<table>
<thead>
<tr>
<th>Title</th>
<th>New algorithm for signed integer comparison in four-moduli superset {2n, 2n+1, 2n+1, 2n+11}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author(s)</td>
<td>Tay, Thian Fatt; Chang, Chip-Hong</td>
</tr>
<tr>
<td>Citation</td>
<td>Tay, T. F., &amp; Chang, C.-H. (2014). New algorithm for signed integer comparison in four-moduli superset {2n, 2n+1, 2n+1, 2n+11}. 2014 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), 519-522.</td>
</tr>
<tr>
<td>Date</td>
<td>2014</td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://hdl.handle.net/10220/25166">http://hdl.handle.net/10220/25166</a></td>
</tr>
<tr>
<td>Rights</td>
<td>© 2014 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. The published version is available at: [<a href="http://dx.doi.org/10.1109/APCCAS.2014.7032833">http://dx.doi.org/10.1109/APCCAS.2014.7032833</a>].</td>
</tr>
</tbody>
</table>
New Algorithm for Signed Integer Comparison in Four-Moduli Superset \( \{2^n, 2^n-1, 2^n+1, 2^{n+1}-1\} \)

Thian Fatt Tay and Chip-Hong Chang
School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore.

Abstract—Sign detection and magnitude comparison are two difficult operations in Residue Number System (RNS). Existing residue comparators tackle only unsigned integer for magnitude comparison. In this paper, a new algorithm for signed integer comparison in the four-moduli supersets, \( \{2^n, 2^n-1, 2^n+1, 2^{n+1}-1\} \) with even \( n \), is proposed. The dynamic range is quantized into equal subranges to facilitate fast sign detection and magnitude comparison simultaneously without the need for full magnitude recovery by Chinese Remainder Theorem (CRT) or sequential Mixed Radix Conversion (MRC). The proposed algorithm can be implemented by using adders only and the operations are less complex than those used in existing RNS magnitude comparators of comparable dynamic range.

I. INTRODUCTION

Residue arithmetic has replaced two’s complement arithmetic for high-speed digital signal processing in some applications due to the merits of RNS, which include parallelism, modularity and fault-tolerance. In RNS, an integer can be decomposed into several smaller residue digits such that arithmetic operations like addition, subtraction and multiplication can be performed on the residue digits of the operands in parallel and independently. With no order of precedence among the residue digits, some operations such as magnitude comparison, sign detection, overflow detection and scaling, are harder to evaluate in RNS when manipulations of residues across modulus channels are inevitable. These operations can limit the speed of RNS-based computations if they cannot be avoided in the critical path.

Data sorting is a fundamental process in electronic finance, data management systems, digital computers and communication systems [1]. Sorting a large pool of data requires iterative magnitude comparisons and data swaps. Unfortunately, the magnitude of an RNS representation cannot be directly accessible from its residue digits. To compare the magnitudes of two unsigned integers in RNS, non-trivial computations are required to compose the residue digits into some weighted positional representation. Comparing two signed integers in RNS requires further delay since sign detection in RNS is another difficult and time-consuming operation. Currently, there is no hardware amendable algorithm for signed integer magnitude comparison in RNS.

The most direct method for magnitude comparison in RNS is to evaluate the magnitudes of two residue representations using CRT or MRC [2]. However, this method is highly inefficient due to the large modulo operation of CRT and the slow sequential computation of MRC. Other approaches can be divided into two general categories: parity check [3]-[5] and range detection [6]-[8]. For parity check approach, the moduli set must be chosen such that all moduli are odd. In [3], core function is introduced to find the parity of a residue representation by an iterative process of descent and lift when ‘critical core’ values are found. In [4], CRT is used to compute the parity of a residue representation. It is computed by adding fractional representations due to the residue digits to avoid large modulo operations. Algorithm [5] is designed for special moduli sets, whereby the parity computation of residue representation is simplified by the number theoretic properties of the moduli. In contrast, range detection approaches are not particularly restricted by the choice of moduli. The range of a residue representation is detected by the Sum of Quotients (SQT) [6], New CRT-II [7] and Mixed-radix CRT methods [8].

All existing residue comparison algorithms are designed for only unsigned integral comparison. In this paper, a signed integer comparison algorithm for the four-moduli supersets, \( \{2^n, 2^n-1, 2^n+1, 2^{n+1}-1\} \) with even \( n \) is proposed. The dynamic range is divided into subranges for residue comparison with its sign directly detected from the most significant bit (MSB) of subrange identifier at no additional cost. It can be implemented completely in combinational circuits without any lookup table.

II. BACKGROUND

An RNS is characterized by a set of pairwise relatively prime moduli \( \{m_1, m_2, ..., m_N\} \). An unsigned integer \( X \) falls within the dynamic range \([0, M-1]\), where \( M = \prod_{i=1}^{N} m_i \), is represented uniquely by an \( N \)-tuple of residues \((x_1, x_2, ..., x_N)\). Each residue \( x_i \) can be computed by finding the non-negative remainder of \( X \) divided by \( m_i \), i.e., \( x_i = X \mod m_i \) or \( x_i = \lfloor X/m_i \rfloor \).

Given the residue representation \((x_1, x_2, ..., x_N)\), its magnitude \( X \) can be evaluated by using CRT [2]:

\[
X = \sum_{i=0}^{N-1} M_i \left[M_i^{-1}\right]_{m_i} x_i
\]

(1)

where \( M_i = M / m_i \) and \( \left[M_i^{-1}\right]_{m_i} \) is the multiplicative inverse of \( [M_i]_{m_i} \).

\( X \) can also be recovered from its residue representation by applying MRC serially. Starting with a baseline two-moduli set \( \{m_1, m_2\} \), \( X \) can be computed from \((x_1, x_2)\) by [2]:

\[
X = \left[m_1^{-1}\right]_{m_2} (x_2 - x_1) \left[m_1\right]_{m_2} + x_1
\]

(2)

where \( \left[m_1^{-1}\right]_{m_2} \) is the multiplicative inverse of \( [m_1]_{m_2} \).

To represent a sign integer \( \hat{X} \) with symmetrical positive and negative ranges in RNS, the residue representation of \( \hat{X} \) can be mapped to that of \( X \) by using the congruence \( \hat{X} \equiv X + [M/2] \mod M - [M/2] \). When \( \hat{X} \geq 0 \), its residue
representation is mapped to the range \([0, (M/2) - 1]\) if \(M\) is even and \([0, (M-1)/2]\) if \(M\) is odd. When \(X < 0\), its residue representation is mapped to the range \([M/2, M - 1]\) if \(M\) is even and \([((M+1)/2), M-1]\) if \(M\) is odd [9]. In this RNS representation, the sign of \(X\) can be detected from the magnitude \(X\) of its residue representation \((x_1, x_2, \ldots, x_d)\) by:

When \(M\) is even, \(\text{sign}(X) = \begin{cases} 0 & \text{if } X \in \left[0, \left(\frac{M}{2}\right) - 1\right] \\ 1 & \text{if } X \in \left[\frac{M}{2}, M - 1\right] \end{cases}\) \hspace{1cm} (3)

When \(M\) is odd, \(\text{sign}(X) = \begin{cases} 0 & \text{if } X \in \left[0, \left(\frac{M-1}{2}\right)\right] \\ 1 & \text{if } X \in \left[\left(\frac{M+1}{2}\right), M - 1\right] \end{cases}\) \hspace{1cm} (4)

III. PROPOSED ALGORITHM

Let \(m_1 = 2^a, m_2 = 2^a - 1, m_3 = 2^a + 1\) and \(m_4 = 2^{a+1} - 1\) be the moduli of the superset \(\{2^a, 2^a - 1, 2^a + 1, 2^{a+1} - 1\}\) with an even value of \(n\) [10]. The dynamic range of this moduli set can be written as \(M = m_1 \times M_{R1}, \) where \(M_{R1} = m_1 \times m_2 \times m_3 \times m_4\) can be quantized into \(m_1\) number of divisions where each division has an equal range of \(M_{R1}\) as shown in Fig. 1.

![Fig. 1: Division of M into subranges M_{R1}.](image)

By definition of residue arithmetic, an unsigned integer \(X\) can be written as:

\[X = \left\lfloor \frac{X}{M_{R1}} \right\rfloor_{m_1} M_{R1} + \left\lfloor \frac{X}{M_{R2}} \right\rfloor_{m_2} M_{R2} + \left\lfloor \frac{X}{M_{R3}} \right\rfloor_{m_3} M_{R3} + \left\lfloor \frac{X}{M_{R4}} \right\rfloor_{m_4} M_{R4}\] \hspace{1cm} (5)

where \(W_i = \left\lfloor \frac{X}{M_{Ri}} \right\rfloor_{m_i} M_{Ri}\) and \(W_i \in [0, m_i - 1]\).

\(W_i\) is the index of the division wherein \(X\) falls. A larger value of \(W_i\) signifies a larger magnitude of \(X\). The magnitude of \(X\) and \(Y\) can be compared by computing \(W_x\) and \(W_y\). If \(W_x > W_y\), it can be concluded that \(X\) is greater than \(Y\) and vice-versa. However, when \(W_x = W_y\), further quantization is required for exact magnitude comparison.

Similar to the quantization of \(M\), each \(M_{Ri}\) range can be further quantized into \(m_i\) equal subdivisions, each of range \(M_{R2} = m_2 \times m_3\), as shown in Fig. 2. By expressing \(\left\lfloor \frac{X}{M_{R2}} \right\rfloor_{m_2} M_{R2}\) in term of its residue, (5) can be rewritten as:

\[X = W_2 M_{R1} + V_2 M_{R2} + \left\lfloor \frac{X}{M_{R3}} \right\rfloor_{m_3} M_{R3} + \left\lfloor \frac{X}{M_{R4}} \right\rfloor_{m_4} M_{R4}\] \hspace{1cm} (6)

where \(V_2 = \left\lfloor \frac{X}{M_{R2}} \right\rfloor_{m_2} M_{R2}\) and \(V_2 \in [0, m_4 - 1]\).

Similarly, \(V_3\) represents the subdivision wherein \(X\) falls. When \(V_x = V_2, V_3\) and \(V_4\) need to be computed for magnitude comparison. In this case, \(V_x > V_3\) indicates that \(X\) is larger than \(Y\) and vice-versa. When \(V_x = V_3, \left\lfloor \frac{X}{M_{R3}} \right\rfloor_{m_3}\) and \(\left\lfloor \frac{X}{M_{R4}} \right\rfloor_{m_4}\) need to be computed and compared ultimately without further quantization of \(M_{R2}\).

![Fig. 2: Division of M_{R1} into subranges M_{R2}.](image)

The proposed algorithm relies on a two-tier quantization for magnitude comparison without fully recovering the magnitudes of \(X\) and \(Y\) from their respective residue representation. This is faster than comparing the magnitudes after the residue-to-binary conversion by CRT or MRC. Typically, signed integer magnitude comparison also requires an additional sign detection operation to determine the signs of \(\tilde{X}\) and \(\tilde{Y}\). If the signs are different, the negative integer is always considered to be smaller than the positive integer. Otherwise, the magnitudes are then compared. By using our proposed algorithm for RNS signed integer comparison, no overhead will be incurred by the sign detection operation. This is because the signs of the integers can be detected by merely checking the most significant bits (MSBs) of \(W_x\) and \(W_y\), as proven by the following derivations.

Since \(m_1\) is even, \(M\) is always even. Hence, the sign of a residue representation can be determined by (3). For sign(\(\tilde{X}\)) = 1, \(X \geq M/2 = m_1 M_{R1}/2 = 2^{a-1} M_{R1}\). In other words, sign(\(\tilde{X}\)) = 1 when \(W_x \geq 2^{a-1}\). Since \(W_x \in [0, 2^a - 1]\), the condition of \(W_x \geq 2^{a-1}\) is detected by the MSB of \(W_x\) i.e.,

\[\text{sign(\(\tilde{X}\)) = } W_{x,a-1} \] \hspace{1cm} (7)

where \(W_{x,i}\) denote the \(i\)-th bit of \(W_x\).

To compare \(\tilde{X}\) and \(\tilde{Y}\), the residue representation \(Y = (y_1, y_2, y_3, y_4)\) of \(\tilde{Y}\) is also decomposed into \(Y = W_2 M_{R1} + V_2 M_{R2} + \left\lfloor \frac{Y}{M_{R3}} \right\rfloor_{m_3} M_{R3} + \left\lfloor \frac{Y}{M_{R4}} \right\rfloor_{m_4} M_{R4}\) \hspace{1cm} (8)

The proposed pseudo code for the comparison of \(\tilde{X}\) and \(\tilde{Y}\) in residue domain is shown in Fig. 3.

**Compare:**(\(x_1, x_2, x_3, x_4, y_1, y_2, y_3, y_4\), \(m_1, m_2, m_3, m_4\)) {

Compute \(W_x, V_x, \left\lfloor \frac{X}{M_{R3}} \right\rfloor_{m_3} M_{R3} + \left\lfloor \frac{X}{M_{R4}} \right\rfloor_{m_4} M_{R4}\);

\[\text{sign(\(\tilde{X}\)) = } W_{x,a-1} ; \text{ sign(\(\tilde{Y}\)) = } W_{y,a-1} ;\]

if (sign(\(\tilde{X}\)) \# sign(\(\tilde{Y}\))) return SC(sign(\(\tilde{X}\)));

else {

if (\(W_x > W_y\)) return \(\tilde{X} > \tilde{Y}\);

else if (\(W_x < W_y\)) return \(\tilde{X} < \tilde{Y}\);

else {

if (\(V_x > V_y\)) return \(\tilde{X} > \tilde{Y}\);

else if (\(V_x < V_y\)) return \(\tilde{X} < \tilde{Y}\);

else {

if (\(V_x = V_y\)) return \(\tilde{X} > \tilde{Y}\);

else return \(\tilde{X} = \tilde{Y}\).

}

}

} \hspace{1cm} \text{SC}(a) \{ \text{if } (a = 0) \text{ return } \tilde{X} > \tilde{Y} ; \text{ else return } \tilde{X} < \tilde{Y} ; \}

**Fig. 3:** Proposed algorithm for sign integer comparison in RNS.
IV. HARDWARE IMPLEMENTATION

The architecture of the proposed algorithm in Section III is shown in Fig. 4 which consists of two stages. Stage 1 computes \( W_{n} \), \( V_{r} \), \( [X]_{m_{2}} \), \( W_{r} \), \( V_{s} \), and \( [Y]_{m_{2}} \) from the residue representations of the two integers \( X \) and \( Y \) to be compared. These computations can be realized by adders only and the derivation of the simplified architecture will be described. Stage 2 implements the sign detection and magnitude comparison of \( X \) and \( Y \) according to the algorithm described in Section III. An XOR gate is used in Stage 2 to check if the signs of \( X \) and \( Y \) are identical and the comparators can be implemented using simple logic gates.

Fig. 4: The architecture of the proposed signed integer magnitude comparator.

From the definitions of (5) and (6), \( W_{n} \), \( V_{r} \) and \( [X]_{m_{2}} \) can only be computed provided that the magnitude \( X \) is known. As the magnitude of \( X \) is not given, \( W_{n}, V_{r} \) and \( [X]_{m_{2}} \) have to be computed from its residues, \( x_{1}, x_{2}, x_{3} \) and \( x_{4} \). The computations can be simplified by the following properties.

Property 1 [10]: Let \( CLS_{s}(x, r) \) denotes the circular left shift by \( r \) bits of an \( n \)-bit binary integer \( x \). Then,

\[
|2^{x}2^{-r}| = CLS_{s}(x, r)
\]

(9)

Property 2 [10]:

\[
|2^{-x}2^{-r}| = |2^{r}(2^{x} - 1 - x)|2^{-r} = |2^{r}2^{-r}| = CLS_{s}(\overline{x}, r)
\]

(10)

where \( \overline{x} \) denotes the one's complement of \( x \).

Property 3: \( |x|_{2} = |(2^{x} - 1 - x)|2^{-r} = |r + 1|_{2} \)

(11)

By taking the modulo \( M_{2} \) operation on both sides of (1), \( [X]_{m_{2}} \) can be calculated from only \( x_{2} \) and \( x_{3} \) as follows.

\[
[X]_{m_{2}} = M_{2}[M_{2}^{-1} x_{2} + M_{2}^{-1} x_{3}]_{m_{2}} \tag{12}
\]

Using (2),

\[
|X|_{m_{2}} = |(2^{r} + 1)^{-x_{3}}(x_{2} - x_{3})|2^{-r} \cdot (2^{r} + 1) + x_{3}
\]

In (13), since \( x_{3} \in [0, 2^{n-1}] \), \( x_{3} \) has to be modularly reduced by \( 2^{n-1} \) before it is deducted from \( x_{2} \). Let \( a_{s} = |x|_{m_{2}} \). By applying Properties 1 and 2, (13) can be rewritten as

\[
[X]_{m_{2}} = CLS_{s}(x_{1}, n-1) + CLS_{s}(\overline{a_{s}}, n-1)|2^{n-1}(2^{n} + 1) + x_{3}
\]

(14)

where \( P_{s} = CLS_{s}(x_{2}, n-1) + CLS_{s}(\overline{a_{s}}, n-1) \).

Since \( P_{s} \) is an \( n \)-bit integer, (14) can be implemented by a mod \( 2^{n-1} \) adder and a simplified carry propagate adder (CPA), as shown in Fig. 5.

Fig. 5: Architecture for \( |X|_{m_{2}} \). Computation.

By comparing (5) with (6), we have

\[
[X]_{m_{2}} = V_{r}M_{2} + [X]_{m_{2}} \tag{15}
\]

\( [X]_{m_{2}} \) can be computed from its residues \((|X|_{m_{2}} , x_{4})\) of the two-moduli set \( \{M_{2}, m_{2}\} \) by using (2).

\[
[X]_{m_{2}} = \left( M_{2}^{2} - 1 \right) x_{3} - [X]_{m_{2}} + V_{r}M_{2} \tag{16}
\]

By comparing (15) and (16), \( V_{r} \) can be expressed as

\[
V_{r} = \left( M_{2}^{2} - 1 \right) x_{3} - |X|_{m_{2}} + x_{4} \tag{17}
\]

Lemma 1: The multiplicative inverse of \( M_{2} \) modulo \( m_{2} = (2^{n-1} - 1) \) is given by

\[
[M_{2}^{-1}]_{m_{2}} = \frac{1}{3}(2^{r+1} - 5) = \sum_{l=0}^{2^{l+1} - 1} 2^{l+1} + 1 \tag{18}
\]

where \( n \) is an even number.

Proof:

\[
\frac{2^{2^{n-1}} - 1}{3}(2^{r+1} - 5)\text{ mod } 2^{2^{n-1} - 1} = 1
\]

Let \( [X]_{m_{2}} \) denote the \( i \)-th bit of \( [X]_{m_{2}} \), and \( J_{s} = \left([X]_{m_{2}, 2n-4} \cdots [X]_{m_{2}, 0}\right)_{2} \) and \( J_{s} = \left([X]_{m_{2}, 2n-4} \cdots [X]_{m_{2}, 0}\right)_{2} \).

Then, \( [X]_{m_{2}} = 2^{n+1}J_{s} + J_{s} \). By substituting (18) into (17),

\[
V_{r} = \left( \sum_{l=0}^{2^{l+1} - 1} 2^{l+1} + 1 \right) \left( x_{3} - 2^{n+1}J_{s} + J_{s} \right) \text{ mod } 2^{2^{n-1}-1}
\]

(19)

From (19), \( V_{r} \) can be computed by a \((3 \times n) / 2, 2^{n+1} - 1)\) multi-operand modular adder (MOMA) [10]. Fig. 6(a) shows the architecture for the computation of \( V_{r} \) with \( n = 4 \). It consists of
four carry save adders with end-around carry (CSAs with EAC) and a mod $2^1$-1 adder.

Finally, $X$ can be computed by (2) from the residues
\[
\left[ X_{1|d_4} , x_1 \right]
\]
of the two-moduli set $\{M_{41}, m_1\}$ as follows:
\[
X = \left\lfloor M_{41} \right\rfloor \left( x_1 - \left\lfloor X_{1|d_4} \right\rfloor \right) \cdot M_{41} + \left\lfloor X_{1|d_4} \right\rfloor
\]
(20)

Comparing (5) and (20), $W_1$ can be expressed as
\[
W_1 = \left\lfloor M_{41} \right\rfloor \left( x_1 + V_x - \left\lfloor X_{1|d_4} \right\rfloor \right)
\]
(21)

Let $T_x = \{V_{x,1} \ldots V_{x,0}\}_2$ and $U_x = \{X_{M_{41},1+1} \ldots X_{M_{41},0}\}_2$.

By applying Property 3, (21) can be rewritten as
\[
W_1 = \left\lfloor M_{41} \right\rfloor \left( x_1 + T_x + \bar{U}_x + 1\right)
\]
(22)

Lemma 2: The multiplicative inverse of $M_{41}$ modulo $m_1 = (2^2 - 1)(2^n+1)$ modulo $2^n$ is given by:
\[
M_{41}^{-1} = \left\lfloor M_{41} \right\rfloor = 1
\]
(23)

Proof: $\left(2^2 - 1\right)\left(2^n+1\right)x_1 \equiv 1 \mod 2^n$.

By substituting (24) into (22),
\[
W_1 = \left\lfloor M_{41} \right\rfloor \left( x_1 + T_x + \bar{U}_x + 1\right)
\]
(24)

From (24), $W_1$ can be computed by a CSA and a mod $2^n$ adder as shown in Fig. 6(b). The computations of $W_1$, $V_x$ and $Y_1$ from the residue representation of the operand $Y$ can be similarly implemented.

VI. COMPLEXITY ANALYSIS

As the delay and complexity of residue comparators are dictated by the largest integers involved in the residue comparison operations, residue comparison algorithms for different moduli sets are typically benchmarked by their maximum size modulo operations as in [5]-[7]. Table I shows the dynamic ranges, maximum operator sizes and support for signed integer comparison (SC) of our proposed and existing residue comparison algorithms [3]-[8]. Methods [3] and [4] are designed for moduli sets with odd moduli only and require redundant modulus. Their operators’ sizes are generally larger for the same dynamic range. Methods [5] and [6] are developed for two pairs of conjugate moduli and $\{2^n, 2^n+1\}$. Their dynamic ranges are all smaller than our moduli set for the same $n$. If their dynamic ranges are made nearer to but lower than ours by increasing $n$, their operators’ sizes are always larger than ours. Methods [7] and [8] can be used for any moduli set. Their operators’ sizes are larger than ours. Besides having the smallest operator size, our algorithm is the only one that is able to compare signed integers in residue representation.

Table I: Performance Evaluation of Residue Comparison Algorithms.

<table>
<thead>
<tr>
<th>Method</th>
<th>Dynamic Range</th>
<th>Operator Size</th>
<th>SC?</th>
</tr>
</thead>
<tbody>
<tr>
<td>[3]</td>
<td>$M$</td>
<td>$\sum_{i=1}^{n} M_i/m_i$</td>
<td>No</td>
</tr>
<tr>
<td>[4]</td>
<td>$\log, MM$</td>
<td>$\sum_{i=1}^{n} M_i/m_i$</td>
<td>No</td>
</tr>
<tr>
<td>[5]</td>
<td>$(2^2^n-1)(2^n-1)/3$</td>
<td>modulo $(2^n+1)$</td>
<td>No</td>
</tr>
<tr>
<td>[6]</td>
<td>$(2^n) (2^n-1)$</td>
<td>modulo $(2^n+1)$</td>
<td>No</td>
</tr>
<tr>
<td>[7]</td>
<td>$(2^2^n-2^n)(2^n+1)$</td>
<td>modulo $(2^n-1)$</td>
<td>No</td>
</tr>
<tr>
<td>[8]</td>
<td>$(2^n-2^n)(2^n+1)$</td>
<td>This, modulo $(2^n+1)$</td>
<td>Yes</td>
</tr>
</tbody>
</table>

ACKNOWLEDGEMENT

The authors would like to acknowledge the support from MOE-AcRF Tier2 grant MOE2013-T2-2-017.

REFERENCES