IMPLEMENTATION OF THE SINGLE MODULUS COMPLEX ALU

Rom-Shen Kao
Fred J. Taylor

University of Florida

ABSTRACT

Recently the complex residue number system, or CIN, has been the subject of intense study. One special embodiment of this theory is the single modulus complex RNS processor which suggests both implementation and performance advantages. In this paper the conjectures are tested in the context of a CMOS gate array design and are found to be valid.

This work was supported under an ARO grant.

I. INTRODUCTION

Over the past several years there has appeared numerous articles advocating the use of the quadratic residue number system or QRNS, for performing complex algebraic operations [1-6]. As a subset of the more general theory of the residue number system (RNS), it has been shown to offer low multiplication complexity but, like other RNS tools, suffers from an inability to efficiently service scaling, magnitude comparison, or number system conversion calls. In an attempt to overcome this objectionable feature, the single modulus QRNS, or SM-QRNS was proposed [6]. Based on Fermat primes \( p = 2^n + 1 \) for \( n = 2, 4, 8, 16 \), the SM-QRNS was shown to offer several advantages over other QRNS embodiments. They were:

i) compared to the general QRNS
   - scaling greatly simplified
   - magnitude comparison greatly simplified
   - sign detection trivial
   - capable of assimilating existing high performance radix-2 hardware into its design

ii) compared to the RNS and conventional binary weighted systems
   - reduced computational complexity
   - reduced computational latency

II. THE SM-QRNS

Let \( \mathbb{Z}_p[i] = \{a+ib | a, b \in \mathbb{Z}_p, i = \text{SQRT}(-1)\} \). The ring \( \mathbb{Z}_p[i] \) is called the ring of Gaussian integers modulo \( p \). A complex CRNS number of the form \( a+ib \) evaluates to \( i \) if \( a = 0 \) and \( b = 1 \). If \( i \notin \mathbb{Z}_p \), then it is considered to be \textbf{imaginary}. That is, there is \( i \not\in \mathbb{Z}_p \) such that \( i^2 = -1 \) mod \( p \) and \( i \) is historically called a nonquadratic root and \(-1\) is a nonquadratic residue. However, if \( i \in \mathbb{Z}_p \), then \( i \) is a \textbf{quadratic root} and \( a+ib \) is real. For example, if \( p = 17 \) and \( i = 4 \in \mathbb{Z}_p \), then \( i^2 = 16 = -1 \) mod 17. For such moduli the concept of an imaginary operation is void. Now define \( Z(p,p) = \mathbb{Z}_p \times \mathbb{Z}_p \) and \((a, b), (c, d) \in Z(p,p) \) with rules of composition given by:

- Addition:
  \[(a, b) + (c, d) = ((a + c) \mod p, (b + d) \mod p)\]
- Multiplication:
  \[(a, b)(c, d) = (ac \mod p, bd \mod p)\]

(1)

Even though addition requires two (2) real adds (as in the CRNS), multiplication requires but two (2) real multiplies (versus 4 products and 2 adds in the CRNS)! The isomorphism between \( \mathbb{Z}_p[i] \) and \( \mathbb{Z}(p,p) \) has been well studied and for \( p \) a Gaussian prime satisfying \( p = 4k + 1 \) (in fact, it can be a product of Gaussian primes). The isomorphism given by \( \phi \), maps a complex number \( z = a + ib \) (\( i = \text{SQRT}(-1) \)) into a two tuple \((z, z^*)\) as follows:

\[
(a, b) \xrightarrow{\phi} (z, z^*)
\]
\[z = (a + jb) \mod p;
\]
\[z^* = (a - jb) \mod p;
\]
\[j^2 = -1 \mod p; j \in \mathbb{Z}_p
\]
\[a = 2^{-1}(z + z^*) \mod p;
\]
\[b = (2j)^{-1}(z - z^*) \mod p
\]

(2)


In (2), \( j \) denotes a quadratic root. In fact there are two such roots (say \( j_1 \) and \( j_2 \)) which are both additive and multiplicative inverses of each other.

A specific choice of Gaussian primes are those which are also Fermat primes of the form \( p = 2^n + 1 \), \( n = 2, 4, 8, 16 \). For the case where \( n = 32 \), \( p \) is a composite Gaussian prime and therefore also an admissible modulus. As such, the critical parameters found in equation 2 take the form:

\[z^2 = 2^{n-1} + 1; j = 2^n/2;
\]
\[j^2 = 2^n + 1 - 2^n/2
\]

(3)

which can be seen to be the essentially radix 2 scaling operations. Based on this result, a SM-QRNS of the type suggested in Figure 1 can be designed. The functional elements will be summarized in the next section.
III. SM-QRNS ARCHITECTURE

The SM-QRNS unit suggested in Figure 1 consists of the following function modules.

1) Negator (see Figure 2): Note, \(-x = (p - x) \mod p = 2^n + 2 + x \mod p\) where \(x\) denotes bitwise complement of \(x\). Then if \(x = \sum a_i 2^i\) it follows that \(-x = \sum b_i 2^i\) where:

\[
\begin{align*}
b_0 &= \overline{a_0} \\
b_1 &= \overline{a_1} \\
b_2 &= \overline{a_1a_2} + \overline{a_2} \\
&\vdots \\
b_n &= \overline{a_1a_2\ldots a_n} \\
&\quad+ (a_1 + a_2 + \ldots + a_{n-1})\overline{a_n}
\end{align*}
\]

ii) Modulo \(p\) Adder (see Figure 3): The \(p\) adder consists of an \(n\)-bit fast carry-lookahead adder, a modulo \(p\) mapping unit, and control logic (compare network). The mapping unit is designed to transfer \(S = A + B\) into \(V = A + B\mod p\) or \(V = S - p\) if \(S > p\), using the following procedure [7]:

a. Starting with the LSB of \(S\), complement all "0's" up to the first encountered "1."

b. Complement the first encountered "1."

c. Complement all other bits of greater significance unchanged.

More specifically:

\[
\begin{align*}
v_0 &= \overline{s_0} \\
v_1 &= \overline{s_1} + \overline{s_0} \overline{s_1} \\
v_2 &= \overline{s_1} \overline{s_2} + (s_0 + s_1)s_2 \\
&\vdots \\
v_{n-1} &= \overline{s_{n-1}} \ldots \overline{s_1} + (s_0 + s_1 + \ldots + s_{n-2})\overline{s_{n-1}} \\
v_n &= 0
\end{align*}
\]

iii) Modulo \(p\) Multiplier (see Figure 4): The modulo \(p\) multiplier consists of an unsigned multiplier and a network to process some exceptions:

a. If both inputs equal \(2^n\), then set the product to 1.

b. If only one input equals \(2^n\) (say \(A\)) then set the product to \(-B\)

IV. SM-QRNS SUPPORT

The SM-QRNS architecture displayed in Figures 1 through 4 consists of a number of common macrocells called modules. They are summarized below and shown in Figure 5.

a. Scaling by \(j\) (eq. 3)

\[
\begin{align*}
jX \mod 2^n/2 & = 2^n/2X_{LO} - X_{HI} \\
x &= 2^n/2X_{HI} + X_{LO}
\end{align*}
\]

b. Scaling by \(2^{-1}\) (eq. 3)

\[
2^{-1}X \mod p = (2^n - 1)X_{LO} + X_{HI} \\
x = 2^nX_{HI} + X_{LO}
\]

c. Scaling by \((2j)^{-1}\) (eq. 3)

\[
(2j)^{-1}X \mod 2^n/2 = 2^n/2X_{HI} - X_{LO} \\
x = 2^nX_{HI} + X_{LO}
\]

V. SM-QRNS GATE ARRAY DESIGN

The SM-QRNS architecture reported in the previous section were tested using TEGAS (T[es]T Generation And Simulation language) and the IGC2000 Macro Library which are parts of the GE Semiconductor Gate Array design tool kit. Each array cell consists of three CMOS devices using 2 micron gate-length and the array itself may consist of up to 54,000 internal transistors. With these building blocks, the following modules were designed:

NEG: Negates modulo \(p\) MTT: Product modulo \(p\)

SUM: Carry lookahead adder \(JX\): scale by \(J\)

MDL: Sum modulo \(p\) ITWO: scale by \(2^{-1}\)

MUL: Carry-save unsigned ITWOJ: scale by \((2j)^{-1}\)

For expository reasons, the MDL module will be developed more fully. The MDL accepts data from the adder and modifies the received data modulo \(p\) if \(S > p\) or passes it unaltered otherwise. A section of this logic is reported in Figure 6. Such data is used to define the gate array.

VI. ARRAY ANALYSIS

The macros used to configure the SM-QRNS are 2, 3, 4, and 5 input NAND gates, 2 input XOR and XNOR gates, 2, 3, and 4 input NOR gates, 2-2 and 2-2-2 AND-OR- inverter. The NEG, SUM, MUL, MUL, MTT, JX, ITWO, and ITWOJ units were designed and integrated into a SM-QRNS machine. An eight bit design, for example, consists of 4157 1/3 array cells where each array cell consisted of 3 p-n channel transistors pairs.

The main purpose for this study was to develop a performance database for SM-QRNS and conventional CRNS designs. To establish a common denominator for comparison, the QRNS and CRNS units are configured so as to have a similar hardware budget with both operating dual modulo \(p\) multipliers (see Figure 1). The basic database used to support the comparative analysis is reported in Table 1.

Assume \(T_c\) is the carry propagation time and \(T_s\) is the sum propagation time for a one-bit adder. The timing analysis for the module MUL unit, in general case is

\[
T_d = \begin{cases} 
(n-1)T_s + (n-1)T_c & \text{if } nT_c < (n-1)T_s \\
(2^n/2)T_c & \text{if } nT_c > (n-1)T_s
\end{cases}
\]
where \( n \) is the bit number of the multiplicands. Based on the equivalent hardware budget assumption, the timing analysis for the QRNS and CRNS systems are stated as follows:

a. Because of the number of gate levels are the same for the case were \( n = 8 \) to 16, assume that the gate delay time for the non-wordlength dependent modules (i.e., non-multiplicuy) is fixed.

b. Assume \( T_C = T_s \), the delay time \( T_m \) for the single modulus multiplier \( MULLT \) is

\[
T_m = (2n - 2)T_C + 230ns
\]

c. The delay time for the QRNS system under worse case is

\[
T_q = T_m + \text{(delay time of non-processing units)}
\]

\[
= (2n - 2)T_C + 230ns + 650ns
\]

\[
= 2nT_C + 2T_C + 880ns
\]

while the delay time for CRNS system under worse case is

\[
T_x = 2T_m + \text{(delay time of non-multiplier units)}
\]

\[
= 2((2n - 2)T_C + 230ns) + 200ns
\]

\[
= 4nT_C + 4T_C + 660ns
\]

The "break even" wordlength is given by

\[
4nT_C + 4T_C + 660ns = 2nT_C + 2T_C + 880ns
\]

\[
2nT_C + 2T_C = 220ns
\]

From the performed simulation, \( T_q \) approximates 15ns. Therefore, the equation shown above becomes reduced to 30n = 190 or \( n = 6.3 \).

Based on the timing studies and data reported in Table 1, the performance of a CRNS and SM-QRNS can be compared. The data represents temporal bounds on the performance of each individual module. For both the SM-QRNS and CRNS it can be seen that the principal bottleneck is at the multiplier level. The table also reports summary data for the integrated SM-QRNS and CRNS designs. This data was produced using TEGAS applied to a complete design rather than simply summing the individual module delays (in accordance to Figure 1). The result is a superior SM-QRNS design.

V. SUMMARY AND CONCLUSIONS

The principal objective of this work was to test the recently published conjecture that an integrated QRNS based unit could outperform an CRNS counterpart. Based on the analysis provided, using relatively slow gate array technology, the conjecture has been verified. While the results will vary somewhat as a function of technology and macro-cell differences, the reported results do represent an unbiased exploration of SM-QRNS question using accepted analysis tools. The foregoing analysis indicates that in order to increase the potential speed advantage of SM architecture, the non-wordlength dependent overhead factors (i.e. scaling, negation and mod p) should be reduced to a minimum. It is noted that using gate-level constructs to implement these modules will present penalties. For example, moving from one fan-out to three could easily add another 1.06ns delay for the N05 macro (5-input NAND gate) under 27°C and 5 V operating conditions. Instead, it may be advantageous to use a high-performance commercially available multiplier chip (used as an attached coprocessor) or high speed semiconductor table lookup (for \( n < 12 \) bits). For example, a 45ns 16 x 16 multiplier chip could be interfaced to a SM-QRNS controller/engine. It would accept operands from the engine, compute the binary weighted product, and return the results to the engine for modulo \( p \) reduction. The table-lookup method would also be a suitable forum in which to develop a faster architecture. For example, using high speed memory device (e.g., 40ns, 64Kx1 static RAM chips) to implement the mapping logic in a \( n = 16 \) QRNS system, the following latencies have been determined:

<table>
<thead>
<tr>
<th>Table-Lookup</th>
<th>Gate Array</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEG 40ns</td>
<td>60ns</td>
</tr>
<tr>
<td>SUM + MUL 250ns</td>
<td>250ns</td>
</tr>
<tr>
<td>JX 40ns</td>
<td>130ns</td>
</tr>
<tr>
<td>ITWO 40ns</td>
<td>100ns</td>
</tr>
<tr>
<td>ITWOJ 40ns</td>
<td>120ns</td>
</tr>
<tr>
<td>MULLT (2n-2)T_C + 230ns</td>
<td>(2n-2)T_C + 230ns</td>
</tr>
</tbody>
</table>

Using the fast lookup method, the non-wordlength overhead bias to \( T_q \) can be decreased from 880ns to 670ns. The "break even" wordlength then becomes \( n = 1.3 \).

Some large and varied macrocell libraries, composed of input more pin gates, multiplexers, register files, and multipliers, are available in the market place. These predefined LSI/VLSI functions are optimized such as the C20000 CMOS gate array from Fujitsu Microelectronics mixes high-speed static memory with random logic. This configuration, which consists of 15,000 uncommitted gates with 6K or 12K bits of RAM, provides both small lookup table and discrete logic design capabilities. The bipolar array is another choice for modular design although it has fewer gates within a single chip (up to 8,000 gates versus 20,000 for CMOS counterparts). Higher speed with fewer problems in driving large fan-out (as many as eight loads) can be found in bipolar arrays which have advantages in designing high throughput signal processing systems.

REFERENCES


Acknowledgement: To JoEllen Wilbur for her assistance.

This work was supported under an ARO grant.

Figure 1 Functional block of the Single Modulus QRNS (SM-QRNS) unit

Table 1 Listing of propagation delay of the CRNS, QRNS system and their modules

<table>
<thead>
<tr>
<th>Modulus $P = 2^n + 1$</th>
<th>$n = 4$</th>
<th>$n = 8$</th>
<th>$n = 16$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module</td>
<td></td>
<td>min</td>
<td>max</td>
</tr>
<tr>
<td>NEG</td>
<td></td>
<td>30ns</td>
<td>50ns</td>
</tr>
<tr>
<td>SUM</td>
<td></td>
<td>60ns</td>
<td>70ns</td>
</tr>
<tr>
<td>MDL</td>
<td></td>
<td>50ns</td>
<td>80ns</td>
</tr>
<tr>
<td>MUL</td>
<td></td>
<td>-</td>
<td>80ns</td>
</tr>
<tr>
<td>MUTT</td>
<td></td>
<td>-</td>
<td>250ns</td>
</tr>
<tr>
<td>JX</td>
<td></td>
<td>70ns</td>
<td>110ns</td>
</tr>
<tr>
<td>ITWO</td>
<td></td>
<td>50ns</td>
<td>60ns</td>
</tr>
<tr>
<td>ITWOJ</td>
<td></td>
<td>70ns</td>
<td>100ns</td>
</tr>
<tr>
<td>QRNS</td>
<td></td>
<td>480ns</td>
<td>740ns</td>
</tr>
<tr>
<td>CRNS</td>
<td></td>
<td>230ns</td>
<td>560ns</td>
</tr>
</tbody>
</table>

24
Figure 2 (A) Circuit diagram of the NEGATOR module
(B) PLA implementation of a 4-bit lookup table for the NEGATOR module

Figure 3 (A) Modulo P Adder
(B) MDL\textsubscript{p} unit implemented by using the table lookup method
(C) PLA Implementation of a 4-bit lookup table for the MDL\textsubscript{p}
Figure 4  Modulo $P$ Multiplier

Figure 5  
(A) Scaling module $j$
(B) Scaling module $\text{inv}(2)$
(C) Scaling module $\text{inv}(2j)$
Figure 6  Circuit diagram of the MDL module (a submodule of Modulo P Adder)