Efficient Fault-Tolerant Arithmetic using a Symmetrical Modulus Replication RNS

Laurent Imbert and Graham A. Jullien

ATIPS Laboratory, Dept. Electrical and Computer Engineering, University of Calgary
2500 University Drive NW, Calgary, AB Canada, T2N 1N4
jullien@enel.ucalgary.ca

Abstract: *This article presents a novel fault-tolerant technique which is based on the previously introduced Modulus Replication Residue Number System (MRRNS). Using the MRRNS mapping technique, we are able to compute using modular arithmetic over identical channels. We can make this system fault tolerant by increasing the number of channels, but by using some elementary polynomial properties we are able to obtain the same level of fault tolerance with an almost 50% decrease in the number of channels. This new system is referred to as the Symmetric MRRNS (SMRRNS), and compared to previously proposed fault-tolerant techniques, the overhead of the SMRRNS system is very small.*

1. Introduction

Many digital signal processing algorithms require large inner-product computations to be performed in real-time (i.e., output data generated at the same rate as the input data are received). Even with current technology, there is a requirement for massively parallel implementations of inner-products in order to be able to achieve real-time processing with high data rates (say, 10^8 samples/sec or higher). There are many methods available to break the computation into parallel sections, both at the algorithmic [1][2] and arithmetic levels. A popularly published technique for parallel arithmetic decomposition is the use of the residue number system (RNS) which allows independent computations over modular computation channels using relatively prime moduli [3]. This technique has been known to adapt readily to a fault-tolerant structure using redundant channels with moduli that are relatively prime to the non-redundant channels (this has been referred to as the Redundant RNS [4]). This technique provides fault tolerance over arithmetic computations, a feature not shared with even the most powerful of modern error correcting codes [5].The major problem with the Redundant RNS technique is the need to build parallel computational channels that are not identical (i.e., compute over different moduli). This has limitations if we require a system, say, where unused channels may be substituted for faulty channels; a powerful technique for increasing the yield of ASIC devices.

There are certainly advantages to be gained by computing over finite rings or fields, and the RNS provides potential benefits when this is taken into account. Another technique is available, however, which allows algorithms to be decomposed into independent residue computations over

identical channels; this has been referred to as the Modulus Replication RNS (MRRNS) [6][7]. In the MRRNS technique, we map the input data samples into polynomials based on their binary representation and a set of indeterminates based on pre-chosen powers of two. Using Lagrange interpolation, we map the *n*th order polynomials into *n*+1 distinct points with which we can compute (add and multiply) in a component-wise fashion.The MRRNS technique, therefore, appears to offer similar advantages to the RNS without the major disadvantage of having to build non-identical channels.

We have recently addressed the problem of fault-tolerant computation over the MRRNS using a similar approach to the Redundant RNS, by adding redundant, but identical, computational channels [8]. In this paper we investigate efficiencies that may be introduced into the fault-tolerant MRRNS system by restricting the data sample polynomials to be even. We refer to this as the Symmetrical MRRNS (SMRRNS) technique.

2. The MRRNS system

The MRRNS [6] system uses the fact that any univariate polynomial of degree *n* can uniquely be represented by its value at *n+1* distinct points. This comes from the following theorem [9].

Theorem 1[Lagrange Interpolation]:

Let $x_0, \ldots, x_n$ be a set of *n*+1 distinct points, and let $y_0, \ldots, y_n$ be given. Then, *p* is the only polynomial of degree *n* or less that has value $p(x_k) = y_k$ for $k = 0, \ldots, n$. It is given by the formula:

$$p(z) = \sum_{i=0}^{n} y_i \frac{\prod_{j \neq i} (z - x_j)}{\prod_{j \neq i} (x_i - x_j)} \tag{1}$$

❑

The MRRNS technique uses theorem 1 to generate *n*+1 points for each of the input data polynomials. The computational task, for example a large inner-product, is decomposed as follows: input data are first mapped into polynomials, which are evaluated at a convenient number of distinct points. It is important that the number of points must be large enough, not only to represent the input polynomials, but more importantly the result of the computation. Then additions, subtraction and multiplications are performed over completely independent channels. Actually, if *p, q* are two polynomials then, for all *x*, we have

$$\begin{aligned} (p+q)(x) &= p(x) + q(x) \\ pq(x) &= p(x)q(x) \end{aligned} \tag{2}$$

Moreover, on each of these independent channels, operations are not performed over the whole integer ring, but either over a direct sum of several small integer rings as in the classical RNS, or over a finite field GF(*p*). Any prime number can be used to define the finite field, but Fermat primes, i.e. numbers of the form $2^{2^n} + 1$, offer great advantages for modular multiplication. The most important advantage of the MRRNS compared to the classical RNS, is that all the channels are perfectly identical, and does not depend on a specific moduli. An interpolation technique is used to recover the final polynomial and a final evaluation step gives the result.

Let us consider the following example, where we compute the result modulo 257 (the 4*th* Fermat prime).

Example 1: To compute the inner-product $79 \times 47 + 121 \times 25 = 6738$ using the MRRNS technique, we first define the polynomials $p_1(x)$, $q_1(x)$, $p_2(x)$ and $q_2(x)$ that correspond to the inputs values 79, 47, 121 and 25, assuming the indeterminate $x = 8$. Let $p_1(x) = -1 + 2x + x^2$, $q_1(x) = -1 + 6x$, $p_2(x) = 1 + 7x + x^2$ and $q_2(x) = 1 + 3x$. Since the degree of the final polynomial is 3, theorem 1 implies that we must evaluate each of these polynomials at $n = 4$ distinct points. Let us use the set $S = \{-1, 1, -2, 2\}$, and assume that the coefficients of the final polynomial belong to the set $\{-128, \ldots, 128\}$, which allow us to perform all the computations over the finite field *GF(257)*. Evaluating $p_1(x)$, $q_1(x)$, $p_2(x)$ and $q_2(x)$ at each point of *S* gives the vectors $u_1 = (-2, -2, -1, 7)$, $v_1 = (-7, 5, -13, 11)$, $u_2 = (-5, 9, -9, 19)$ and $v_2 = (-2, 4, -5, 7)$. According to equations (2), we perform the componentwise products and sum (modulo 257) to obtain $w_1 = u_1 \otimes_{257} v_1 = (14, 10, 13, 77)$, $w_2 = u_2 \otimes_{257} v_2 = (10, 36, 45, -124)$ and $w = w_1 \oplus_{257} w_2 = (24, 46, 58, -47)$. An interpolation algorithm, such as Newton's divided differences [10], is used to recover the final result:

$$r(x) = (-1 + 2x + x^2)(-1 + 6x) + (1 + 7x + x^2)(1 + 3x) = 2 + 2x + 33x^2 + 9x^3 \qquad (3)$$

which is evaluated at $x = 8$ to obtain the result of our inner-product $r(8) = 6738$.

In this example, we have used the embedded finite field *GF(257)*. Another solution would have been to define a classical RNS for each channel using the direct product of several small integers, such as $Z_3 \times Z_5 \times Z_7$ [7].

3. Adding fault-tolerance to the MRRNS

To make a computation fault-tolerant we need to add some redundancy. In the MRRNS, this can be done by adding extra-channels. Since the MRRNS system uses replicated identical channels, the redundant channels are also identical. This is a major advantage over using a redundant RNS

system [11][4], where all of the channels are different from one another. In the previous example, in order to obtain a polynomial of degree 3, we have used vectors of length 4, the minimum required value. By adding extra-channels, that is by evaluating the input polynomials at more points than the minimum required, we can build a system which can allow us to detect and correct errors. A similar technique was proposed in [11] using the polynomial residue number system, but we shall see in section 4 that the MRRNS offers advantages which can provide a much more efficient fault-tolerant system.

In the following, we present the properties of a basic fault-tolerant MRRNS system and provide important theorems for error detection and error correction.

3.1 Single fault detection

In the following we will consider all numbers as polynomials. We shall denote $y_k = p(x_k)$, the value of the polynomial *p* at the point $x = x_k$ and refer to it as a vector when represented by its values $(y_0, y_1, \ldots, y_n)$.

Definition 1: If *u, v* are two vectors of elements in any finite ring *R*, we denote *d*(*u,v*) the distance between *u* and *v* as the number of coordinates in which *u* and *v* differ. It is easy to prove that *d* is a metric on *R*, that is, for all $u, v, w \in R$ we have: $d(u, v) = 0$ iff $x = y$; $d(u, v) = d(v, u)$; and $d(u, w) \leq d(u, v) + d(v, w)$.

Thus, if *u* is the correct result and *v* is the vector obtained at the end of the computations, *d(u,v)* give the number of errors that have occurred. If we represent the result as $v = u + e$, the number of error is also given by *w*(*e*), the number of non-zero elements of the error vector *e*. These definitions can be seen as an extension of the *Hamming distance* and *Hamming weight* used in classical coding theory [5].

We see that if we expect the result to be a polynomial of degree *n*, we need at least *n*+1 distinct points. In order to detect a single error, we now considering a set of *n*+2 distinct points, leading to *n*+2 independent channels. The idea is quite simple. Since *n*+1 points are sufficient to represent polynomials of degree *n*, and since we are expecting a result of degree at most *n*, the highest order coefficient of the interpolated polynomial must clearly be equal to zero if no error has occurred during the computations. Otherwise, we can conclude that at least one error has occurred. The following theorem clarifies this error detection property.

Theorem 2: Let *u* and *v* be the correct and computed vectors respectively, p_u and p_v the corresponding polynomials, and assume that $d(u, v) \leq 1$. If $\deg(p_v) = n$ then $p_v = p_u$ is the correct result. Otherwise, $\deg(p_v) > n$ and we can conclude that at least one error has occurred.

Proof: Let $x_0, \ldots, x_{n+1}$ be the set of n+2 distinct points, and let $v = (v_0, \ldots, v_{n+1})$ be the computed vector. From theorem 1, p_u is the unique polynomial of degree n or less that has value $u_0, \ldots, u_{n+1}$. Then, if $\deg(p_v) = n$, it is clear that $p_v = p_u$. Otherwise, under the assumption that $d(u, v) \leq 1$, there exists a unique $0 \leq j \leq n + 1$, such that $v_j \neq u_j$ and $v_i = u_i$, for $i \neq j$. The polynomial p_v is at least of degree n since it has value $v_i = u_i$ for the subset of n+1 distinct points $S' = S - \{x_j\}$. Hence, since it differs from u at position j, its degree $\deg(p_v) \geq n$.

❑

3.2 Single fault correction

The same idea is used to correct a single error. By adding two extra-channels, we can now either detect two errors, or detect and correct one error. Again, this error-correction property comes from theorem 1. The following example illustrates the fault detection/correction algorithm.

Example 2: Let us compute the product of the polynomials $p(x) = 1 = 2x + 3x^3$ and $q(x) = 2 - x$. Since the result, $r(x)$, is a polynomial of degree 3, we consider a set of $4 + 2 = 6$ distinct points. Let $S = \{-4, -2, -1, 1, 2, 4\}$. Evaluating p and q at each of these points gives the vectors $u = (57, 17, 6, 2, 9, 41)$ and $v = (6, 4, 3, 1, 0, -2)$. We compute the product $w_k = u_k \times v_k$ for $k = 0, \ldots, 5$, which gives $w = (342, 68, 18, 2, 0, -82)$. The interpolation step then computes $r(x) = 2 - 5x + 8x^2 - 3x^3 + 0x^4 + 0x^5$, which is the correct result. Note the zero higher order coefficients for a correct result.

Now, let us assume that one error has occurred, say on channel 2, and that $w = (342, -\mathbf{71}, 18, 2, 0, -82)$. Since we have no idea in which channel the error occurred, we independently eliminate each of them one by one and interpolate the reduced vector corresponding to each of the 6 subsets of 5 points of S. For the first channel, this leads to the fourth order polynomial $46 - 38x + 98x^2 + 30x^3 + 123x^4$. Since this is higher than third order, we know that an error has occurred. The same technique is applied to the 5 other channels and the only polynomial of degree 3 (the only correct computation) is given by removing the second channel (the channel in error).

3.3 Generalization

The previous results for single error detection and correction can easily be generalized. The following theorem holds.

Theorem 3: If we assume that the result of the computation(s) is a polynomial of degree at most *n-1*, and if $d \geq c$ are two positive integers, then a MRRNS system composed of $n + d + c$ channels is able to detect up to d and correct up to c errors.

Proof: The detection part of the proof is directly derived from the proof of theorem 2. For the correction, the proof comes from the fact that by removing a subset of c channels, one are provided with a set of $n + d$ points able to detect up to d errors. Actually, if we remove c correct channels, the vector we interpolate with respect to the subset of $n + d$ corresponding points still contains c channels in error. Since $c \leq d$, all of them are detected.

❑

In the following section, we present an efficient mapping technique that leads to a very low overhead fault-tolerant system.

4. The Symmetrical-MRRNS mapping technique

The SMRRNS is an extension of the MRRNS which uses polynomial parity properties by mapping the input values into even polynomials only. It uses the fact that the result of any computation involving only additions, subtractions and multiplications is also an even polynomial. Let p, q be two polynomials such that $p(-x) = p(x)$, $q(-x) = q(x)$, then

$$\begin{aligned} (p \pm q)(-x) &= (p \pm q)(x) \\ pq(-x) &= pq(x) \end{aligned} \tag{4}$$

In the SMRRNS, we can take advantage of these properties by choosing the set S such that

$$S = \{x_0, \ldots, x_n\} \qquad \text{with } |x_i| \neq |x_j| \qquad \forall i \neq j. \tag{5}$$

If we assume that the result of the computational task is the vector $w = (w_0, \ldots, w_n)$ then, since r, the interpolated polynomial corresponding to w, is an even polynomial, $r(-x) = r(x)$ for all $x \in S$. If $0 \notin S$, this allows us to extend w to the vector:

$$w' = (w_0, w_1, \ldots, w_{n-1}, w_n, w_n, w_{n-1}, \ldots, w_1, w_0). \tag{6}$$

If $0 \in S$, and assuming that w_n corresponds to $r(0)$, we have:

$$w' = (w_0, w_1, \ldots, w_{n-1}, w_n, w_{n-1}, \ldots, w_1, w_0). \tag{7}$$

The vectors in (6) and (7) corresponds to those we obtain if we evaluate our polynomials at each point of the extended set $S' = \{x_0, \ldots, x_n, -x_n, \ldots, -x_0\}$. Using theorem 1, the final polynomial is then computed from this extended vector w' with respect to the extended set S'.

In the following we will assume, without loss of generality, that 0 belongs to the set S and that it corresponds to the value w_n. Using this mapping technique, n channels in the SMRRNS are

almost equivalent to a classical MRRNS composed of $2n + 1$ channels. From theorem 3, we see that these *n* extra-channels can be used to detect and correct up to $\lfloor n/2 \rfloor$ errors. For example, if we expect the result to be a fourth order polynomial, we need only compute over 5 channels, and obtain a fault-tolerant system which is able to detect and correct up to 2 errors.

If we assume that only 1 error has occurred, say in channel *k*, then the value w_k is in error, and since $w'_k = w'_{2n-k}$, both are incorrect in *w'* (if w_n is the channel in error then *w'* contains only one error: w'_n). The correcting algorithm consists of removing one by one all the pairs of values (w'_k, w'_{2n-k}) for $k = 0, \ldots, n$, and using the resulting vector to compute a polynomial according to the corresponding subset of *S'*. This is best explained using the following example.

Example 3: Let us compute the product $324 \times 179 = 57996$ with the ability to correct one error. We first map the values into even polynomials assuming the indeterminate $x = 8$. Let

$$\begin{aligned} p(x) &= 5x^2 + 4 \\ q(x) &= 5x^2 - 13 \end{aligned} \tag{8}$$

Since the result $r(x) = p(x) \times q(x)$ is of degree 4, we need at least 5 points. According to eqn. (5), we define $S = \{8, 4, 2, 1, 0\}$. We also assume that the coefficients of the final polynomial belong to the set $\{-128, \ldots, 128\}$, which allows us to perform all the computations over the finite field *GF(257)*.

The evaluation of *p* and *q* (modulo 257) at each points of *S* gives the vectors:

$$\begin{aligned} u &= (67, 84, 24, 9, 4) \\ v &= (-78, 35, -1, -10, -13) \end{aligned} \tag{9}$$

and the componentwise product $w = u \otimes_{257} v$ gives:

$$w = (-86, 113, -24, -90, -52) \tag{10}$$

This vector is interpolated with respect to *S* to obtain the even polynomial $r(x) = 15x^4 - 53x^2 - 52$, and evaluated at $x = 8$ to compute the final result $r(8) = 57996$.

Let us now assume that one error has occurred, say on channel 2, and that we have obtained the vector

$$err = (-86, \mathbf{120}, -24, -90, -52) \tag{11}$$

The extended vector

$$err' = (-86, \mathbf{120}, -24, -90, -52, -90, -24, \mathbf{120}, -86) \quad (12)$$

now contains 2 errors that are easily detected. By interpolating eqn. (12) according to the set $S' = \{8, 4, 2, 1, 0, -1, -2, -4, -8\}$, we obtain the eighth order even polynomial $109x^8 - 68x^6 + 122x^4 + 56x^2 - 52$.

Since we have no idea in which channel the error has occurred, we will try to correct each of these one by one. Let first assume that the error occurred in the first channel. To correct it, we remove the values err'_0 and err'_8 from err' as discussed previously. This gives the vector $err'_1 = (120, -24, -90, -52, -90, -24, 120)$, that we interpolate with respect to the subset of S': $S'_1 = \{4, 2, 1, 0, -1, -2, -4\}$. The result is the polynomial $r_1(x) = -92x^6 - 39x^4 + 93x^2 - 52$, which is an even polynomial, but of degree 6. Since the correct result is a polynomial of degree four, we clearly the channel in error is still represented in the vector. So we now apply the same technique to the second channel, leading to the interpolation of the vector

$$err'_2 = (-86, -24, -90, -52, -90, -24, -86) \quad (13)$$

according to the subset $S' = \{8, 2, 1, 0, -1, -2, -8\}$. We now obtain the fourth order (even) polynomial

$$r(x) = 15x^4 - 53x^2 - 52 \quad (14)$$

which is clearly the correct result.

It is important to note that the correct result can only be given by correcting the channel in error. In other words, if we use the same technique to correct channels 3, 4, and 5 in the previous example, we will obtain polynomials of degree 6. The reason is quite easy to understand. When removing a pair of correct values, the vector we interpolate still contains 2 wrong values. And since we have 7 points, from theorem 3, we are able to detect both errors.

5. Discussion and Conclusions

A classical and widely used approach to arithmetic fault-tolerance is modular redundancy [13][14]. In this technique, several computations are performed in parallel using identical sets of data, and their results are compared. If no error has occurred, all the copies agree; otherwise, a vote is used to determine the correct result. This very general technique can be applied to any computational task, but the hardware overhead is relatively large; i.e., 200% for a single error correction [11]. Moreover, it does not take advantage of the specific properties of both the inputs values and the computational task. In contrast, the SMRRNS technique only requires a 2-channel (typically not more than 40%) overhead in computing the main algorithm, and a relatively small increase in the reverse mapping procedure. Since the mapping complexity is independent of the

size of the main computation, this overhead reduces as the main computational complexity increases. The SMRRNS technique also uses identical independent computational channels, which allows for easy yield increase ASIC design and perfectly matched critical paths for efficient pipelining. As with many other algorithmic-based fault tolerant systems, the fault detection/ correction part of the system (the interpolation structure in our case) has to be provided with separate fault tolerant circuitry. A suitable approach is to use the classical bit-level systolic technique with in-line parity checking [15].

In this paper we have discussed extensions to the Modulus Replication RNS (MRRNS) technique for computing over independent computational channels based on a polynomial decomposition of the input data samples. The first extension is formed by adding redundant channels for fault detection and correction and we have proved that a MRRNS system composed of $n + d + c$ channels is able to detect up to d and correct up to c errors. The second extension is formed by reducing the overall number of channels by restricting the input polynomials to be even with symmetrical evaluation points. Combining both extensions we have provided examples that demonstrate fault tolerant computation using fewer channels than required by the MRRNS technique for non fault-tolerant computation. As an example we have shown that if we expect the result to be a fourth order polynomial, we need only compute over 5 channels to obtain a fault-tolerant system which is able to detect and correct up to 2 errors. In contrast, a previously published paper [16] required 5 channels to compute fourth order polynomial results without any fault tolerant capabilities.

6. References

[1] H. Krishna Garg. *Digital Signal Processing Algorithms*. CRC Press, 1998.

[2] K.-H. Huang and J. A. Abraham. Algorithmic-based fault tolerance for matrix operations. *IEEE Transactions on Computers*, 33(6):518-528, June 1984.

[3] M. A. Soderstrand, W. K. Jenkins, G. A. Jullien and F. D. Taylor. *Residue Number System Arithmetic: Modern Applications in Digital Signal Processing*. IEEE Press Reprint Series. IEEE Press, 1986.

[4] W. K. Jenkins. The design of error checkers for self-checking residue number arithmetic. *IEEE Transactions on Computers*, C-32(4):388-396, April 1983.

[5] R. W. Hamming. *Coding and information theory.* Prentice-Hall, Englewood Cliffs, N.J., 1980.

[6] N. M. Wigley and G. A. jullien. On modulus replication for residue arithmetic computations of complex inner products. *IEEE Transactions on Computers*, 39(8):1065-1076, August 1990.

[7] N. M. Wigley, G. A. Jullien and D. Reaume, “Large Dynamic Range Computations over Small Finite Rings.” *IEEE Trans. on Computers*, Vol. 43, No.1, pp. 78-86, 1994.

[8] L. Imbert and G. A. Jullien, “Fault-Tolerant Computation of Large Inner Products”, *IEE Electronics Letters*, (to be published).

[9] R. E. Blahut. *Fast algorithms for digital signal processing*. Addison Wesley, 1985.

[10] D. E. Knuth. *The Art of Computer Programming, Vol 2: Semi-numerical Algorithms*. Addison Wesley, Reading, MA, third edition, 1997.

[11] P. E. Beckmann and B. R. Musicus. Fast fault-tolerant digital convolution using a polynomial residue number system. *IEEE Transaction on Signal Processing*, 41(7):2300-2313, July 1993.

[12] C. Schurgers, F. Catthoor and M. Engels. Optimized map turbo decoder. In *Proceedings of SiPS 2000, IEEE Workshop on Signal Processing Systems - Design and Implementation*, pages 245-254, Piscataway, N.J., USA, 2000.

[13] J. M. Tahir, S. S. Dayland, R. N. Gorgui-Naguib and O. R. Hinton. Fault tolerant arithmetic unit using duplication and residue codes. *Integration, The VLSI Journal*, Vol 18(2-3), pages 187-200, June 1995.

[14] E. E. Swartzlander. Fault-Tolerant arithmetic via time-shared TMR. In *Proceedings of the SPIE, The International Society for Optical Engineering*, volume 3807, pages 84-92, 1999.

[15] M. Taheri, G. A. Jullien, and W. C. Miller, “High Speed Signal Processing Using Systolic Arrays Over Finite Rings.” *IEEE Journal of Selected Areas in Communications*, Vol. 6, No. 3, April, pp. 504-512, 1988

[16] G. A. Jullien, W. Luo, and N. M. Wigley, “High Throughput VLSI DSP Using Replicated Finite Rings”, *Journal of VLSI Signal Processing*, 14, pp. 207-220, 1996.

.