Low-Power FIR Digital Filters Using Residue Arithmetic

William L. Freking and Keshab K. Parhi
Department of Electrical and Computer Engineering
University of Minnesota
200 Union St. S.E.
Minneapolis, MN 55455-0154
{freking, parhi@ece.umn.edu}

Abstract
This paper demonstrates that residue arithmetic can result in implementation of low-power FIR digital filters. It is shown that, for word-lengths up to 32 bits, the power consumption of residue-arithmetic-based FIR filters is dramatically less than two’s-complement-based FIR filters. The power reduction is possible since the use of residue arithmetic transforms the filtering problem into multiple smaller word-length filters for various moduli which are operated in parallel. These compact filters can be operated with lower supply voltage for a specified sample speed, thus obtaining decreased power consumption compared to binary. Power reduction factors for residue arithmetic implementation become increasingly favorable as the system word-length is increased.

1 Introduction
A number of techniques have been devised to implement low-power FIR digital filters. Many of these methods impose constraints on coefficient selection and are therefore not general. Likewise, methods which attain low power consumption through hardware reduction techniques are typically restricted to fixed-coefficient filters. However, the task of identifying power reduction techniques which are valid for programmable-coefficient FIR filters has received less attention. This paper addresses this issue with the application of the residue number system (RNS).

The RNS is a numerical representation which is traditionally regarded for its high-speed implementation attributes. The RNS decomposes a given algorithm into smaller word-length, parallel, modular arithmetic calculations. A general RNS system is illustrated in Figure 1. The RNS paradigm exhibits several drawbacks such as I/O conversion overhead and difficulties associated with sign detection and magnitude comparison. Although these inherent shortcomings prevent general application of RNS methodology, algorithms which are chiefly composed of multiplications and additions with few or no conditionals are well suited for RNS implementation. This specialized architectural category is perhaps best exemplified by the FIR filter [1].

A brief introduction to the fundamentals of RNS arithmetic is provided below followed by a discussion of various approaches to implement RNS FIR digital filters. Section II briefly describes how the RNS can be utilized to reduce the complexity of certain arithmetic operations. Section III describes the low power properties of RNS FIR filters. Hardware cost, switching activity, and voltage supply reduction are examined along with their implications for low-power operation.

Figure 1: RNS implementation is characterized by parallel modulo arithmetic versions of the particular algorithm and I/O conversion.

1.1 Principles of RNS Arithmetic
RNS arithmetic is based on the centuries-old observation that integers specified within a finite range
may be represented uniquely by a set composed by performing the modulo operation of the integer with each member of a set of special moduli [2]. The members of this new set are referred to as residues. The moduli set possesses two salient properties: the product of the moduli is greater than the largest integer to be represented and the moduli are mutually prime. The latter condition dictates that no moduli share factors, i.e., the greatest common denominator of all possible moduli pairs is 1.

The mathematical operations of addition and multiplication on operands converted to residue-set representation are easily deduced from the procedural definition of the modulo operation; no intermediate transformations to and from integer representation are demanded. Consider the integers $X$ and $Y$ as well as their sum, $S$. Let the $i$-th residue of $X$ be denoted as $x_i$. The modulo operation is defined such that

$$X = Q_X m_i + x_i.$$  

Since

$$S = X + Y = (Q_X + Q_Y) m_i + x_i + y_i = Q_S m_i + s_i,$$

it is clear that the addition operation is achieved by

$$s_i = |x_i + y_i|_{m_i},$$

where the notation $|a|_{m_i}$ signifies the modulo operation on $a$ with modulus $m_i$. Multiplication, by an analogous argument, is realized by

$$p_i = |x_i y_i|_{m_i}.$$  

While the conversion process from integer to residue sets is obvious, the reverse conversion is somewhat more abstruse. Fortunately, algorithms such as the Chinese Remainder Theorem (CRT) and mixed-radix conversion are available for this purpose (see [3]).

1.2 RNS FIR Filter Implementation

FIR filter implementation is slightly different within RNS-decomposed channels compared to a binary implementation in that modular arithmetic must be used. A block diagram of both a standard binary and a RNS direct-form FIR filter unit cell is shown in Figure 2. Small dynamic ranges permit the use of lookup-table-based modular arithmetic in RNS implementations. However, as the dynamic range is increased, lookup tables grow as the square of the modulus whereas functional logic grows logarithmically with respect to modulus. Thus, to exploit the RNS for larger dynamic ranges (typically 16 bits or greater), functional logic is often chosen.

Modular multiplications dominate the complexity of functional-logic RNS FIR filter implementations. Many architectures have been proposed to efficiently implement modular multiplication [5][6] and modular inner-product computations[7]. Common to all such algorithms is the necessity to perform modular reduction operations, either embedded at bit or word level within the multiplication operation or as a post-processing step, thus incurring a hardware penalty.

A different approach can be taken by observing that the modulo reduction operations need not be performed within each multiplier or multiply-accumulate operation in a residue-decomposed FIR filter. Rather, the modulo reduction operation can be deferred to a later point in the computation, preferably to the end of the residue decomposed FIR calculation [8]. Thus, standard binary multipliers and adders can be used as opposed to modular architectures.

Deferment of the reduction has a definite drawback – the word-length of the computation has the potential to grow with each accumulation operation in the FIR filter. Rounding is not permissible within the residue-decomposed filter, and dynamic scaling of the RNS range can be a costly operation since it is dependent on information in all the residue channels.

An elegant solution to this problem was proposed in [8]. It was observed that integers which are multiples of the modulus can be added or subtracted from the result at any point in the computation without affecting the final modulo-reduced result. The resultant method requires a conditional subtraction of a correction factor after each multiply-accumulate operation, and thereby limits word-length growth to the word-length at the multiplier output. A further modification was proposed in [9] in which a different choice of correction factor results in a reduction of the correction logic. Both implementations are shown in Figure 3.
2 Moduli Selection for Hardware Reduction

Functional logic approaches to RNS system implementation motivate investigation of moduli selection. Related is the question of optimal moduli selection for reduced memory consumption in RNS systems. It is well known that a set of a large number of small moduli produce small memory sizes. This solution is not only adequate, but optimal for any logic circuit having a complexity that is proportional to the square of the number of input bits.

However, it can be shown that, in general, any complexity function which meets criteria of being a positive-slope, strictly increasing function can result in hardware reduction if RNS implementation is used. In particular, the optimal solution can be determined to be the case where the dynamic range is as equally divided as possible. Further, this optimal solution is a decreasing function with respect to the number of moduli. Thus, the observation concerning moduli selection with regard to minimum memory size can be generalized to a wide variety of functional logic implementations.

3 RNS for Low Power FIR Filters

3.1 Hardware Cost

Hardware cost has a direct bearing on total power consumption in most architectures. An important fact in RNS FIR filter architectures is that conversion overhead remains constant as the number of filter taps is increased. Therefore, if hardware reduction is possible relative to binary implementation within the primary filter logic, power reductions should be possible for a sufficient number of taps.

In order to illustrate hardware reduction in the filter logic, consider an example of a full-adder-based RNS FIR filter implementation. From Figure 2, it is clear that the total complexity for a binary FIR unit cell in terms of full adders is \( n^2 + n \) (where \( n \) is the number of input bits) assuming a binary array multiplier with no rounding operation. For the RNS case, the implementation in Figure 3(b) is utilized with assumptions similar to the binary case. The RNS complexity function is therefore \( n^2 + 2n_1 \). Assuming an optimal moduli set and noting that the RNS representation must have a range of slightly greater than \( 2n \) bits (the word-length of the multiplication results), we find that the least possible RNS complexity for the given case is:

\[
C_R = r \left( \frac{4n^2}{r^2} + 2 \frac{2n}{r} \right) = 4 \frac{n^2}{r} + 4n.
\]

Thus, in order that the RNS case exhibit a hardware reduction in comparison to the binary case, the following condition must be satisfied:

\[
4 \frac{n^2}{r} + 4n < n^2 + n,
\]

which simplifies to:

\[
r > 4 \left( \frac{n}{n - 3} \right).
\]

Therefore, in this particular case, an optimal RNS system of at least 5 moduli can obtain a hardware reduction for sufficiently large \( n \). Note that the number of moduli required to obtain a reduction decreases as \( n \) increases. This fact is due to the increasing dominance of the multiplier complexity (which is mitigated by the RNS) over the addition complexity. Regardless of the binary range, a greater advantage is garnered by increasing the number of moduli, which translates to larger word-length architectures receiving the most benefit.

Comparisons of the hardware cost per FIR unit cell for RNS and binary implementations are shown in Table 1. Both binary-array and Booth-multiplier implementations are shown for each case. The increased benefit of RNS representation as the binary word-length increases is very evident.

### Table 1. Hardware costs for FIR unit cells (number of full adders)

<table>
<thead>
<tr>
<th>Word Length</th>
<th>Binary Array</th>
<th>Binary R4-Booth</th>
<th>RNS Array</th>
<th>RNS R4-Booth</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>80</td>
<td>169.5</td>
<td>56</td>
<td>154.5</td>
</tr>
<tr>
<td>12</td>
<td>168</td>
<td>246</td>
<td>108</td>
<td>216</td>
</tr>
<tr>
<td>16</td>
<td>288</td>
<td>328</td>
<td>176</td>
<td>284</td>
</tr>
<tr>
<td>24</td>
<td>624</td>
<td>520</td>
<td>360</td>
<td>446</td>
</tr>
<tr>
<td>32</td>
<td>1088</td>
<td>700</td>
<td>608</td>
<td>588</td>
</tr>
</tbody>
</table>

3.2 Switching Activity Reduction

RNS implementation lends itself to reduced switching activity which translates into reduced power consumption. Switching activity is reduced due to the
confined range of allowed bit combinations in residue representation – namely only values values from 0 to \( m_i - 1 \) are permitted. Bit probabilities, which are closely linked to switching probabilities within digital circuits, are shown in Figure 4 for several 4-bit moduli. Although in the chosen FIR filter implementation modulo corrections are not made within the multiplication operations, there is still reduced switching activity in the multipliers due to residue representation at the inputs. This fact leads to power reduction when compared to binary multipliers of the same word-length. Simulations using the HEAT power estimation tool [11] indicate switching activity reductions of up to 38% are possible in 4×4-bit multipliers due to residue representation (see Table 2).

![Probability of "1" Vs. Bit Position](image)

Figure 4: Probability of 1 at each bit position for various moduli.

<table>
<thead>
<tr>
<th>Modulus</th>
<th>Power (in mW)</th>
<th>Power Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>216</td>
<td>38.29%</td>
</tr>
<tr>
<td>11</td>
<td>243</td>
<td>30.57%</td>
</tr>
<tr>
<td>13</td>
<td>283</td>
<td>19.14%</td>
</tr>
<tr>
<td>15</td>
<td>320</td>
<td>8.57%</td>
</tr>
<tr>
<td>16 (binary)</td>
<td>350</td>
<td>0.00%</td>
</tr>
</tbody>
</table>

### Table 3. Voltage reductions due to RNS representation with respect to word-length

<table>
<thead>
<tr>
<th>Word Length</th>
<th>Voltage Reduction</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>0.572</td>
</tr>
<tr>
<td>12</td>
<td>0.500</td>
</tr>
<tr>
<td>16</td>
<td>0.409</td>
</tr>
<tr>
<td>24</td>
<td>0.353</td>
</tr>
<tr>
<td>32</td>
<td>0.297</td>
</tr>
</tbody>
</table>

### Table 4. Normalized average power consumed by FIR units after voltage supply reduction (assuming 32 bit word-length sample rate)

<table>
<thead>
<tr>
<th>Word Length</th>
<th>Binary Array</th>
<th>RNS Array</th>
<th>Binary R4-Booth</th>
<th>RNS R4-Booth</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>0.074</td>
<td>0.051</td>
<td>0.051</td>
<td>0.046</td>
</tr>
<tr>
<td>12</td>
<td>0.154</td>
<td>0.056</td>
<td>0.099</td>
<td>0.050</td>
</tr>
<tr>
<td>16</td>
<td>0.265</td>
<td>0.050</td>
<td>0.162</td>
<td>0.044</td>
</tr>
<tr>
<td>24</td>
<td>0.574</td>
<td>0.060</td>
<td>0.331</td>
<td>0.051</td>
</tr>
<tr>
<td>32</td>
<td>1.000</td>
<td>0.057</td>
<td>0.559</td>
<td>0.048</td>
</tr>
</tbody>
</table>

### 3.3 Supply Voltage Reduction

The nature of the RNS decomposition of an algorithm into smaller word-length partitions can be exploited for low-power operation when high sample rates are not demanded. In particular, FIR filters which do not require fine-grain pipelining of the multipliers can benefit substantially. Maintaining the same sample speed as a comparable word-length binary implementation, the supply voltage can be reduced [12] due to a shortened critical path length within the parallel residue channels. Supply voltage reductions increase rapidly with larger word-length (see Table 3) due to the fact that the critical path in the RNS case increases approximately as the log of equivalent binary word-length, while the critical path increases linearly with word-length in the binary case. Power consumption is reduced as the square of the voltage reduction factor, resulting in dramatic reductions in power consumption as demonstrated in Table 4. In addition, if further power reduction is required, it is found that RNS pipelining overhead in the primary logic is not substantially different from the binary implementation. Linear reduction factors in critical path length with respect to the level of pipelining are possible in both RNS and binary cases.

### 3.4 Simulation Results

Simulations using the HEAT power estimation tool were conducted for a programmable-coefficient FIR filter with 16 bit coefficients and 32 bit dynamic range implemented in both binary and RNS with deferred reduction. Both implementations utilized array multipliers followed by ripple-carry adders for the accumulation operation. Results indicate that for standard 1.2 micron CMOS technology with operational frequency of 50 MHz, the FIR unit cell dissipates 26.2 mW on average. In comparison, the RNS implementation, including contributions from all residue channels, dissipates only 3.8 mW.
Although power dissipation per tap is much lower than the comparable binary implementation, the I/O conversion overhead of the RNS implementation must also be considered. A straightforward implementation of the CRT and input conversion circuitry was found to dissipate approximately 400mW using the above parameters. Because this overhead remains constant relative to the number of filter taps, it follows that increasing the number of taps will be beneficial to the RNS design. Figure 5 illustrates the dependence on number of taps of the overall RNS power reduction factor. It is clear, for this example, that the power overhead is completely compensated above 20 taps, with more taps yielding increasingly favorable power reductions. Note voltage reduction was not exploited, which would reduce power consumption further.

Figure 5: RNS power reduction factor with respect to number of taps.

### 4 Conclusions

It has been demonstrated that RNS representation has significant power reduction properties when applied to programmable FIR digital filter implementation. Compared to binary representation, a suitably chosen RNS can offer hardware reduction to any combination of operations meeting certain criteria. It was observed that hardware reduction is most dramatic as the word-length of the equivalent binary system increases. Limited bit pattern combinations of RNS data were found to lead to smaller switching activities, particularly in the multipliers. Finally, supply voltage reduction, which can dramatically reduce power consumption, is possible for RNS FIR filters which require less demanding sample rates.

### Acknowledgment

This research was supported by the Office of Naval Research under grant number N00014-96-1-0931.

### References


