New algorithm for signed integer comparison in four-moduli superset $\{2n, 2n - 1, 2n + 1, 2n + 1 - 1\}$

Tay, Tian Fatt; Chang, Chip-Hong

2014

Tay, T. F., & Chang, C.-H. (2014). New algorithm for signed integer comparison in four-moduli superset $\{2n, 2n - 1, 2n + 1, 2n + 1 - 1\}$. 2014 IEEE Asia Pacific Conference on Circuits and Systems (APCCAS), 519-522.

<https://hdl.handle.net/10356/105041>

<https://doi.org/10.1109/APCCAS.2014.7032833>

© 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: [<http://dx.doi.org/10.1109/APCCAS.2014.7032833>].

Downloaded on 26 Apr 2023 20:09:03 SGT

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 system, 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 fully 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 integer 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 $\{m_1, m_2, \dots, m_N\}$. 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 $(x_1, x_2, \dots, x_N)$. Each residue x_i can be computed by finding the non-negative remainder of X divided by m_i , i.e., $x_i = X \bmod m_i$ or $x_i = \lfloor X \rfloor_{m_i}$.

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

$$X = \sum_{i=1}^N M_i \left\lfloor M_i^{-1} \right\rfloor_{m_i} x_i \Big|_M \quad (1)$$

where $M_i = M / m_i$ and $\left\lfloor M_i^{-1} \right\rfloor_{m_i}$ is the multiplicative inverse of M_i modulo 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]:

$$X = \left\lfloor m_1^{-1} \right\rfloor_{m_2} (x_2 - x_1) \Big|_{m_2} \cdot m_1 + x_1 \quad (2)$$

where $\left\lfloor m_1^{-1} \right\rfloor_{m_2}$ is the multiplicative inverse of m_1 modulo m_2 .

To represent a sign integer $\tilde{X}$ with symmetrical positive and negative ranges in RNS, the residue representation of $\tilde{X}$ can be mapped to that of X by using the congruence $\tilde{X} \equiv \lfloor X + \lfloor M/2 \rfloor \rfloor_M - \lfloor M/2 \rfloor$. When $\tilde{X} \geq 0$, 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 $\tilde{X} < 0$, 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 $\tilde{X}$ can be detected from the magnitude X of its residue representation $(x_1, x_2, \dots, x_N)$ by:

$$\text{When } M \text{ is even, } \text{sign}(\tilde{X}) = \begin{cases} 0 & \text{if } X \in [0, (M/2)-1] \\ 1 & \text{if } X \in [M/2, M-1] \end{cases} \quad (3)$$

$$\text{When } M \text{ is odd, } \text{sign}(\tilde{X}) = \begin{cases} 0 & \text{if } X \in [0, (M-1)/2] \\ 1 & \text{if } X \in [(M+1)/2, M-1] \end{cases} \quad (4)$$

III. PROPOSED ALGORITHM

Let $m_1 = 2^n$, $m_2 = 2^n - 1$, $m_3 = 2^{n+1}$ and $m_4 = 2^{n+1} - 1$ be the moduli of the superset $\{2^n, 2^n - 1, 2^{n+1}, 2^{n+1} - 1\}$ 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 $M_{R1} = m_2 \times m_3 \times m_4$. M 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} .

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

$$X = \lfloor X/M_{R1} \rfloor_{m_1} M_{R1} + |X|_{M_{R1}} = W_x M_{R1} + |X|_{M_{R1}} \quad (5)$$

where $W_x = \lfloor X/M_{R1} \rfloor_{m_1}$ and $W_x \in [0, m_1 - 1]$.

W_x is the index of the division wherein X falls. A larger value of W_x signifies a larger magnitude of X . The magnitude of X and Y can be compared by computing and comparing 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_{R1} range can be further quantized into m_4 equal subdivisions, each of range $M_{R2} = m_2 \times m_3$, as shown in Fig. 2. By expressing $|X|_{M_{R1}}$ in term of its residue, (5) can be rewritten as:

$$\begin{aligned} X &= W_x M_{R1} + \left\lfloor |X|_{M_{R1}} / M_{R2} \right\rfloor_{m_4} M_{R2} + |X|_{M_{R2}} \\ &= W_x M_{R1} + V_x M_{R2} + |X|_{M_{R2}} \end{aligned} \quad (6)$$

where $V_x = \left\lfloor |X|_{M_{R1}} / M_{R2} \right\rfloor_{m_4}$ and $V_x \in [0, m_4 - 1]$.

Similarly, V_x represents the subdivision wherein X falls. When $W_x = W_y$, V_x and V_y need to be computed for magnitude comparison. In this case, $V_x > V_y$ indicates that X is larger than Y and vice-versa. When $V_x = V_y$, $|X|_{M_{R2}}$ and $|Y|_{M_{R2}}$ need to be computed and compared ultimately without further quantization of M_{R2} .

Fig. 2: Division of M_{R1} into subranges M_{R2} .

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 $\text{sign}(\tilde{X}) = 1$, $X \geq M/2 = m_1 M_{R1} / 2 = 2^{n-1} M_{R1}$. In other words, $\text{sign}(\tilde{X}) = 1$ when $W_x \geq 2^{n-1}$. Since $W_x \in [0, 2^n - 1]$, the condition of $W_x \geq 2^{n-1}$ is detected by the MSB of W_x , i.e.,

$$\text{sign}(\tilde{X}) = W_{x, n-1} \quad (7)$$

where $W_{x,i}$ denote the i -th bit of W_x

To compare $\tilde{X}$ and $\tilde{Y}$, the residue representation $Y \equiv (y_1, y_2, y_3, y_4)$ of $\tilde{Y}$ is also decomposed into

$$Y = W_y M_{R1} + V_y M_{R2} + |Y|_{M_{R2}} \quad (8)$$

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  $W_x, V_x, |X|_{M_{R2}}, W_y, V_y$ , and  $|Y|_{M_{R2}}$ ;
   $\text{sign}(\tilde{X}) = W_{x, n-1}$ ;  $\text{sign}(\tilde{Y}) = W_{y, n-1}$ ;
  if ( $\text{sign}(\tilde{X}) \neq \text{sign}(\tilde{Y})$ ) return SC( $\text{sign}(\tilde{X})$ );
  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 ( $|X|_{M_{R2}} > |Y|_{M_{R2}}$ ) return  $\tilde{X} > \tilde{Y}$ ;
        else if ( $|X|_{M_{R2}} < |Y|_{M_{R2}}$ ) return  $\tilde{X} < \tilde{Y}$ ;
        else return  $\tilde{X} = \tilde{Y}$ ; } } }
  }
SC( $a$ ) { if ( $a = 0$ ) return  $\tilde{X} > \tilde{Y}$ ; 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_x , V_x , $|X|_{M_{R2}}$, W_y , V_y , and $|Y|_{M_{R2}}$ 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 $\tilde{X}$ and $\tilde{Y}$ according to the algorithm described in Section III. An XOR gate is used in Stage 2 to check if the signs of $\tilde{X}$ and $\tilde{Y}$ 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_x , V_x and $|X|_{M_{R2}}$ can only be computed provided that the magnitude X is known. As the magnitude of X is not given, W_x , V_x and $|X|_{M_{R2}}$ 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 $CLS_n(x, r)$ denotes the circular left shift by r bits of an n -bit binary integer x . Then,

$$2^r x \Big|_{2^n-1} = CLS_n(x, r) \quad (9)$$

Property 2 [10]:

$$\left| -2^r x \right|_{2^n-1} = \left| 2^r (2^n - 1 - x) \right|_{2^n-1} = \left| 2^r \bar{x} \right|_{2^n-1} = CLS_n(\bar{x}, r) \quad (10)$$

where $\bar{x}$ denotes the one's complement of x .

$$\text{Property 3: } \left| -x \right|_{2^n} = \left| (2^n - 1 - x) + 1 \right|_{2^n} = \left| \bar{x} + 1 \right|_{2^n} \quad (11)$$

By taking the modulo M_{R2} operation on both sides of (1), $|X|_{M_{R2}}$ can be calculated from only x_2 and x_3 as follows.

$$|X|_{M_{R2}} = \left| M_2 \left| M_2^{-1} \right|_{m_2} x_2 + M_3 \left| M_3^{-1} \right|_{m_3} x_3 \right|_{M_{R2}} \quad (12)$$

Using (2),

$$|X|_{M_{R2}} = \left| \left((2^n + 1)^{-1} \right) \Big|_{2^n-1} (x_2 - x_3) \right|_{2^n-1} \cdot (2^n + 1) + x_3$$

$$= \left| 2^{n-1} (x_2 - x_3) \right|_{2^n-1} \cdot (2^n + 1) + x_3 \quad (13)$$

In (13), since $x_3 \in [0, 2^n]$, x_3 has to be modularly reduced by 2^n-1 before it is deducted from x_2 . Let $a_x = |x_3|_{2^n-1}$. By applying *Properties 1* and *2*, (13) can be rewritten as

$$\begin{aligned} |X|_{M_{R2}} &= \left| CLS_n(x_2, n-1) + CLS_n(\bar{a}_x, n-1) \right|_{2^n-1} (2^n + 1) + x_3 \\ &= 2^n P_x + P_x + x_3 \end{aligned} \quad (14)$$

where $P_x = \left| CLS_n(x_2, n-1) + CLS_n(\bar{a}_x, n-1) \right|_{2^n-1}$.

Since P_x 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 $|X|_{M_{R2}}$ Computation.

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

$$|X|_{M_{R1}} = V_x M_{R2} + |X|_{M_{R2}} \quad (15)$$

$|X|_{M_{R1}}$ can be computed from its residues ($|X|_{M_{R2}}$, x_4) of the two-moduli set $\{M_{R2}, m_4\}$ by using (2).

$$|X|_{M_{R1}} = \left\| M_{R2}^{-1} \right|_{2^{n+1}-1} (x_4 - |X|_{M_{R2}}) \Big|_{2^{n+1}-1} M_{R2} + |X|_{M_{R2}} \quad (16)$$

By comparing (15) and (16), V_x can be expressed as

$$V_x = \left\| M_{R2}^{-1} \right|_{2^{n+1}-1} (x_4 - |X|_{M_{R2}}) \Big|_{2^{n+1}-1} \quad (17)$$

Lemma 1: The multiplicative inverse of M_{R2} modulo $m_4 = (2^{2n}-1)$ modulo $(2^{n+1}-1)$ is given by

$$\left| M_{R2}^{-1} \right|_{m_4} = \frac{1}{3} (2^{n+1} - 5) = \sum_{i=1}^{n/2-1} 2^{2i+1} + 1 \quad (18)$$

where n is an even number.

$$\text{Proof: } \left| (2^{2n} - 1) \times \frac{1}{3} (2^{n+1} - 5) \right|_{2^{n+1}-1}$$

$$= \left| \frac{1}{3} (2^{3(n+1)-2} - 5 \times 2^{2(n+1)-2} - 2^{n+1} + 5) \right|_{2^{n+1}-1} = 1$$

Let $|X|_{M_{R2},i}$ denote the i -th bit of $|X|_{M_{R2}}$, and $J_{x1} = (|X|_{M_{R2},n} \cdots |X|_{M_{R2},0})_2$ and $J_{x2} = (|X|_{M_{R2},2n-1} \cdots |X|_{M_{R2},n+1})_2$. Then, $|X|_{M_{R2}} = 2^{n+1} J_{x2} + J_{x1}$. By substituting (18) into (17),

$$\begin{aligned} V_x &= \left\| \left(\sum_{i=1}^{n/2-1} 2^{2i+1} + 1 \right) (x_4 - 2^{n+1} J_{x2} - J_{x1}) \right\|_{2^{n+1}-1} \\ &= \left| (2^3 + \dots + 2^{n-1} + 1) (x_4 - J_{x2} - J_{x1}) \right|_{2^{n+1}-1} \\ &= \left| CLS_{n+1}(x_4, 3) + CLS_{n+1}(\bar{J}_{x2}, 3) + CLS_{n+1}(\bar{J}_{x1}, 3) \right. \\ &\quad \left. + \dots + CLS_{n+1}(\bar{J}_{x1}, n-1) + x_4 + \bar{J}_{x2} + \bar{J}_{x1} \right|_{2^{n+1}-1} \end{aligned} \quad (19)$$

From (19), V_x 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_x with $n = 4$. It consists of

four carry save adders with end-around carry (CSAs with EAC) and a mod 2^5-1 adder.

Finally, X can be computed by (2) from the residues $(|X|_{M_{R1}}, x_1)$ of the two-moduli set $\{M_{R1}, m_1\}$ as follows:

$$X = \left\| M_{R1}^{-1} \right\|_{m_1} \left(x_1 - |X|_{M_{R1}} \right) \cdot M_{R1} + |X|_{M_{R1}} \quad (20)$$

Comparing (5) and (20), W_x can be expressed as

$$W_x = \left\| M_{R1}^{-1} \right\|_{2^n} \left(x_1 + V_x - |X|_{M_{R2}} \right) \quad (21)$$

$$\text{Let } T_x = (V_{x,n-1} \dots V_{x,0})_2 \text{ and } U_x = (|X|_{M_{R2,n-1}} \dots |X|_{M_{R2,0}})_2.$$

By applying *Property 3*, (21) can be rewritten as

$$W_x = \left\| M_{R1}^{-1} \right\|_{2^n} (x_1 + T_x + \bar{U}_x + 1) \quad (22)$$

Lemma 2: The multiplicative inverse of M_{R1} modulo $m_1 = (2^{2n}-1)(2^{n+1}-1)$ modulo 2^n is given by:

$$\left\| M_{R1}^{-1} \right\|_{m_1} = 1 \quad (23)$$

$$\text{Proof: } \left\| (2^{2n}-1)(2^{n+1}-1) \times 1 \right\|_{2^n} = \left\| 2^{3n+1} - 2^{2n} - 2^{n+1} + 1 \right\|_{2^n} = 1$$

By substituting (23) into (22),

$$W_x = \left\| (x_1 + T_x + \bar{U}_x + 1) \right\|_{2^n} \quad (24)$$

From (24), W_x can be computed by a CSA and a mod 2^n adder as shown in Fig. 6(b). The computations of W_y , V_y and $|Y|_{M_{R2}}$ from the residue representation of the operand Y can be similarly implemented.

Fig. 6: Architectures for (a) V_x computation with $n=4$ (b) W_x computation

V. 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 $\{2^n, 2^n-1, 2^n+1\}$. 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.

Method	Dynamic Range	Operator Size	SC?
[3]	M	$\sum_{i=1}^N w_i M / m_i$	No
[4]	M	$\log_2 NM$	No
[5]	$(2^{2n+2}-1)(2^{2n}-1)/3$	modulo $(2^{2n+1}-1)$	No
[6]	$(2^n)(2^{2n}-1)$	modulo $(2^{2n}-1)$	No
[7]	$(2^{2n+1}-2^n)(2^{2n}-1)$	modulo $(2^{2n}-1)$	No
[8]		modulo $[(2^{2n+1}-2^n) + (2^{2n}-1)]$	No
This		modulo $(2^{n+1}-1)$	Yes

VI. CONCLUSION

A signed integer magnitude comparison algorithm in RNS has been presented. The algorithm is designed based on the four-moduli superset $\{2^n, 2^n-1, 2^n+1, 2^{n+1}-1\}$ with even values of n . The proposed algorithm can be implemented by using only adders and can handle sign detection in residue representation without additional overhead. For comparable dynamic range, the operation size of our proposed algorithm is the smallest among the existing algorithms.

ACKNOWLEDGEMENT

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

REFERENCES

- [1] S. W. Cheng, "A high-speed magnitude comparator with small transistor count," *Proc. IEEE Int. Conf. Electron. Circuits Systems*, vol. 3, Dec. 2003, pp. 1168-1171.
- [2] N. S. Szabó and R. I. Tanaka, *Residue Arithmetic and Its Applications to Computer Technology*. New York: McGraw-Hill, 1967, pp. 27-32.
- [3] D. D. Miller, R. E. Altschul, J. R. King and J. N. Polky, "Analysis of the residue class core function of Akushskii, Burcev and Pak," in *Residue Number System Arithmetic, Modern Applications in Digital Signal Processing*, M. A. Soderstrand, W. C. Jenkins, G. A. Jullien, and F. J. Taylor, eds. New York, IEEE Press, paper 7-2, 1985, pp. 390-401.
- [4] M. Lu and J. Chiang, "A novel division algorithm for the Residue Number System," *IEEE Trans. Comput.*, vol. 41, no. 8, pp. 1026-1032, Aug. 1992.
- [5] L. Sousa, "Efficient method for magnitude Comparison in RNS based on two pairs of conjugate moduli," *Proc. 18th IEEE Symp. Computer Arithmetic*, Montpellier, France, Jun. 2007, pp. 240-250.
- [6] S. Bi and W. J. Gross, "The mixed-radix Chinese Remainder Theorem and its applications to residue comparison," *IEEE Trans. Comput.*, vol. 57, no. 12, pp. 1624-1632, Dec. 2008.
- [7] Y. Wang, X. Song, and M. Aboulhamid, "A new algorithm for RNS magnitude comparison based on new Chinese Remainder Theorem II," *Proc. 9th Great Lakes Symp. VLSI*, Michigan, USA, Mar. 1999, pp. 362-365.
- [8] G. Dimauro, S. Impedovo, and G. Pirlo, "A new technique for fast number comparison in Residue Number System," *IEEE Trans. Comput.*, vol. 42, No. 5, pp. 608-612, May 1993.
- [9] T. F. Tay, C. H. Chang, and J. Y. S. Low, "Efficient VLSI implementation of 2^n scaling of signed integer in RNS $\{2^n-1, 2^n, 2^n+1\}$," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 21, no. 10, pp. 1936-1940, Oct. 2013.
- [10] B. Cao, T. Srikanthan, and C. H. Chang, "Efficient reverse converters for the four-moduli sets $\{2^n, 2^n-1, 2^n+1, 2^{n+1}-1\}$ and $\{2^n, 2^n-1, 2^n+1, 2^{n+1}-1\}$," *Proc. IEE Comput. Digit. Tech.*, vol. 152, no. 5, pp. 687-696, Sep. 2005.