RNS IMPLEMENTATION OF FIR FILTERS BASED ON DISTRIBUTED ARITHMETIC USING FIELD-PROGRAMMABLE LOGIC

A. García †, U. Meyer-Bäse †, A. Lloris †, F. J. Taylor ‡

† Department of Electronics and Computer Technology
University of Granada
18071 Granada, Spain
‡ High-Speed Digital Architecture Laboratory
University of Florida
Gainesville, FL 32611-6130, USA

ABSTRACT
Field-programmable logic (FPL) densities and performance have steadily improved, allowing DSP solutions to be integrated on a single FPL chip. The primary limitation of FPLs, in DSP-centric applications, is their intrinsically weak arithmetic performance compared to DSP microprocessors and ASICs. In some cases, distributed arithmetic (DA) has been used to mask FPL arithmetic inadequacies. The Residue Number System (RNS) has demonstrated an ability to support high-bandwidth arithmetic with limited resources. This paper presents a methodology for merging distributed arithmetic with the residue number systems to achieve high-performance FPL solutions.

1. INTRODUCTION
The FPL community is aggressively promoting FPLs as a viable DSP technology. The competitive FPL-enabled DSP designs to date, however, have been essentially restricted to regular SAXPY (S=AX+Y) class applications having fixed coefficients (e.g., FIRs and DFTs). This class of DSP applications lends itself to FPL implementation since the general purpose arithmetic weakness can be masked by implementing arithmetic as direct look-up table (LUT) operations. The facilitating technology, called distributed arithmetic (DA), implements FIR-class tasks with a set of bit-serial data shifts, adder, and LUT operations [5-7]. While achieving high-bandwidth in some designs, the resulting designs have typically been low-precision. Extending the dynamic range of an FPL-enabled DSP processor using residue arithmetic was recently reported [1]. The RNS achieves its arithmetic processing advantage by performing high-bandwidth, TLU-intensive calculations within small wordlength channels. There is, therefore, possible synergy between the RNS and new FPL device families (e.g., Altera’s FLEX [2]) which include collections of small wordlength, built-in tables. The potential fusion of DA and RNS technologies has been explored in the literature for FIR and systolic array structures [3, 4, 8, 9]. These attempts meet with limited success but also demonstrated a systemic weakness. Specially, DA architecture possesses some operations which are RNS-hostile. In this paper, a methodology is presented which overcomes this barrier and provides a design paradigm for implementing high-bandwidth high-precision DSP-centric objects with commercially available FPL devices. This thesis is presented by first providing a brief review of the facilitating technologies followed by specific design recommendations and demonstrations.

2. RNS BACKGROUND
Arithmetic in the RNS [10-12] is defined in terms of a basis set \{m_1, m_2, ..., m_L\} of relatively prime integers. The dynamic range of an RNS system is given by M, where:

\[ M = \prod_{b=1}^{L} m_b \] (1)

For any integer \(0 \leq X < M\), \(X\) is uniquely represented by the \(L\)-tuple \([s_1, s_2, ..., s_L]\) where \(s_i = X \mod m_i\), \(i=1, 2, ..., L\). Signed RNS numbers can also be defined over the range \(-M/2 < X < M/2\). Arithmetic is defined over the ring of integers modulo \(M\), where for \(\oplus\) representing either addition, subtraction or multiplication, and for \(0 \leq X, Y, Z < M\):

\[ Z = (X \oplus Y) \mod M = \{z_1, ..., z_L\} \]

\[ z_j = (s_j \oplus y_j) \mod m_j \] (2)

In practice, the modulo \(m_j\) operations are performed concurrently as table lookups within small wordlength ROM or RAM. It should also be appreciated that each RNS digit is of equal significance and there are no carry propagation requirements between channels. This, in concept, is well suited for FPL assimilation.

3. DISTRIBUTED ARITHMETIC
Conventional DA [6] reduces the \(2\)'s complement computation of the inner product or SAXPY form:

\[ y = \sum_{k=1}^{K} a_k x_k = -\sum_{n=1}^{N} \left( \sum_{k=1}^{K} a_k x_k \right) \cdot 2^{-n} - \sum_{k=1}^{K} a_k x_k \cdot 0 \] (3)

where \(x_k \in \{-1, 1\}\) and \(x_k[j]\) denotes the \(j\)th bit of \(x_k\) (\(j=0\) is the MSB):

---

† FPL devices include field programmable gate arrays (FPGA) and complex programmable logic devices (CPLD).
\[ x_k = -y_0 + \sum_{n=1}^{N-1} x_n [2^n] \]  

The 2^K possible values of \( \Sigma a_k x_k[n] \) can be stored in a 2^K x N-bit LUT \( \Phi \), which is addressed by the vector \( x_k[N-1]x_k[N-2]\ldots x_k[1] \) during the F^T DA cycle. Equation (3) can be rewritten as:

\[ y = \Phi(x[N-1])2^{(N-1)} + \Phi(x[N-2])2^{(N-2)} + \ldots + \Phi(x[0])2^{(0)} \]  

The production of y is seen to require repeated calls to the table \( \Phi \) followed by a shift-addr (scaled accumulation). The first N-1 table lookups are shift added with the last being subtracted from the accumulator. The main advantage of DA lies in its capability to compute equation (3) with a 2^K-word look-up table and a shift-accumulator in N TLU cycles. The speed and complexity of bilateral DA design is defined by the parameters N and K, and the speed and size of available look-up tables.

4. PROPOSED DA-RNS SCHEME

It is possible to consider the fusion of a DA-based structure with the RNS since the RNS, like 2's complement, is a modular system (calculations modulo M versus modulo 2^K for 2's complement). An RNS implementation of equation (3) would be given by the L-tuple \( \{y_1, y_2, \ldots, y_L\} \), where:

\[ y_i = \left( \sum_{k=1}^{K} c_k (x_k \mod m_i) \right) \mod m_i \]  

If \( N = \left[ \log_2 m_i \right] \), then it is clear that:

\[ x_k \mod m_i = \sum_{n=0}^{N-1} x_n (\mod m_i) [2^n] \]  

where \( (x_k \mod m_i)[2^n] \) is the \( n \)-th bit of \( x_k \mod m_i \) (note that now \( j = 0 \) is the LSB). Following the argument used to develop equation (5), it follows that:

\[ y_i = \left( b^{N-1} \Phi_1(x[N-1]) \right) \mod m_i + b^{N-2} \Phi_1(x[N-2]) \mod m_i + \ldots + \left( \Phi_1(x[0]) \right) \mod m_i \]  

where \( y_i = y \mod m_i \), \( \Phi_i \) is given by:

\[ \Phi_i(x[0]) = \left( \sum_{k=0}^{K} c_k (x_k \mod m_i)[2^n] \right) \mod m_i \]  

and \( g[n] \) is the vector formed with the \( n \)-th bits of the \( K \) inputs for the modulo \( m_i \) channel, similar to the 2's complement case. The induced architecture is reported in Figure 1. Historically, DA implementation, whether in 2's complement or RNS, uses a LSB-first protocol. Unfortunately, the standard architecture creates a RNS conundrum since it requires that the TLU data be successively scaled by \( 2^i \), \( i \in [0, N] \). While being a benign operation in 2's complement, such is not the case in the RNS where each scale factor must be interpreted to be \( 2^i \mod m_i \). Each distinct scaling operation would therefore require a separate TLU. Alternatively, the accumulator output can be scaled by \( 2^i \) which is equally awkward to implement in the RNS, requiring an specific TLU too. Figure 1 represents an MSB first RNS-DA system consisting of a fixed table and scaled modular accumulator. The architecture of the scaled modulo adder is shown as the insert to Figure 1. Notice that it contains two residue input operands, three carry bits, and a residue output that is explained by the theorem shown below.

**Theorem:** Let be \( 0 \leq y = m_i \), with \( N = [\log_2 m_i] \). Defining the adder as \( \text{acc}(2x+y) \mod m_i \), it is a function of \( C_i \) (MSB of \( N+1 \) bit output of first adder), \( c_1 \), \( c_2 \), and \( c_3 \) (carry out of the first, second and third adder, respectively) whose values are given by the following table:

<table>
<thead>
<tr>
<th>( C_i )</th>
<th>( c_1 )</th>
<th>( c_2 )</th>
<th>( c_3 )</th>
<th>( \text{acc}(2x+y) \mod m_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>( \text{sum}_1 )</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>( \text{sum}_2 )</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>( \text{sum}_3 )</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Not possible</td>
</tr>
</tbody>
</table>

The proof, that is not shown for the sake of brevity, is straightforward and involves the study of three cases, namely:

1. \( 2x+y < m \)
2. \( m \leq 2x+y < 2m \)
3. \( 2m \leq 2x+y < 3m \)

5. DESIGN EXAMPLE

The most aggressively studied DA DSP application is FIR filtering. The same is true for RNS-enabled DSP solutions. With the structure shown in Figure 1 it is possible to architected an FIR filter using the presented DA-RNS scheme. If defined by an L modulus RNS system, the FIR would consist of L channels where each channel contains a look-up table and a modified \( (2x+y) \mod m_i \) accumulator. From a practical viewpoint, each modulus would be bounded by \( m_i \geq 2^8 \) (8-bit). An FIR having a 24-bit dynamic range would therefore contain three or four channels.

A modern FPGA device contains dedicated tables. The Altera FLEX 10K device [2] includes assorted logic elements (LE) and blocks providing a 2^8 tables with a fast carry chain support, and a number of 2K-bit ROMs or RAMs, called EABs. These
The design objectives of this work are several 8-tap and 10-tap FIR filters, with 24-bit and 32-bit full dynamic ranges. For those filters with 24-bit dynamic range, a four 6-bit modulus set was chosen, namely \( m_i = \{63, 61, 59, 57\} \), suitable for implementation with an Altera FLEX 10K device. On the other hand, the 32-bit dynamic range is covered with five moduli, \( m_i = \{256, 63, 61, 59, 57\} \). Thus, the 8-bit modulus channel does not limit the performance of the 6-bit modulus channels, and 32-bit dynamic range is handled without lack of speed. For maximum performance, three-input multiplexers in Figure 1 were implemented as a couple of two-input multiplexers, one for \( \text{sum}_1 \) and \( \text{sum}_2 \) and the other selecting between this output and \( \text{sum}_3 \). All of the study cases were implemented as an RNS-enabled filter as well as a 2's complement DA FIR through VHDL synthesis. The results are reported in Table 2 for device speed grades 4 and 3, indicated in brackets. The advantage of the RNS-enabled design becomes increasingly apparent when higher order filtering was required, as stated above, due to the maintained performance with the RNS bused structure. Although larger resources are required for the proposed scheme, binary alternative is faster just for the simplest case. When a faster device is considered, performance improvement is more noticeable for binary arithmetic, since longer carry chains make more evident the reduction in the device characteristic time.

6. SUMMARY

An RNS enhanced DA architecture based on an innovative MSB-first scaled accumulator is presented. As a design example, a full-precision FIR filter was presented and compared with a binary implementation for several numbers of taps and coefficient precision cases. The RNS-enabled design was shown to have a higher sustained bandwidth, with up to 35% improvement.

7. ACKNOWLEDGEMENTS

A. Garcia and Dr. A. Lloris were supported by the Dirección General de Enseñanza Superior under project PB96-1397. Dr. U. Meyer-Blase and Dr. F. Taylor were supported under a Max Kade Foundation award.

Figure 1. Distributed Arithmetic scheme for SAXPY RNS applications.
### Table 2. FIR filter comparison.

<table>
<thead>
<tr>
<th>Item</th>
<th>Speed</th>
<th>LE use</th>
<th>EAB use</th>
</tr>
</thead>
<tbody>
<tr>
<td>8-tap, 24-bit FIR filter</td>
<td>DA-RNS: 3.28MSPS (-4)</td>
<td>442</td>
<td>4 @ 6144 bits</td>
</tr>
<tr>
<td></td>
<td>DA-RNS: 4.22MSPS (-3)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Binary: 3.27MSPS (-4)</td>
<td>170</td>
<td>2 @ 4096 bits</td>
</tr>
<tr>
<td></td>
<td>Binary: 4.61MSPS (-3)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8-tap, 32-bit FIR filter</td>
<td>DA-RNS: 3.28MSPS (-4)</td>
<td>554</td>
<td>5 @ 8192 bits</td>
</tr>
<tr>
<td></td>
<td>DA-RNS: 4.22MSPS (-3)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Binary: 2.44MSPS (-4)</td>
<td>202</td>
<td>3 @ 6144 bits</td>
</tr>
<tr>
<td></td>
<td>Binary: 3.79MSPS (-3)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10-tap, 32-bit FIR filter</td>
<td>DA-RNS: 2.78MSPS (-4)</td>
<td>648</td>
<td>16 @ 32768 bits</td>
</tr>
<tr>
<td></td>
<td>DA-RNS: 3.60MSPS (-3)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Binary: 2.34MSPS (-4)</td>
<td>224</td>
<td>12 @ 24576 bits</td>
</tr>
<tr>
<td></td>
<td>Binary: 3.09MSPS (-3)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 8. REFERENCES


