Fault-Tolerant Computations Within Complex FIR Filters

P. Chan, Graham A. Jullien, Laurent Imbert, V.S. Dimitrov, G.H. Mcgibney

FAULT-TOLERANT COMPUTATION WITHIN COMPLEX FIR FILTERS

P. Chan[1], G.A. Jullien[1], L. Imbert[2], V.S. Dimitrov[1], G.H. McGibney[3]

[1]Advanced Technology Information Processing Systems Laboratory, University of Calgary
2500 University Drive NW, Calgary, Alberta, CANADA, T2N 1N4

[2]Centre National de la Recherche Scientifique, France and
ATIPS Laboratory, Centre for Information Security and Cryptography, University of Calgary
2500 University Drive NW, Calgary, Alberta, CANADA, T2N 1N4

[3]TRLabs
Suite 280, 3553 - 31 Street NW, Calgary, Alberta, CANADA, T2L 2K7

ABSTRACT

In this paper we propose an architecture for the implementation of fault-tolerant computation within a high throughput multirate equalizer for an asymmetrical wireless LAN. The area overhead is minimized by exploiting the algebraic structure of the Modulus Replication Residue Number System (MRRNS). We demonstrate that for our system the area cost to correct a fault in a single computational channel is 82.7%. Generalized results for single error correction showing significant area savings are also presented.

1. INTRODUCTION

Distortion within the transmission channel in a wireless system, such as multipath interference and frequency selective fading, is typically corrected using an adaptive equalizer which includes a programmable Finite Impulse Response (FIR) filter. As the target data rate for the system increases, the FIR filter becomes more and more difficult to implement. The filter must have a high clock frequency to match the symbol rate and the number of filter taps must be significantly increased in order to suppress the added multipath interference in a high bandwidth system. The filter should also make use of fault-tolerant computations in order to provide a high level of reliability. These are requirements for
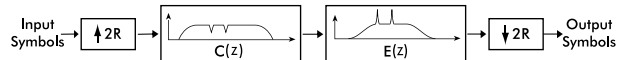
Fig. 1. Post-equalization operation.

a next generation wireless LAN [1], patented and developed at TRLabs Calgary, which is capable of data rates greater than 1Gb/s. This system requires a 128th order complex adaptive filter with 8-bit complex inputs and a 400MHz clock frequency.

This wireless LAN is an asymmetric system where the major portion of the signal processing occurs within the base station, allowing the remote terminals to be simple RF transceivers. The post-equalization operation for the uplink from a terminal to the base station is shown in Figure 1. The transmitted symbols travel through a variable expander, $\uparrow 2R$, the wireless channel, $C(z)$, which includes the RF transceiver, the adaptive filter, $E(z)$, and the variable decimator, $\downarrow 2R$. A digital signal processor (DSP) is used to compute $E(z)$ for both the in-phase (I) and quadrature (Q) channels, such that $E(z)$ approximates the inverse of $C(z)$.

Finite ring processors have been proposed, used, and extensively researched for several decades, with most of the work originating from the Residue Number System (RNS) [2]. Using two versions of the RNS, the MRRNS in conjunction with the Quadratic RNS (QRNS), it is possible to construct a complex FIR filter using identical, replicated channels [3, 4]. The MRRNS architecture also allows fault tolerance to be efficiently implemented through the addition of redundant computational channels [5].

The underlying number system for the FIR filter

is reviewed in Section 2, and the fault-tolerant technique used with the MRRNS is reviewed in Section 3. Section 4 describes the initial architecture for the filter, and introduces the modifications required to implement fault-tolerant computation. Generalized results for adding single fault correction capability are presented in Section 5.

2. NUMBER SYSTEM

Four mapping operations are required to map real and imaginary data into the number system used in the FIR filter. These operations are presented below in the order that they are performed. A complete background on the concepts involved can be found in [2, 3, 4, 6, 7, 8].

2.1. Polynomial Map

The first mapping operation for the MRRNS represents numbers as polynomials of indeterminates which are powers of two. The nth order polynomial representation of a complex number in terms of a single known indeterminate, x, can be expressed as follows:

$$a + jb = \sum_{k=0}^{n} (a_k + jb_k)x^k \qquad (1)$$

2.2. RNS Map

The polynomial coefficients a_k and b_k are mapped into finite rings, $R(m_i) = \{S : \oplus, \otimes\}$, using the set of relatively prime moduli, $M = \{m_0, m_1, \ldots, m_{L-1}\}$, where $S = \{0, 1, \ldots, m_i - 1\}$, and the symbols $\oplus$ and $\otimes$ represent addition and multiplication modulo m_i, respectively. Thus, a coefficient can be written as an L-tuple of residue digits, $a_k = (a_{k,0}, a_{k,1}, \ldots, a_{k,L-1})$, where $a_{k,i}$ is a_k modulo m_i. The result of a computation, modulo the product of the elements of M, can be recovered using the Chinese Remainder Theorem (CRT) [6].

2.3. QRNS Map

For prime moduli of the form $4K + 1$, where K is an integer, there exists a solution, $x = j_i$, and its additive inverse, $-j_i$, to the monic quadratic $x^2 + 1 = 0$ in the modular quadratic ring, $QR(m_i) = \{S : \oplus, \otimes\}$, such that j_i is an element of $R(m_i)$. Both j_i and its multiplicative inverse will belong to $QR(m_i)$ [9]. Although an extension field cannot be built based on a solution of the monic quadratic, an extension ring can be generated. The extension element can be written as $AQ_i = (A_i^\circ, A_i^*)$, with $A_i^\circ = r_i \oplus j_i \otimes i_i$ (normal) and $A_i^* = r_i \oplus (-j_i) \otimes i_i$ (conjugate). The real and imaginary components are r_i and i_i, respectively,

where $r_i, i_i, A_i^\circ, A_i^* \in R(m_i)$. Addition and multiplication over the quadratic ring are computed as:
Addition:

$$AQ_i \oplus BQ_i = (A_i^\circ \oplus B_i^\circ, A_i^* \oplus B_i^*) \qquad (2)$$

Multiplication:

$$AQ_i \otimes BQ_i = (A_i^\circ \otimes B_i^\circ, A_i^* \otimes B_i^*) \qquad (3)$$

The real and imaginary results, R_i and I_i, of a computation can be recovered from the normal and conjugate components of the result, C_i° and C_i^*, as:

$$\begin{bmatrix} R_i \\ I_i \end{bmatrix} = \frac{1 - m_i}{2} \begin{bmatrix} 1 & 1 \\ -j_i & j_i \end{bmatrix} \begin{bmatrix} C_i^\circ \\ C_i^* \end{bmatrix} \qquad (4)$$

where (4) is computed over $R(m_i)$.

In this system, the moduli in the set M for the RNS map are chosen such that each complex coefficient $c_{k,i} = a_{k,i} + jb_{k,i}$ can mapped into normal and conjugate components, $c_{k,i}^\circ$ and $c_{k,i}^*$. This allows complex multiplication to be performed without interaction between the two components, requiring only two integer multiplication operations, compared to four integer multiplication and two integer addition operations that would be required to perform the same computation directly with the real and imaginary data [7]. Furthermore, the normal and conjugate components may now be handled in independent, identical channels.

2.4. Polynomial Evaluation Map

The nth order polynomial for a QRNS channel can be represented by evaluating the polynomial at a set of points, $X = \{x_0, x_1, \ldots, x_{p-1}\}$, where $p \geq n + 1$. This can be implemented as a matrix multiplication against the Vandermonde matrix of all the possible roots of the ideal [10]. The condition that $p \geq n+1$ ensures that no actual reduction occurs during the interpolation phase (multiplication against the inverse Vandermonde matrix). Note that the multiplication of two polynomials results in an output polynomial that is of higher degree than the input polynomials. As such, the minimum number of points needed to represent an nth order input polynomial, $p = n + 1$, may not be sufficient to represent the output polynomial of a system. Thus, p must be chosen based on the order of the output polynomial [11].

The MRRNS representation of the normal channel polynomial can be written as the following p-tuple:

$$\sum_{k=0}^{n} c_{k,i}^\circ x^k = \sum_{k=0}^{n} (c_{k,i}^\circ x_0^k, c_{k,i}^\circ x_1^k, \ldots, c_{k,i}^\circ x_{p-1}^k) \qquad (5)$$

This allows computations for each polynomial coefficient to be performed independently using the same set of moduli. The dependence of the system's dynamic range on the chosen moduli set is thus reduced, as the restrictions on the dynamic range in an RNS system now apply to individual coefficients rather than the entire system [4].

3. FAULT TOLERANCE

Fault tolerance within a MRRNS architecture is implemented through the addition of redundant channels. Since this is accomplished by increasing the number of points in the set X, the redundant channels are identical to those in the original system. This is an advantage over the Redundant RNS (RRNS) [12], where the redundant channels must compute over additional moduli.

Consider an output polynomial of degree d, which can be uniquely represented by $d + 1$ points. If the computation is performed over $d + 2$ channels, it is possible to detect a fault in single channel. The interpolation of the polynomial using $d + 2$ points results in a polynomial of degree $d + 1$. However, the expected result is of degree d, which implies that the highest order coefficient must be zero if no errors occurred in the computation. It is proved in [5] that a fault in a single channel will always result in a non-zero value for the highest order coefficient.

If a second channel is added, the ability to detect a single error in $d + 2$ channels can be used to correct an error occurring in one of the $d + 3$ channels. The polynomial interpolation can be performed using the $d + 3$ unique subsets of $d + 2$ channels. If an error is present in only one channel, then one of the subsets will contain only those channels which were error free. Since it is possible to detect a single error using $d + 2$ channels, the interpolation which yields the correct result can be identified. This is generalized in [5], where it is shown that a system with $d + 2e + 1$ channels can detect and correct up to e errors.

As with many other algorithmic implementations of fault tolerance [13], this technique does not provide coverage for all hardware in the system. Faults occurring in the mapping stages prior to the polynomial evaluation map and in the corresponding reverse mapping stages are not detected in this implementation. It also does not guarantee that errors in the interpolation maps will be detected, as the highest order coefficient may not be affected. However, these components require relatively little area in comparison to the computational channels for a large inner product, and a general technique such as Triple Modular Redundancy
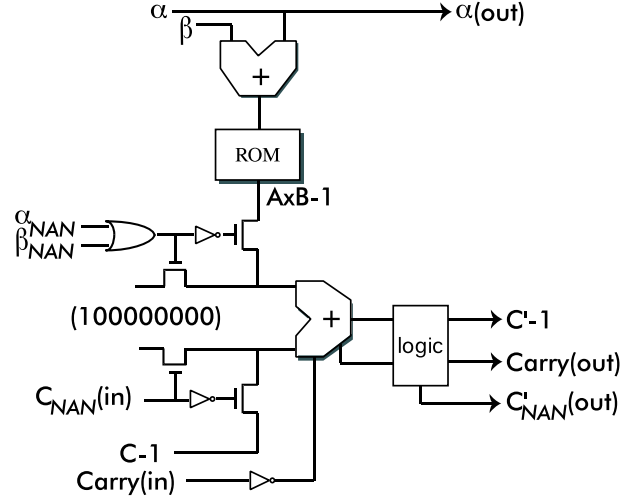

Fig. 2. Fermat ALU.

(TMR) [14] could be used to provide fault tolerance for these sections without incurring a large area overhead in the context of the entire system.

4. ARCHITECTURE

Since the implementation of fault tolerance only affects the polynomial evaluation map, computational channels, and polynomial interpolation map, only these components of the FIR filter are discussed. All three of these components are constructed using a simplified Multiply-Accumulate (MAC) cell, which performs the function $A \otimes B \oplus C$. Previous research has shown that an efficient MAC structure can be constructed using moduli which are Fermat primes, of the form $2^{2^t} + 1$, where t is an integer [3, 15]. This is known as the half index domain MAC, or the Fermat ALU, and exploits the properties of the Fermat primes with index calculus for the multiplier and diminished-1's [16] for the non-index modulo accumulator. Figure 2 shows a modulo 257 MAC cell, where the inputs α and β are A and B mapped into the index domain.

Based on input and dynamic range requirements for the FIR filter in the gigabit wireless LAN, we have chosen the order of the input polynomial to be $n = 1$ and the moduli set to be $M = \{17, 257\}$. We base our analysis on a modulo 257 QRNS channel as it contains the timing critical paths in the system, and will therefore give more pessimistic results than the equivalent modulo 17 hardware. Figure 3 shows the structure of a modulo 257 QRNS channel before (a) and after (b) the addition of fault-tolerant capability. The input and output of the system shown is either the normal or conjugate component of the QRNS polynomial coefficients.
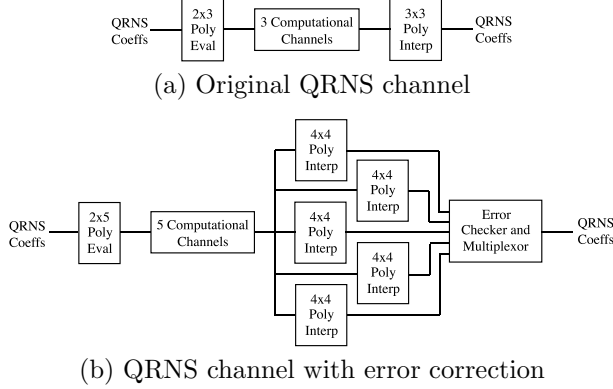
(a) Original QRNS channel

(b) QRNS channel with error correction

Fig. 3. Modulo 257 QRNS channel.

The modulo 257 MAC cell will be used as the basis for our area comparison. Note that the inputs to each chain of MAC cells must be mapped for index calculus multiplication. The area required to perform this operation can be approximated as 1.3 MAC cells per MAC chain. It is also necessary to implement programmable filter coefficients in the computational channels, which incurs an area overhead of approximately 30% over a simple MAC chain.

The result of an inner product computation taking first degree polynomials as inputs will be a second degree polynomial, which requires three computational channels to represent. The polynomial evaluation can be computed by multiplication against a 3×3 Vandermonde matrix. However, since the x^2 coefficient of the input polynomials is known to be zero, one row of the Vandermonde matrix may be discarded. Thus, this operation requires a 2×3 array of MAC cells. Similarly, the polynomial interpolation map is a 3×3 array of MAC cells. The total area for one modulo 257 QRNS channel in the 128th order FIR filter can thus be approximated as 524.6 MAC cells.

In order to detect and correct a single fault, two redundant computational channels are added to the system. The evaluation map is now a 5×5 Vandermonde matrix, which is reduced to a 2×5 array of MAC cells using the same optimization as in the above case. The interpolation map requires five 4×4 inverse Vandermonde matrices, which correspond to the 4×4 Vandermonde matrices which may be generated from each subset of four points. Each of these 4×4 arrays of MAC cells operates on a unique subset of four computational channels. An additional component is required to select the correct result by checking the x^3 coefficients of each interpolated polynomial. Since it is expected that either all x^3 coefficients are zero (no faults), or only one x^3 coefficient is zero (single fault), all other cases

result in the error checker indicating that a fault has occurred which cannot be corrected. For this system, the area of this component is approximately equal to 1.5 MAC cells. The total area is thus 958.6 MAC cells, which is an increase of 82.7% from the original system.

5. GENERALIZATION

We generalize the results for single error correction for an hth order FIR filter which maps its inputs to n^{th} order polynomials. A QRNS channel in this system requires $2n+1$ computational channels, which results in a $(n+1) \times (2n+1)$ array of MAC cells for the polynomial evaluation map, and a $(2n+1) \times (2n+1)$ array for the interpolation map. The area of the original system can thus be approximated as follows:

$$A_O = 6n^2 + 13.5n + 5.9 + 2.6hn + 1.3h \qquad (6)$$

To correct up to one error, $2n + 3$ computational channels are required. The polynomial evaluation map is expanded to be a $(n + 1) \times (2n + 3)$ array of MAC cells, and the interpolation map will use $(2n+3)$ arrays consisting of $(2n + 2) \times (2n + 2)$ MAC cells. Synthesis of the interpolation stage for values of n in the interval $[1, 4]$ show that the additional area required to determine and select the correct result can be approximated as $0.4n^3 - 2.5n^2 + 6.6n - 3$ MAC cells. Note that this approximation is based on the 400MHz clock frequency for our system, and that a lower clock frequency may allow the logic to be optimized for area, while a higher clock frequency may incur addition area costs in order to meet timing constraints. This becomes more of an issue for larger values of n, where the logic is more complex. However, the effect of inaccuracy in this term is minimal, as the total area for the system is dominated by the computational channels and polynomial maps. Thus, the percentage increase in area to implement fault tolerance is approximated as follows:

$$A_I = 100 \times \frac{8.4n^3 + 26.7n^2 + 47n + 19.1 + 2.6h}{A_O}\% \quad (7)$$

Equation (7) is graphed for several values of n in Figure 4. It can be seen that when $h \gg n$, this approach to implementing fault tolerance offers significant area savings when compared to a general technique such as TMR, which has an area overhead of 200% [12].

6. CONCLUSION

This paper has presented a detailed analysis of the cost of implementing single fault correction capability in a
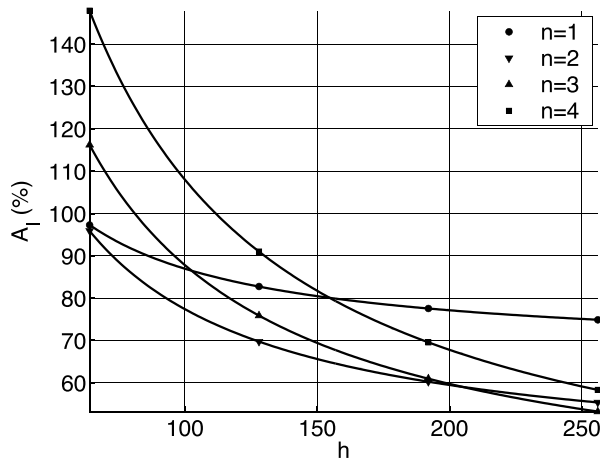
Fig. 4. Area cost of single error correction.

FIR filter using the MRRNS. The fault-tolerant architecture makes use of the algebraic properties of the MRRNS, and has been shown to provide significant area savings when compared with general techniques. This architecture also requires few additional components to be designed, as identical redundant channels are used, and the polynomial mapping stages are simply expanded from the original components.

A case study has been presented for a 128th order FIR filter which maps its inputs to first order polynomials, and it has been shown that the area cost associated with correcting a single fault in this system is 82.7%.

7. REFERENCES

[1] G.H. McGibney, *Wireless Networking with Simple Terminals*, Ph.D. thesis, Dept. Electrical and Computer Engineering, University of Calgary, Alberta, Canada, 2000.

[2] N.S. Szabo and R.I. Tanaka, *Residue Arithmetic and Its Applications to Computer Technology*, McGraw-Hill, New York, NY, 1967.

[3] M. Shahkarami, *Exploiting Redundancy in Modulus Replication Inner Product Processors*, Ph.D. thesis, Dept. Electrical Engineering, University of Windsor, Ontario, Canada, 1999.

[4] N.M. Wigley and G.A. Jullien, "On modulus replication for residue arithmetic computations of complex inner products," *IEEE Transactions on Computers*, vol. 39, no. 8, pp. 1065–1076, Aug. 1990.

[5] L. Imbert, V.S. Dimitrov, and G.A. Jullien, "Fault-tolerant computations over replicated finite rings," *IEEE transactions on Circuits and Systems I: Fundamental Theory and Applications*, vol. 50, no. 7, pp. 858–864, July 2003.

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

[7] S.H. Leung, "Application of residue number systems to complex digital filters," *Fifteenth Asilomar Conference on Circuits, Systems & Computers*, pp. 70–74, 1981.

[8] W.K Jenkins and J.V. Krogmeier, "The design of dual-mode complex signal processors based on quadratic modular number codes," *IEEE transactions on Circuits and Systems*, vol. 34, no. 4, pp. 354–364, Apr. 1987.

[9] T. Nagell, *Introduction to Number Theory*, Chelsea, New York, NY, 1964.

[10] G.A. Jullien, W. Luo, and N.M. Wigley, "High throughput VLSI DSP using replicated finite rings," *Journal of VLSI Signal Processing*, vol. 14, no. 2, pp. 207–220, 1996.

[11] N.M. Wigley, G.A. Jullien, and D. Reaume, "Large dynamic range computations over small finite rings," *IEEE transactions on Computers*, vol. 43, no. 1, pp. 78–86, Jan. 1994.

[12] W.K. Jenkins, "The design of error checkers for self-checking residue number arithmetic," *IEEE Transactions on Computers*, vol. 32, no. 4, pp. 388–396, Apr. 1983.

[13] P.E. Beckmann and B.R. Musicus, "Fast fault-tolerant digital convolution using a polynomial residue number system," *IEEE transactions on Signal Processing*, vol. 41, no. 7, pp. 2300–2313, July 1993.

[14] P.K. Lala, *Self-Checking and Fault-Tolerant Digital Design*, Morgan Kaufmann, San Francisco, CA, 2001.

[15] M. Shahkarami, G.A. Jullien, R. Muscedere, B. Li, and W.C. Miller, "General purpose FIR filter arrays using optimized redundancy over direct product polynomial rings," *Thirty-Second Asilomar Conference on Signals, Systems & Computers*, vol. 2, pp. 1209–1213, 1998.

[16] L.M. Leibowitz, "A simplified binary arithmetic for the fermat number transform," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 24, no. 5, pp. 356–359, Oct. 1976.