



ATIPS Laboratory

Dept. Electrical and
Computer Engineering

*Fault Tolerant Computation of
Large Inner Products*

Submitted to Electronics Letters

Laurent Imbert and Graham A. Jullien

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

Keywords: Fault tolerant computation, Inner products, Polynomial rings

Fault Tolerant Computation of Large Inner Products

Laurent Imbert and Graham A. Jullien

Abstract: In this paper we introduce a new technique for applying fault tolerance to Modulus Replication RNS computations by adding redundancy to the independent computational channels. This technique provides a low-overhead solution to fault tolerant large inner product computations.

Introduction: In the MRRNS system, we represent numbers as polynomials of indeterminates which are powers of 2. As an example, if we use the indeterminate $x = 8$, we have:

$$79 = x^2 + x + 7 = x^2 + 2x - 1 = 2x^2 - 6x - 1 = \dots \quad (1)$$

The MRRNS system makes use of the fact that every polynomial of degree n can be uniquely defined by its values at $n + 1$ distinct points, and that closed arithmetic operations can be performed over completely independent channels [1]. Another advantage of using the reduced dynamic range channels, is the ability to implement the arithmetic over finite fields $\text{GF}(p)$ [2]. If we choose p as a Fermat prime, i.e. a number of the form $2^{2^n} + 1$, modular multiplication can be much simplified [3].

As in [4] we express our algorithm in term of matrix transformations and denote M_p and M_p^{-1} as the matrices used for the evaluation and the interpolation steps respectively.

In the following, we shall use m_k to denote the vector composed of the k^{th} row of M_p^{-1} , and S for the set of N distinct points. It is important that N must be large enough, not

only to represent the input polynomials, but more importantly the result of the computation(s).

The technique is very efficient for computing many additions and multiplications in between the mapping and recovery stages; particularly where we can restrict the number of cascaded multiplications to one in any signal flow path. Fortunately, inner product computations, which are heavily used in DSP algorithms, allow this restriction for an arbitrary inner product length.

VLSI implementation advantages include reducing the interconnect span in the computational data path; this leads to easier testing and lower power [3]. Since the computations are independent we can also purposely skew the clocks between the independent data paths, thus reducing the clock current spike.

One of the potential advantages of independent computations is the ability to perform fault detection and correction at a much reduced complexity compared to more classical computational fault tolerant techniques. Although this has been previously explored at the circuit level [5], no one has yet taken advantage of the algebraic structure of the MRRNS. In this letter we will open the exploration of a new technique for fault tolerance using the independent computational structure of the MRRNS system.

Error detection: The detection technique we propose is based on the fact that we can easily compute the constant term of the final polynomial since it only depends on the constant term of each of the initial polynomials involved in the computation.

Definition 1: If u, v are two vectors in $GF(p)^n$, we denote $d(u, v)$ the distance between u and v as the number of coordinates in which u and v differ.

Thus, if u is the correct result and v is the vector obtained after the inner product, $d(u, v)$ gives the number of errors. This definition can be seen as an extension of the *Hamming distance* used in classical coding theory. The following theorem can be used to detect a single error.

Theorem 1: Let w, z be the correct and computed vectors respectively, and let

$Q(x) = q_0 + q_1x + q_2x^2 + \dots + q_nx^n$ be the final polynomial. Let us assume $0 \notin m_1$

(this is the case if $0 \notin S$), and $d(w, z) \leq 1$, i.e. that at most one error occurred. Then

$$d(w, z) = 1 \text{ iff } m_1 \cdot z \neq q_0.$$

Proof: See [6].

Error correction: Let us consider the following example. We compute the inner product $79 \times 47 + 121 \times 25 = 6738$ with correction for one channel in error. We first define four polynomials that correspond to the values 79, 47, 121, 25; let the indeterminate $x = 8$, and assign the following polynomial representations: $79 \rightarrow -1 + 2x + x^2$, $47 \rightarrow -1 + 6x$, $121 \rightarrow 1 + 7x + x^2$ and $25 \rightarrow 1 + 3x$. Since the final polynomial is third order, 4 distinct points are sufficient for its representation, but in order to correct for one error, we add a redundant channel; i.e., we compute over 5 points. We will let $S = \{-1, 1, -2, 2, 3\}$. If we express the evaluation step in terms of matrix operations, the vectors composed of the polynomial coefficients now have 5 coordinates, the last one

being clearly equal to 0. We also assume that the coefficients of the final polynomial belong to the set $\{-128, \dots, 128\}$, which allows us to compute over $GF(257)$. Evaluating the polynomials at these 5 points gives the vectors $u_1 = (-2, 2, -1, 7, 14)$, $v_1 = (-7, 5, -13, 11, 17)$, $u_2 = (-5, 9, -9, 19, 31)$ and $v_2 = (-2, 4, -5, 7, 10)$. The component-wise operations then give: $w_1 = u_1 \otimes v_1 = (14, 10, 13, 77, -19)$, $w_2 = u_2 \otimes v_2 = (10, 36, 45, -124, 53)$ and $w = w_1 \oplus w_2 = (24, 46, 58, -47, 34)$.

The final result is recovered by computing

$$q = M_{257}^{-1} \times w = \begin{bmatrix} -128 & 1 & 77 & 128 & -77 \\ 85 & -85 & -107 & 107 & 0 \\ 75 & -22 & -107 & 22 & 32 \\ 43 & -43 & 107 & -107 & 0 \\ -75 & -107 & 30 & 107 & 45 \end{bmatrix} \times \begin{bmatrix} 24 \\ 46 \\ 58 \\ -47 \\ 34 \end{bmatrix} = (2, 2, 33, 9, 0), \quad (2)$$

which corresponds to the coefficients of the final polynomial

$Q(x) = 2 + 2x + 33x^2 + 9x^3 + 0x^4$, and gives the result $Q(8) = 6738$. Note the zero highest order coefficient for a correct result.

Let us assume that one error occurred, say in the second channel, and that the result obtained is $z = (24, -24, 58, -47, 34)$ instead of w . Since we have no idea in which channel the error occurred, we correct each of them one by one. For the first channel, we assume that $z = (y, -24, 58, -47, 34)$ and we look for the value y such that $z \cdot m_5 = 0$. This inner product generates the highest degree coefficient of Q , which must be 0. Substituting this value of y in z allows us to compute the final polynomial coefficients. This gives the vector $(119, -37, 68, 83, 0)$ which corresponds to a third order polynomial,

but the constant term does not match the expected value, $q_0 = 2$. Using the same technique, we assume now that the error occurred in the second channel. Hence, we define a new $z = (24, y, 58, -47, 34)$ and we look for the value y such that $z \cdot m_5 = 0$. This leads us to solve the linear congruence $-107y \equiv -39 \pmod{257}$, which gives $y = 46$, the correct result.

If we consider a sequential implementation of this algorithm, one can stop the computations as soon as the correct result is obtained. We have proved that the correct solution can only be given by correcting the channel in error. In the other cases, we are adding a new error in our computed result, and we have proved that there exists no vector z verifying $d(w, z) = 2$, $m_n \cdot z = 0$ and $m_1 \cdot z = q_0$. The reader interested in the details of this proof can consult [6].

Although this algorithm seems complicated, the amount of computation required for each channel is very small. It basically consists of solving the linear congruence $z_k y \equiv b \pmod{p}$, where $b = m_n \cdot z - m_n \cdot z_k$. Since $m_n \cdot z$ has already been computed in the detection step, we need only one addition to define b . The solution of the linear congruence is given by $y = z_k^{-1} \times b \pmod{p}$, where z_k^{-1} , the modular inverse of z_k modulo p , is stored in a table of $(p-1) \times \lceil \log_2(p-1) \rceil$ bits. Moreover, we only need to perform the small inner product $m_1 \cdot z$ to decide whether the obtained vector corresponds to the correct result or not. Hence, the total cost for each channel is just one addition, and $n+1$ multiplications over $GF(p)$.

Conclusions: In this letter we have disclosed a new technique for detecting and correcting errors in a MRRNS computational environment. The detection technique requires a very low overhead redundant computational channel; the correction uses a full redundant channel. In the case of large inner product computations, the increase in reverse mapping overhead is very small.

References:

- 1 WIGLEY, N.M., JULLIEN, G.A. and REAUME, D., 'Large Dynamic Range Computations over Small Finite Rings' IEEE Trans. on Computers, 1994, 43, 1, pp. 78-86.
- 2 LIDL, R. and NIEDERREITER H., 'Introduction to finite fields and their applications', Cambridge University Press, Cambridge, England, revised edition, 1994.
- 3 JULLIEN, G.A., LUO, W., and WIGLEY, N.M., 'High Throughput VLSI DSP Using Replicated Finite Rings', Journal of VLSI Signal Processing, 1996, 14, pp. 207-220
- 4 BLAHUT, R.E., 'Fast Algorithms for Digital Signal Processing', Addison Wesley, 1985. Blahut book
- 5 JULLIEN, G.A, BIZZAN, S.S., WIGLEY, N.M., and MILLER, W.C., 'Using Redundant Finite Rings for Fault Tolerant Signal Processors', Proceedings of SPIE, 1994, Advanced Signal Processing: Algorithms, Architectures, and Implementations V, Volume: 2296, pp. 785-796.
- 6 IMBERT, L. and JULLIEN, G.A., 'Fault Tolerant Computation of Large Inner Products', Research report RR-01.A201, ATIPS Laboratory, Dept. Electrical and Computer Engineering, University of Calgary, Calgary, AB, Canada, February 2001. Accessible electronically at <http://www.enel.ucalgary.ca/~imbert>.