A MORE EFFICIENT RESIDUE ARITHMETIC IMPLEMENTATION OF THE FFT

Fred J. Taylor

Department of Electrical Engineering
University of Florida
Gainesville, Florida 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 multiplier is designed which makes use of these new results and, in contrast 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. These 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 new 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 multilword 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 \( Z_m \), \( m = \left\{ 0, 1, \ldots, M-1 \right\} \) 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 \( X = \{x_1, x_2, \ldots, x_l\} \) where \( x_i = X \mod p_i \). 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 \( X = x \mod M \) if \( x \geq 0 \) and \( M - |X| \mod M \) 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 awkward 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

\[
X = \bar{x}_1 + \bar{x}_2 p_1 + \bar{x}_3 p_1 p_2 + \cdots + \bar{x}_l p_1 \cdots p_{l-1}
\]

(1)

or equivalently, by the \( l \)-tuple \( \langle \bar{x}_1, \bar{x}_2, \ldots, \bar{x}_l \rangle \)

of mixed radix (MR) digits \( \{x_i \in [0, p_{i-1}-1]\} \). Since the mixed radix number system is a weighted number system, magnitude comparison is straightforward but arithmetic is slow.

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

\[
X = \sum_{i=1}^{l} n_i (m_i^{-1} x_i) \mod p_i \mod M
\]

(2)

CH2146-9/85/0000/0243$01.00 © 1985 IEEE
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 \( \log_p \)-level modulo \( M \) adder tree.

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

\[
Z = X \oplus Y \quad \text{mod} \quad p_1 \\
Z = (X \oplus Y) \mod p_1
\]

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 the 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 CMOS, moduli sizes from 8 to 12 bits in width, can be achieved with data rates of 33 to 100 MOPS. The data wordlength 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], Krogh 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: \((a+ib) + (c+id) = (a+c) \mod p + i(c+d) \mod p\).

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

**Definition 2:** \( Z_{p^i} \equiv Z \). Let \((a,b), (c,d) \in Z_{p^i}\), 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 Krogh and Jenkins [Kro83] have defined an isomorphism between \( Z[i] \) and \( Z_{p^i} \). It was premised on the use of prime moduli of the form \( p_i = 4n+1 \). It is known if \( p \) is a Gaussian prime \( p = 4n+1 \), then the congruence \( x^2 \equiv -1 \mod p \) 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

\[ p = p_1^\alpha_1 p_2^\alpha_2 \cdots p_n^\alpha_n \]

then there exists a \( k \in \mathbb{Z}_p \) such that \( k^2 \equiv -1 \mod p \) if and only if each \( p_i \) of the form \( 4k+1 \) (i.e., Gaussian primes) [Her75]. Finally, \( k^2 \equiv -1 \mod (p^i) \) has two solutions in \( \mathbb{Z}_p \) 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 \( k \in \mathbb{Z}_p \) with \( k^2 \equiv -1 \mod p \), then \( Z[i] \) can be shown to be isomorphic to \( Z_{p^i} \) under

\[ \phi = ((a+kb) \mod p, (a-b) \mod p) \]

\[ \phi^{-1} = (2^{-1}(x+y)) \mod p + i(2^{-1}k^{-1}(x-y)) \mod p \]

In summary, the QRNS, quadratic residue can be found which satisfy \( x^2 \equiv -1 \mod p \) where \( p \) is a prime of the form \( p_i = 4n+1 \). Let \( j_1 \) and \( j_2 \) denote the two quadratic residues, then it is known that \( j_1 \) and \( j_2 \) 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 modular 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 \( X^2 = -1 \mod p \). This modified system the CRNS word, \( x + jy \) is represented as

\[ x + jy \rightarrow m + nj \sqrt{a} ; j^2 = -1 \mod p \]

However, the dynamic range is reduced by a factor of \( \sqrt{a} \) in this system (vs. \( \sqrt{a} = 1 \) 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 \( N \) so as to not diminish the effective dynamic range too strongly.

IV. IMPLEMENTING DFT'S IN THE CRNS

The design of contemporary DFT systems is of existing in the CRNS was originally studied by Tseng et al. [Ts79] and [Tay81] Taylor and Huang. This work remains valid today except that different complexity and throughput parameters, which reflect recent advances in CRNS 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 method proposed by Taylor and Huang [Tay81], one obtains the data shown in Table 1. Here \( T = A + M \) (adds + multipies) and \( S = \text{number of scaling calls}. \) 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 0.
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: \( j_1^* = -j_1 \)) 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 subtrahsors \( S_3 \) and \( S_4 \) are combined with the multipliers \( M_7 \) and \( M_8 \), respectively, to define two identical hybrid units for implementing \(-j_1(a-b)\). That is, upon receipt of an address \( v \), the precomputed value of \((-jv) \text{ mod } p_i\) will be read from memory. If units \( S_3, S_4, M_7, \) and \( M_8 \) are implemented as distinct modular table lookup mappings, then a "split" architecture will result.

The throughput/multiplier 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 third 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

\[
A = (L + 2K + b + 4)
\]

\[
K = (n + 1)/8
\]

\[
M/2 < (2^n - 1)
\]

\[
L = \text{number of moduli}
\]

\[
b = \text{bound on largest moduli in bits}
\]

Other residue to decimal conversion algorithms are permissible 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 CRNS 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 \((x_i, x_i')\). The scaled version of \( X \), say \( x_i/X/M \), is given by

\[
i \quad \text{QRNS} \quad (x_i, x_i') \quad (7i)
\]

\[
i \quad \text{QRNS to CRNS} \quad X_{Ri} = 2^{L_i-1}(x_i + x_i') \quad (7ii)
\]

\[
X_{RI} = 2^{L_i-1}(x_i - x_i') \quad (7iii)
\]

\[
\text{QRNS to Decimal; } i = 1,2,...,L
\]

\[
X_R = (\sum \frac{m_i}{1024}x_Ri) \mod M \quad (7iv)
\]

\[
X_I = (\sum \frac{m_i}{1024}x_II) \mod M \quad (7v)
\]

\[
\text{Scaled CRNS (CRNS); } i = 1,2,...,L
\]

\[
X_R' = X_R \quad (7vi)
\]

\[
X_I' = X_I \quad (7vii)
\]

Note, the above equations produces fractional values of \( X_R' \) and \( X_I' \) which must be converted into an integer word (this will be addressed later in this section).

v) Scaled Decimal CRNS to QRNS

\[
X_i = X_R + j_3X_I; \quad X_i^* = X_R - j_3X_I \quad (7v)
\]

The key to the conversion process is the scaling by \( M' \). A potential problem can be uncovered when equation 7(vi) is interpreted too literally (see Figure 2a). 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+52 = 58 and 52+35 = 87 = 22 mod 65. If the unscaled sums are then scaled and rounded, the correct values of [58/10] = 6 and [22/10] = 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 65 adder would be needed (i.e., 5.8 and 8.7 mod 65 = 2.2). However, it would require the same number of bits of precision to add a mod 65 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 a successful RNS design is an efficient modulo M adder. For any non-trivial QRNS realization, \( 2^n \) will exceed any practical table lookup address space where \( n = \lceil \log_2 n_{\text{max}} \rceil + 1 \). 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 \( n > n_{\text{max}} \) bit adder and use combinatorial logic to test if \( S > M \). If not, \( S \) mod \( M = S - M = S + K \) where \( K = 2^n - M \) with the mod 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 \( 2^k - 2 \) Method: Taylor [Tay84] has published on the design of a mod \( 2^k - 2 \) adder. In Table 3, the prime number decomposition of \( 2^{2n} - 1 \) 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 [Tay84] was based on the use of the mixed radix conversion algorithm or MRC. For the QRNS 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.

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 \( n_{\text{max}} \) bits of significance. It may be tempting to restrict \( M' \) to be a radix \( -2 \) value, such as \( M' = 2^q \) for \( q = n_{\text{max}}/2 \) with \( n_{\text{max}} \) 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, \( n_{\text{max}} \) or \( n_{\text{max}}/2 \) 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 admissible QRNS moduli of the form \( p_i = \lambda_i \cdot 2^{n_i} + 1 \) data conversion can be realized as follows [Ban72].

\[
X_i = X \mod p_i = 2^{(-1)^j a_j} j = 0, 1, ..., n_M
\]

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 \( n_i \) bits are available for encoding and that for \( X \in Z_M \):

\[
X = \lambda_i 2^{n_i}; i = 0, 1, ..., T = \lceil n_M/n_i \rceil
\]

Then \( X \mod p_i = (\lambda_i \mod p_i), \lambda_i = (\lambda_i 2^{n_i}) \mod p_i, j = 0, 1, ..., T \). The values of \( \lambda_i \) 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-converter will be estimated to be \( 4L \) TLC's. This is a moduli dependent value having the range shown in Table 5.

<table>
<thead>
<tr>
<th># of Moduli</th>
<th>TLC Delay Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-2</td>
<td>5</td>
</tr>
<tr>
<td>3-4</td>
<td>6</td>
</tr>
<tr>
<td>5-8</td>
<td>7</td>
</tr>
<tr>
<td>9-16</td>
<td>8</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

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 1)

<table>
<thead>
<tr>
<th>Level Item</th>
<th>CRNS</th>
<th>QRNS</th>
<th>SCALING</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Butterfly Adds</td>
<td>(N/16)4</td>
<td>(N/16)4</td>
<td>2NA</td>
</tr>
<tr>
<td>1 Butterfly Adds</td>
<td>(N/16)4</td>
<td>(N/16)4</td>
<td>2NA</td>
</tr>
<tr>
<td>1 Multiples</td>
<td>(N/12)4</td>
<td>(N/6)4</td>
<td>Q</td>
</tr>
<tr>
<td>1 Recombining Adds</td>
<td>(N/6)4</td>
<td>(N/6)4</td>
<td>Q</td>
</tr>
</tbody>
</table>

with compression: 9RN-5N 5.5RN-1.5N 2RN-4N

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

\[
\text{OH} = \frac{(\text{Operations Count})}{(\text{Scaling Count})} 
\]

Radix-4 FFT: \( \text{OH} = 2.75 \) (vs. 4.5 for the CRNS)

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 realization (which is trivial to achieve in the CRNS), a transfer rate of 25 MFFS/sec. can be achieved. However, the possibility of a prohibitively 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 inaccuracy. In previous analysis the computational complexity of a scaling operation was quite high (typ. 20-25 lookup cycles). In this work a significant 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 autoescalor can be designed to run in two lookup cycles in limited hardware. This is offset by the fact that twice as many CRNS 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 autoescalor. 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

### ITEM RADIX

<table>
<thead>
<tr>
<th>T (Adds and Multiplies)</th>
<th>2</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without Absorption</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6N-4N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9N-5N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2Nn</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2Nn</td>
<td></td>
<td></td>
</tr>
<tr>
<td>With Absorption</td>
<td></td>
<td>4</td>
</tr>
<tr>
<td>6N-4N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9N-5N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2Nn-4N</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2Nn-4N</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

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

### Complex Input FFT Table Lookup Count

<table>
<thead>
<tr>
<th>DELAY</th>
<th>#OF</th>
<th>#OF</th>
<th>#OF</th>
<th>ORGANIZATION</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MUL</td>
<td>ADD</td>
<td>SUB</td>
<td>AT</td>
</tr>
<tr>
<td>5</td>
<td>12</td>
<td>11</td>
<td>11</td>
<td>4</td>
</tr>
<tr>
<td>26</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>7,8,..,12</td>
<td>3,6,7,10,11</td>
<td>2,3,6,7,10,11</td>
</tr>
<tr>
<td>48</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>4,5,6</td>
<td>5,6,10</td>
<td>5,6,7,10,11</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>7,8,9</td>
<td>5,11</td>
<td>5,6,10</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>10,11,12</td>
<td>7,8</td>
<td>5,6,10</td>
</tr>
<tr>
<td>16</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>2</td>
<td>2,10</td>
<td>2,10</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>3,11</td>
<td>3,11</td>
<td>8</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>4,12</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
</tbody>
</table>

**TABLE 2a:** Number of Arithmetic Hardware Units Required vs. Throughput for a Radix-4 CoNS FFT Butterfly
<table>
<thead>
<tr>
<th>DELAY</th>
<th>#OF MULT</th>
<th>#OF ADD</th>
<th>#OF SUB</th>
<th>#OF SCALARS</th>
<th>ORGANIZATION AT MULT</th>
<th>ADD SUB</th>
<th>-k</th>
<th>LEVELS</th>
</tr>
</thead>
<tbody>
<tr>
<td>a(#)</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>2</td>
<td>s 1.2, ... 6</td>
<td>1.2,3,4,5,6,7,8</td>
<td>1.2,3,4,5,6,7,8</td>
<td>(53,77)(54,66)</td>
</tr>
<tr>
<td>b(#)</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>0</td>
<td>s 1.2, ... 8</td>
<td>1.2,3,4,5,6,7,8</td>
<td>1.2,3,4,5,6,7,8</td>
<td>NO</td>
</tr>
<tr>
<td>a(#)</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>1</td>
<td>s 1.3</td>
<td>1.3,5,7</td>
<td>1.3,5,7</td>
<td>NO</td>
</tr>
<tr>
<td>a(#)</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>s 1.2,3,4</td>
<td>1.2,3,4,5,6,7</td>
<td>1.2,3,4,5,6,7</td>
<td>NO</td>
</tr>
<tr>
<td>4b(#)</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>s 1.2,3,4,5,6,7,8</td>
<td>1.2,3,4,5,6,7</td>
<td>1.2,3,4,5,6,7</td>
<td>NO</td>
</tr>
<tr>
<td>8b(#)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>s 1.2,3,4,5,6,7,8</td>
<td>1.2,3,4,5,6,7</td>
<td>1.2,3,4,5,6,7</td>
<td>-k</td>
</tr>
<tr>
<td>8b(#)</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>s 1.2,3,4,5,6,7,8</td>
<td>1.2,3,4,5,6,7</td>
<td>1.2,3,4,5,6,7</td>
<td>NO</td>
</tr>
</tbody>
</table>

# combined adder(subtractor)/scalar (i.e., hybrid units (53,77) and (54,66) do {-1,(a-b)}mod2)
split architecture separate adder(subtractor)/scalar (i.e., distinct 53,54,77,78 units)
NO: No operation -k = [-1,(a-b)]mod2

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

PRIME FACTORS OF 2^4-1
q ∈ {17,30}

<table>
<thead>
<tr>
<th>p</th>
<th>Prime Factorization</th>
</tr>
</thead>
<tbody>
<tr>
<td>12 = 2^2 * 3</td>
<td>22^2, 1+31 = 89 * 603</td>
</tr>
<tr>
<td>13 = 2^3 * 3</td>
<td>22^3, 1+3 = 29 * 833</td>
</tr>
<tr>
<td>18 = 2^3 * 3</td>
<td>22^3, 1+4 = 17608</td>
</tr>
<tr>
<td>31 = 2^3 * 13</td>
<td>22^3, 1+5 = 3011</td>
</tr>
<tr>
<td>36 = 2^3 * 31</td>
<td>22^3, 1+6 = 3046</td>
</tr>
<tr>
<td>42 = 2^3 * 31</td>
<td>22^3, 1+7 = 3047</td>
</tr>
<tr>
<td>51 = 2^3 * 31</td>
<td>22^3, 1+8 = 3048</td>
</tr>
</tbody>
</table>

Prime Factorization

Table 3

Table 7: Numerical Summary of URNS

<table>
<thead>
<tr>
<th>N POINTS</th>
<th>TYPE</th>
<th>AN WITH SCALA *</th>
<th>SCALING</th>
<th>MODULI L</th>
<th>SCALING</th>
<th>TOTAL OPERATIONS</th>
<th>WORK FACTOR</th>
</tr>
</thead>
<tbody>
<tr>
<td>128</td>
<td>radix-4 FFT</td>
<td>4927</td>
<td>1280 * A</td>
<td>1-2</td>
<td>5</td>
<td>11327</td>
<td>.77</td>
</tr>
<tr>
<td></td>
<td>radix-4 FFT</td>
<td>2976</td>
<td>716 * A</td>
<td>5-6</td>
<td>7</td>
<td>13687</td>
<td>.59</td>
</tr>
<tr>
<td>1024</td>
<td>radix-4 FFT</td>
<td>54784</td>
<td>16384 * A</td>
<td>1-2</td>
<td>5</td>
<td>13682</td>
<td>.59</td>
</tr>
<tr>
<td>2048</td>
<td>radix-4 FFT</td>
<td>120832</td>
<td>36864 * A</td>
<td>5-6</td>
<td>7</td>
<td>37888</td>
<td>.47</td>
</tr>
</tbody>
</table>

[ ] denotes requirement based on previously published MRC scaler.

249
This work was supported under an ARO grant.