A MORE EFFICIENT RESIDUE ARITHMETIC IMPLEMENTATION OF THE FFT

Fred J. Taylor

Department of Electrical Engineering
University of Florida
Gainesville, FL 32611

ABSTRACT

After 20 years, the FFT remains restricted in its real time capabilities. To overcome this throughput obstacle, fast residue arithmetic units are studied based on several recent innovations in the field of complex finite rings. A dedicated machine is designed which makes use of these new results and is compared to conventional FFT designs. Using high speed semiconductor memory to implement the required residue arithmetic mappings, speed and complexity metrics of a basic FFT unit are shown to be improved. However, the derived architecture and arithmetic introduce a new and challenging set of magnitude scaling problems. They are resolved with the result being an integrated residue arithmetic FFT system capable of supporting very high real time data rates.

I. INTRODUCTION

The Discrete Fourier Transform (DFT) has for a long time been the privileged domain of the Fast Fourier Transform (FFT). Countless papers have been written on this subject with almost a like number of hardware and software products servicing this area (e.g. TRW chip sets, ILS software package etc.). The now venerable FFT owes its longevity to its relatively simple mathematical infrastructure, superb data structure, and a comparative economy of multiplications. Nevertheless, the FFT can often fall short of the real-time needs of many signal processing tasks. Since the FFT is a numeric-intensive operation, a fast real-time application requires the use of very fast, small wordlength computational elements. However, to achieve needed precision within a small wordlength environment, fast multiword procedures must be sought. One method, which has received a considerable amount of recent attention is the residue number system (RNS). This arithmetic system allows one to cover a large integer dynamic range by paralleling a system of small wordlength data paths. However, because the FFT maps complex into complex numbers, multiple real and imaginary data paths (along with their interconnection) may be required. The RNS also presents other implementation problems that are not found in weighted number systems. In this work, a synergism between recent advancements in the DFT, RNS, and the complex RNS is achieved. The advantages and disadvantages of this marriage is also developed and analyzed.

II. THE RNS

In a recent tutorial on this subject, the origin of the RNS was traced back 1700 years in history [Tay84]. It was also noted that residue number system (RNS) is of particular interest to computer scientists and engineers because of the parallel "carry-free" nature of its arithmetic [Gar59,Sza67,Tay84]. The residue representation of an integer in the residue class $Z_M (Z_M = (0,1,\ldots,M-1)$ is expressed in terms of a moduli set P where $P = (p_1,p_2,\ldots,p_L)$ is a set of relatively prime integers and $M = p_1 p_2 \cdots p_L$. It is well known that X has the unique L-tuple representation given by $X \rightarrow (X_1,X_2,\ldots,X_L)$ where $X_i = X \bmod p_i$. Alternatively, any integer $X \in [-M/2, M/2]$ can be uniquely represented as the L-tuple $(X_1,X_2,\ldots,X_L)$, where $X_i = X \bmod p_i$ if $x \geq 0$ and $(M - |X|) \bmod p_i$ if $x < 0$. If X exceeds its dynamic range limitation, a non-recoverable error due to register overflow will occur. Also, since the RNS is an integer system, division is not closed. Therefore, some operations are ackward in the RNS. For example, magnitude comparison in the RNS is known to be difficult to achieve [Gar59,Sza67,Tay84]. Finally, all reported overflow detection algorithms make use of the Chinese Remainder Theorem (CRT) and mixed-radix conversion (MRC) algorithm [Sza67].

In the mixed-radix system, any integer $X \in [0,M-1]$ can be expressed uniquely as

$$X = \overline{X}_1 + \overline{X}_2 p_1 + \overline{X}_3 p_1 p_2 + \ldots + \overline{X}_L p_1 \cdots p_{L-1} \qquad (1)$$

or equivalently, by the L-tuple $\langle \overline{X}_1, \overline{X}_2, \ldots, \overline{X}_L \rangle$ of mixed radix (MR) digits $(\overline{X}_i \in [0, p_i - 1])$. Since the mixed radix number system is a weighted number system, magnitude comparision is straight-forward but arithmetic is slow.

The celebrated Chinese Remainder Theorem, or CRT, can synthesize X from its RNS L-tuple as follows [Sza67]

$$X = \left(\sum_{i=1}^{L} m_i (m_i^{-1} X_i) \bmod p_i \right) \bmod M \qquad (2)$$

where $m_i = M/p_i$ and m_i^{-1} denotes the multiplicative inverse of m_i modulo p_i. Unlike the sequentially computed MRC, the CRT sum of product terms can be computed separately and then concurrently combined in a $\log_2 L$-level modulo M adder tree.

The major attraction of the RNS is the manner in which it performs arithmetic. For ϕ denoting the operations +, -, or *, it follows that if $X \in Z_M$, $Y \in Z_M$ and $Z \in Z_M$ (Z_M = residue class of integers modulo M), then

$$Z = X \phi Y \xrightarrow{RNS} (Z_1, \ldots, Z_L) ;$$
$$Z_i = (X_i \phi Y_i) \bmod p_i \qquad (3)$$

That is, each digit in the L-tuple representation of Z can be computed concurrently without regard to the value of the other digits. This **carry-free** arithmetic is in sharp contrast to traditional weighted number systems where the value of a digit is predicated on the value of those of lesser significance.

The RNS mappings suggested in equations 1 through 3 are not normally directly supported with conventional digital hardware. However, something better exists and it is high-speed high-density ROM or RAM. Using such devices, in ECL, bipolar, or HMOS, moduli sizes from 5 to 12 bits in width, can be achieved with data rates of 33 to 100 MOPS. The data wordwidth can be adjusted simply by adding or subtracting parallel RNS data paths.

Taylor and Huang [Tay81] detailed the operations count requirement for the radix-2 and 4 FFT, plus several other NTT-DFTs. In the case of the FFT, a high hardware and operations count was reported due principally to the overflow scaling requirement and the need to support complex arithmetic. A similar problem occurs in conventionally fixed point architecture where multiple real data paths, arithmetic, and storage must be used to support operations over a complex field. Recently, several advancements have been made to this area by Leung [Leu81], Krogmeier and Jenkins [Kro83], Soderstrand and Poe [Sod84] and Taylor et al. [Tay85]. This work relates the traditional complex RNS (or CRNS) system to that built upon quadratic (or QRNS) residues. As a result, it puts the question of applying complex RNS arithmetic into new light.

III. GENERAL FRAMEWORK FOR COMPLEX RESIDUE NUMBER

Definition 1: Let $Z[i] = \{a + ib | a,b \in Z; i = SQRT(-1)\}$. Z[i] is called a Gaussian ring whose rules of composition are:

Addition: $(a + ib) + (c + id) = (a + c) \bmod p + i(c + d) \bmod p$.

Multiplication: $(a + ib)(c + id) = (ac - bd) \bmod p + i\,(ad + bc) \bmod p$.

Definition 2: $Z_{pxp} = Z_p x Z_p$. Let (a,b), (c,d) $\in Z_{pxp}$, with rules of composition:

Addition: (a,b) + (c,d) = ((a + c) mod p, (b + c) mod p).

Multiplication: (a,b) (c,d) = (ac mod p, bd mod p).

Leung [Leu81] and Krogmeier and Jenkins [Kro83] have defined an isomorphism between $Z_p[i]$ and Z_{pxp}. It was premised on the use of **prime** moduli of the form $p_i = 4n+1$. It is known that if p is a Gaussian prime $p = 4n + 1$, then the congruence $x^2 = -1 \bmod p$ has an integer solution and has been called a quadratic residue (although this is a special case) [Her75]. Furthermore, let p be an odd integer with a prime decomposition

$$p = p_1^{e_1} p_2^{e_2} \ldots p_n^{e_n}$$

then there exists a $k \in Zp$ such that $k^2 \equiv -1 \bmod p$ if and only if each p_i of the form $4k\pm1$ (i.e., Gaussian primes) [Her75]. Finally, $k^2 \equiv -1 \bmod (p^e)$ has two solutions in Z_{p^e} if -1 is a quadratic residue of p and no solution if -1 is a quadratic nonresidue mod p [Arm80].

Thus, if p is a positive odd integer such that there exists a k in Z_M with $k^2 = -1 \bmod p$, then $Z_p[i]$ can be shown to be isomorphic to Z_{pxp} under

$$\phi = ((a+kb) \bmod p, (a-kb) \bmod p)$$
$$\phi^{-1} = (2^{-1}(x+y)) \bmod p + i(2^{-1}k^{-1}(x-y)) \bmod p \qquad (4)$$

In summary, in the QRNS, quadratic residue can be found which satisfy $x^2 = -1 \bmod p_i$ where p_i is a prime of the form $p_i = 4n + 1$. Let j_{1i} and j_{2i} denote the two quadratic residues, then it is known that j_{1i} and j_{2i} are both additive and multiplicative inverses of each other.

The utility of the QRNS is suggested in Definition 2. Observe that even though QRNS addition requires two real RNS add (like the CRNS), multiplication also requires but two real RNS operations (vs. real multiplies and two real adds in the CRNS). Since each RNS operation can be thought of as requiring a dedicated table lookup each. **The QRNS can be predicted to require 4 (i.e., 6-2) less tables per complex multiply.**

Soderstrand [Sod84] has offered another extension to this body of knowledge by introducing a so called like-QRNS type system which trades moduli choice for reduced dynamic range. The motivation for this variation on a theme is the limited QRNS moduli choice rule (i.e., Gaussian or composite Gaussian primes). To create a more robust choice of moduli, Soderstrand simply sought roots mod p that when squared yields a negative number. Since the RNS is modular system based on the roots to X^2 -amodp. In this modified system the CRNS word, x + jy is represented as

$$x + jy \dashrightarrow m + n\hat{j}\sqrt{a} \; ; \; \hat{j}^2 = -a \bmod p \qquad (5)$$

However, the dynamic range is reduced by a factor of $\sqrt{a}$ in this system (vs. $\sqrt{1} = 1$ in the QRNS). This is its principal disadvantage. It should be assumed that in most applications using complex

arithmetic, that the dynamic range for the reals should equal that for the imaginary. As a result, if the modified QRNS is used, care must be taken in the choice of $\sqrt{a}$ so as to not diminish the effective dynamic range too strongly.

IV. IMPLEMENTING DFT'S IN THE CRNS

The design of contemporary DFT systems in the RNS was originally studied by Tseng et al. [Tse79] and [Tay81] Taylor and Huang. This work remains valid today except that different complexity and throughput parameters, which reflect recent advances in RNS technology, must be used.

The traditionally more efficient radix-4 FFT is constructed with a basic computational unit which has 4 complex inputs and 4 complex outputs. Using the analysis methods presented by Taylor and Huang [Tay81], one obtains the data shown in Table 1. Here T = A + M (adds + multiplies) and S = number of scaling calls. The radix-2 RNS FFT operations are also summarized in Table 1. The architecture of a radix-4 CRNS DFT consists of the following operations per butterfly:

1. 12 real multiplies at level 1.
2. 6 real add/sub at level 2.
3. 8 real add/sub at level 3.
4. 8 real add/sub at level 4.

Now consider the QRNS architecture abstracted in Figure 1. It can be noted that the requirements are:

1) 6 real multiplies at level 1.
2) 8 real add/sub + 2 multiplies (note: $\bar{J}_i^* = -\bar{J}_i$) at level 2.
3) 8 real add/sub at level 3.

Referring again to Figure 1, it can be noted that in an integrated architecture, the subtractors S3 and S4 are combined with the multipliers M7 and M8 respectively, to define two identical hybrid units for implementing $-\bar{J}_i(a-b)$. That is, upon receipt of an address v, the precomputed value of $(-jv) \bmod p_i$ will be read from memory. If units S3, S4, M7, and M8 are implemented as distinct modular table lookup mappings, then a "split" architecture will result.

The throughput/complexity tradeoffs for the QRNS are summarized in Table 2. What is of principal interest is that for the integrated architecture, the table count decreases from 34 (CRNS) to 22 (QRNS) and is now a three stage vs. a four stage process. For a split architecture, the table lookup count increases to 24 and the level metric increases to four. The savings of 12 tables, where each table generally consists of multiple chips, represents an important reduction in system cost plus offers a board real estate savings. Finally, the savings of one level in the butterfly translates to a speed enhancement for the QRNS.

In [Tay81], the number of table lookup or computed operations, per scaling call, was computed for a then typical moduli set. The scaling algorithm was based on the standard MRC formula. The table lookup operations per scaling count, denoted A, was shown to be given by

$$\begin{aligned} &A = (L + 2K + b + 4) \\ &K = (n+1)/8 \\ &M/2 < (2^n - 1)^2 \\ &M = \text{RNS dynamic range} \\ &L = \text{number of moduli} \\ &b = \text{bound on largest moduli in bits} \end{aligned} \tag{6}$$

Other residue to decimal conversion algorithms are permissable as well. Most notable of these is the Chinese Remainder Theorem (CRT) (see eq. 2).

V. SCALING IN THE QRNS

A QRNS poses several special problems due to the fact that

1. Coupling of data prior to the formal CRT conversion is required. This is due to the need to engage in a QRNS to CRNS conversion.
2. Coupling of post CRT data is needed to reconstruct a CRNS data base.
3. Coupling of post CRNS data is required to construct a QRNS data base.

Formally, if (X_i, X_i^*) be QRNS digits of a complex integer X, the CRNS representation of the same shall be denoted (XR_i, XZ_i). The scaled version of X, say $X' \to X/M$, is given by

i) QRNS $\quad (X_i, X_i^*)$ (7i)

ii) QRNS to CRNS
$$XR_i = 2_i^{-1}(X_i + X_i^*) \qquad XI_i = 2_i^{-1} j_i^{-1}(X_i - X_i^*) \tag{7ii}$$

iii) QRNS to Decimal; $i = 1,2,\ldots,L$
$$XR = \left(\sum m_i (m_i^{-1} XR_i)_{p_i} \right) \bmod M \qquad XI = \left(\sum m_i (m_i^{-1} XI_i)_{p_i} \right) \bmod M \tag{7iii}$$

iv) Scaled Decimal (CRNS); $i = 1,2,\ldots,L$
$$XR' = \frac{XR}{M'} = \left(\sum \frac{m_i}{M'} (m_i^{-1} XR_i)_{p_i} \right) \bmod M \qquad XI' = \frac{XI}{M'} = \left(\sum \frac{m_i}{M'} (m_i^{-1} XI_i)_{p_i} \right) \bmod M \tag{7iv}$$

Note, the above equation produces fractional values of XR' and XI' which must be converted into an integer word (this will be addressed later in this section).

v) Scaled Decimal CRNS to QRNS
$$X_i' = XR_i' + \bar{J}_i XI_i' \; ; \; X_i'^* = XR_i' - \bar{J}_i XI_i' \tag{7v}$$

The key to the conversion process is the scaling by M'. A potential problem can be uncovered when equation 7(iv) is interpreted too literally (see Figure 2.a). For example, suppose $p_1 = 5$, $p_2 = 13$, $M = 65$, $M' = 10$, the unscaled CRT partial products (location a,b,c,d) would be 13, 52, 45, and 35, respectively. When paired and added, the unscaled sum would be 13+45 = 58 and 52+35 = 87 $\equiv$ 22 mod 65. If the unscaled sums are then scaled and rounded, the correct values of $[58/10]_R = 6$ and $[22/10]_R = 2$ will result. If the scaled CRT partial product outputs are used directly, then the adders would be presented with (13/10 + 45/10) = 5.8 and (52/10 + 35/10) = 8.7. Whereas the

rounded mod p value of the first sum (namely 5.8) will produce the correct result, the second will not. To correct this flaw, a modulo 6.5 adder would be needed (i.e., 5.8 mod 6.5 = 5.8 and 8.7 mod 6.5 = 2.2). However, it would require the same number of bits of precision to do a mod 6.5 add as a mod 65 add. Therefore, nothing significant is gained and, in fact, some interpretability is lost. The moral of all this is that the key to an successful RNS design is an efficient modulo M adder. For any non-trivial QRNS realization, $2n_m$ will exceed any practical table lookup address space where $n_m = [\log_2 p_{max}]^+$. Therefore, the mod M mapping will have to be computed instead of being looked up. This can pose a serious design and throughput obstacle unless this question is successfully managed. Several viable solutions are now offered and they are:

1. Direct Approach: To compute the sum S, use a fast $n > n_m$ bit adder and using combinational logic to test if S > M. If not, S mod M = S otherwise S mod M = S-M = S+K where $K = 2^n - M$ with the n_mth bit ignored. This will require a second add cycle using the same hardware. The total mod M adder delay presumably can be made to not exceed the table lookup delay.
2. Mod 2^s-2^t Method: Taylor [Tay84] has published on the design of a mod 2^s-2^t adder. In Table 3, the prime number decomposition of 2^n-1 is summarized for potentially meaningful wordlengths. These moduli could be used to build a "like" QRNS system after Soderstrand [Sod84].

Based on this analysis, a revised temporal magnitude scaling budget can be derived. The scaler proposed in [Tay81] was based on the use of the mixed radix conversion algorithm or MRC. For the CRNS realization, with respect to P = {13, 17, 23, 25, 27, 29, 31}, a temporal overhead of S = 21 TLC's was incurred per scaling call. Using the CRT, in the manner suggested, the count found in Table 4 will result.

Table 4
Speed Complexity

Item	TLC's	Table Count	No. of Operands
QRNS to Decimal (Post QRNS multiplies to CRT partial products)	1	L	2/table
Sum of CRT Partial Products mod M	$\log_2 L = L_2$	Binary Adders + Logic	2/add
Scale *	2*	2L*	$1/n_m$ bits
Reconstruct QRNS Word	1	L	2/table
Total	$4+L_2$	4L	

* estimated

The 3rd operation in Table 4 may appear to be an undesired surprise. The data presented by a mod M or equivalently mod M/M' adder, has n_m bits of significance. It may be tempting to restrict M' to be a radix -2 value, such as $M' = 2^q$ for $q = n_m/2$ with n_m even. This would simplify the scaling operation by allowing it to be realized as a binary shift. Furthermore, the product of two numbers in $Z_{M'}$ would remain in Z_M which is scaled back into $Z_{M'}$. Nevertheless, n_m or $n_m/2$ is a value which will exceed the address space requirements of contemporary semiconductor memory. Therefore, something must be done to map this long word into mod p_i counterparts.

Consider the output of the CRT subsystem to be $X \in Z_M$ (can also be interpreted in $Z_{M/M'}$). Then for admissable QRNS moduli of the form $p_i = 4k_i+1 = (2^{n_i})+1$ data conversion can be realized as follows [Ban72].

$$X_i = X \bmod p_i = \Sigma(-1)^j \alpha_j;\ j = 0,1,\ldots,n_M \qquad (8)$$

For an arbitrary p_i, such as in the like QRNS case [Sod84], the problem becomes more complex and should be treated as a table lookup statement. Suppose that tables of width n_t-bits are available for encoding and that for $X \in Z_M$,

$$X = \Sigma \alpha_i 2^{in_t};\quad i = 0, 1, \ldots, T = [n_M/n_t]^+ \qquad (9)$$

Then $X \bmod p_i = (\Sigma\beta_i) \bmod p_i$, $\beta_i = (\alpha_i 2^{in_t}) \bmod p_i$, $i = 0, 1, \ldots, T$. The values of β_i can be recombined using a custom mod p_i adder or a table lookup three of depth $\log_2 T$. As a result, for the purposes of analysis, the throughput delay of the scaler-encoder will be estimated to be $4+L_2$ TLC's. This is a moduli dependent value having the range shown in Table 5.

Table 5
Moduli Range

# of Moduli	TLC Delay Units
1-2	5
3-4	6
5-8	7
9-16	8
⋮	⋮

Thus, it should be remembered, that these figures should be compared to the 21 TLC's required of the six moduli MRC system previously reported.

VI. CRNS - QRNS COMPARISON

The main advantage of the QRNS and like QRNS systems, is a reduced multiplication budget and, through the use of the CRT, a lower scaling burden. These new parameters must be factored into the previous CRNS-DFT analysis. The data presented in the CRNS case can be updated as in Table 6.

Table 6
CRNS-QRNS Comparison (ref Table I)

without compression [Tay81]:

Level	Item	CRNS	QRNS	SCALING
1	Butterfly Adds	N(16)/4	N(16)/4	2NA
1	Butterfly Adds	N(16)/4	N(16)/4	2NA
1	Multiplies	N(12)/4	N(6)/4	
1	Recombining Adds	N(6)/4	0	
		9rN-5N	5.5rN-1.5N	2rNA

with compression: 9rN-5N 5.5rN-1.5N (2rN-4N)A

The result of all this is a reduced arithmetic complexity and scaling overhead. The previously computed CRNS overhead ratio values, in light of this analysis now becomes for a radix-4 FFT

OH = (Operations Count)/(Scaling Count)

Radix-4 FFT: OH = 2.75 (vs. 4.5 for the CRNS) (10)

Based on the data reported in Table 7, for 40 nsec TLC's, a 1024 pt FFT can be computed in less than 7 ms. However, this is not a fair appraisal of the system since some sort of block processing would normally be used. For a pipelined reqlization (which is trivial to achieve in the RNS), a transform rate of 25 MFFTs/sec. can be achieved. However, the possibility of a prohibitly high hardware cost must be born. The truth lies somewhere in between.

One last remark on the data reported in Table 7. A hybrid parameter called "work factor" has been developed. It measures, as a ratio, the amount of time used for DFT arithmetic versus magnitude scaling. The larger this number, the better. All radix-4 FFTs tested are below unity. However, smaller length FFTs exhibit a comparatively superior work factor.

VII. SUMMARY AND RECOMMENDATIONS

The significant advancements made in residue arithmetic warranted a reinvestigation of RNS-DFTs reported in the late 70's and early 80's. The data and analysis reported in this paper supports earlier claims that the RNS may represent a superior FFT architecture. This thesis is historically questionable due to a high scaling incumberance. In previous analysis the computational complexity of a scaling operation was quite high (typ. 20-25 lookup cycles). In this work a signficant reduction in scaling complexity has been reported (4-fold decrease typ.). Also, the introduction of the QRNS and like QRNS systems significantly reduce the FFT arithmetic count. This in turn translates to a major reduction in arithmetic hardware (which in this case is almost one to one with table lookup semiconductor memory chips). However, the reduced arithmetic count artificially inflates the overhead scaling ratio. This, by itself, is of no concern. However, it is shown in this work that the architecture of the QRNS scaler is moderately complex. The key to fast CRT converters/scalers is reported to be an efficient mod M adder. Based on this analysis, it would appear that a CRT based CRNS autoscaler can be designed to run in two lookup cycles in limited hardware. This is offset by the fact that twice as many CRNS autoscalers than QRNS scalers are needed. Therefore, the complexity issue becomes a "wash". It can also be seen that a radix-4 QRNS butterfly cycle can be as short as three memory cycles vs. four for a CRNS design which the CRNS butterfly unit is about 20% more complex. Therefore, as a result, the following observation are made

1. The conventional CRNS-FFT will probably be slightly more complex but run slightly faster than its QRNS counterpart.
2. The key problem, in all designs considered, is that of implementing fast autoscaler. Several viable approaches are recommended.

Based on this analysis, it can be seen that the RNS-FFT approach has become increasingly attractive and, for modest transform lengths, can be realized in fast hardware. It would also appear that either a conventional QRNS or QRNS based design would be represented in a viable prototype development project.

Acknowledgements: The author wishes to express his appreciation to the help of Mr. Sousa, Papadourakis, Skavantzos, and Stouraitis on this project.

BIBLIOGRAPHY

Gar59 H.L. Garner, "The Residue Number System," IRE Trans. Electronic Computers, Vol EC-8, pp 140-147, June 1959.

Sza67 N.S. Szabo and R.I. Tanaka, Residue Arithmetic and its Applications to Computer Technology, New York: McGraw-Hill, 1967.

McC79 J.H. McClellan and C. M. Rader, Number Theory in Digital Signal Processing, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1979.

Tay83 F.J. Taylor, Digital Filter Design Handbook, Marcel-Dekker, 1983.

Jen77 W.K. Jenkins and B.J. Leon, "The Use of Residue Number Systems in the Design of Finite Impulse Response Digital Filters," IEEE Trans. Circuits and Systems, Vol. CAS-24, No. 4, pp. 191-201, April, 1977.

Tay81 F.J. Taylor and A.S. Ramnarayanan, "An Efficient Residue-to-Decimal Converter," IEEE Trans. Circuits and Systems, Vol. CAS-28, No.12, pp. 1164-1169, Dec. 1981.

Tay82 F.J. Taylor and C.H. Huang, "An Autoscale Residue Multiplier," IEEE Trans. Computer, Vol. C-31, No. 4, pp. 321-325, April 1982.

Jul78 G.A Jullien, "Residue Number Scaling and Other Operations Using ROM Arrays," IEEE Trans. Computers, Vol. C-27, No. 4, pp. 325-336, April, 1978.

Aga78 D.P. Agarwal, "Modulo 2**n+1 Arithmetic Logic," IEEE J. Electronic Circuits and Systems, Vol. 2, No. 6, pp. 186-188, Nov. 1978.

Ram80 Ramnarayan, "Practical Realization of Mod p, p Prime Multiplier," Electronics Letters, Vol. 19(15), pp. 466-467, June 1980.

Sod80 M.A. Soderstrand and C. Vernia, "General Moduli P Multiplier with RNS Arithmetic Applications," Proc. of the 1980 IEEE Intl. Symposium on Circuits and Systems, pp. 375-378, April 28-30, 1980.

Leu81 Shu-Hung Leung, "Application of Residue Number System to Complex Digital Filters," Proc. 15th Asilomar Conf. Circuits Systems and Computers, Pacific Grove, CA, pp. 70-74, Nov. 1981.

Kro83 J.J. Krosmeier and W.K. Jenkins, "Error Detection and Correction in the Quadratic Number System," Proc. 26th Midwest Symp. on Ckts. and Sys., Pueblo, MX, 1983.

Sod84 M.A. Soderstrand and G.D. Poe, "Applications of a Quadratic-Like Complex Residue Number System to Ultrasonics," Proc. 1984 ICASSP, San Diego, CA, March 1984.

Bar79 A. Baraniecka and G.A. Jullien, "Hardware Implementation of Convolution Using Number Theoretic Transforms," Proc. 1979 IEEE Intl. Conf. on Acoustics, Speech and Signal Processing, pp. 490-494, April 2-4, 1979.

Tse79 B.D. Tseng, G.A. Jullien and W.C. Miller, "Implementation of FFT Structures Using the Residue Number System," IEEE Trans. Computers, Vol. C-28, No.11, pp. 831-845, Nov. 1979.

Tay81 F.J. Taylor and C.H. Huang, "A Comparison of DFT Algorithms Using a Residue Architecture," Computer and Electrical Engineering (England), Vol. 8, No. 3, pp. 161-171, Sept. 1981.

Her75 I.N. Herstein, Topics in Algebra, 2nd ed., Southwest Book Co., 1975.

Arm80 E.P. Armendariz and S. McAdam, Elementary Number Theory, MacMillan, 1980.

Tay84 F.J. Taylor, "Residue Arithmetic: A Tutorial with Examples," IEEE Computer Magazine, May 1984.

Tay84i F.J. Taylor and M. Sousa, "On the Structure of the Complex Residue Number System," submitted to IEEE Trans. on Compt., Nov. 1984.

Ban72 D.K. Banerji and T.A. Brzozowski, "On Translation Algorithms in Residue Number Systems," IEEE Trans. on Compt., C-21, No. 12, pp. 1281-1286, Dec. 1972.

Tay85 F.J. Taylor et al., "A Radix-4 FFT Using Complex Arithmetic, (in print for IEEE T. on Compt.).

ITEM / RADIX	T (Adds and Multiplies) 2	T (Adds and Multiplies) 4	S (Scaling Calls) 2	S (Scaling Calls) 4
Without Absorption	6Nr-4N	9Nr-5N	2Nr	2Nr
With Absorption	6Nr-4N	9Nr-5N	2Nr-4N	2Nr-4N

Absorption implies that input/output magnitude scaling operations are imbedded in first/last stage arithmetic operations.

Complex Input FFT Table Lookup Count

Table 1

DELAY Δns	#OF MULT	#OF ADD	#OF SUB	ORGANIZATION AT	MULT	ADD	SUB	LEVELS
Δ	12	11	11	Δ	1,2,...,12	1,2,3,4,5,6,7,8,...,11	1,2,3,4,5,6,7,8,...,11	4
2Δ	6	6	6	Δ	1,2,...,6	1,2,4,5,7,9	1,4,5,8,9	4
				2Δ	7,8,...,12	3,6,7,10,11	2,3,6,7,10,11	
4Δ	3	3	3	Δ	1,2,3	1,4,9	4,8	4
				2Δ	4,5,6	2,6,10	1,5,9	
				3Δ	7,8,9	3,5,11	2,6,10	
				4Δ	10,11,12	7,8	3,7,11	
8Δ	2	2	2	Δ	1,9	1,9	6	4
				2Δ	2,10	2,10	7	
				3Δ	3,11	3,11	8	
				4Δ	4,12	4	1,9	
				5Δ	5	5	2,10	
				6Δ	6	6	3,11	
				7Δ	7	7	4	
				8Δ	8	8	5	

TABLE 2a: Number of Arithmetic Hardware Units Required vs. Throughput for a Radix-4 CRNS FFT Butterfly

DELAY Δns	#OF MULT	#OF ADD	#OF SUB	#OF SCALARS	ORGANIZATION AT	MULT	ADD	SUB	-%	LEVELS
Δ(#)	6	8	6	2	Δ	1,2,...,6	1,2,3,4,5,6,7,8	1,2,5,6,7,8	(S3,M7)(S4,M8)	3
Δ(!)	8	8	8	0	Δ	1,2,...,6,7,8	1,2,3,4,5,6,7	1,2,3,4,5,6,7,8	NO	4
2Δ(#)	3	4	3	1	Δ	1,3,5	1,3,5,7	1,5,7	(S3,M7)	3
					2Δ	2,4,6	2,4,6,8	2,6,8	(S4,M8)	
2Δ(!)	4	4	4	0	Δ	1,3,5,7	1,3,5,7	1,3,5,7	NO	4
					2Δ	2,4,6,8	2,4,6,8	2,4,6,8	NO	
4Δ(#)	2	2	2	1	Δ	1,5	1,5	1,5	NO	3
					2Δ	2,6	2,6	2,6	NO	
					3Δ	3	3,7	7	(S3,M7)	
					4Δ	4	4,8	8	(S4,M8)	
4Δ(!)	2	2	2	0	Δ	1,5	1,5	1,5	NO	4
					2Δ	2,6	2,6	2,6	NO	
					3Δ	3,7	3,7	3,7	NO	
					4Δ	4,8	4,9	4,8	NO	
8Δ(#)	1	1	1	1	Δ	1	1	1	NO	3
					2Δ	2	2	2	NO	
					3Δ	3	3	5	NO	
					4Δ	4	4	6	NO	
					5Δ	5	5	7	NO	
					6Δ	6	6	8	NO	
					7Δ	NO	7	NO	(S3,M7)	
					8Δ	NO	8	NO	(S4,M8)	
8Δ(!)	1	1	1	0	Δ	1	1	1	NO	4
					2Δ	2	2	2	NO	
					3Δ	3	3	3	NO	
					4Δ	4	4	4	NO	
					5Δ	5	5	5	NO	
					6Δ	6	6	6	NO	
					7Δ	7	7	7	NO	
					8Δ	8	8	8	NO	

\# combined adder(subtractor)/scalar (i.e., hybrid units (S3,M7) and (S4,M8) do $(-J_i(a-b))\text{mod}p_i$)
! split architecture separate adder(subtractor)/scalar (i.e., distinct S8,S4,M7,M8 units)
NO: No operation -% = $-J_i(a-b)\text{mod}p_i$

TABLE 2b: Number of Arithmetic Hardware Units Required vs. Throughput for a Radix-4 QRNS FFT Butterfly

PRIME FACTORS OF 2^q-1

$q \in [12,30]$

$2^{12}-1=3*3*5*7*13$
$2^{13}-1=8191\#$
$2^{14}-1=3*43*127$
$2^{15}-1=7*31*257$
$2^{16}-1=13*17*257$
$2^{17}-1=131071\#$
$2^{18}-1=3*3*3*7*19*73$
$2^{19}-1=524287\#$
$2^{20}-1=3*5*5*11*31*41$
$2^{21}-1=77*127*337$
$2^{22}-1=3*23*89*683$
$2^{23}-1=3*23*89*683\#$
$2^{24}-1=47*178481\#$
$2^{25}-1=31*601*1801\#$
$2^{26}-1=3*2731*8191\#$
$2^{27}-1=7*73*262657\#$
$2^{28}-1=3*5*29*43*113*127$
$2^{29}-1=233*1103*2089\#$
$2^{30}-1=3*3*7*11*31*151*331$

Prime Factorization

Table 3

\# too large of moduli for use with commercially available high-speed memory

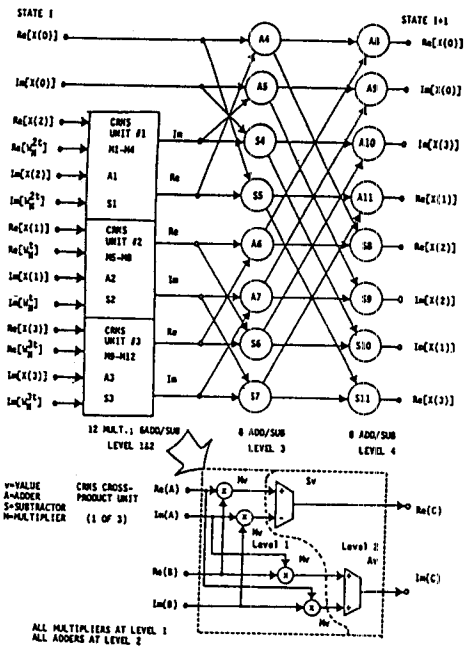
Table 7

Numerical Summary of QRNS

L MODULI QRNS or like QRNS OFTs

N POINTS	TYPE	ARITH AR	SCALING SC*A	MODULI L	SCALING MULT. A	TOTAL OPERATIONS T = AR + SC*A	WORK FACTOR WF = AR/(SC*A)
128	radix-4 FFT	4927	1280*A	1-2	5	11327	.77
				3-4	6	12607	.74
				5-8	7	13887	.55
512	radix-4 FFT	24576	7168*A	1-2	5	60416	.68
				3-4	6	67584	.57
				5-8	7	74752	.48
1024	radix-4 FFT	54784 [54784]	16384*A [16384]*[A]	1-2	5	136524	.67
				3-4	6	153088	.56
				5-8	7	169472[436224]	.47
2048	radix-4 FFT	120832	36864*A	1-2	5	303152	.65
				3-4	6	342016	.54
				5-8	7	378880	.47

[ ] denotes requirement based on previously published MRC scaler.



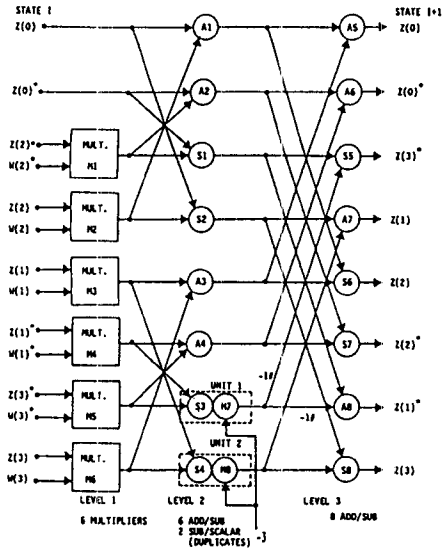


FIGURE 1:

Far left:
CRNS RADIX-4 FFT Butterfly Unit

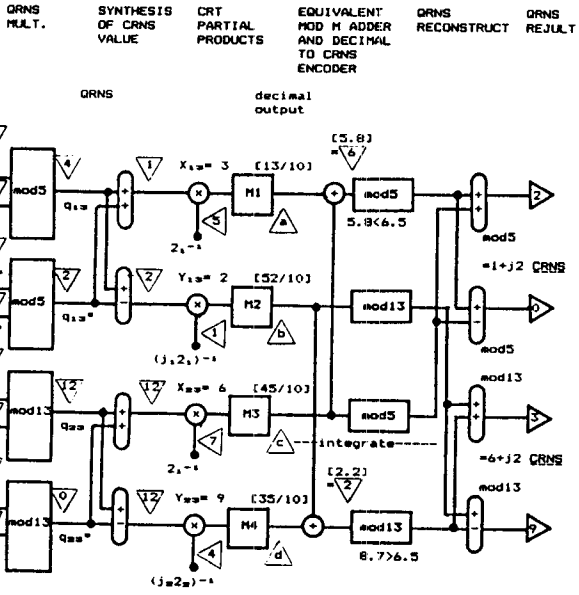
Near left:
QRNS RADIX-4 FFT Butterfly Unit

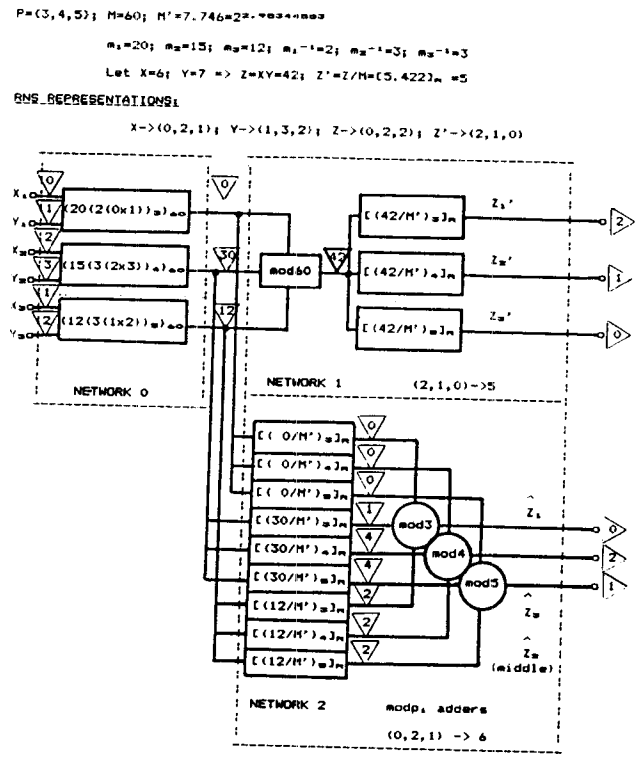
note: the reduced complexity of the QRNS version

FIGURE 2:

Below left:
Scaled CRT Options

Below near left:
QRNS Autoscale Model





Summary:

M1=(m₁(m₁⁻¹X₁₂)mod5)/10
M2=(m₁(m₁⁻¹Y₁₂)mod13)/10
M3=(m₂(m₂⁻¹X₂₂)mod5)/10
M4=(m₂(m₂⁻¹Y₂₂)mod13)/10

Given: 5.8+j2.2=6+j2 (approx.)

I: QRNS CONSTRUCTION
- i: (5.8+3(2.2))mod5 = 2.4 ->2
- ii: (5.8-3(2.2))mod5 = -.9 ->4
- iii: (5.8+5(2.2))mod13= 3.8 ->4
- iv. (5.8-5(2.2))mod13=-5.8 ->8

II: RECONSTRUCTION
- i: X1=(2₁⁻¹(2+4))mod5 ->3
- ii: Y1=((2₁j₁)⁻¹(2-4))mod5 ->3
- iii: X2=(2₂⁻¹(4+8))mod13 ->6
- iv: Y2=((2₂j₂)⁻¹(4-8))mod13->10

≠ CRNS

This work was supported under an ARO grant.