Efficient Residue Comparison Algorithm for General Moduli Sets

Shaoqiang Bi, Warren J. Gross
Dept. Electrical and Computer Engineering, McGill University.
Montreal, Quebec, Canada
email: shaoqiang.bi@mail.mcgill.ca, wjgross@ece.mcgill.ca

Abstract

In this paper, we propose new residue comparison algorithms for general moduli sets and present an efficient ROM-free residue comparator for $\{2^n+1,2^n,2^n-1\}$ using the smallest modulo operation without introducing redundant modulus. Compared to the residue comparator based on the fastest known residue-to-binary (R/B) converter, our design is faster and reduces the hardware close to half.

INTRODUCTION

The carry-free property of the residue number systems (RNS) has made it attractive as a basis for high-speed arithmetic operation [1]. However, due to the non-position nature of the RNS, magnitude comparison between residue numbers is much more complex than that in the weighted number system. This difficulty prevents a wide variety of general purpose computations from taking advantage of the residue arithmetic.

There are several known techniques for residue comparison. One is to use the Chinese remainder theorem (CRT) to convert a residue number to a binary number. However the CRT is based on large modulo operations which make a direct implementation of the CRT inefficient. A different technique based on the core function requires 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 solution using parity checking has been proposed in [3] which assumes that all moduli of the moduli set are odd and ROM look-up tables are mandatory to determine the operands parity. In [4], a different technique uses the diagonal function to compare residue numbers. It requires a large modulo operation which is usually implemented using look-up tables. Another technique based on the New CRT utilizes $t=\lfloor n/2 \rfloor+1$ level modulo multipliers in sequence [5]. All of these algorithms are rather complex and there is no feasible VLSI design of residue comparators have been presented for these algorithms.

In this paper, based on the improved CRT [6], [7], we present new residue comparison algorithms for general moduli sets and the most popular three-moduli set $M1=\{2^n+1,2^n,2^n-1\}$. The proposed algorithms use the smallest modulo operation, exclude the utilization of ROM look-up tables and do not introduce any redundant modulus. This is contrasted with all the previous algorithms, wherein the moduli must be kept reasonably small to avoid excessive cost of the comparators realized by ROM's. Based on the new algorithms, we propose an area-reduced and high-speed design of a residue comparator for $M1$ which is efficient, practical and easy to implement in VLSI. Compared to the residue comparator based on the fastest known residue-to-binary (R/B) converter, our design is faster and reduces the hardware close to half.

BACKGROUND MATERIALS

Let $\{P_1,P_2,\cdots,P_n\}$ be a set of positive numbers all greater than 1. The P_i's are called moduli and the n-tuple set $\{P_1,P_2,\cdots,P_n\}$ is called the moduli set. In order to avoid redundancy, the moduli of a residue number system must be pair-wise 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 $X=(x_1,x_2,\cdots,x_n)$. Such a representation is unique for any integer $X \in [0,M-1]$, where $M=P_1P_2\cdots P_n$ is the dynamic range of the moduli set $\{P_1,P_2,\cdots,P_n\}$ [1]. To convert a residue number $(x_1,x_2,...,x_n)$ into its binary representation X, the CRT is widely used.

Chinese Remainder Theorem the binary number X is computed by $X=\left|\sum_{i=1}^{n} N_i \left|N_i^{-1}\right|_{P_i} x_i\right|_M$ where $N_i=M/P_i$, and $\left|N_i^{-1}\right|_{P_i}$ is the multiplicative inverse of $|N_i|_{P_i}$.

The CRT requires a large modulo operation of size M that is not efficient. In [6] and [7], a modulo reduction technique has been proposed to reduce the modulo base of the CRT and results in the improved CRT algorithm as follows.

Theorem 1 Given $\{P_1,P_2,...,P_n\}$, the residue number $(x_1,x_2,\ldots,x_n)$ is converted to the binary number X by

$$X=\sum_{m=1}^{n-2}\left(\left\lfloor \sum_{i=1}^{n} w_i x_i' \Big/ \prod_{i=2}^{m+1} P_i \right\rfloor_{P_{m+2}} \prod_{i=1}^{m+1} P_i\right) + P_1\left|\sum_{i=1}^{n} w_i x_i'\right|_{P_2} + x_1 \qquad (1)$$

where $n>1$, $w_1=\left(N_1\left|N_1^{-1}\right|_{P_1}-1\right)\Big/P_1$, $w_i=\dfrac{N_i}{P_1}$, $x_1'=x_1$, $x_i'=\left|N_i^{-1}x_i\right|_{P_i}$, for $i=2,3,\ldots,n$. $\lfloor \bullet \rfloor$ is the floor function.

The improved CRT can be used to derive efficient R/B algorithm for $M1$ as follows [6].

Theorem 2 For the three-moduli set $M1=\{2^n+1, 2^n, 2^n-1\}$, we have

$$X=(2^n+1)2^n A_X+(2^n+1)B_X+x_1 \qquad (2)$$

0-7803-9197-7/05/$20.00 © 2005 IEEE.

where $n>1$, $A_X = \left\| \left\lfloor \frac{K_x}{2^n} \right\rfloor \right\|_{2^n-1}$, $B_X = |K_x|_{2^n}$, and $K_x = \left(2^{2n-1}-1\right)x_1 + \left(2^n-1\right)^2 x_2 + 2^{2n-1}x_3$.

NEW RESIDUE COMPARISON ALGORITHMS

Theorems 1 and 2 considerably reduce 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 which are highly concurrent and suitable for VLSI implementation.

We rewrite (1) as follows.

$$X = \sum_{m=1}^{n-2}\left(\alpha_{m+1}\prod_{i=1}^{m+1}P_i\right) + \alpha_1 P_1 + \alpha_0 \tag{3}$$

where $\alpha_{m+1} = \left\| \left\lfloor \sum_{i=1}^{n} w_i x_i' \Big/ \prod_{i=2}^{m+1} P_i \right\rfloor \right\|_{P_{m+2}}$, $\alpha_1 = \left| \sum_{i=1}^{n} w_i x_i' \right|_{P_2}$, $\alpha_0 = x_1$.

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

Similarly, we can define the kernel set $E(y) = (\beta_{n-1},...,\beta_1,\beta_0)$ for $Y=(y_1,y_2,\ldots,y_n)$. With Definition 1, we introduce a new residue comparison algorithm for the general moduli set as follows.

Theorem 3 Given any two positive integers $X=(x_1,x_2,\ldots,x_n)$ and $Y=(y_1,y_2,\ldots,y_n)$ with the general moduli set $\{P_1,P_2,...,P_n\}$, we can do the comparison using their kernel sets $E(x) = (\alpha_{n-1},...,\alpha_1,\alpha_0)$ and $E(y) = (\beta_{n-1},...,\beta_1,\beta_0)$. Without losing generality, assuming α_i and β_i are the first occurring pair of non-equal elements in $E(x)$ and $E(y)$ respectively, namely, $\alpha_i \neq \beta_i$, and $\alpha_j = \beta_j$ for $n > j > i \geq 0$, we have: if $\alpha_i > \beta_i$, then $X > Y$, else $X < Y$. However, if $\alpha_i = \beta_i$ for $n > i \geq 0$, then we have $X = Y$.

(Proof is omitted due to lack of space)

With Theorem 3, the comparison of two residue numbers is simplified to comparing their kernel sets. The calculation of each element α_i (or β_i) only requires a small size modulo operation which is based on P_i. Moreover, the comparisons of α_i and β_i can be done in parallel. Thus, Theorem 3 results in a parallel and high-speed operation.

Based on Theorem 3, we can directly derive a new residue number comparison algorithm for the most popular three moduli set $M1$ as Theorem 4.

Theorem 4 Given any two positive integers $X=(x_1,x_2,x_3)$ and $Y=(y_1,y_2,y_3)$ with the moduli set $M1=\{2^n+1,2^n,2^n-1\}$, we can do the comparison using their kernel sets $E(x) = (\alpha_2,\alpha_1,\alpha_0) = (A_X,B_X,x_1)$ and $E(y) = (\beta_2,\beta_1,\beta_0) = (A_Y,B_Y,y_1)$, where A_X, B_X, A_Y and B_Y are defined in Theorem 2. For the first occurring $\alpha_i \neq \beta_i$, namely $\alpha_j = \beta_j$ where $3 > j > i \geq 0$, we have: if $\alpha_i > \beta_i$, then $X > Y$, else $X < Y$. However, if $\alpha_i = \beta_i$ for $3 > i \geq 0$, then we have $X = Y$.

(Proof is omitted due to lack of space)

Theorem 2 has given the definitions of A_X, B_X, A_Y and B_Y. However, it can be further simplified for efficient VLSI implementation.

Theorem 5 Given the three-moduli set $M1=\{2^n+1,2^n,2^n-1\}$, the residue number (x_1,x_2,x_3) is converted into the binary number X by

$$X = \left(2^n+1\right)2^n A_X + \left(2^n+1\right)B_X + x_1 \tag{4}$$

where $n>1$ and

$$A_X = \begin{cases} |T_{X1}+T_{X2}+T_{X3}|_{2^n-1} = |Z|, & x_1 \in \left[0, 2^n-1\right], x_2 \geq x_1 \\ |T_{X1}+T_{X2}+T_{X3}-1|_{2^n-1} = |Z-1|, & x_1 \in \left[0, 2^n-1\right], x_2 < x_1 \\ \left|\left(2^{n-1}-1\right)+T_{X2}+T_{X3}\right|_{2^n-1} = |Z'|, & x_1 = 2^n \end{cases}$$

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

where

$T_{X1} = \left|2^{n-1}x_1\right|_{2^n-1} = x_{1,0}x_{1,n-1}\cdots x_{1,1}$, $T_{X2} = \left|-x_2\right|_{2^n-1} = \overline{x}_{2,n-1}\cdots\overline{x}_{2,0}$ and $T_{X3} = \left|2^{n-1}x_3\right|_{2^n-1} = x_{3,0}x_{3,n-1}\cdots x_{3,1}$.

(Proof is omitted due to lack of space)

In the following, we use an example to show the improvement of the new residue comparison algorithms over the previous algorithms [4], [5] in the literature.

Example 1 Compare two residue numbers X=58=(4, 2, 2) and Y=261=(0, 5, 2) with the moduli set {9, 8, 7}.

Based on the diagonal function in [4], we have

$D(X) = \left|s_1x_1 + s_2x_2 + s_3x_3\right|_{SQ} = \left|106\times4 + 167\times2 + 109\times2\right|_{191} = 21$

$D(Y) = \left|s_1y_1 + s_2y_2 + s_3y_3\right|_{SQ} = \left|106\times0 + 167\times5 + 109\times2\right|_{191} = 98$

$D(X) < D(Y) \Rightarrow X < Y$.

Based on the algorithm in [5], we have

$X_0 = x_3 + \left|k_0\left(x_2 - x_3\right)\right|_{P_2} P_3 = 2 + \left|(-1)\times(2-2)\right|_8 7 = 2$

$Y_0 = y_3 + \left|k_0\left(y_2 - y_3\right)\right|_{P_2} P_3 = 2 + \left|(-1)\times(5-2)\right|_8 7 = 37$

$\left|k_1\left(x_1 - X_0\right)\right|_{P_1} = \left|5(4-2)\right|_9 = 1$, $\left|k_1\left(y_1 - Y_0\right)\right|_{P_1} = \left|5(0-37)\right|_9 = 4$

$\left|k_1\left(x_1 - X_0\right)\right|_{P_1} < \left|k_1\left(y_1 - Y_0\right)\right|_{P_1} \Rightarrow X < Y$

Based on Theorem 4 and Theorem 5,

since $x_1 \in \left[0, 2^3-1\right], x_2 < x_1$, we have

$A_X = \left|T_{X1}+T_{X2}+T_{X3}-1\right|_{2^n-1} = \left|010+101+001-1\right|_7 = 000$

since $y_1 \in \left[0, 2^3-1\right], y_2 > y_1$, we have

$A_Y = \left|T_{Y1}+T_{Y2}+T_{Y3}\right|_{2^n-1} = \left|000+010+001\right|_7 = 011$

Then, we have $A_X < A_Y \Rightarrow X < Y$.

The above residue number comparison requires a modulo-191 addition if using the diagonal function in [4]. The length of the modulo addition is 8-bit. By using the algo-

rithm 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 look-up tables 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 can be implemented very efficiently without using any ROM look-up tables.

PROPOSED RNS COMPARATOR FOR M1

Based on the formulas of A_X in Theorem 5, we have $Z = T_{X1} + T_{X2} + T_{X3}$ and $Z' = \left(2^{n-1} - 1\right) + T_{X2} + T_{X3}$. Since the only difference between Z and Z' is the first item, T_{X1} 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 array, one stage of n-bit carry-save adder (CSA) with end-around-carry (EAC) and one n-bit 1's complement adder. The calculation of $|Z-1|_{2^n-1}$ can be implemented simply using a modulo decrementer. The detail structure of the A_X and B_X generator can be found in Figure 1. The second MUX array uses the carry-out signal C_{out} of the subtractor $x_2 - x_1'$ as its selecting signal. Here, x_1' consists of the n-bit least significant bits (LSB) of x_1. Namely, we have $x_1 = 2^n x_{1,n} + x_1'$. 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 the difference D gives the value of B_X. Based on the definition of B_X in Theorem 5, this property can be easily verified.

Based on the proposed generator, we can compare the magnitudes of X and Y using Theorem 4. As shown in Figure 2, we use a 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. 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 Figure 2. The other signal is C_{XY}. Based on Theorem 4, we can see that if E_1=0 then C_{XY}=C_1, else if E_1=1 and E_2=0 then C_{XY}=C_2, else if E_1=1, E_2=1 and E_2=0 then C_{XY}=C_3. Two MUX's connected in a cascade way as shown in Figure 2 can implement the logic for C_{XY}.

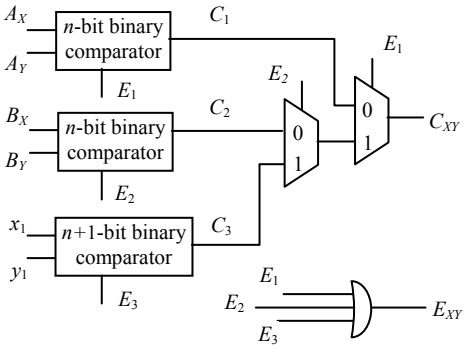


Figure 2. The proposed residue comparator.

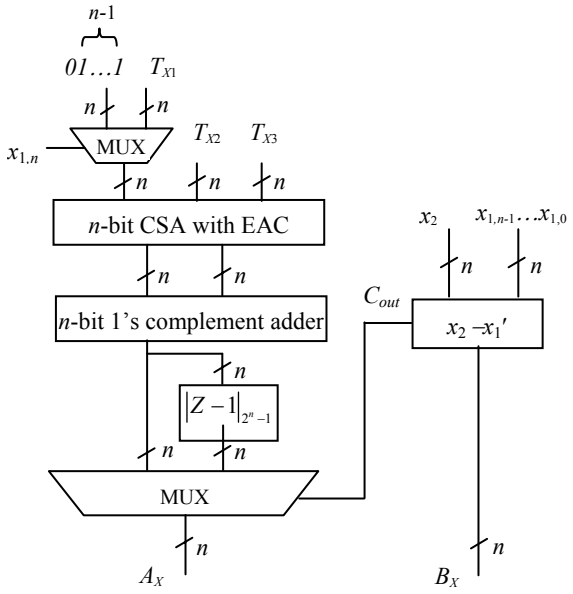
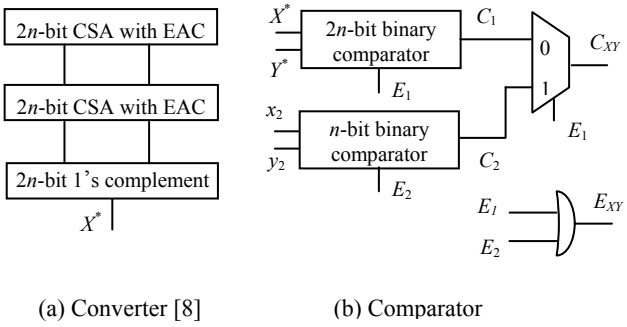


Figure 1. The A_X and B_X generator.

PERFORMANCE EVALUATION

In the literature, some fast designs of R/B converter for $M1$ have been presented recently. The CRT-based residue comparison algorithms have benefited from this technical innovation. For the convenience of comparison, we show a residue comparator in Figure 3 which is based on Piestrak's CRT converter [8] (shown in Figure 3 (a)) that is one of the fastest known VLSI converters for $M1$. Based on $X = x_2 + 2^n \left| A + B + C - x_1 \right|_{2^{2n}-1} = x_2 + 2^n X^*$, Piestrak has suggested to use two stages of $2n$-bit CSA's with EAC and one $2n$-bit 1's complement adder to compute X^*. Then, we can compare two numbers X and Y as shown in Figure 3 (b).

Table 1 summarizes the comparison of the proposed residue comparator with the comparator in Figure 3. It can be seen that the proposed comparator is faster and reduces the area close to half compared to the design in Figure 3. The reason for such improvement is that modulo $2^n - 1$ is the only required modulo operation for the new residue comparator. On the other hand, Piestrak's converter and its



(a) Converter [8] (b) Comparator

Figure 3. The residue comparator based on [8].

Table 1. Performance comparison of CRT-based residue comparators.

Comparator	OR/ AND	XOR/ XNOR	FA	INV	MUX	n-bit 1's complement adder	$2n$-bit 1's complement adder	n/n+1-bit binary comparator	$2n$-bit binary comparator	Delay
[8]-based	$4n$-2	$4n$	$8n$	$4n$	1	–	2	1	1	t_{INV}+2t_{FA}+t_{MUX} +$t_{1CA(2n)}$+$t_{BC\,(2n)}$
Proposed	$\lfloor n\log n \rfloor$	–	$4n$	$2n$	$6n$+2	2	–	3	–	$\lceil \log n \rceil$ t_{OR}+t_{FA}+5t_{MUX}+$t_{1CA(n)}$+$t_{BC(n+1)}$

modifications are based on the formula $\left|A+B+C-x_1\right|_{2^{2n}-1}$, where the final modulo summation requires to use $2n$-bit 1's complement adders. Thus the proposed comparator reduces the modulo size by half. Since the modulo part is the critical path of the residue comparator, a more efficient design is obtained compared to all residue comparators based on CRT converters.

Table 2 is the performance summary of the previous residue comparison algorithms for $M1$. 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 modulo 2^n-1 is the only required modulo operation in the proposed residue comparison algorithm and all of the previous algorithms are more complex than our new algorithm. Moreover, they all assume the use of ROM look-up tables that are expensive and not suitable for low power designs. Besides, due to the complexity and the ROM-based property of the previous residue comparison algorithms, there is no feasible VLSI design of residue comparators having been presented for these algorithms. In summary, our new algorithm is the best residue comparison algorithm based on the criteria listed in Table 2, which uses the smallest modulo operation, excludes the utilization of ROM look-up tables and does not introduce any redundant modulus.

Table 2. Comparison of different algorithms.

	Complexity	ROM based	Only for odd moduli	Redundant moduli	Feasible VLSI design
[2]	$2^{2n}\sum_{i=1}^{3} w_i + 2^n(w_3-w_1)-w_2$	yes	yes	yes	no
[3]	$\log(2^{3n}-2^n)n$	yes	yes	yes	no
[4]	Modulo ($3\cdot 2^{2n}-1$)	yes	no	no	no
[5]	Modulo ($2^{2n}-1$)	yes	no	no	no
Proposed Algorithm	Modulo (2^n-1)	no	no	no	yes

CONCLUSION

The proposed residue comparison algorithm and its application to $M1$ provide a novel way to design size-reduced, fast and ROM-free residue comparators using the smallest modulo operation without introducing any redundant modulus. Other residue algorithms such as sign test, overflow detection and division might benefit too. It is expected that the proposed technique will have many other applications in RNS study.

REFERENCES

[1] N. Szabo and R. Tanaka, *Residue Arithmetic and its Applications to Computer Technology*. New York: 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," 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.

[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, August 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," *IEEE Ninth Great Lakes symposium on VLSI*, pp. 362-365, 1999.

[6] Shaoqiang Bi, Wei Wang and Asim Al-Khalili, "Modulo deflation in (2^n +1, 2^n, 2^n –1) converters", *Proc. IEEE International Symposium on Circuits and Systems*, vol. 2, pp. 429-432, 2004.

[7] Shaoqiang Bi, Wei Wang and Asim Al-Khalili, "New modulo decomposed residue-to-binary algorithm for general moduli sets", *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing*, vol. 5, pp. 141-144, 2004.

[8] S. J. Piestrak, "A high-speed realization of a residue to binary number system converter," *IEEE Trans. CAS-II*, vol.42, pp. 661-663, Oct. 1995.