

*On Modulus Replication for  
Residue Arithmetic  
Computations of Complex  
Inner Products*

*N. Wigley, G. A. Jullien*

*IEEE Trans. on Computers (Special Issue on  
Computer Arithmetic) Vol. 39, No. 8, August  
1990, pp. 1065-1076*

# On Modulus Replication for Residue Arithmetic Computations of Complex Inner Products

N. Wigley, G.A. Jullien

VLSI Research Group, University of Windsor  
Windsor, Ontario, Canada N9B 3P4

## Abstract

Residue Number Systems require the selection of ring moduli whose product is greater than the predicted dynamic range of the computation being performed. The restriction that the moduli be relatively prime usually limits the set of available moduli and hence the maximum dynamic range. This is particularly the case when small moduli are to be considered for efficient hardware implementation. Severe restrictions occur when algebraic constraints, such as those posed by the necessity to implement quadratic residue rings, are a factor.

This paper presents a technique for coding weighted magnitude components (e.g. bits) of numbers directly into polynomial residue rings, such that repeated use may be made of the same set of moduli to effectively increase the dynamic range of the computation. This effectively limits the requirement for large sets of relatively prime moduli. For practical computations over quadratic residue rings, at least 6-bit moduli have to be considered; we show, in this paper, that 5-bit moduli can be effectively used for large dynamic range computations.

**Key Words:** Quadratic residue rings, Polynomial rings, Residue Number Systems, Complex Arithmetic, Inner Product Computations.

## 1. Introduction

Residue Number Systems (RNS) have been treated as somewhat of a curiosity over the past three decades [1,2]. They promise carry-free computation (with the implication of high speed implementations) but are notoriously cumbersome in implementing operations that are not closed over the finite ring structures used. Operations such as scaling, division, sign detection, and relative magnitude determination fall into this category. For some specialized applications, however, RNS hardware has many advantages. Digital Signal Processing (DSP) algorithms, where non-closed operations are limited, are perfect candidates. There are some algebraic restrictions in the development of RNS solutions, the most notable being the requirement that ring moduli be relatively prime. Coupled with hardware limitations (e.g. the need to use only small moduli) this effectively limits the dynamic range of computation, and leads, in turn, to the increased use of inefficient scaling operations.

There are a few applications where the use of the RNS can be combined with the exploitation of algebraic properties of certain ring or field structures. Examples are Number Theoretic Transforms (NTTs), used for their convolution property [3], and Quadratic Residue Number Systems (QRNS), used for efficient complex arithmetic implementations [4]. These applications place even more demand on the moduli set, generating conflicts with the requirement for efficient hardware. In the case of QRNS implementations, it is almost mandatory to use moduli from a 6-bit set (moduli  $\leq 64$ ) in order to guarantee a reasonably practical dynamic range. For some particularly efficient hardware solutions [5], this hardware limitation effectively reduces the efficiency.

Another approach, using direct-product rings, is the polynomial residue number system (PRNS) [7], whereby a quotient ring is formed as the quotient of a formal-polynomial ring by an ideal generated by a polynomial which splits completely over either the base ring or an extension of it. For example, under certain circumstances, the ring  $Z_M[X]/(F(X))$  is isomorphic

to the direct product of many copies of  $Z_M$ , the number of copies being equal to the degree of the polynomial  $F$ .

In this paper we consider a completely new approach to the problems of the coupling of relatively prime moduli to the dynamic range computation. Our approach is based on coding weighted magnitude components (e.g. bits) of numbers into polynomial residue rings, such that repeated use may be made of the same set of moduli to effectively reduce the coupling between the dynamic range of the computation and the moduli set. We will demonstrate the procedure using the problem of obtaining large computational dynamic range from a QRNS with a small number of small moduli (<33); the approach, however, is by no means limited to this example. In order to demonstrate the approach effectively, we will consider inner product computations using complex vectors. This will allow us to consider both dynamic range and computational requirements associated with the QRNS approach.

We start with a brief mathematical introduction.

## 2. Quadratic Residue Number Systems

The QRNS is based on the principles of the RNS with the application of a special ring structure to allow the emulation of complex arithmetic calculations. We deal with the following algebraic constructs.

### 2.1 Finite Rings

In residue systems we deal with rings, or fields, that are used for the actual implementation and rings that are isomorphic to direct products of implementation rings or extensions of them.  $| \circ |_{m_k}$  is the usual notation to indicate the operation of residue reduction where  $\circ \in \{+, x\}$  and  $m_k$  is the modulus of the reduction operation.

We will define the ring (field) of computation by:

$$\text{Base Ring(Field): } R(m_k) \text{ or GF}(m_k) = \{S: \oplus_{m_k}, \otimes_{m_k}\}; S = \{0, 1, \dots, m_k-1\}$$

where the symbols  $\oplus_{m_k}$  and  $\otimes_{m_k}$  correspond to modulo  $m_k$  addition and multiplication, respectively. We will define the ring, to which the computation is finally mapped, as:

$$\text{Direct Product Ring: } R(M) = \{\overline{S} : \overline{\oplus}, \overline{\otimes}\}; \overline{S} = \{0, 1, \dots, M-1\}; M = \prod_{k=1}^L m_k.$$

Where it is not obvious by context which ring the operation is being performed over, a subscript will be employed. Thus  $\oplus_m$  indicates addition over the ring with modulus  $m$ . Note that direct product rings are not rings over which actual implementation of the digital signal processing algorithm takes place; they are the direct product of the corresponding implementation rings. We have therefore used the symbols  $\overline{\oplus}$  and  $\overline{\otimes}$  rather than the expected symbols  $\oplus_M$  and  $\otimes_M$ . Results become available over these rings following the appropriate isomorphic mapping (e.g. Chinese Remainder Theorem).

## 2.2 Residue Number System

This is a brief introduction using the above notation. A digit in the residue number system is represented by an  $L$ -tuple of residues [2]:

$$X = (x_0, x_1, \dots, x_{L-1}) \tag{1}$$

where  $x_i = |X|_{m_i}$  is the  $i$ th residue and  $m_i$  is the  $i$ th modulus. Closed computations (addition, multiplication) over the implementation rings map to the direct product ring via the Chinese Remainder Theorem (CRT). We will write this succinctly as:

$$A \overline{\oplus} B \Leftrightarrow \{a_0 \oplus b_0, a_1 \oplus b_1, \dots, a_L \oplus b_L\}; \quad A \overline{\otimes} B \Leftrightarrow \{a_0 \otimes b_0, a_1 \otimes b_1, \dots, a_L \otimes b_L\}$$

with  $A, B \in R(M)$ ;  $a_k, b_k \in R(m_k)$  and  $R(M) \cong \otimes \prod R(m_k)$  (direct product).

The isomorphism ( $\cong$ ) between  $R(M)$  and the direct product of  $\{R(m_k)\}$  means that calculations over  $R(M)$  can be effectively carried out, over each  $R(m_k)$ , independently and in parallel. A final mapping (e.g. CRT) to  $R(M)$  is performed at the end of a chain of calculations. We have therefore broken down a calculation set in a large dynamic range,  $M$ , to a set of  $L$  calculations set in small dynamic ranges given by the  $\{m_i\}$ . . This is the main advantage of using the RNS over a conventional weighted value numbering system (e.g. binary).

The final mapping is found from the CRT:

$$X = \sum_{k=1}^L \hat{m}_k \otimes_M \left[ x_k \otimes_{m_k} (\hat{m}_k)^{-1} \right] \quad (2)$$

with  $\hat{m}_k = M / m_k$ ,  $X \in R(M)$ ,  $x_k \in R(m_k)$  and  $(\bullet)^{-1}$  the multiplicative inverse operator. We have also used the notation  $\sum_M$  to indicate summation over the ring  $R(M)$ .

### 2.3 Quadratic Residue Number System

The Quadratic Residue Ring is defined by using the solution of the polynomial  $x^2+1=0$  ;  $x = j = \sqrt{-1}$  , such that  $j \in R(m_k)$ .  $j$  will have two solutions in  $R(m_k)$ . Unlike the generation of extension fields by adjoining the solution of an irreducible polynomial, the QR ring is defined with a reducible polynomial. Effectively we represent  $j$  with an indeterminate  $x$  and then take the algebraic quotient modulo the polynomial  $x^2 + 1$ . We choose a modulus  $M$  such that  $x^2 + 1$  factors over the ring  $Z_M$ .

We formally define the QR and direct product QR rings as:

*Quadratic Residue Ring:*

$$QR(m_k) = \{S: \oplus, \otimes\}; S = \{A^\circ, A^*\} \text{ with } A^\circ, A^* \in R(m_k) \text{ and } A^\circ = a^r + ja^i, A^* = a^r - ja^i; j = \sqrt{-1}; a^r, a^i, j \in R(m_k), m_k = \prod_i p_i^{e_i}, p_i = 4k+1, \text{ a prime. } A^\circ \text{ will be referred to as the } \textit{normal}$$

component of element  $\mathbf{A} = (A^\circ, A^*)$  and  $A^*$  as the *conjugate* component of element  $\mathbf{A}$ . The

multiplication and addition operators both compute component-wise.

*Direct Product Quadratic Ring:*

$$\text{QR}(M) = \{ S : \overline{\oplus}, \overline{\otimes} \}; S = \{(A_1^\circ, A_1^*), \dots, (A_L^\circ, A_L^*)\}; M = \prod_{k=1}^L m_k.$$

We effectively form a residue number system as a direct product of a set of rings that support the QR requirements.

Following the QR mapping, we are able to carry out all closed complex calculations with only component-wise addition and multiplication. This both reduces the number of base field operations required for multiplication and effectively separates the two channels of computation.

### 3. Complex Inner Product Computation

Consider the problem of generating the inner product

$$z = \sum_{m=0}^{N-1} c(m)h(m) \tag{3}$$

of two complex sequences  $c(m)$  and  $h(m)$ , using a quadratic residue representation. We write the complex sequences as polynomials:

$$c(m) = c^r(m) + xc^i(m) \Big|_{x=j}; \quad h(m) = h^r(m) + xh^i(m) \Big|_{x=j}$$

and, as in section 2.3, map the polynomials to the ring  $Z_M \times Z_M$  by:

$$\begin{aligned} c^r(m) + xc^i(m) &\rightarrow (c^r(m) + jc^i(m), c^r(m) - jc^i(m)) = (C^\circ, C^*) \\ h^r(m) + xh^i(m) &\rightarrow (h^r(m) + jh^i(m), h^r(m) - jh^i(m)) = (H^\circ, H^*) \end{aligned} \tag{4}$$

We wish to know if this mapping is 1-1. Suppose that the components  $c^r(m)$ ,  $c^i(m)$ ,  $h^r(m)$  and

$h^i(m)$  all satisfy a common bound:

$$-2^\gamma + 1 \leq c^r(m), c^i(m), h^r(m), h^i(m) \leq 2^\gamma - 1.$$

It then follows from (3) that the components of the inner product  $z$ ,  $z^r$  and  $z^i$ , satisfy the bound

$$-2N(2^\gamma - 1)^2 \leq z^r, z^i \leq 2N(2^\gamma - 1)^2.$$

To map the sequences  $\{c(m)\}$ ,  $\{h(m)\}$ , and  $\{z(m)\}$  into  $Z_M \times Z_M$  in a 1-1 fashion it will suffice to have the following bound:

$$M/2 > 2N(2^\gamma - 1)^2 \tag{5}$$

which, assuming for example that  $\gamma = 8$ , becomes

$$M > 4N(2^8 - 1)^2 = 260100N \tag{6}$$

Let us now write  $M$  as a product of relatively prime numbers  $M = \prod_{k=1}^L m_k$ , where, in order to allow the QR map (4), we require the existence of solutions of the equation  $x^2 \oplus_{m_k} 1 = 0$  for each  $m_k$ . If, in addition, we make the requirement  $m_k \leq 32$  (as discussed in the introduction), then we have as eligible factors  $m_k = 2, 5, 10, 13, 17, 25, 26$ , and  $29$ . Since the  $m_k$  have to be relatively prime, and the prime 2 cannot occur to a power higher than one, the largest possible  $M$  is given by  $17 \times 25 \times 26 \times 29$ .

No additional or better choices for  $m_k$  can be used, as all prime divisors of the  $m_k$  must be of the form  $4n + 1$ , except that 2 may occur to a single power. Note that 5 and 13 are included, as well as 2; but 9 and 21 are ineligible. Thus we have the following bound on the possibilities for the blocklength  $N$  of the inner product, assuming  $\gamma = 8$ :

$$N_{\max} \leq \frac{M}{4(2^\gamma - 1)^2} \leq \frac{17 \times 25 \times 26 \times 29}{260100} \leq 1.23 \tag{7}$$

Clearly we must resort to one or more of: smaller wordlengths for the sequence elements; a larger selection for the values of  $m_k$  (i.e. include integers with six or more bits); change the

method of mapping the Gaussian integers into finite rings. Assuming that we cannot reduce the sequence wordlength, and that we have hardware restrictions associated with moduli  $>32$ , we will consider the third alternative. In the next section we introduce a new mapping technique that allows repeated use of the limited moduli set, uncoupling the problem of limited dynamic range (represented by blocklength of the inner product calculation).

## 4. Bit Mapping

We shall give a method involving a mapping of the input sequences into finite rings which consists of direct products of RNS rings with small moduli. The same modulus may be repeated several times. This will permit a significant increase in the allowable blocklength  $N$  and/or the wordlength but *without* demanding a corresponding increase in the *size* of the moduli.

### 4.1 An Example

We begin with an example. Suppose that we wish to multiply two complex numbers,  $c = 237 - j225$  and  $h = -162 + j211$ . This corresponds to the inner product with blocklength  $N = 1$ .

Step 1

Expand the two complex numbers in octal:

$$c(m) = \sum_{k=0}^{d_c} c_k(m) 8^k = \sum_{k=0}^{d_c} c_k^r(m) 8^k + j \sum_{k=0}^{d_c} c_k^i(m) 8^k \quad (8)$$

where  $c_k^r(m)$  and  $c_k^i(m) \in \{0,1,\dots,7\}$ . There is a similar expression for  $h(m)$ . The coefficients are given in table 1.

Step 2

$c_k^r(m)$	5	5	3	0	0
$c_k^i(m)$	-1	-4	-3	0	0
$h_k^r(m)$	-2	-4	-2	0	0
$h_k^i(m)$	3	2	3	0	0

Table 1

0	-2	1	0	0
-3	-1	5	0	0
0	6	0	0	0
-4	-1	-4	0	0
1	6	8	0	0
-8	4	-2	0	0
-7	4	-7	0	0
3	5	3	0	0
-7	-14	-4	0	0
-12	-5	10	0	0
5	-9	5	0	0
-9	1	-9	0	0

Table 2

(Convert to RNS). Choose the modulus  $M = 6409$  which factors into three smaller moduli  $M = 13 \cdot 17 \cdot 29$  which we denote by  $m_1, m_2,$  and  $m_3$  respectively. We can then consider the elements of Table 1 as elements

of  $Z_{13}, Z_{17},$  and  $Z_{29},$  since they all lie in the interval  $[-6, 6],$  and therefore lie between  $\left\{ \frac{m_k-1}{2} \right\}$  and  $\left\{ \frac{m_k-1}{2} \right\}$  inclusive,  $k = 1, \dots, 3.$

Step 3

(Convert to QRNS). Perform the QRNS map on the real and imaginary parts, as indicated by (4). We do this three times, using  $j_1 = 5$  for  $m_1 = 13,$   $j_2 = 4$  for  $m_2 = 17,$  and  $j_3 = 12$  for  $m_3 = 29.$  The results are given in table 2. The first four rows indicate the results of  $c$  and  $h$  modulo 13; the second four rows give the results modulo 17, etc. Note that all residues modulo  $m_k$  are taken in the interval  $-\frac{m_k-1}{2} \leq \text{residue} \leq \frac{m_k-1}{2}.$

Step 4

(Polynomial Map). Map the five-dimensional vectors of table 2 by multiplying on the right by the matrices given in table 3. This table has three  $5 \times 5$  matrices, corresponding to the moduli

$m_k = 13, 17$  and  $29$ . For instance, the second row of table 2, namely  $[-3, -1, 5, 0, 0]$ , multiplied by the fifth column  $[1, 2, 4, -5, 3]$  of table 3, yields, when taken modulo 13, the result:

$[-3] \oplus_{13} [-2]1 \oplus_{13} 20 \oplus_{13} 0 \oplus_{13} 0 = [15] \equiv 2$ , which is row two, column five of table 4. An explanation of the entries of table 3 will be given in section 4.2.

Step 5

(Multiply) Carry out the multiplication of  $c$  and  $h$  together by multiplying the representations of  $c$  and  $h$  in table 4. This is done by multiplying the coordinates of  $c$  (in the first two rows of each of the three blocks of table 4) by the coordinates of  $h$  (in the third and fourth rows of each block), the multiplications being done coordinate-wise, and the results being reduced by the

appropriate modulus. For instance in row 2 column 3 of block 1 of table 4 there is the entry  $-3$ . It corresponds to

1	1	1	1	1
-2	-1	0	1	2
4	1	0	1	4
5	-1	0	1	-5
3	1	0	1	3
<hr/>				
1	1	1	1	1
-2	-1	0	1	2
4	1	0	1	4
-8	-1	0	1	8
-1	1	0	1	-1
<hr/>				
1	1	1	1	1
-2	-1	0	1	2
4	1	0	1	4
-8	-1	0	1	8
-13	1	0	1	-13

Table 3

-5	3	0	-1	0
6	3	-3	1	2
1	-6	0	6	-1
-5	6	-4	4	4
<hr/>				
4	3	1	-2	-6
-7	3	-8	-6	-8
8	-1	-7	7	7
5	1	3	-6	8
<hr/>				
5	3	-7	4	7
9	3	-12	-7	-11
14	-10	5	1	7
11	10	-9	12	-14

Table 4

the entry two rows beneath it, namely  $-4$ . Multiply them

together and reduce the result modulo 13, obtaining  $-1$ . This is table 5, row 2 column 3.

Step 6

-5	-5	0	-6	0
-4	5	-1	4	-5
-2	-3	-7	3	-8
-1	3	-7	2	4
12	-1	-6	4	-9
12	1	-8	3	9

Reverse the polynomial map. We now have the product  $z$  of  $c$  with  $h$ , and to evaluate it we proceed backwards. The inverses of the matrices of table 3 are given in table 6. Multiply table 5 on the right by the respective matrices of table 6, and get the results of table 7.

Table 5

Step 7

Reverse the QRNS decomposition (4). The forward and inverse operations, can be expressed by using the matrices:

$$\begin{bmatrix} 1 & j_k \\ 1 & -j_k \end{bmatrix} \tag{9}$$

$$\frac{1-m_k}{2} \begin{bmatrix} 1 & 1 \\ -j_k & j_k \end{bmatrix}$$

which are inverse to each other. Operating on the elements of table 7 with the matrices of (9) (matrix multiplication this time on the *left* ) yields the elements of table 8.

0	-1	-6	1	6
0	-5	5	-2	2
1	0	2	0	-3
0	5	5	2	2
0	1	-6	-1	6
<hr/>				
0	-7	-5	7	5
0	5	-5	3	-3
1	0	3	0	-4
0	-5	-5	-3	-3
0	7	-5	-7	5
<hr/>				
0	-12	6	12	-6
0	9	-9	5	-5
1	0	6	0	-7
0	-9	-9	-5	-5
0	12	6	-12	-6

Table 6

0	0	1	6	0
-1	-6	6	-1	6
<hr/>				
-7	-4	-5	7	-5
-7	6	7	2	-6
<hr/>				
-6	-7	13	-5	9
-8	4	13	-3	-3

Table 7

6	-3	-3	-4	3
4	-2	6	2	2
<hr/>				
-7	1	1	-4	3
0	3	7	7	-2
<hr/>				
-7	13	13	-4	3
-12	8	0	12	-14

Table 8

Step 8

Reverse the RNS representation by using the Chinese Remainder Theorem. Table 8 consists of three  $2 \times 5$  matrices. We combine the three representations of a given row and column. For instance row one, column one has the three representations  $[6, -7, -7]$ . Denote these for the time being by  $\alpha$ ,  $\beta$ , and  $\gamma$ .

The mixed radix version is given by the formula:

$$D = 13 \cdot 17 A + 13 B + C$$

with:  $C = \alpha \mid_{13}$ ;  $B = 4 \otimes_{17} (\beta \oplus_{17} [-C])$ ;  $A = [-8 \otimes_{29} \gamma] \oplus_{29} [-12 \otimes_{29} B] \oplus_{29} [8 \otimes_{29} C]$

Thus with  $[\alpha, \beta, \gamma] = [6, -7, -7]$  we have:  $C = 6$  ;  $B = -1$ ;  $A = 0$ .

Thus  $D = 13 \cdot 17 \cdot A + 13 \cdot B + C = 13 \cdot 17 \cdot 0 + 13(-1) + 6 = -7$ , which is in row one, column one of table 9. The other entries in the table are computed in a similar manner.

In this section *all* of the residues  $r$  modulo  $m_k$  *must* lie in the interval:

$$-\frac{m_k - 1}{2} \leq r \leq \frac{m_k - 1}{2}.$$

-7	-16	-16	-4	3
17	37	58	41	15

Table 9

Step 9

Use the entries in table 9 as the coefficients for the octal expansion of the answer (even though some of the entries do not lie in the set  $\{0,1,\dots,7\}$ ):

$$z = -7 - 16 \cdot 8 - 16 \cdot 8^2 - 4 \cdot 8^3 + 3 \cdot 8^4 + j(17 + 37 \cdot 8 + 58 \cdot 8^2 + 41 \cdot 8^3 + 15 \cdot 8^4)$$

Thus the final answer is:

$$z = 9081 + 86457j.$$

## 4.2 The General Case

Step 1

We choose a positive integer  $\gamma$ . Given the two complex sequences  $c(m)$  and  $h(m)$  we expand them in powers of the radix  $2^\gamma$ :

$$c(m) = \sum_{k=0}^{d_c} c_k(m) 2^{\gamma k} = \sum_{k=0}^{d_c} c_k^r(m) 2^{\gamma k} + j \sum_{k=0}^{d_c} c_k^i(m) 2^{\gamma k}$$

where  $c_k^r(m)$  and  $c_k^i(m) \in \{0, 1, \dots, 2^\gamma - 1\}$ . (If, e.g.,  $c_k^r(m) < 0$ , then expand  $-c_k^r(m)$ ). A similar expression holds for  $h(m)$ .

We replace the complex number  $j$  with the indeterminate  $x$ , and the radix  $2^\gamma$  with another indeterminate,  $y$ . Thus we can write

$$c(m) = P_m(x, y) \Big|_{\substack{x=j \\ y=2^\gamma}}$$

where  $P_m$  is a polynomial of degree  $\leq 1$  in  $x$  and of degree  $\leq d_c$  in  $y$ , and has coefficients which belong to  $\{0, 1, \dots, 2^\gamma - 1\}$ . We map the polynomials  $P_m(x, y)$  representing  $c(m)$ , and also those which represent  $h(m)$ , to the ring  $Z_M[x, y]$  in a 1-1 fashion by ensuring that  $M$  is large enough. The conditions on  $M$  will be detailed below.

Process the coefficients of the polynomials representing  $c(m)$  and  $h(m)$  by treating them as arrays  $A_1(i, s, l, m)$ , where  $i = 1, 2$  signifies the number  $c$  or  $h$ ;  $s = 1, 2$  signifies its real or imaginary part;  $l = 0, 1, \dots, d$  signifies the  $l$ -th coefficient of the polynomial; and  $m = 0, 1, \dots, N-1$ , signifies the  $m$ -th term of the sequences  $c(m)$  and  $h(m)$ . The integer  $d$  is called the "replication factor", and at present must satisfy  $d \geq \max(d_c, d_h)$ .

Step 2

Factor  $M = \prod_{k=1, \dots, K} m_k$  into a product of relatively prime factors. Use the isomorphism  $Z_M \cong Z_{m_1} \times Z_{m_2} \times \dots \times Z_{m_K}$  to convert the arrays  $A_1(i, s, l, m)$  into arrays  $A_2(k, i, s, l, m)$  where  $k = 1, \dots, K$  is the index of the modulus  $m_k$ . This step will involve no computation unless  $2^l \geq m_k$ , and even then it will only require a reduction of the coefficients modulo  $m_k$ .

Step 3

Convert the arrays  $A_2(k, i, s, l, m)$  into arrays  $A_3(k, i, s, l, m)$  by the usual QRNS map. This map is always an isomorphism. One uses the first matrix of step 7 in section 4.1, with respect to the variable  $s$ .

Step 4

(The polynomial map). At this point we have represented the numbers  $c(m)$  and  $h(m)$  in the ring  $R^2[y]$  where  $R$  is an abbreviation for  $Z_{m_1} \times Z_{m_2} \times \dots \times Z_{m_K}$  and  $R^2$  denotes the cross-product ring  $R \times R$ . This cross-product arises from the QRNS representation. The polynomials are represented by the arrays  $A_3(k, i, s, l, m)$ .

We now choose, according to [6], a polynomial  $g(y)$  of degree  $d$  (an integer to be chosen later, but which already has the requirement  $d \geq d_c, d_h$ ). A description of the polynomial  $g(y)$  will be given below, but for the moment assume that it factors completely over  $R$ :

$$g(y) = (y - r_1)(y - r_2) \dots (y - r_d)$$

and that the roots  $\{r_i\}$  satisfy the condition that their differences  $\{r_i - r_j; i \neq j\}$  be invertible in  $Z_M$ . *This condition is the analog of the RNS condition that the moduli be relatively prime.*

Let  $(g(y))$  denote the ideal in  $R[y]$  which is generated by the polynomial  $g(y)$ . Then, as proved in [6], there exists an isomorphism

$$R^2[y] / (g(y)) \cong R^2 \times R^2 \times \dots \times R^2, \tag{10}$$

where the right hand side (RHS) has  $d$  factors. (This is why  $d$  is called the "replication factor"). The map is carried out by evaluation of the polynomials in the LHS at the respective roots  $y = r_i$ , thus obtaining the vector on the RHS.

We first assume the polynomials representing the sequence elements  $c(m)$  and  $h(m)$  are elements of the ring  $\mathbb{R}^2[y] / (g(y))$ . This assumption is correct provided that the degrees of the polynomials are always less than  $d$ , the degree of  $g(y)$ . Thus we get the requirement  $d > \max(d_c, d_h)$ .

We then convert the array  $A_3(k,i,s,l,m)$  to an array  $A_4(k,i,s,e,m)$ , where  $e = 1, 2, \dots, d$ , by means of the formula

$$A_4(k,i,s,e,m) = \sum_{f=1}^d m_k A_3(k,i,s,l,m) \otimes_{m_k} r_e^{l-1}$$

which expresses the isomorphism (10).

In the example of section 2 we used  $d = 5$  and  $g(y) = (y + 2)(y + 1)y(y - 1)(y - 2)$ . The effect of evaluating the polynomials at the values  $y = 0, \pm 1$ , and  $\pm 2$  is given by right-multiplication of the vectors of Table 2 by the matrices of Table 3, as can be seen by inspection. The elements of Table 3 are of the form  $f^{l-1}$ , taken modulo  $m_k$ , for  $f = -2, -1, \dots, +2$  and  $l = 1, \dots, 5$ . The inverse matrices are given in Table 6.

Thus we use five consecutive integers for the roots of  $g(y)$ . Any other choice for the roots of  $g(y)$  would change the matrices of Tables 3 and 6, but would not affect the outcome in any meaningful way. The roots of  $g(y)$ , however, must always be distinct and their differences must be invertible in  $\mathbb{Z}_M$ .

Step 5

We now carry out the multiplications and additions of the inner product. Note that addition and

multiplication in a ring which is a direct product ring are given by component-wise addition and multiplication. Thus

$$A_5(k,s,e) = \sum_{m=0}^{N-1} A_4(k,1,s,e,m) \otimes_{m_k} A_4(k,2,s,e,m)$$

It is this formula which yields the power of the method. The whole problem has been reduced to performing some additions and multiplications in some very small finite rings, with *no* cross-channel communication. The number of different rings  $Z_{m_k}$  is very small, usually three or four, but several copies of each may be made. For complex data there will be  $2d$  copies; for real data,  $d$  copies. (See section 5 for typical values of  $d$ ).

Steps 6-8

These steps simply reverse the process. Instead of having to handle two sequences of data, we are now reduced to two numbers, the real and imaginary parts of  $z$ , represented by the array  $A_5(k,s,e)$ .

We transform  $A_5(k,s,e)$  to an array  $A_6(k,s,e)$  by inverting the polynomial map. This amounts to right-multiplication by the appropriate matrices of table 6. In the general case the entries of these matrices can be found by means of Lagrange Interpolation Polynomials (LIP) (see [6]); the coefficients of the LIPs corresponding to the roots of  $g(y)$  will be the entries of the matrices.

It is at this stage that we get the critical lower bound on the degree  $d$  of  $g(y)$ . Since the polynomials representing  $z^r$  and  $z^i$  must have degree  $< d$  in order to avoid an error in step 6 (undoing the polynomial map), we investigate this degree. Because the polynomials representing  $c(m)$  and  $h(m)$  have degrees  $\leq d_c$  and  $d_h$  respectively, the inner product  $z$  will be represented by a polynomial of degree  $\leq d_c + d_h$ . Thus we require  $d > d_c + d_h$ . This is the final requirement on  $d$ .

We then transform  $A_6(k,s,e)$  to an array  $A_7(k,s,e)$  using the inverse map of the QRNS map. See

the second matrix of step 2 of section 4.1. Use the variable  $s$  for matrix multiplication.

Finally we use the mixed radix conversion to undo the RNS representations of our two real numbers. This process is well-known [2]. We obtain an array  $A_g(s,l)$  of elements of  $Z_M$ . These elements are the coefficients of polynomials in  $Z_M[y]$ .

The result now is two polynomials, or at least an array  $A_g(s,l)$ ,  $s = 1,2$ ;  $l = 1,2,\dots,d$  which represents the polynomials. These numbers are to be used as coefficients of the radix- $2^\gamma$  expansion of  $z^r$  and  $z^i$ :

$$z^r = \sum_{l=0}^d A_g(1,l)2^{\gamma l}, \quad z^i = \sum_{l=0}^d A_g(2,l)2^{\gamma l}.$$

It is important to note that the coefficients  $A_g(s,l)$  do *not* have to satisfy the inequality  $0 \leq A_g(s,l) \leq 2^\gamma - 1$ . There is no reason to expect them to be the usual coefficients of the radix- $2^\gamma$  expansion of  $z$ .

## 5. Size Restrictions

We must be careful that we obtain the correct answer in the case of *homomorphisms* which are not *isomorphisms*. There are three maps, which, together with their inverses, may cause trouble.

1. The map of the coefficients of the radix- $2^\gamma$  expansions of the sequences  $c(m)$  and  $h(m)$  into  $Z_M$ . These are also the coefficients of the polynomials  $P_m(x,y)$  and  $Q_m(x,y)$  which represent  $c(m)$  and  $h(m)$ .

Since we always have:  $0 \leq c_k^r(m), c_k^i(m), h_k^r(m), h_k^i(m) < 2^\gamma - 1$ , there is no problem here provided that we have  $2^\gamma - 1 \leq M$ .

We must also ensure that the return map from  $Z_M$  to  $Z$  yields no error. This situation requires a somewhat more delicate analysis. Suppose that the inner product  $z$  is

represented by the polynomial

$$Z(x,y) = \sum_{m=0}^{N-1} P_m(x,y) Q_m(x,y).$$

Then the coefficients of this polynomial, as already remarked, will *not* have coefficients in the interval  $[0, 2^\gamma - 1]$ . In fact

$$Z(x,y) = \sum_{m=0}^{N-1} \sum_{k=0}^{d_c} \sum_{k'=0}^{d_h} (c_k^r(m) + xc_k^i(m)) (h_{k'}^r(m) + xh_{k'}^i(m)) y^{k+k'},$$

where we have expanded the polynomials  $P_m$  and  $Q_m$ . The left-hand side can be written  $z = z^r + xz^i$ , and the coefficients of  $z^r$  and  $z^i$  can be estimated in size. For fixed  $k + k'$  the right hand side cannot have more than  $1 + \min(d_c, d_h)$  terms multiplying the factor  $y^{k+k'}$ , as the domains of  $k$  and  $k'$  are given by  $k \in \{0, 1, \dots, d_c\}$  and  $k' \in \{0, 1, \dots, d_h\}$ . Since  $m$  runs through a total of  $N$  values, we get

$$0 \leq |z_k^r|, |z_k^i| \leq 2N(2^\gamma - 1)^2 [1 + \min(d_c, d_h)], \quad (11)$$

where the factor 2 comes from the two summands of the real (or imaginary) part,  $N$  is the size of the domain of  $m$ ,  $(2^\gamma - 1)^2$  is the (most pessimistic) bound on the coefficients, and  $[1 + \min(d_c, d_h)]$  arises from the smaller of the domains of  $k$  and  $k'$ . Thus we shall require the modulus  $M$  to satisfy

$$M/2 > 2N(2^\gamma - 1)^2 [1 + \min(d_c, d_h)]. \quad (12)$$

Note that this is a great improvement over equation (5) since we are dealing with much smaller values of  $\gamma$  than  $\gamma = 8$ .

2. The RNS map. This is always an isomorphism of  $Z_M$  and  $Z_{m_1} \times Z_{m_2} \times \dots \times Z_{m_K}$ , and thus causes no problem.
3. The QRNS map. This is also an isomorphism and causes no problem.

4. The polynomial map. Any error made under this map has to belong to the ideal  $(g(y))$ , i.e. must be a polynomial multiply of  $g(y)$ . But as explained earlier, this is taken care of by the requirement  $d > d_c + d_h$ . The only polynomial in the ideal  $(g(y))$  of degree  $< d$  is the zero polynomial. The requirement  $d > d_c + d_h$  on the replication factor tells us the minimum size of  $d$  in terms of the number of bits of the input data. Given this information on the size of  $c(m)$  and  $h(m)$ , we must then make  $2d$  (for complex data,  $d$  for real data) copies of the ring  $\mathcal{R}$  in order to carry out the arithmetic of the inner product.

We now discuss the assumption made earlier, namely that there exists a polynomial  $g(y)$  whose roots have invertible differences over  $Z_M$ . This is easily seen to be the case if and only if each prime  $p$  dividing  $M$  satisfies the inequality  $p \geq d$ . For if each  $p$  satisfies the inequality, then a set of  $d$  consecutive integers will have its differences invertible ( $p$  could not divide any of the differences since the roots are so close together). Conversely if  $p < d$  for some divisor  $p$  of  $M$  then the set of  $d$  roots will, when considered as residues modulo  $p$ , have one or more duplications, and the corresponding differences will be zero modulo  $p$ .

## 6. Benefits of the Method

Let us now sum up the restrictions on the parameters of the problem and determine what word- and block-lengths are available for a given selection of moduli.

The first restriction is (12). The second is the minimum bound for the replication factor  $d$ , and comes from point 4 of section 5. The third is the requirement on the roots of  $g(y)$ , and the fourth is the restriction on the moduli for the QRNS map. Altogether we have

- 1)  $M > 4N(2^\gamma - 1)^2[1 + \min(d_c, d_h)]$
- 2)  $d > d_c + d_h$

- 3) the differences of the roots of  $g(y)$  are invertible in  $Z_M$ , so  $p \geq d$
- 4)  $p \equiv 1 \pmod{4}$  for any  $p$  dividing  $M$  (Note that by conditions 2) and 3) the case  $p = 2$  is now meaningless).

Let us now assume that the input data  $c(m)$  and  $h(m)$  have, in addition to a sign bit,  $b_c$  and  $b_h$  bits respectively. Then it is easy to show that it is sufficient to choose  $\gamma$ ,  $d_c$  and  $d_h$  such that

$$5) \quad \gamma(1 + d_c) \geq b_c \quad \gamma(1 + d_h) \geq b_h.$$

For comparison of the present method (Modulus Replication Residue Number System, or MRRNS) with the usual (QRNS) method, we reproduce (5), which now takes the form

$$1') \quad M > 4N(2^{b_c} - 1)(2^{b_h} - 1). \quad (14)$$

### 6.1 Example 1 (Eight-bit input data)

Let  $b_c$  and  $b_h$  both equal 8. Then (14) yields, as in (6),

$$M > 4N(2^{b_c} - 1)^2 = 260100N \quad (15)$$

By restricting the moduli to be less than 32 we saw before that the maximum permissible blocklength is  $N_{\max} \leq 1.23$ .

By contrast, let us examine various possibilities of  $\gamma$  under the present (MRRNS) method.

Let  $\gamma = 1$ . Then  $d_c = d_h = 7$  and thus  $d = 15$ . Since  $p$  must be  $\geq 15$  for primes  $p$  dividing  $M$ , we have  $m_k = 17, 29, \dots$ . If we restrict ourselves to moduli less than 32, then 1) becomes

$$M = 17 \times 29 > 4N(1 + 7) = 32N$$

and we have  $N_{\max} = \left\lfloor \frac{17 \times 29}{32} \right\rfloor = 15$ . Although this is certainly an improvement over  $N_{\max} =$

1, we can do better. Note that this choice of  $\gamma$  entails  $2d = 30$  replications of the ring  $R$ .

Let  $\gamma = 2$ . Then  $d_c = d_h = 3$  and  $d = 7$ . Since  $p \geq 7$  we can add 13 to the list of possibilities for  $m_k$ . Restricting ourselves again to 5-bit moduli we have

$$M = 13 \times 17 \times 29 > 4N3^2(1 + 3) = 144N$$

which implies  $N_{\max} = \left\lfloor \frac{13 \times 17 \times 29}{144} \right\rfloor = 44$ .

Let  $\gamma = 3$ . Since we required eight bits for  $c(m)$  and  $h(m)$ , condition 5) above says we must take  $d_c = d_h = 2$  and  $d = 5$ . Since  $p \geq d = 5$ , we can add 5 to the list of possible prime factors of the moduli  $m_k$ . If, on the other hand, we restrict the moduli to 5-bit numbers then we can add 25 as an additional value of  $m_k$  to be added to the earlier choices of 13, 17, and 29. Then 1) becomes

$$M = 13 \times 17 \times 25 \times 29 > 4N7^2(1 + 2) = 588N$$

which yields  $N_{\max} = \left\lfloor \frac{13 \times 17 \times 25 \times 29}{588} \right\rfloor = 272$ .

With  $\gamma = 4$  we can take  $d_c = d_h = 1$  and  $d = 3$ . The condition  $p \geq 3$  yields no new choices for  $m_k$ , and thus 1) becomes

$$M = 13 \times 17 \times 25 \times 29 > 4N(15)^2(1 + 2) = 2700N,$$

and hence  $N_{\max} = \left\lfloor \frac{13 \times 17 \times 25 \times 29}{2700} \right\rfloor = 59$ . Thus we are receiving diminishing returns for

our efforts, except that the replication number  $d = 3$  has been reduced from  $d = 5$ .

The next values of  $\gamma$ , namely  $\gamma = 5, 6,$  and  $7$ , still require  $d_c = d_h = 1$  and  $d = 3$ . Thus the only change is to increase the denominator of the fraction that yields  $N_{\max}$ , so the only changes are for the worse.

Finally, we investigate  $\gamma = 8$ . The factor  $(2^\gamma - 1)^2$  in condition 1) would appear to make this a very discouraging choice. Indeed, we get, with  $c(m)$  and  $h(m)$  still having eight bits each,  $d_c = d_h$

= 0, which yields  $d = 1$ . Hence there is, in this instance, no replication of the ring  $R$  other than duplication for the purpose of QRNS. Moreover, 1) becomes

$$M > 4N(2^\gamma - 1)^2(1 + 0) = 260100N.$$

Note that this is precisely inequality (14). Thus the usual method of using RNS and QRNS without MRRNS is simply a form of MRRNS with  $\gamma =$  maximum bit-length of the input data.

## 6.2 Example 2 (Twelve-bit input data)

Here we have  $b_c = b_h = 12$ . First let us examine the requirements made by using the old method. Condition (14) becomes

$$M > 4(2^{12} - 1)^2N = 67076100N.$$

Allowing all possible six-bit moduli  $m_k = 2, 5^2, 13, 17, 29, 37, 41, 53$ , we have  $N_{\max} \leq \left\lfloor \frac{2 \times 13 \times 17 \times 25 \times 29 \times 37 \times 41 \times 53}{67076100} \right\rfloor = 384$ . Of course this supposes the use of a very large value ( $2.57 \times 10^{10}$ ) for  $M$ , with the attendant problems of hardware design.

For the MRRNS method, reasonable choices of  $\gamma$  are  $\gamma = 1, 2, 3$  and  $4$ . The best choice can be shown to be  $\gamma = 4$ . Then  $d_c = d_h = 2$  and  $d = 5$ . As moduli we have the choices  $m_k = 5^2, 13, 17, 29, 37, \dots$ , and condition 1) becomes

$$M > 4N(2^4 - 1)^2(1 + 2) = 2700N.$$

Using only 5-bit moduli we have for  $M$  a maximum value  $M = 13 \times 17 \times 25 \times 29$  and  $N_{\max} = 59$ . If we allow 6-bit moduli, then by using *all* of them (i. e. 13, 17, 25, 29, 37, 41 and 53) we get  $N_{\max} = 44771203$ .

## 6.3 Example 3 (Input data of One and Twelve-bits)

Additional improvements can be made if one of the input variables, say  $c(m)$ , has a shorter word-length than the other variable  $h(m)$ . As an extreme example suppose that  $c(m)$  always

takes on the values 0 or  $1^1$ , so that  $b_c = 1$ . Let us assume that  $h(m)$  is a sequence of 12 bit integers, so  $b_h = 12$ .

For the usual method inequality (14) becomes

$$M > 4N(2^{12} - 1)(2^1 - 1) = 16380N.$$

Using only 5-bit moduli 17, 25, 26 and 29 we obtain  $N_{\max} = 19$ .

For the MRRNS system requirements 1) and 2) become

$$1) \quad M > 4N(2^\gamma - 1)^2(1 + 0)$$

$$2) \quad d > 0 + d_h$$

With  $\gamma = 1$  we have  $d_h = 11$ ,  $d = 12$ ,  $M > 4N$ ,  $m_k = 13, 17, 29, \dots$ , and thus with 5-bit moduli  $N_{\max} = \left\lfloor \frac{13 \times 17 \times 29}{4} \right\rfloor = 1602$ . This is with 12 replications of the QRNS system.

With  $\gamma = 2$  we obtain  $d_h = 5$ ,  $d = 6$  and  $M > 36N$ . Since  $p \geq d$ , we have  $m_k = 5^2, 13, 17, 29, \dots$ , and thus with 5-bit moduli  $N_{\max} \leq \left\lfloor \frac{13 \times 17 \times 25 \times 29}{36} \right\rfloor = 4450$ .

With  $\gamma = 3$  we have  $d_h = 3$ ,  $d = 4$  and  $M > 196N$ . There is no increase in the list of eligible moduli. Thus the only changes are a smaller value of  $N_{\max}$  and a decrease in the replication factor.

Consequently, for  $\gamma = 2$  we get the optimal value of  $N_{\max} = 4450$ . We use the moduli 13, 17, 25 and 29 and twelve replications of the ring  $\mathcal{R}$  (2 for QRNS, 6 for MRRNS). By contrast, the usual method would use the moduli 26, 17, 25 and 29 and only two replications for the QRNS; and the maximum blocklength allowable would be  $N_{\max} = 19$ .

---

<sup>1</sup> Useful in m-sequence correlation systems

## 7. Computational Bounds and Scaling

### 7.1 Computational Bounds, an Example

Let us consider a fixed problem, that of computing the DFT. We shall assume that the input data are of nine bits each, and that the radix is  $2^3$ . This means that each input integer can be written as a polynomial of degree  $\leq 2$  in the radix 8. The coefficients of the polynomial will be integers in the interval  $[0,7]$ . The output will consist of similar polynomials, but of degree 4. The degree of the quotient map (the replication factor  $d$ ) must therefore be 5. We will discuss both the Modulus Replication Residue Number System (MRRNS) and the direct QRNS methods.

#### 7.1.1 MRRNS technique

Let us assume that, for a fixed integer  $n$ , the sequence  $h(m)$  is the  $m$ th term in the  $n$ th exponential sequence (basis set) for the DFT, namely

$$h(m) = e^{2\pi i n m / N}, \quad 0 \leq m, n \leq N-1.$$

First, since we know the sequence  $h$ , we can get a vast improvement on the inequality (12).

Using the definition of the DFT we have, as before,

$$z_k(n) = \sum_{i+j=k} \sum_{m=0}^{N-1} c_i(m) h_j(m, n) \quad 0 \leq n \leq N-1 \quad (16)$$

Let us write these variables in terms of their real and imaginary parts:

$$z_k(n) = z_k^r(n) + jz_k^i(n) \quad c_i(m) = c_i^r(m) + jc_i^i(m) \quad h_i(m) = h_i^r(m) + jh_i^i(m)$$

We wish to obtain a bound on the integers  $z_k^r(n)$  and  $z_k^i(n)$  in order to establish the necessary dynamic range, and hence the minimum allowable value for the modulus  $M$ .

It is a simple matter to compute the coefficients  $h_1^r(m)$  and  $h_1^i(m)$  from their values as the real and imaginary parts of  $e^{2\pi i n m / N}$ . We then examine the equation

$$z_k^r(m) = \sum_{m=0}^{N-1} \sum_{i+j=k} c_i^r(m) h_1^r(m) - c_i^i(m) h_1^i(m)$$

and a similar equation for  $z_k^i(m)$ . Since we have the inequalities  $0 \leq c_1^r(m) \leq 7$  and  $0 \leq c_1^i(m) \leq 7$ , it follows that the expression for  $z_1^r(m)$  is maximized by  $c_1^r(m)$  identically equal to 7 and  $c_1^i(m)$  identically equal to 0; the dynamic range required by  $z_1^r(m)$  is thus  $\leq 2 \times 7 \times U$ , where  $U = \sum_{m=0}^{N-1}$

$$\sum_{i+j=k} h_1^r(m).$$

The number  $U$  can be computed directly from the trigonometric values of  $h(m)$ , and empirical evidence suggests that it grows linearly with the blocklength  $N$ . For  $N = 64$  one obtains  $U = 669$ , so that  $M > 2 \times 7 \times 669 = 9366$ . Since  $29 \times 25 \times 13 = 9425$ , we find that these three moduli, namely 29, 25 and 13 will allow a sufficient dynamic range to guarantee accuracy of the mappings to the direct product rings and return. Thus by utilizing ten copies of the ring  $Z_{29} \times Z_{25} \times Z_{13}$  we can embed the DFT with any blocklength  $N \leq 64$  and nine-bit word length.

### 7.1.2 Direct QRNS technique

If we compare this with the usual QRNS method of computing the DFT, we are led to the following observations. Assume that the modulus to be used is again denoted by  $M = \prod m_k$ . The choice of  $M$  will be different, of course. In order to accommodate the nine-bit data and blocklength  $N$  we must have a dynamic range which will allow the largest of the real and imaginary parts of the terms

$$z(n) = \sum_{m=0}^{N-1} c(m)h(m)$$

Since the values of  $h(m)$  are given as roots of unity that have been scaled to nine-bit complex integers, it follows that  $|h(m)| \leq 2^9$ . It thus makes sense to use the  $L^1$  norm on  $h$ , and therefore the  $L^\infty$  norm on  $c$ . Thus we get  $\|h\|_1 = 2^9 N$ , and

$$|x(n)|, |y(n)| \leq |z(n)| \leq (2^9\sqrt[9]{2}) (2^9N) = 2^{18}\sqrt[9]{2}N$$

With  $N = 64$  we thus obtain  $|z(n)| \leq 23726566.4$ , and we need  $|z(n)| < M/2$ , where  $M$  is a product of moduli which are relatively prime, and each prime divisor of  $M$  (except 2) must be of the form  $4k+1$ . Restricting our attention to moduli  $< 64$  (for architectural reasons), we are limited to the list  $m_k \in \{2,13,17,25,29,37,41,53,61\}$ . A reasonable choice is  $M = 61 \times 53 \times 41 \times 29 \times 13 = 49972481$ . Thus, since we are using the QRNS method, we shall require two copies of the ring  $Z_{61} \times Z_{53} \times Z_{41} \times Z_{17} \times Z_{13}$ . We will use these two equivalent systems in a comparison study in section 8.

## 7.2. Scaling

We now turn our attention to scaling. Let us continue examination of the above example of the DFT with nine-bit input data and moduli 29, 25 and 13. An examination of the method as given in section 4.1 shows that a typical sample of output will be represented as a polynomial

$$z = \sum_{k=0}^4 (A_{1k} + A_{2k}m_1 + A_{3k}m_1m_2)Y^k \quad (17)$$

where  $Y$  will be set equal to  $2^Y = 8$ . If we choose, say  $m_1 = 17$ ,  $m_2 = 25$  and  $m_3 = 29$ , then the values of  $A_{1k}$ ,  $A_{2k}$ , and  $A_{3k}$  will satisfy the restrictions  $|A_{ik}| \leq (m_i-1)/2$ , i.e.

$$|A_{1k}| \leq 8 \qquad |A_{2k}| \leq 12 \qquad |A_{3k}| \leq 14$$

In actual fact the DFT output data will be given by

$$|z(p)| = \left| \sum_{j=0}^{N-1} c(p-j)h(j) \right| \leq \|c\| \|h\| \leq 2^9 N 2^9 \sqrt[9]{2} = 2^{23.6}$$

Let us suppose that we decide to limit the output data to nine bits. The maximum possible value of (17) is

$$(8 + 12 \times 17 + 14 \times 17 \times 25) \sum_{k=0}^4 2^{3k} = 28844322 = 2^{24.8}$$

which agrees fairly closely with the (more accurate) estimate of  $2^{23.6}$ . Let us suppose then that we scale by neglecting any term which is  $< 2^{11}$  absolute value. We examine the fifteen terms of (17) and find that the terms involving  $A_{10}$ ,  $A_{11}$ ,  $A_{12}$ ,  $A_{20}$ , and  $A_{21}$  will all be  $< 2^{11}$  in absolute value. Thus we can discard five of the fifteen terms of (17) in compiling our final answer.

## 8. VLSI Implementation

It is difficult to completely generalize the possible implementation techniques and their ramifications on the efficiency of the, essentially, competing methods of MRRNS and direct QRNS. We can, however, perform a reasonably accurate comparison of the implementation of the DFT example, used in the previous section, considering the 3 implementation stages:

- 1) Map from binary complex components to the appropriate set of rings. This involves mapping to RNS, followed by the QRNS mapping:

$$(c^j \oplus_{m_k} (j \otimes_{m_k} c^j), (c^j \oplus_{m_k} (-j \otimes_{m_k} c^j)) \Rightarrow (C_{m_k}^{\circ}, C_{m_k}^*)$$

and, in the case of the MRRNS, a polynomial mapping using pre-determined powers of the roots of the polynomial  $g(y)$ .

- 2) Perform, component-wise, the inner product computation over all of the rings.
- 3) Reverse the mapping. This involves inverting the polynomial mapping (for the MRRNS technique), inverting the QRNS mapping:

$$\left( 2^{-1} \otimes_{m_k} [C_{m_k}^{\circ} \oplus_{m_k} C_{m_k}^*] \right), \left( (2 \otimes_{m_k} j)^{-1} \otimes_{m_k} [C_{m_k}^{\circ} \oplus_{m_k} (-C_{m_k}^*)] \right) \Rightarrow c^j, c^i$$

and converting back to binary.

We can take these 3 steps and perform a direct comparison between the two techniques. In order to have some meaningful silicon area comparisons (other than the usual questionable complexity analyses) we will select the bit-steering ROM technique[8], and implement the computation as, mainly, linear bit-level systolic arrays. The comparison is based on breaking the computation into a set of pipelined inner product steps between arbitrary data and fixed coefficients. We choose this approach because the method has proved efficient, and we have direct access to working cell designs with known silicon area. We will perform an (Area.Period) product comparison since the techniques will have slightly different throughput rates. We will assume that only the data stream,  $\{c\}$ , is to be mapped on-line; the coefficient stream,  $\{h\}$ , is mapped off-line. In order to obtain a certain amount of generality we will initially use  $L_M$  for the number of MRRNS moduli, and  $L_D$  for the number of direct QRNS moduli. The number of BIPSP<sub>m</sub>[8] cells is obtained from the  $L_M$  and  $L_D$  multipliers. A 6-bit modulus inner product step requires a 12 unit cell area, a 5-bit modulus requires a 5 unit cell area and a 4-bit modulus requires a 2.4 unit cell area<sup>1</sup>. A unit cell area is defined as the area of a pipelined 5-bit BIPSP<sub>m</sub> cell. We will use these figures in a final calculation for the silicon area calculation. A comparison of required inner product steps, for both techniques, is presented in Table 10. The steps required have been optimized by assuming that one of the  $g(y)$  roots is zero. This yields simplifications in the matrix multiplications required for the polynomial forward mapping and to a lesser extent, the reverse polynomial mapping (see the example in section 4.1). The direct QRNS residue mapping requires  $B$ , BIPSP<sub>m</sub> cells[8], where  $B$  is the length of the input real, or imaginary, complex data in bits. The scaling complexity is taken from the number of accumulate/fixed coefficient multiply steps required for a mixed radix conversion[9] (or MRC/scale conversion). We will assume that the final output has the same precision as the input, and use a generalization (approximate) of the result for the

---

<sup>1</sup>Based on experimental design measurements

MRRNS (from the example); i.e. we reduce  $L_M$  by 1. For the MRC conversion we will use  $d'$  as the reduced number of bit mapped channels. For the direct QRNS technique, we will scale by half of the moduli ( $S = \frac{L_D}{2}$  [9] to the lowest integer value  $\lfloor \frac{L_D}{2} \rfloor$ ). We will assume that an MRC coded result is acceptable for the output. In order to keep the formulations as simple as possible, we have not taken into account the bit size of each ring modulus. We will do this in the calculation by assigning an appropriate weight to the modulus number variable ( $L_M^w, L_D^w$ ); the weight will include the number of bit-level cells for a complete inner product step. We will interpret  $(L_M-1)$  as  $L_M^w \frac{(L_M-1)}{L_M}$ ; similarly products of  $L_M$  will have the weight multiplied only once.

Step	MRRNS	Direct QRNS
Forward mapping	Hard-wire map of $2d_c$ bit-level components. No steps required. QRNS map of $2(d_c+1)L_M$ steps. Polynomial map of $2d_c(d-1)L_M$ steps.	Residue conversion of $2BL_D$ BIPSP <sub>m</sub> cells (B is number of data bits). QRNS conversion of $2L_D$ steps.
Inner product computation	$N2L_M d$ steps.	$N2L_D$ steps.
Reverse mapping and scaling	Reverse polynomial map of $2(L_M-1)(2d-1)(\frac{d-1}{2})$ steps. Reverse QRNS map of $2d(L_M-1)$ steps. MRC convert of $(\frac{(L_M-1)L_M}{2})2d'$ steps.	Reverse QRNS map of $2L_D$ steps. MRC convert of $2(L_D - \lfloor \frac{L_D}{2} \rfloor)(\lfloor \frac{L_D}{2} \rfloor) + (L_D - \lfloor \frac{L_D}{2} \rfloor - 1)(L_D - \lfloor \frac{L_D}{2} \rfloor)$ steps.

Table 10 A comparison of the inner product steps for the MRRNS and direct QRNS techniques

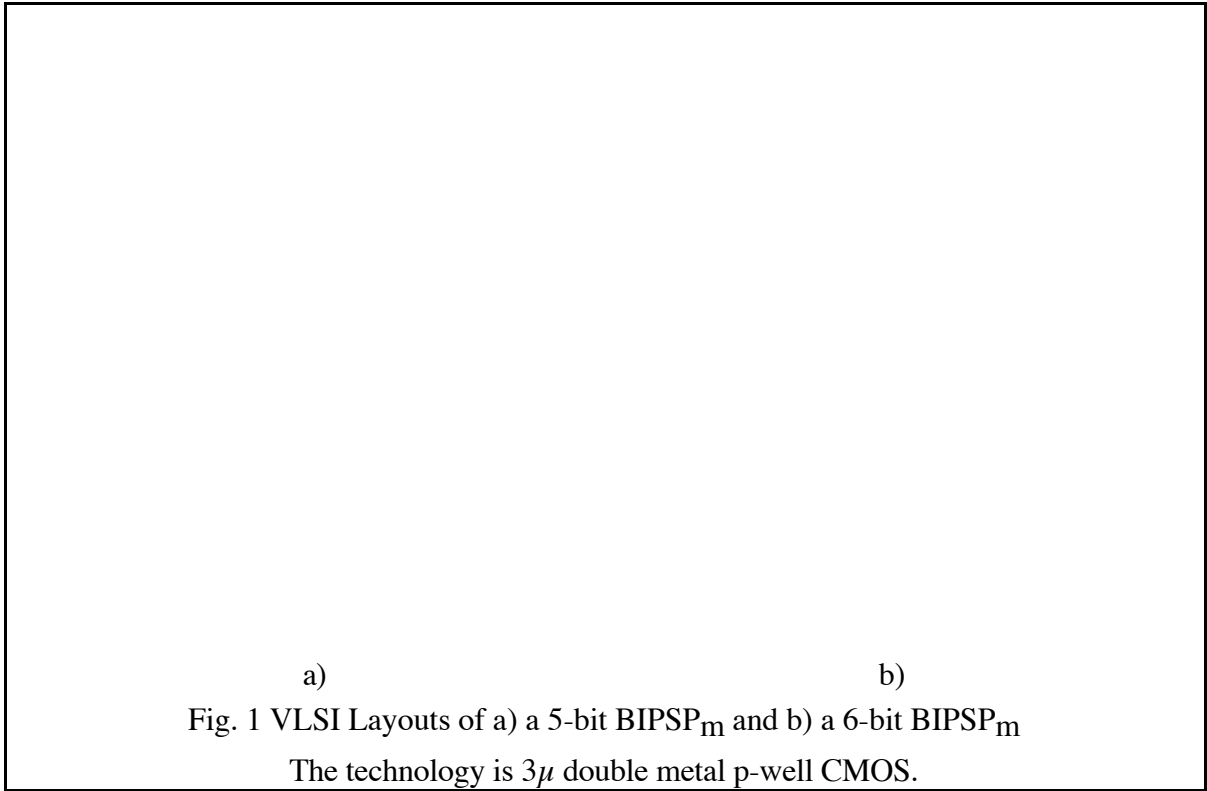
We can now produce specific silicon area comparisons. The BIPSP<sub>m</sub> cells are shown in Fig. 1

for a 5-bit and 6-bit cell. The 6-bit cell runs about 10% slower than the 5-bit cell, so we use a factor of 1.11 in the (Area.Period) product of the direct QRNS system, since it requires the use of 6-bit cells. Note that the system will run at the throughput rate of its slowest channel. We will use a unit (Area.Period) product as that (Area.Period) produced by a 5-bit BIPSP<sub>m</sub> cell. The formulations of Table 10 are converted into (Area.Period) products in Table 11. The parameters of the example are:

$$d_c = 3, d = 5, d' = 4, L_M = 3, L_M^W = 12.4, L_D = 5, L_D^W = 43.4, N = 64.$$

The final results are an (Area.Period) product of 9010.66 for the MRRNS technique, and an (Area.Period) product of 7974.27 for the direct QRNS technique. It appears that the direct QRNS technique has a slight edge over the MRRNS method (about 12%). This is a much more meaningful figure than a generalized complexity analysis, but it still hides the structural benefits of the MRRNS technique. If we impose the condition that a binary output is required, then the QRNS conversion will have to allow multiple base extensions to a binary ring modulus [10], and this will increase the complexity of conversion by about a factor of 2. The (Area.Period) products will now be approximately the same. The significant advantage that the MRRNS technique has over the direct QRNS is the replication of the same three rings. In wafer scale systems, where it is necessary to build redundancy in order to compensate for bad dies, the redundancy replication factor is only 20% of the inner product computation array cells; for the direct QRNS, the replication factor is 100% (assuming a single replacement channel for each of the different rings). The design work is also simplified because of the replication of the same computational array.

It is clear that the MRRNS method holds structural advantages that can be exploited in massively parallel arrays, and that the price for this replication of structure is not as high as one expects, when taking into account actual silicon parameters, and may even hold (Area.Period) product advantages when yield increase, through redundancy, is taken into account.



Step	MRRNS	Direct QRNS
Forward mapping (Area.Period)	Hard-wire map of $2d_c$ bit-level components. No steps required. QRNS map: 99.2. Polynomial map: 297.6.	Residue conversion: 151.8. QRNS conversion: 96.34.
Inner Product computation (Area.Period)	7936.	6166.27.
Reverse mapping and scaling (Area.Period)	Reverse polynomial map: 297.6. Reverse QRNS map: 82.66. MRC convert: 297.6.	Reverse QRNS map: 96.34. MRC convert: 1463.52.
Total (Area.Period)	<b>9010.66</b>	<b>7974.27</b>

Table 11 (Area.Period) product comparisons for the MRRNS and direct QRNS techniques

## 9. Conclusions

This paper has introduced a new method of coding complex integers (Moduli Replication Residue Number System - MRRNS) which transforms an inner product computation to a massively parallel computation on very small rings. The mapping allows replication of the same rings to take place, unlike conventional Residue Number Systems where all the ring moduli have to be relatively prime. Although this same feature is present in computations using algebraic integers, our mapping technique is trivial, unlike that of the algebraic integer coding requirements. Rather than use mapping based on cyclotomic polynomials, and over-spanded approximations to complex numbers, our technique allows direct mapping of groups of bits in the real and imaginary parts of the complex number.

We have demonstrated flexibility in the mapping, where the sequences involved in the inner product calculation have different integer word lengths, and several examples show the power of the technique.

The paper has also made direct comparisons between the MRRNS and direct QRNS methods, and, on the basis of (Area.Period) product, we have shown that, although the MRRNS technique appears to require considerable redundancy in its implementation, the two methods require virtually the same silicon resources. When yield increase through redundancy is taken into account, the MRRNS looks much the better choice.

## 10. References

- [1] Szabo, N.S. and Tanaka, R. I. (1967) "Residue Arithmetic and Its Applications to Computer Technology", McGraw-Hill Publishing Co.
- [2] Soderstrand, M.A., Jenkins, W.K., Jullien, G.A. and Taylor, F.J. (eds) (1986) "Residue Number System Arithmetic: Modern Applications in Digital Signal Processing", IEEE Press, New York, NY.

- [3] Pollard, J.M. (1971). "The Fast Fourier Transform in a Finite Field", Math. Comp., Vol. 25, April, pp. 365-374.
- [4] W.K. Jenkins and J.V. Krogmeier (1987). "The Design of Dual-Mode Complex Signal Processors Based on Quadratic Modular Number Codes," IEEE. Trans. Circuits and Systems, (invited paper), Vol. CAS-34, No. 4, April, pp. 354-364.
- [5] G.A. Jullien, P.D. Bird, J.T. Carr, M. Taheri, W.C. Miller, (1989) "An Efficient Bit-Level Systolic Cell Design for Finite Ring Digital Signal Processing Applications," Journal for VLSI Signal Processing, (in print).
- [6] Wigley, N. and Jullien, G.A. (1989). "Packing Bits into Direct Product Representations of Finite Polynomial Rings." Submitted to IEEE Trans. on Information Theory, October.
- [7] Stouraitis, T., and Skavantzios, A., (1988). "Parallel Decomposition of Complex Multipliers." Proc. 22d. Asilomar Conf. Circ. Sys. Comp., pp. 379-383, December.
- [8] Taheri, M., Jullien, G.A. and Miller, W.C. (1988). "High Speed Signal Processing Using Systolic Arrays Over Finite Rings." IEEE Transactions on Selected Areas in Communications, VLSI in Communications III, Vol. 6, No. 3, April, pp. 504-512.
- [9] Jullien, G.A. (1978). "Residue Number Scaling and Other Operations Using ROM Arrays." IEEE Transactions on Computers, Vol. C-27, April, pp. 325-336.
- [10] Senoy, A.P., and Kumaresan, R. (1987). "Residue to Binary Conversion for RNS Arithmetic Using only Modular Look-Up Tables." Submitted for publication to IEEE Trans. Circuits and Systems

## Author Affiliation

N. Wigley, G.A. Jullien

VLSI Research Group, University of Windsor  
Windsor, Ontario, Canada N9B 3P4

## Acknowledgements

The authors acknowledge financial support from the Natural Sciences and Engineering Research Council of Canada to carry out this research work.