The Mixed-Radix Chinese Remainder Theorem and Its Applications to Residue Comparison

Shaoqiang Bi and Warren J. Gross, *Member*, *IEEE*

Abstract—The Chinese remainder theorem (CRT) and the mixed-radix conversion (MRC) are two classic theorems used to convert a residue number to its binary correspondence for a given moduli set $\{P_n, \ldots, P_2, P_1\}$. The MRC is a weighted number system, and it requires operations modulo P_i only, and hence, magnitude comparison is easily performed. However, the calculation of the mixed-radix coefficients in the MRC is a strictly sequential process and involves complex divisions. Thus, the residue-to-binary (R/B) conversions and residue comparisons based on the MRC require a large delay. In contrast, the R/B conversion and residue comparison based on the CRT are fully parallel processes. However, the CRT requires large operations modulo $M = P_n, \ldots, P_2P_1$. In this paper, a new mixed-radix CRT is proposed that possesses both the advantages of the CRT and the MRC, which are parallel processing, small operations modulo P_i only, and the efficiency of making modulo comparison. Based on the proposed CRT, new residue comparators are developed for the three-moduli set $\{2^n - 1, 2^n, 2^n + 1\}$. The FPGA implementation results show that the proposed modulo comparators are about 20 percent faster and smaller than one of the previous best designs.

Index Terms—Chinese remainder theorem, mixed-radix conversion, residue comparator, FPGA.

1 Introduction

Interest in the residue number system (RNS) in the face of standard number systems can be explained by the emergence of application-specific integrated circuits (ASICs) that benefit from the speed, area, and power advantage of the RNS. Specifically, the RNS has been receiving significant attention for high-speed digital signal processing (DSP) computation with high precision for the intrinsic properties of the RNS such as carry-free operations, parallelism, and modularity. The RNS is defined in terms of a set of mutually prime moduli that are independent of each other. Since there is no carry propagation among arithmetic operations based on each modulus, it is easy to implement RNS computations in parallel, thus resulting in very high-speed and low-power VLSI implementations [1].

However, due to the nonposition nature of the RNS, the magnitude comparison between residue numbers is much more complex than that in the weighed number system. Other residue arithmetic functions such as sign test, overflow detection, and division suffer from the same difficulty. This difficulty prevents a wide variety of general-purpose computations from taking advantage of the residue arithmetic. To do the residue number comparison, the traditional techniques use the Chinese remainder theorem (CRT) or the mixed-radix conversion (MRC) [1]. A direct implementation of the CRT is inefficient since it is based on a large modulo M operation, where M is the dynamic range of the RNS. The MRC is a strictly sequential process and requires a long delay. Some techniques based on the core function [2], parity checking [3], or the diagonal function [4] have been proposed to compare the magnitudes of residue numbers. The core functions require an iterative process of descent and lifting to find the critical core value. An improved version of this technique was presented in [2] to avoid the iterative process at the cost of a redundant modulus. A different solution [3] to do the residue number comparison assumes that all moduli of the moduli set are odd and ROM lookup tables (LUTs) are mandatory to resolve the difficulty in the determination of the operand parity. The diagonal function [4] requires a large modulo SQ operation, which is usually implemented using large ROM LUTs.

Another interesting technique is to do the residue comparison based on the New CRT [5], which combines the CRT and the MRC to reduce the residue computation delay. Other similar techniques [6], [7] can also be used to compare residue numbers. These techniques depend on ROMs that are addressed by the residue to get the mixed-radix (MR) representation for the ith orthogonal projection of X. Then, a $log_2 n$-level modulo adder tree is used to get the MR digits x'_i. However, Mohan has pointed out in [8] that the last stage has carry propagation from the modulo adder network of one residue to another and introduces a relatively large delay. There is a large body of research on these methods in the literature [9], [10], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27].

Despite the theoretical validity of these algorithms, the VLSI design of residue comparators faces challenges due to the complexity and the ROM-based property of these algorithms. It is important to develop new residue comparison algorithms and propose VLSI comparators that are moduli parity independent, minimizing the utilization of ROM LUTs, and do not introduce any redundant modulus.

In this paper, a new MR CRT is proposed that possesses both the advantages of the CRT and the MRC, which are parallel processing, small operations modulo P_i only, and the efficiency of making residue comparison. Based on the proposed CRT, new residue comparators are developed for the three-moduli set $\{2^n - 1, 2^n, 2^n + 1\}$. The FPGA

- *S. Bi is with Xilinx Inc., 2100 Logic Drive, San Jose, CA 95124. E-mail: shaoqiang.bi@xilinx.com.*
- *W.J. Gross is with the Department of Electrical and Computer Engineering, McGill University, 3480 University Street, Suite 633, Montreal, QC H3A 2A7, Canada. E-mail: warren.gross@mcgill.ca.*

Manuscript received 24 July 2006; revised 11 July 2007; accepted 16 June 2008; published online 25 July 2008.
Recommended for acceptance by F. Lombardi.
For information on obtaining reprints of this article, please send e-mail to: tc@computer.org, and reference IEEECS Log Number TC-0286-0706.
Digital Object Identifier no. 10.1109/TC.2008.126.

implementation results show that the proposed modulo comparators are about 20 percent faster and smaller than one of the previous best designs.

The remaining sections of this paper are organized as follows: In Section 2, a background overview of the RNS and different CRT algorithms is provided. In Section 3, the MR CRT is proposed. The proof of its correctness is established based on the modified CRT. Then, a new residue comparison theorem is proposed based on the MR CRT in Section 4. The VLSI implementations of new residue comparators for $\{2^n-1, 2^n, 2^n+1\}$ are presented, and the area cost and performance of the proposed comparator are evaluated and compared with previous designs in Section 5 followed by the conclusion.

2 Background Materials

Let $\{P_n, \ldots, P_2, P_1\}$ be a set of positive numbers all greater than one. The P_i's are called moduli, and the n-tuple set $\{P_n, \ldots, P_2, P_1\}$ is called the moduli set. In order to avoid redundancy, the moduli of an RNS must be pairwise relatively prime. For an integer number X, we have $x_i = X \bmod P_i$ (denoted as $|X|_{P_i}$). Thus, a number X in RNS can be represented as $X = (x_n, \ldots, x_2, x_1)$. Such a representation is unique for any integer $X \in [0, M-1]$, where $M = P_n, \ldots, P_2P_1$ is the dynamic range of the moduli set $\{P_n, \ldots, P_2, P_1\}$ [1]. To convert a residue number $(x_n, \ldots, x_2, x_1)$ to its binary representation X, the MRC and the CRT are widely used.

Theorem 1 (MRC [1]). *A number X can be computed by the formula*

$$X = \sum_{i=1}^{n} v_i a_i, \tag{1}$$

where $n > 1$, $v_i = \prod_{j=1}^{i-1} P_j$ for $2 \le i \le n$, $v_1 = 1$, and a_i, which are called the MR digits, are computed by the formulas $Y_1 = X$, $Y_i = (Y_{i-1} - a_{i-1})|P_{i-1}^{-1}|_{P_i}$, and $a_i = |Y_i|_{P_i}$.

We list a_1, a_2, and a_3 as follows:

$$\begin{aligned} a_1 &= x_1, \\ a_2 &= \Big|(x_2 - a_1)\big|P_1^{-1}\big|_{P_2}\Big|_{P_2}, \\ a_3 &= \Big|((x_3 - a_1)\big|P_1^{-1}\big|_{P_3} - a_2)\big|P_2^{-1}\big|_{P_3}\Big|_{P_3}. \end{aligned}$$

MR representation is of great importance in residue computation for the following two related reasons [1]: 1) the MR system is a weighted number system, and hence, magnitude comparison is easily performed, and 2) the MRC procedure requires operations modulo P_i only. However, the computation of the MR digits is a strictly sequential process and is not as "parallel" as the CRT method. The residue-to-binary (R/B) conversion and the residue comparison based on the MRC has a long delay and is not suitable for high-speed design. In contrast, the CRT is a fully parallel process.

Theorem 2 (CRT). *The binary number X is computed by*

$$X = \left|\sum_{i=1}^{n} N_i \big|N_i^{-1}\big|_{P_i} x_i\right|_M, \tag{2}$$

where $n > 1$, $N_i = M/P_i$, and $|N_i^{-1}|_{P_i}$ is the multiplicative inverse of $|N_i|_{P_i}$ defined by $\|N_i^{-1}|_{P_i} N_i|_{P_i} = 1$.

It can be noted that the CRT requires a binary inner product operation followed by a large modulo M operation that is not efficient. This inefficiency makes the CRT-based RNS algorithms such as residue comparison and R/B conversion slow and complex. This real drawback makes VLSI design very difficult, especially for general moduli sets. In the literature, there exist extensive studies of the CRT, and some good CRT theorems have been proposed [28], [7].

Theorem 3 (New CRT II [28]). *The following algorithm, translate, finds the correct decimal representation of the RNS number $X = (x_1, x_2, \ldots, x_n)$.*

Algorithm: $translate((x_1, x_2, \ldots, x_n), X)$
if $n = 2t > 2$ *(n is an even number greater than 2)*
then
$translate((x_1, \ldots, x_t), L_1)$, $M_1 = \prod_{i=1}^{t} P_i$
$translate((x_{t+1}, \ldots, x_n), L_2)$, $M_2 = \prod_{i=t+1}^{n} P_i$
$findno(L_1, L_2, M_1, M_2, X)$
end if
if $n = 2t + 1 > 2$ *(n is an odd number greater than 2)*
then
$translate((x_1, \ldots, x_t), L_1)$, $M_1 = \prod_{i=1}^{t} P_i$
$translate((x_{t+1}, \ldots, x_n), L_2)$, $M_2 = \prod_{i=t+1}^{n} P_i$
$findno(L_1, L_2, M_1, M_2, X)$
end if
if $n = 2$ **then**
$findno(x_1, x_2, P_1, P_2, X)$
end if
if $n = 1$ **then**
$X = |x_1|_{P_1}$
end if

Procedure findno *is defined as follows:*

Algorithm: $findno(x_1, x_2, P_1, P_2, X)$
find a k_0 such that $|k_0 P_2|_{P_1} = 1$
$X = x_2 + |k_0(x_1 - x_2)|_{P_1} P_2$

It can be noted that the New CRT II is designed using a divide-and-conquer approach. Each modulo multiplier in the New CRT II is bounded by size $\sqrt{M}$. Thus, efficient designs can be obtained based on the New CRT II for general moduli sets. However, the New CRT II utilizes $\log n$-level modulo multipliers in sequence, which means that the total delay caused by the modulo operations increases with the number of the moduli that is $O(\log n)$ times.

Theorem 4 (modified CRT [7]). *Given the moduli set $\{P_n, \ldots, P_2, P_1\}$, the residue number $x_n, \ldots, x_2, x_1$ is converted into the binary number X by*

$$X = x_1 + P_1 \left|\sum_{i=1}^{n} w_i x_i'\right|_{P_n \ldots P_2}, \tag{3}$$

where $n > 1$, $w_1 = (N_1|N_1^{-1}|_{P_1} - 1)/P_1$, $w_i = N_i/P_1$, $x_1' = x_1$, and $x_i' = |N_i^{-1}|_{P_i} x_i$, for $i = 2, 3, \ldots, n$.

Comparing the CRT and the modified CRT, it can be noted that the modified CRT reduces the modulo base by P_1. Thus, it leads to an efficient converter design. However, for the

moduli sets with a large size, the modified CRT is still slow. If the modulo base can be further reduced and the delay becomes independent of the size of the moduli sets, then a more efficient converter design can be obtained. Bi et al. in [29] have proposed a modulo reduction theorem that can be used to develop the new MR CRT in the next section.

Theorem 5 (modulo reduction theorem [29]). *Given the integers $K, P_n, \ldots, P_2, P_1$ and $n > 1$, we have*

$$|K|_{P_n \ldots P_2 P_1} = |K|_{P_1} + \sum_{m=1}^{n-1} \left[\prod_{i=1}^{m} P_i \left| \left\lfloor \frac{K}{\prod_{i=1}^{m} P_i} \right\rfloor \right|_{P_{m+1}} \right]. \quad (4)$$

In the next section, we will propose the new MR CRT. We need to use the following properties:

Lemma 1. $|2^{n_0} X|_{2^n-1} = x_{n-n_0-1} \ldots x_0 x_{n-1} \ldots x_{n-n_0}$ *for an* n-bit *binary number* X.

Lemma 2. $|-X|_{2^n-1} = \overline{x}_{n-1} \ldots \overline{x}_0$ *for any nonzero* n-bit *binary number* X.

3 The New Mixed-Radix CRT

In this section, we propose a novel MR CRT.

Theorem 6. *Given $\{P_n, \ldots, P_2, P_1\}$, the magnitude of a residue number $X = (x_n, \ldots, x_2, x_1)$ is calculated as follows:*

$$X = \sum_{m=1}^{n-2} \left[\alpha_{m+1} \prod_{i=1}^{m+1} P_i \right] + \alpha_1 P_1 + \alpha_0, \quad (5)$$

where $\alpha_{m+1} = \left| \left\lfloor \sum_{i=1}^{m+2} \gamma_i x_i / \prod_{i=2}^{m+1} P_i \right\rfloor \right|_{P_{m+2}}$, $\alpha_1 = |\gamma_1 x_1 + \gamma_2 x_2|_{P_2}$, $\alpha_0 = x_1$, $n > 1$, $\gamma_1 = (N_1 |N_1^{-1}|_{P_1} - 1)/P_1$, $\gamma_i = M |N_i^{-1}|_{P_i} / P_1 P_i$, *and* $M = P_n, \ldots, P_2 P_1$ *for* $i = 2, 3, \ldots, n$. *The floor function is indicated by* $\lfloor \bullet \rfloor$.

Proof. The modulo operation of Theorem 4 can be decomposed using Theorem 5 as follows:

$$\begin{aligned} X &= x_1 + P_1 \left| \sum_{i=1}^{n} w_i x_i' \right|_{P_n \ldots P_2} \\ &= x_1 + P_1 \left\{ \left| \sum_{i=1}^{n} w_i x_i' \right|_{P_2} + \sum_{m=1}^{n-2} \left[\left| \left\lfloor \frac{\sum_{i=1}^{n} w_i x_i'}{\prod_{i=2}^{m+1} P_i} \right\rfloor \right|_{P_{m+2}} \prod_{i=2}^{m+1} P_i \right] \right\} \\ &= x_1 + P_1 \left| \sum_{i=1}^{n} w_i x_i' \right|_{P_2} + P_2 P_1 \left| \left\lfloor \frac{\sum_{i=1}^{n} w_i x_i'}{P_2} \right\rfloor \right|_{P_3} \\ &\quad + \cdots + P_{n-1} \ldots P_2 P_1 \left| \left\lfloor \frac{\sum_{i=1}^{n} w_i x_i'}{P_{n-1} \ldots P_3 P_2} \right\rfloor \right|_{P_n}. \end{aligned}$$

Notice that

$$\begin{aligned} \sum_{i=1}^{n} w_i x_i' &= w_1 x_1' + \frac{N_2}{P_1} x_2' + \frac{N_3}{P_1} x_3' + \cdots + \frac{N_n}{P_1} x_n' \\ &= \frac{N_1 |N_1^{-1}|_{P_1} - 1}{P_1} x_1 + P_3 P_4 \ldots P_n |N_2^{-1}|_{P_2} x_2 + P_2 P_4 \\ &\quad \ldots P_n |N_3^{-1}|_{P_3} x_3 + \cdots + P_2 P_3 \cdots P_{n-1} |N_n^{-1}|_{P_n} x_n \\ &= \gamma_1 x_1 + \gamma_2 x_2 + \cdots + \gamma_n x_n, \end{aligned}$$

where $\gamma_1 = (N_1 |N_1^{-1}|_{P_1} - 1)/P_1$, $\gamma_i = M |N_i^{-1}|_{P_i} / P_1 P_i$, and $M = P_n, \ldots, P_2 P_1$ for $i = 2, 3, \ldots, n$. Thus, we have

$$\begin{aligned} X &= x_1 + P_1 |\gamma_1 x_1 + \gamma_2 x_2|_{P_2} + P_2 P_1 \left| \left\lfloor \frac{\gamma_1 x_1 + \gamma_2 x_2 + \gamma_3 x_3}{P_2} \right\rfloor \right|_{P_3} \\ &\quad + P_3 P_2 P_1 \left| \left\lfloor \frac{\gamma_1 x_1 + \gamma_2 x_2 + \gamma_3 x_3 + \gamma_4 x_4}{P_2 P_3} \right\rfloor \right|_{P_4} \\ &\quad + \cdots + P_{n-1} \ldots P_2 P_1 \left| \left\lfloor \frac{\gamma_1 x_1 + \gamma_2 x_2 + \cdots + \gamma_n x_n}{P_2 P_3 \ldots P_{n-1}} \right\rfloor \right|_{P_n} \\ &= \alpha_0 + \alpha_1 P_1 + \sum_{m=1}^{n-2} \left[\alpha_{m+1} \prod_{i=1}^{m+1} P_i \right]. \end{aligned}$$

□

Theorem 6 decomposes the large modulo $M = P_n, \ldots, P_2 P_1$ operation of the CRT to a number of small operations modulo P_i only. Theorem 6 provides an MR form of the CRT that converts residue numbers to weighted numbers, and hence, magnitude comparison is easily performed. This New CRT is different from the MRC process. The calculation of the MR coefficients in the MRC is a strictly sequential process, whereas all α_i's of Theorem 6 can be computed in a fully parallel way. Hence, Theorem 6 possesses both the advantages of the CRT and the MRC, which are parallel processing, small operations modulo P_i only, and the efficiency of making residue comparison, and thus leads to efficient VLSI designs of residue comparators.

4 Residue Number Comparison

The MR CRT considerably reduces the complexity of the CRT by decomposing the large modulo M operation to several small modulo operations in parallel. In this section, we use this parallelism to present new residue comparison algorithms for $\{2^n - 1, 2^n, 2^n + 1\}$ that are highly concurrent and suitable for VLSI implementation.

Definition 1. *We define $\alpha_{n-1}, \ldots, \alpha_1, \alpha_0$ in Theorem 6 as the kernel set of $X = (x_n, \ldots, x_2, x_1)$ and denote it as $E(x) = (\alpha_{n-1}, \ldots, \alpha_1, \alpha_0)$.*

Similarly, we can define the kernel set $E(y) = (\beta_{n-1}, \ldots, \beta_1, \beta_0)$ for $Y = (y_n, \ldots, y_2, y_1)$. With Definition 1, the comparison of two residue numbers is simplified to comparing their kernel sets [30], [9]. Given any two residue numbers $X = (x_n, \ldots, x_2, x_1)$ and $Y = (y_n, \ldots, y_2, y_1)$ with the general moduli set $\{P_n, \ldots, P_2, P_1\}$, we can do the comparison using their kernel sets $E(x) = (\alpha_{n-1}, \ldots, \alpha_1, \alpha_0)$ and $E(y) = (\beta_{n-1}, \ldots, \beta_1, \beta_0)$. Without losing generality, assuming α_h and β_h are the first occurring pair of nonequal elements in $E(x)$ and $E(y)$ respectively, namely, $\alpha_h \neq \beta_h$ and $\alpha_j = \beta_j$ for $n > j > h \geq 0$, we have that if $\alpha_h > \beta_h$, then $X > Y$; else, $X < Y$. However, if $\alpha_h = \beta_h$ for $n > h \geq 0$, then we have $X = Y$.

Based on Theorem 6 and Definition 1, we can directly derive a new residue number comparison algorithm for the most popular three-moduli set $\{2^n, 2^n + 1, 2^n - 1\}$ as the following.

Theorem 7. *Given any two positive integers $X = (x_3, x_2, x_1)$ and $Y = (y_3, y_2, y_1)$ with the moduli set $\{2^n - 1, 2^n, 2^n + 1\}$, we can do the comparison using their kernel sets $E(x) = (\alpha_2, \alpha_1, \alpha_0) = (A_X, B_X, x_1)$ and $E(y) = (\beta_2, \beta_1, \beta_0) = (A_Y, B_Y, y_1)$. For the first occurring $\alpha_h \neq \beta_h$, namely,*

Example 1

Compare the size of two residue numbers $X = 169 = (7, 1, 1)$ and $Y = 38 = (2, 6, 3)$ with the moduli set $\{9, 8, 7\}$

Solution 1: Diagonal Function in [4]

- $SQ = Q_1 + Q_2 + Q_3$, $Q_i = M/P_i$, $|-s_iP_i|_{SQ} = 1$, $i = 1, 2, 3$
- $Q_1 = 56$, $Q_2 = 63$, $Q_3 = 72$, $SQ = 191$
- $s_1 = 106$, $s_2 = 167$, $s_3 = 109$
- $D(X) = |s_1x_1 + s_2x_2 + s_3x_3|_{SQ} = |106 \times 7 + 167 \times 1 + 109 \times 1|_{191} = 63$
- $D(Y) = |s_1y_1 + s_2y_2 + s_3y_3|_{SQ} = |106 \times 2 + 167 \times 6 + 109 \times 3|_{191} = 13$
- $D(X) > D(Y) \Rightarrow X > Y$

Solution 2: New CRT in [5]

- $|k_0P_3|_{P_2} = |k_0 \times 7|_8 = 1 \Rightarrow k_0 = -1$
- $|k_1P_2P_3|_{P_1} = |k_1 \times 56|_9 = 1 \Rightarrow k_1 = 5$
- $X_0 = x_3 + |k_0(x_2 - x_3)|_{P_2}P_3 = 1 + |(-1) \times (1 - 1)|_8 \times 7 = 1$
- $Y_0 = y_3 + |k_0(y_2 - y_3)|_{P_2}P_3 = 3 + |(-1) \times (6 - 3)|_8 \times 7 = 38$
- $|k_1(x_1 - X_0)|_{P_1} = |5 \times (7 - 1)|_9 = 3$
- $|k_1(y_1 - Y_0)|_{P_1} = |5 \times (2 - 38)|_9 = 0$
- $|k_1(x_1 - X_0)|_{P_1} > |k_1(y_1 - Y_0)|_{P_1} \Rightarrow X > Y$

Solution 3: Theorems 6 and 7

- $T_{X_1} = x_{1,0}x_{1,2}x_{1,1} = 111$, $T_{Y_1} = y_{1,0}y_{1,2}y_{1,1} = 001$
- $T_{X_2} = \overline{x}_{2,2}\overline{x}_{2,1}\overline{x}_{2,0} = 110$, $T_{Y_2} = \overline{y}_{2,2}\overline{y}_{2,1}\overline{y}_{2,0} = 001$
- $T_{X_3} = x_{3,0}x_{3,2}x_{3,1} = 100$, $T_{Y_3} = y_{3,0}y_{3,2}y_{3,1} = 101$
- Since $x_1 \in [0, 2^3 - 1]$, $x_2 < x_1$, we have
- $A_X = \left|T_{X_1} + T_{X_2} + T_{X_3} - 1\right|_{2^n-1} = |111 + 110 + 100 - 1|_7 = 010$
- Since $y_1 \in [0, 2^3 - 1]$, $y_2 < y_1$, we have
- $A_Y = \left|T_{Y_1} + T_{Y_2} + T_{Y_3}\right|_{2^n-1} = |001 + 001 + 101|_7 = 000$
- $A_X > A_Y \Rightarrow X > Y$

Fig. 1. Example 1.

$\alpha_j = \beta_j$, where $3 > j > h \geq 0$, we have that if $\alpha_h > \beta_h$, then $X > Y$; else, $X < Y$. However, if $\alpha_h = \beta_h$ for $3 > h \geq 0$, then we have $X = Y$. Here, A_X and B_X are defined as follows (A_Y and B_Y can be calculated in the same way):

$$X = (2^n + 1)2^nA_X + (2^n + 1)B_X + x_1, \tag{6}$$

where $n > 1$, and

$$A_X = \begin{cases} |T_{X_1} + T_{X_2} + T_{X_3}|_{2^n-1}, \\ \quad \text{for } x_1 \in [0, 2^n - 1], x_2 \geq x_1, \\ |T_{X_1} + T_{X_2} + T_{X_3} - 1|_{2^n-1}, \\ \quad \text{for } x_1 \in [0, 2^n - 1], x_2 < x_1, \\ \left|(2^{n-1} - 1) + T_{X_2} + T_{X_3}\right|_{2^n-1}, \\ \quad \text{for } x_1 = 2^n, \end{cases}$$

$$B_X = \begin{cases} |x_2 - x_1|_{2^n}, & \text{for } x_1 \in [0, 2^n - 1], \\ x_2, & \text{for } x_1 = 2^n, \end{cases}$$

where

$$\begin{array}{lllll} T_{X_1} & = & |2^{n-1}x_1|_{2^n-1} & = & x_{1,0}x_{1,n-1} \ldots x_{1,1}, \\ T_{X_2} & = & |-x_2|_{2^n-1} & = & \overline{x}_{2,n-1} \ldots \overline{x}_{2,0}, \\ T_{X_3} & = & |2^{n-1}x_3|_{2^n-1} & = & x_{3,0}x_{3,n-1} \ldots x_{3,1}. \end{array}$$

Proof. For $\{2^n, 2^n + 1, 2^n - 1\}$, referred to Theorem 6, we have $X = (2^n + 1)2^nA_X + (2^n + 1)B_X + x_1$, where $n > 1$, and

$$\begin{aligned} A_X &= \left|\left\lfloor \frac{(2^{2n-1} - 1)x_1 + (2^n - 1)^2x_2 + 2^{2n-1}x_3}{2^n} \right\rfloor\right|_{2^n-1} \\ &= \left|\left\lfloor 2^{n-1}x_1 + 2^nx_2 - 2x_2 + 2^{n-1}x_3 + \frac{x_2 - x_1}{2^n} \right\rfloor\right|_{2^n-1}, \\ B_X &= \left|(2^{2n-1} - 1)x_1 + (2^n - 1)^2x_2\right|_{2^n} \\ &= |x_2 - x_1|_{2^n}. \end{aligned}$$

1. If $x_1 \in [0, 2^n - 1]$, namely, $x_{1,n} = 0$, since $x_2 \in [0, 2^n - 1]$, we have $x_2 - x_1 \in [-(2^n - 1), 2^n - 1]$:

   I. For $x_2 \geq x_1$, $x_2 - x_1 \in [0, 2^n - 1]$; then, $(x_2 - x_1)/2^n \in [0, 1)$. Thus, $\lfloor (x_2 - x_1)/2^n \rfloor = 0$. We have

   $$\begin{aligned} A_X &= \left|2^{n-1}x_1 + 2^nx_2 - 2x_2 + 2^{n-1}x_3\right|_{2^n-1} \\ &= \left|2^{n-1}x_1 - x_2 + 2^{n-1}x_3\right|_{2^n-1} \\ &= |T_{X_1} + T_{X_2} + T_{X_3}|_{2^n-1}, \\ B_X &= |x_2 - x_1|_{2^n}, \end{aligned}$$

   where

   $$\begin{array}{lllll} T_{X_1} & = & |2^{n-1}x_1|_{2^n-1} & = & x_{1,0}x_{1,n-1} \ldots x_{1,1} \quad \textit{by Lemma 1}, \\ T_{X_2} & = & |-x_2|_{2^n-1} & = & \overline{x}_{2,n-1} \ldots \overline{x}_{2,0} \quad \textit{by Lemma 2}, \\ T_{X_3} & = & |2^{n-1}x_3|_{2^n-1} & = & x_{3,0}x_{3,n-1} \ldots x_{3,1} \quad \textit{by Lemma 1}. \end{array}$$

   II. For $x_2 < x_1$, $x_2 - x_1 \in [-(2^n - 1), 0)$; then, $(x_2 - x_1)/2^n \in (-1, 0)$. We have

   $$\begin{aligned} A_X &= \left|2^{n-1}x_1 + 2^nx_2 - 2x_2 + 2^{n-1}x_3 - 1\right|_{2^n-1} \\ &= |T_{X_1} + T_{X_2} + T_{X_3}|_{2^n-1}, \\ B_X &= |x_2 - x_1|_{2^n}. \end{aligned}$$

2. If $x_1 = 2^n$, namely, $x_{1,n} = 1$, since $x_2 \in [0, 2^n - 1]$, we have $x_2 - x_1 \in [-2^n, -1]$; thus, $\lfloor (x_2 - x_1)/2^n \rfloor \in [-1, 0)$. We have

   $$\begin{aligned} A_X &= \left|2^{n-1}x_1 + 2^nx_2 - 2x_2 + 2^{n-1}x_3 - 1\right|_{2^n-1} \\ &= \left|2^{n-1}2^n - 1 - x_2 + 2^{n-1}x_3\right|_{2^n-1} \\ &= \left|(2^{n-1} - 1) + T_{X_2} + T_{X_3}\right|_{2^n-1}, \\ B_X &= |x_2 - x_1|_{2^n} \\ &= |x_2 - 2^n|_{2^n} \\ &= x_2. \end{aligned}$$

In conclusion, Theorem 7 holds for any residue number (x_3, x_2, x_1). □

The example shown in Fig. 1 demonstrates the improvement of the new residue comparison algorithms over the previous algorithms [4], [5] in the literature.

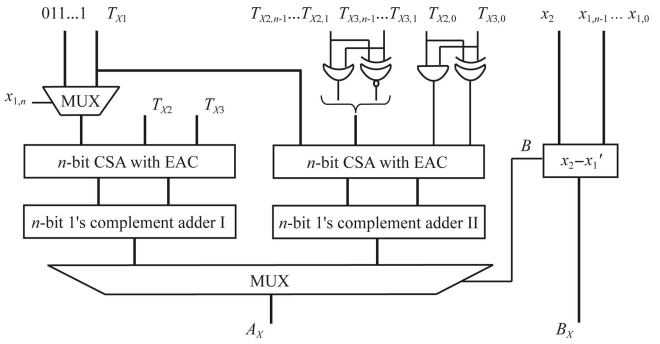


Fig. 2. A_X and B_X Generator 1.

When using the diagonal function in [4], the above residue number comparison needs a modulo-191 addition, which is not the low cost modulus of $2^n \pm 1$. The length of the modulo addition is 8 bits. By using the algorithm in [5], which is based on the New CRT, two levels of 4-bit modulo multipliers in series are required. Both the modulo-191 addition and the two modulo multipliers are implemented using ROM LUTs as suggested by the authors [4], [5]. If using the proposed algorithms, the same residue comparison requires only one 3-bit modulo-7 addition, which is a low cost modulus and can be implemented very efficiently without using any ROM LUTs.

With Theorem 7, we can implement a high-speed design of the residue number comparator for the three-moduli set $\{2^n-1, 2^n, 2^n+1\}$. There are different ways to do the implementations. The approaches using a parallel structure (high-speed design) and using a cascade structure (cost-effective design) are evaluated in the following section.

5 New RNS Comparators for $\{2^n-1, 2^n, 2^n+1\}$

In this section, based on Theorem 7, we present two high-speed and cost-effective residue comparators for the three-moduli set $\{2^n-1, 2^n, 2^n+1\}$.

Based on Theorem 7, to compare two residue numbers (x_3, x_2, x_1) and (y_3, y_2, y_1), we need to calculate four values: A_X, B_X, A_Y, and B_Y. Here, we only present how to derive the values of A_X and B_X since we can get A_Y and B_Y by the same way. It is easy to see that A_X is on the critical path that determines the delay of the comparator. Based on Theorem 7, the formulas of A_X can be rewritten as follows:

$$A_X = \begin{cases} |Z|_{2^n-1}, & for\ x_1 \in [0, 2^n-1], x_2 \geq x_1, \\ |Z-1|_{2^n-1}, & for\ x_1 \in [0, 2^n-1], x_2 < x_1, \\ |Z'|_{2^n-1}, & for\ x_1 = 2^n, \end{cases} \quad (7)$$

where $Z = T_{X_1} + T_{X_2} + T_{X_3}$, and $Z' = (2^{n-1}-1) + T_{X_2} + T_{X_3}$.

It is noted that there is a modulo decrement operation $|Z-1|_{2^n-1}$ in (7) that serves as a building block in the proposed residue comparators. There are different ways to implement the proposed comparators. One way is to do the calculations of $|Z-1|_{2^n-1}$ and $|Z|_{2^n-1}$ in parallel and leads to a high-speed concurrent design. The other way is to compute $|Z|_{2^n-1}$ at first and then use a modulo decrementer to calculate $|Z-1|_{2^n-1}$, which reduces the hardware resources and results in a cost-effective design.

In the following sections, we first present two design schemes of the A_X and B_X generator in Figs. 2 and 3, respectively. Fig. 5 is the structure of a new modulo decrementer. Then, we introduce the residue comparators, as shown in Fig. 6.

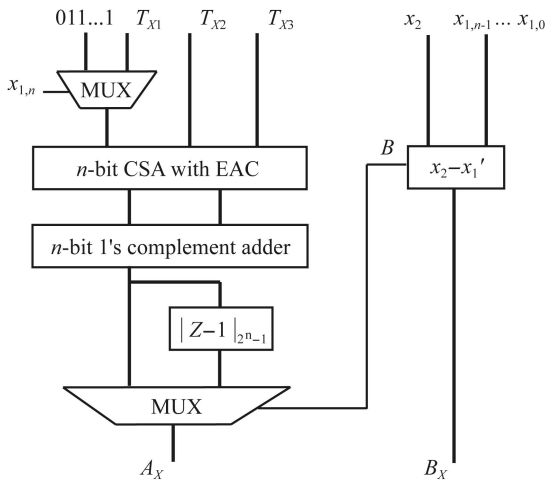


Fig. 3. A_X and B_X Generator 2.

5.1 Generator with Parallel Structure—Generator 1

We use the following three steps to present the parallel structure of Generator 1, which is used to calculate the A_X and B_X.

5.1.1 Calculation of $|Z|_{2^n-1}$ and $|Z'|_{2^n-1}$

Based on (7), we can see that the only difference between Z and Z' is the first item, T_{X_1} for Z and $2^{n-1}-1$ for Z'. We can integrate the calculation of $|Z|_{2^n-1}$ and $|Z'|_{2^n-1}$ using one mux (multiplexer) array, one stage of n-bit carry-save adder (CSA) with end-around-carry (EAC) and one n-bit 1's complement adder, as shown in Fig. 2. The detail structure of CSA with EAC can be found in [31]. The selecting signal of the mux array is $x_{1,n}$. When $x_{1,n} = 1$, namely, $x_1 = 2^n$, we have the output of the mux array as $2^{n-1}-1$; otherwise, we have the output as T_{X_1}. And the n-bit 1's complement adder can be efficiently implemented as an n-bit carry-propagation adder (CPA) with EAC, as suggested in [31], or using the parallel-prefix adder architecture, as presented in [32]. The former is a cost-effective design, but the delay is large, whereas the latter provides a high-speed design built at extra cost of complexity. Besides the above two design schemes, there exist many other choices in the literature for the 1's complement adder whose area and delay strongly depends on its implementation.

5.1.2 Calculation of $|Z-1|_{2^n-1}$

According to (7), if we compute the value of $|Z-1|_{2^n-1}$ simply as $|T_{X_1} + T_{X_2} + T_{X_3} - 1|_{2^n-1}$, then we have to manage four operands. We can simplify the computation as the following. Based on Lemma 2, we have

$$\begin{aligned} &|T_{X_1} + T_{X_2} + T_{X_3} - 1|_{2^n-1} \\ &\quad = |T_{X_1} + T_{X_2} + T_{X_3} + (11\ldots10)_2|_{2^n-1}. \end{aligned}$$

It may be noted that the full adder (FA) cells having logic one as one input can be simplified to an exclusive-NOR (XNOR) gate and an OR gate, while the FAs having logic zero

Example 2
Find the decrement results $Y = \|Z-1\|_{2^n-1}$ of the 4-bit numbers $Z = (0100)_2$ and $(0000)_2$.
Solution for $Z = (0100)_2$
• $Y_0 = Z_0 \oplus (Z_3 + Z_2 + Z_1 + Z_0) = 0 \oplus (0+1+0+0) = 1$
• $Y_1 = \overline{Z_1 \oplus Z_0} = \overline{0 \oplus 0} = 1$
• $Y_2 = \overline{Z_2 \oplus (Z_1 + Z_0)} = \overline{1 \oplus (0+0)} = 0$
• $Y_3 = Z_3 \oplus (Z_2 + Z_1 + Z_0) = 0 \oplus (1+0+0) = 0$
• $Y = (0011)_2$
Solution for $Z = (0000)_2$
• $Y_0 = Z_0 \oplus (Z_3 + Z_2 + Z_1 + Z_0) = 0 \oplus (0+0+0+0) = 0$
• $Y_1 = \overline{Z_1 \oplus Z_0} = \overline{0 \oplus 0} = 1$
• $Y_2 = \overline{Z_2 \oplus (Z_1 + Z_0)} = \overline{0 \oplus (0+0)} = 1$
• $Y_3 = Z_3 \oplus (Z_2 + Z_1 + Z_0) = 0 \oplus (0+0+0) = 1$
• $Y = (1110)_2$

Fig. 4. Example 2.

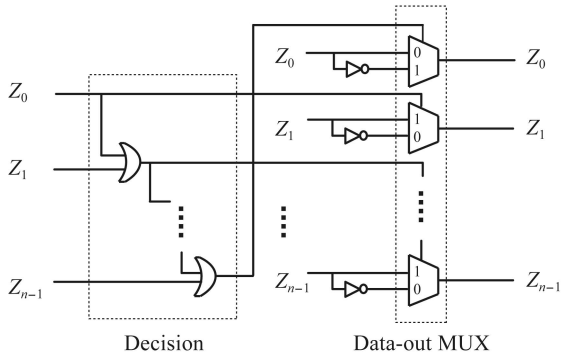


Fig. 5. The mux-based modulo decrementer.

as one input can be reduced to an exclusive-OR (XOR) gate and an AND gate. Then, instead of arranging four operands in two stages of CSAs and one n-bit 1's complement adder to compute $|T_{X_1} + T_{X_2} + T_{X_3} - 1|_{2^n-1}$, we can perform the summation of $|T_{X_1} + T_{X_2} + T_{X_3} + (11\cdots10)_2|_{2^n-1}$, as shown in Fig. 2. The n FA cells of the first stage of CSA can be reduced to n XOR/XNOR gates and n AND/OR gates, thus saving hardware.

5.1.3 Decision of A_x and B_x

Knowing the value of $|Z|_{2^n-1}$, $|Z'|_{2^n-1}$, and $|Z-1|_{2^n-1}$, we can determine A_X using a mux array. Based on (7), we can see that A_X equals to $|Z-1|_{2^n-1}$ when $x_1 \in [0, 2^n-1]$ and $x_2 < x_1$. Otherwise, A_X takes the value of $|Z|_{2^n-1}$ or $|Z'|_{2^n-1}$. Accordingly, the mux array uses the borrow out signal B of the subtractor $x_2 - x'_1$ as its selecting signal. Here, x'_1 consists of the n-bit least significant bits (LSBs) of x_1. Then, we have $x_1 = 2^n x_{1,n} + x'_1$. If $x_1 \in [0, 2^n-1]$ and $x_2 < x_1$, namely, $x_{1,n} = 0$ and $x_2 < x'_1$, then there is a borrow output from the subtractor. Accordingly, A_X has the value of $|Z-1|_{2^n-1}$, which is the output of 1's complement adder 2. If $x_1 \in [0, 2^n-1]$ and $x_2 \geq x_1$, namely, $x_{1,n} = 0$ and $x_2 > x'_1$, there is no borrow, and A_X takes the output of 1's complement adder 1 as its value. Then, we have $A_X = |Z|_{2^n-1}$. When $x_1 = 2^n \Rightarrow x_{1,n} = 1$, $x'_1 = 0$, there is no borrow either. Thus, A_X still takes the output of 1's complement adder 1 as its value, namely, $A_X = |Z'|_{2^n-1}$.

If we represent the subtraction $x_2 - x'_1$ as $x_2 - x'_1 = 2^n B + D$, where B is the borrow and D is the difference, then B_X is the difference D of the subtractor $x_2 - x'_1$, namely, $B_X = D = |x_2 - x'_1|_{2^n}$. For example, given $x'_1 = 6 = (110)_2$ and $x_2 = 1 = (001)_2$, we have $x_2 - x'_1 = -5 = -2^3 + (011)_2$, where $B_X = D = (011)_2$. Based on Theorem 7, this property can be easily verified as follows:

Given $X = (x_3, x_2, x_1) = (|X|_{2^n-1}, |X|_{2^n}, |X|_{2^n+1})$, we know that $x_1 = |X|_{2^n+1} = 2^n x_{1,n} + x'_1$ and $x_2 = |X|_{2^n}$. If $x_1 \in [0, 2^n-1]$, namely, $x_{1,n} = 0$, we have

$$B_X = |x_2 - x_1|_{2^n} = \left|x_2 - \left(2^n x_{1,n} + x'_1\right)\right|_{2^n} = \left|x_2 - x'_1\right|_{2^n} = D.$$

If $x_1 = 2^n$, namely, $x'_1 = 0$, we have

$$B_X = x_2 = |x_2|_{2^n} = \left|x_2 - x'_1\right|_{2^n} = D.$$

The whole design scheme of A_X and B_X Generator 1 can be found in Fig. 2. It consists of n OR/AND gates, n XOR/XNOR gates, n inverters, $2n$ muxes, $3n$ FAs, and two n-bit 1's complement adders. The delay of Generator 1 is the sum of the delay of an XOR t_{XOR}, the delay of an FA t_{FA}, the delay of an n-bit 1's complement adder $t_{1CA(n)}$, and the delay of a mux t_{mux}, i.e., $t_{Generator1} = t_{XOR} + t_{FA} + t_{1CA(n)} + t_{mux}$.

5.2 Generator with Cascade Structure—Generator 2

Another way of computing A_X and B_X is to arrange the calculations of $|Z|_{2^n-1}$, $|Z'|_{2^n-1}$ and $|Z-1|_{2^n-1}$ in a cascade structure, as shown in Fig. 3. There are also three steps in Generator 2. The first step is the calculation of $|Z|_{2^n-1}$ and $|Z'|_{2^n-1}$ which has the same implementation scheme as the step 1 in Generator 2. The second step is the calculation of $|Z-1|_{2^n-1}$. Knowing that $|Z-1|_{2^n-1} = \|Z|_{2^n-1} - 1|_{2^n-1}$, we can build in a modulo 2^n-1 decrementer following the first step. Depending on $x_{1,n} = 0$ or 1, the input of the modulo decrementer is $|Z|_{2^n-1}$ or $|Z'|_{2^n-1}$, and the corresponding output is $|Z-1|_{2^n-1}$ or $|Z'-1|_{2^n-1}$. Based on (7), we decide the correct values of A_X and B_X using a mux array in the third step. The selecting signal is the borrow out signal B of the subtractor $x_2 - x'_1$. If $x_{1,n} = 0$ and $x_2 < x'_1$, then there is a borrow out from the subtractor. Accordingly, A_X has the value of $|Z-1|_{2^n-1}$, which is the output of the modulo decrementer. If $x_{1,n} = 0$ and $x_2 \geq x'_1$, there is no borrow, and A_X takes the output of the 1's complement adder as its value, namely, $A_X = |Z|_{2^n-1}$. When $x_1 = 2^n \Rightarrow x_{1,n} = 1$ and $x'_1 = 0$, there is no borrow out. Thus, A_X still takes the output of the 1's complement adder as its value, namely, $A_X = |Z'|_{2^n-1}$. As to the computation of B_X, we have discussed it in the third step of Generator 1. Namely, the difference of the subtractor $x_2 - x'_1$ gives the value of B_X.

There are different ways to design the modulo 2^n-1 decrementer of $|Z-1|_{2^n-1}$ [10]. An efficient design has been given in [33].

Proposition 1. *Given any n-bit unsigned binary input $Z = Z_{n-1}\ldots Z_1 Z_0$, we get its modulo decrement result $Y = |Z-1|_{2^n-1}$ as follows:*

$$Y = |Z-1|_{2^n-1} \tag{8}$$

$$= \begin{cases} \overline{Z_j \oplus (Z_{j-1} + \cdots + Z_1 + Z_0)}, & 1 \leq j \leq n-1, \\ Z_0 \oplus (Z_{n-1} + \cdots + Z_1 + Z_0), & j = 0. \end{cases} \tag{9}$$

The decrement operation is illustrated by the example in Fig. 4.

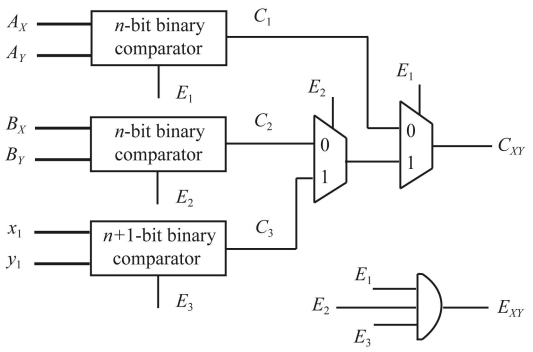


Fig. 6. The proposed residue comparator.

Based on Proposition 1, we can implement a mux-based modulo $2^n - 1$ decrementer, as shown in Fig. 5. In (9), the XNOR and the XOR operations can be implemented using a mux array. The series of OR operations $Z_{n-1} + \cdots + Z_1 + Z_0$ can be implemented using $\log n$ stages of OR gates, which is called as the decision module in Fig. 5. Then, the delay introduced by this mux-based modulo decrementer is only the delay of $\lceil \log n \rceil$ OR gates plus the delay of a mux, where $\lceil \bullet \rceil$ is the ceiling function. The required hardware consists of $\lceil \frac{n}{2} \log n \rceil$ OR gates, n inverters, and n muxes. For implementation in an FPGA, the data flow in each configurable logic block (CLB) is from left to right. The LUT for combinational logic implementation is on the left, and the muxes are on the right. The design in Fig. 5 pairs up OR gates and muxes. This structure has the advantage that it can be easily placed into the CLB and achieve the best case performance.

The Generator 2 with cascade structure in Fig. 3 consists of $\lceil \frac{n}{2} \log n \rceil$ OR gates, $3n$ muxes, n inverters, $2n$ FAs, and one n-bit 1's complement adder. The delay of Generator 2 is the sum of the delay of $\lceil \log n \rceil$ OR gates t_{OR}, the delay of an FA t_{FA}, the delay of an n-bit 1's complement adder $t_{1CA(n)}$, and the delay of three muxes t_{mux}, i.e., $t_{Generator2} = \lceil \log n \rceil t_{OR} + t_{FA} + t_{1CA(n)} + 3t_{mux}$.

Knowing the values of A_X, B_X, A_Y, and B_Y based on either Generator 1 or Generator 2, we can compare the magnitudes of X and Y using Theorem 7. As shown in Fig. 6, we use an n-bit binary comparator to compare A_X and A_Y. In the case of $A_X = A_Y$, we have $E_1 = 1$. If $A_X > A_Y$, then $C_1 = 1$, and $E_1 = 0$. If $A_X < A_Y$, then $C_1 = 0$, and $E_1 = 0$. Similarly, the other two binary comparators are used to compare B_X and B_Y and x_1 and y_1. There are two output signals of the proposed residue comparator. One is E_{XY}, which is used to indicate $X = Y$ when $E_{XY} = 1$. Since we have $E_1 = E_2 = E_3 = 1$ in the case of $X = Y$, we can generate E_{XY} using a three-input AND gate, as shown in Fig. 6. The other signal is C_{XY}. If $X > Y$, then $C_{XY} = 1$. If $X < Y$, then $C_{XY} = 0$. Based on Theorem 7, we can see that $C_{XY} = C_1$ if $E_1 = 0$, $C_{XY} = C_2$ if $E_1 = 1$ and $E_2 = 0$, or $C_{XY} = C_3$ if $E_1 = 1$ and $E_2 = 1$. Two muxes connected in a cascade way as shown in Fig. 6 can implement the logic for C_{XY}. The proposed residue comparator consists of two A_X and B_X generators, three binary comparators, one three-input AND gate, and two muxes. The delay of the proposed residue comparator is the sum of the delay of an A_X and B_X generator, the delay of an $(n+1)$-bit binary comparator $t_{BC(n+1)}$, and the delay of two muxes, i.e., $t_{Comparator} = t_{Generator1or2} + t_{BC(n+1)} + 2t_{mux}$.

5.3 Performance Evaluation

In this section, we present the performance evaluation of our new residue comparators and compare it with previous related algorithms.

Based on Theorem 7, modulo $2^n - 1$ is the only required modulo operation in the proposed residue comparison algorithm for $\{2^n - 1, 2^n, 2^n + 1\}$. Table 1 is the summary of the performances of the previous residue comparison algorithms. The "complexity" refers to the largest modulo operations or the largest integers involved in the residue comparison operation, which indicates the delay and the complexity of the residue comparators. It can be noted that the previous algorithms are more complex than our new algorithm. On the contrary, the proposed algorithm uses the smallest modulo operation and does not introduce any redundant modulus. In summary, our new algorithm is the best residue comparison algorithm based on the criteria listed in Table 1.

Recently, many fast R/B converter designs have been presented for $\{2^n - 1, 2^n, 2^n + 1\}$ in the literature. The CRT-based residue comparison algorithms have benefited from this technical innovation. For the convenience of comparison, we show a residue comparator in Fig. 7 that is one of the fastest existing VLSI converters for $\{2^n - 1, 2^n, 2^n + 1\}$. Based on $X = x_2 + 2^n |(A_1 + A_2 + C_{n+n}) + 2^n (B_1 + B_2)|_{2^{2n}-1} = x_2 + 2^n (B_X + 2^n A_X)$, it has been proposed in [34] as Converter III, which uses two n-bit adders to compute A_X and B_X, as shown in Fig. 7a. Then, we can compare two numbers X and Y, as shown in Fig. 7b.

To get a practical performance measure, the proposed comparators and the comparator based on Converter III in [34] are implemented using the most advanced Xilinx FPGA technology. The synthesis and implementation tools are Xilinx Synthesis Tool (XST) and Xilinx Integrated Software (ISE) flow 9.2i. The target technology is a Xilinx Virtex-5

TABLE 1
Performance Evaluation of Different Residue Comparison Algorithms

Algorithms	Complexity	Only for odd moduli	Redundant moduli
[2]	$2^{2n}(w_3 + w_2 + w_1) + 2^n(w_3 - w_1) - w_2$	yes	yes
[3]	$\log(2^{3n} - 2^n)n$	yes	yes
[4]	Modulo$(3 \cdot 2^{2n} - 1)$	no	no
[5]	Modulo$(2^{2n} - 1)$	no	no
Proposed	Modulo$(2^n - 1)$	no	no

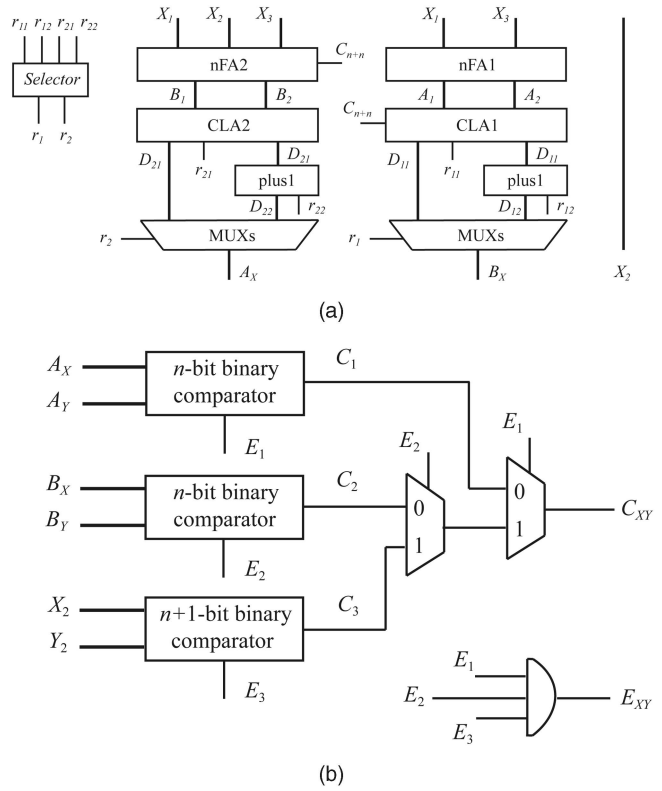


Fig. 7. The residue comparator based on [34]. (a) Converter III [34]. (b) Comparator.

xc5vlx50t-ff665-1 FPGA. The performance evaluation is carried out in terms of area and delay at the layout level. The reported area is evaluated using the number of occupied slices. The results are presented in Table 2. It can be seen that C_2, which is based on Generator 1, is the fastest design. C_2 is faster than C_1, which is based on Converter III in [34], by around 20 percent for small and middle wordwidth. For a large wordwidth such as 64 bits, the percentage drops to around 12 percent. The reason is that the FPGA routing becomes more difficult when the wordwidth is large. When the design becomes more complex, the available FPGA routing resource is comparatively decreased. As a consequence, more signals cannot be routed locally and large interconnection delay is introduced. For the applications where area is of prime importance, C_3, which is based on Generator 2, is the best choice since it consumes around 20 percent less hardware resource than the comparator in Fig. 7. The reason for such improvement is that the final modulo summation of the converter in [34] based on $2^{2n}-1$. The new residue comparator is derived from Theorem 7, where modulo 2^n-1 is the only required modulo operation. Thus, the proposed residue comparator reduces the modulo size by half. Since the modulo part is the critical path of the residue comparator, a high-speed implementation is obtained compared to the previous works.

6 Conclusion

In this paper, a new MR CRT has been proposed that possesses both the advantages of the CRT and the MRC, which are parallel processing, small operations modulo P_i only, and the efficiency of making comparison. Based on the proposed CRT, new residue comparators have been developed for the three-moduli set $\{2^n-1, 2^n, 2^n+1\}$. The FPGA implementation results show that the proposed modulo comparators are about 20 percent faster and smaller than one of the previous best designs.

References

[1] N. Szabo and R. Tanaka, *Residue Arithmetic and Its Applications to Computer Technology*. McGraw-Hill, 1967.

[2] 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," *Residue Number System Arithmetic, Modern Applications in Digital Signal Processing*, M.A. Soderstrand, W.C. Jenkins, G.A. Jullien, and F.J. Taylor, eds., pp. 390-402, IEEE Press, 1985.

[3] M. Lu and J. Chiang, "A Novel Division Algorithm for the Residue Number System," *IEEE Trans. Computers*, vol. 41, no. 8, pp. 1026-1032, Aug. 1992.

[4] G. Dimauro, S. Impedovo, and G. Pirlo, "A New Technique for Fast Number Comparison in Residue Number System," *IEEE Trans. Computers*, vol. 42, no. 5, pp. 608-612, May 1993.

[5] Y. Wang, X. Song, and M. Aboulhamid, "A New Algorithm for RNS Magnitude Comparison Based on New Chinese Remainder Theorem II," *Proc. Ninth Great Lakes Symp. VLSI (GSLVLSI '99)*, pp. 362-365, 1999.

[6] C.H. Huang, "A Fully Parallel Mixed-Radix Conversion Algorithm for Residue Number Applications," *IEEE Trans. Computers*, vol. 32, no. 4, pp. 398-402, Apr. 1983.

[7] W. Wang, M.N.S. Swamy, M.O. Ahmad, and Y. Wang, "A Study of Residue-to-Binary Converters for Three-Moduli Sets," *IEEE Trans. Computers*. vol. 50, no. 2, pp. 235-243, Feb. 2003.

[8] P.V.A. Mohan, "Comments on "Residue-to-Binary Converters Based on New Chinese Remainder Theorem"," *IEEE Trans. Circuits and Systems, Part II*, vol. 47, no. 12, p. 1541, Dec. 2000.

[9] P.V.A. Mohan, "Evaluation of Fast Conversion Techniques for Binary-Residue Number Systems," *IEEE Trans. Computers*, vol. 45, no. 10, pp. 1107-1109, Oct. 1998.

[10] P.V.A. Mohan, "Comments on 'Breaking the 2n-Bit Carry-Propagation Barrier in Residue to Binary Conversion for the $[2^n-1, 2^n, 2^n+1]$ Moduli Set'," *IEEE Trans. Computers*, vol. 48, no. 8, p. 1031, Aug. 2001.

TABLE 2
Complexity and Delay Comparison of CRT-Based Residue Comparators

Word-width	8-bit		16-bit		32-bit		64-bit	
Comparators	Area (slices)	Delay (ns)	Area (slices)	Delay (ns)	Area (slices)	Delay (ns)	Area (slices)	Delay (ns)
C_1: [34]-based	178	11.87	360	17.95	660	30.12	1218	55.25
C_2: Generator I	154	8.93	321	14.66	583	24.74	1057	48.66
C_3: Generator II	138	10.83	301	17.12	526	29.42	995	49.28
$\frac{C_1-C_2}{C_1}\times 100\%$	13.48	24.76	10.8	18.37	11.67	17.87	13.22	11.92
$\frac{C_1-C_3}{C_1}\times 100\%$	22.47	8.78	16.39	4.64	20.3	2.34	18.31	10.81

[11] W.K. Jenkins and J.V. Krogmeier, "The Design of Dual-Mode Complex Signal Processors Based on Quadratic Modular Number Codes," *IEEE Trans. Circuits and Systems,* vol. 34, pp. 354-364, 1987.

[12] S. Andraros and H. Ahmad, "A New Efficient Memoryless Residue to Binary Converter," *IEEE Trans. Circuits and Systems,* vol. 35, pp. 1441-1444, Nov. 1988.

[13] M. Bhardwaj, A.B. Premkumar, and T. Srikanthan, "Breaking the 2n-bit Carry Propagation Barrier in Residue to Binary Conversion for the $\{2^n-1, 2^n, 2^n+1\}$ Modula Set," *IEEE Trans. Circuits and Systems, Part I,* vol. 45, no. 9, pp. 998-1002, Sept. 1998.

[14] M. Bhardwaj, T. Srikanthan, and T. Clarke, "A Residue Converter for the 4-Moduli Superset $\{2^n-1, 2^n, 2^n+1, 2^{n+1}+1\}$," *Proc. 14th IEEE Symp. Computer Arithmetic,* Apr. 1999.

[15] A. Mohan, *RNS: Algorithms and Architectures.* Kluwer Academic Publishers, 2002.

[16] B. Bernardson, "Fast Memoryless, over 64 Bits, Residue to Decimal Converter," *IEEE Trans. Circuits and Systems,* vol. 32, pp. 298-300, Mar. 1985.

[17] K.M. Ibrahim and S. Saloum, "An Efficient Residue to Binary Converter Design," *IEEE Trans. Circuits and Systems,* vol. 35, pp. 1156-1158, Sept. 1988.

[18] G. Bi and E. Jones, "Fast Conversion between Binary and Residue Numbers," *Electronics Letters,* vol. 24, pp. 1195-1197, Sept. 1988.

[19] A. Dhurkadas, "Comments on 'An Efficient Residue to Binary Converter Design'," *IEEE Trans. Circuits and Systems, Part II,* vol. 37, pp. 849-850, 1990.

[20] K.A. Dhurkadas, "A High Speed Realization of a Residue to Binary Number System Converter," *IEEE Trans. Circuits and Systems II,* vol. 45, no. 3, pp. 446-447, Mar. 1998.

[21] D. Gallaher, F.E. Petry, and P. Srinivasan, "The Digit Parallel Method for Fast RNS to Weighted Number System Conversion for Specific Moduli $(2^k-1, 2^k, 2^k+1)$," *IEEE Trans. Circuits and Systems, Part II,* vol. 44, pp. 53-57, Jan. 1997.

[22] R. Conway and J. Nelson, "Fast Converter for 3 Moduli RNS Using New Property of CRT," *IEEE Trans. Computers,* vol. 48, no. 8, pp. 852-860, Aug. 1999.

[23] Y. Wang, M.N.S. Swamy, and M.O. Ahmad, "Three Number Moduli Sets Based Residue Number Systems," *IEEE Trans. Computers,* vol. 46, no. 2, pp. 180-183, Feb. 1999.

[24] P.V.A. Mohan, "Comments on 'The Digit Parallel Method for Fast RNS to Weighted Number System Conversion for Specific Moduli $(2^k-1, 2^k, 2^k+1)$'," *IEEE Trans. Circuits and Systems, Part II,* vol. 47, no. 9, pp. 972-974, 2000.

[25] Z. Wang, G.A. Jullien, and W.C. Miller, "An Improved Residue to Binary Converter," *IEEE Trans. Computers,* vol. 47, no. 9, pp. 1437-1440, Sept. 2000.

[26] W. Wang, M. Swamy, M. Ahmad, and Y. Wang, "A High-Speed Residue-to-Binary Converter for Three-Moduli $(2^k, 2^k-1, 2^k-1-1)$ RNS and a Scheme for Its VLSI Implementation," *IEEE Trans. Circuits and Systems, Part II,* vol. 47, no. 12, pp. 1576-1581, Dec. 2000.

[27] W. Wang, M.N.S. Swamy, M.O. Ahmad, and Y. Wang, "A Note on "A High-Speed Residue-To-Binary Converter for Three-Moduli $(2^k, 2^k-1, 2^k-1-1)$ RNS and a Scheme for Its VLSI Implementation"," *IEEE Trans. Circuits and Systems, Part II,* vol. 49, Mar. 2002.

[28] Y. Wang, "Residue-to-Binary Converters Based on New Chinese Remainder Theorems," *IEEE Trans. Computers,* vol. 49, no. 3, pp. 197-206, Mar. 2000.

[29] S. Bi, W. Wang, and A. Al-Khalili, "New Modulo Decomposed Residue-to-Binary Algorithm for General Moduli Sets," *Proc. IEEE Int'l Conf. Acoustics, Speech and Signal Processing (ICASSP '04),* vol. 5, pp. 141-144, 2004.

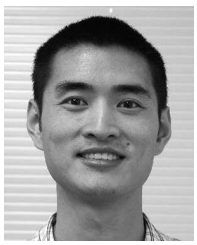
[30] B. Vinnakota and V.V. Rao, "Fast Conversion Techniques for Binary-Residue Number Systems," *IEEE Trans. Computers,* vol. 41, no. 12, pp. 927-929, Dec. 1994.

[31] S.J. Piestrak, "A High-Speed Realization of a Residue to Binary Number System Converter," *IEEE Trans. Computers,* vol. 44, no. 10, pp. 661-663, Oct. 1995.

[32] L. Kalampoukas, D. Nikolos, C. Efstathiou, H.T. Vergos, and J. Kalamatianos, "High-Speed Parallel-Prefix Modulo 2^n-1 Adders," *IEEE Trans. Computers,* vol. 49, no. 7, pp. 673-680, July 2000.

[33] S. Bi, "Low Power Modulo Reduction Technique and Its Application in Residue-to-Binary Converters," MEng thesis, Concordia Univ., Montreal, Mar. 2004.

[34] Y. Wang, X. Song, M. Aboulhamid, and H. Shen, "Adder Based Residue to Binary Number Converters for $\{2^n-1, 2^n, 2^n+1\}$," *IEEE Trans. Computers,* vol. 50, no. 7, pp. 1772-1779, July 2002.



Shaoqiang Bi received the BSc and MEng degrees in electronics and communications engineering from Huazhong University of Science and Technology in 1996 and 1999, respectively, and the MSc degree in electrical and computer engineering from Concordia University in 2004. He currently works at Xilinx Inc., San Jose, California, as a senior product verification engineer. His research interests are in computer arithmetic and FPGAs.

Warren J. Gross received the BASc degree in electrical engineering from the University of Waterloo, Waterloo, Ontario, Canada, in 1996, and the MASc and PhD degrees from the University of Toronto, Toronto, Ontario, in 1999 and 2003, respectively. Currently, he is an assistant professor with the Department of Electrical and Computer Engineering, McGill University, Montreal, Quebec. In the summers of 2004 and 2005, he was a visiting professor at the Université de Bretagne-Sud. Lorient, France. His research interests are in the design and application of signal processing microsystems and custom computer architectures. He is a member of the Design and Implementation of Signal Processing Systems Technical Committee of the IEEE Signal Processing Society. He served as the general chair of the Sixth Analog Decoding Workshop. He has served on the Program Committees for the IEEE Workshop on Signal Processing Systems, the IEEE Symposium on Field-Programmable Custom Computing Machines, and the International Conference on Field-Programmable Logic and Applications. He is a member of the IEEE and the IEEE Computer Society and is a licensed Professional Engineer in the Province of Ontario.