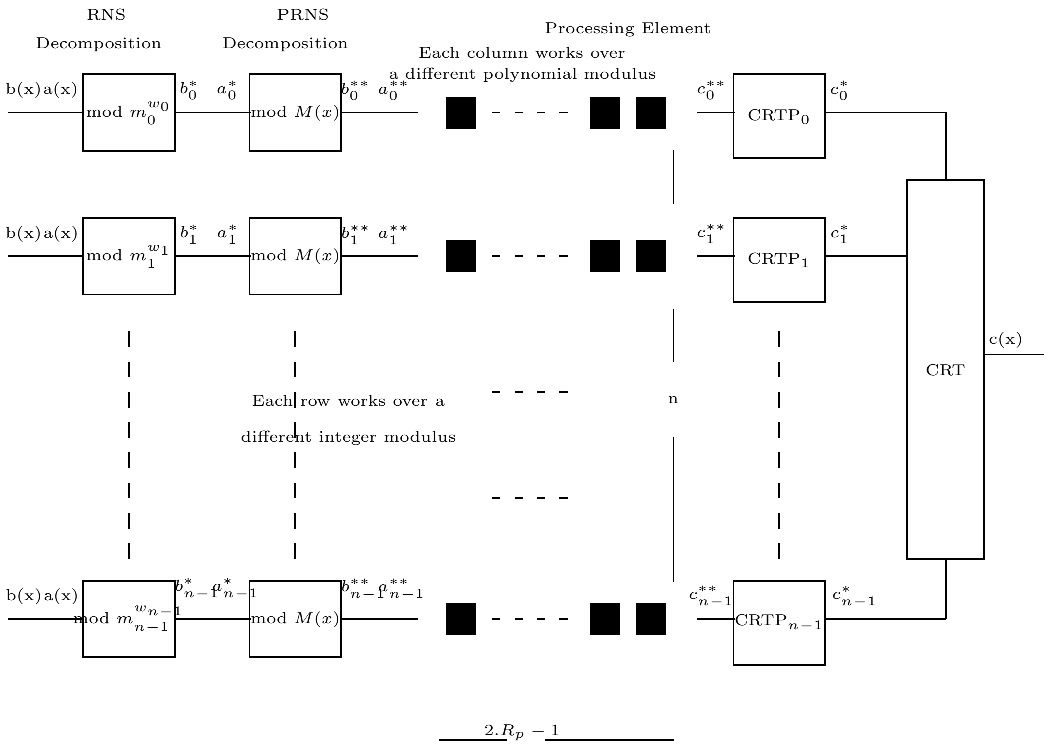

Figure 1: Linear Convolver Using RNS and PRNS

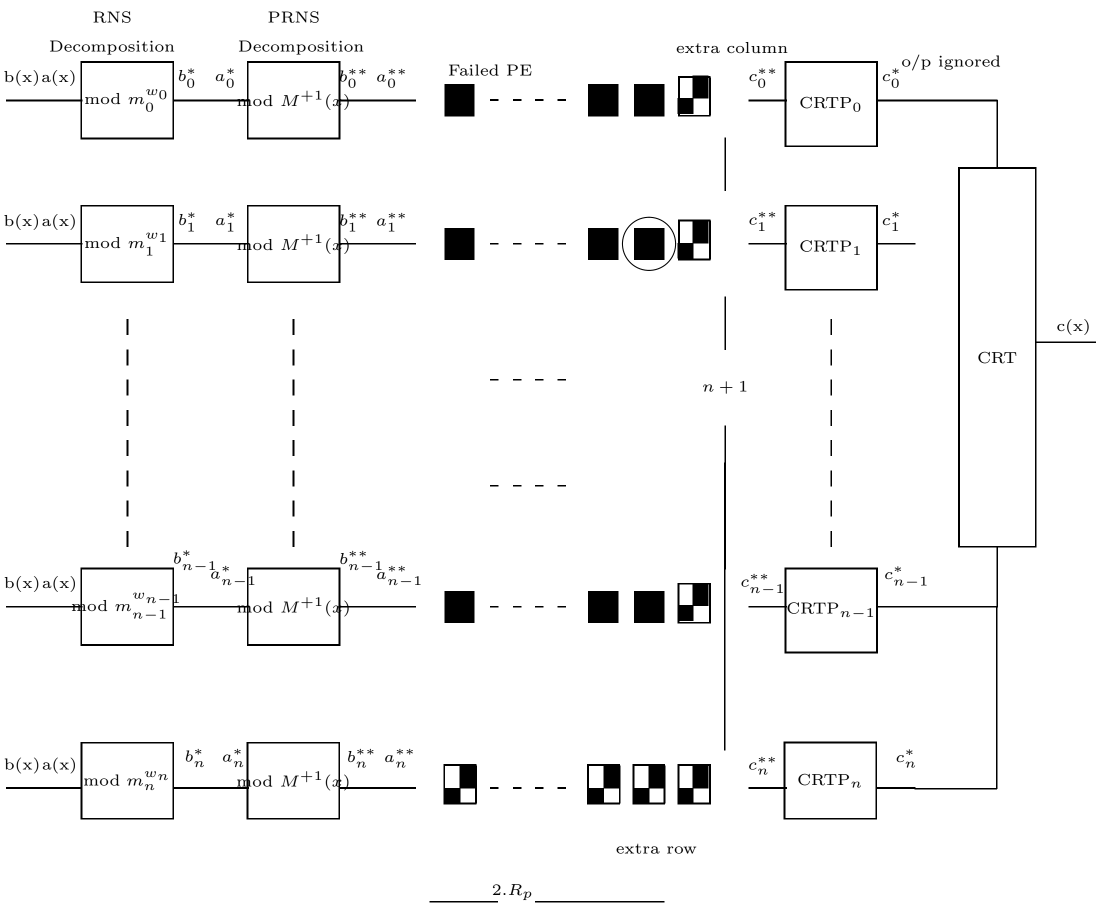

Figure 2: Linear Convolver Using RNS and PRNS with Single Error Correction