A High-Speed Realization of Chinese Remainder Theorem

Shuangching Chen and Shugang Wei
Department of Computer Science
Gunma University
Tenjin-cho 1-5-1, Kiryu-shi, Gunma 376-8515
Japan

Abstract: This paper proposes a novel technique for the Chinese remainder theorem (CRT) with the moduli $(2^n - 1, 2^n, 2^n + 1)$. Hardware implementation of the proposed CRT algorithm utilizes two kinds of parallel adders. One is referred to as modulo signed-digit m adder (MSDA) which performs a fast propagation-free addition and allows for the annihilation of carry or borrow chains using redundant binary number representation. Another implements parallel prefix adder which is the evolution of carry-lookahead adder (CLA). Compared to 16-digit Piestrak's high-speed converter , the computation time is shorten by 34%.

Key–Words: Residue number system, Signed-digit number, Carry-lookahead adder, Parallel prefix adder, Chinese remainder theorem,

1 Introduction

The residue number system (RNS) is an integer number system whose most important property is that additions, subtractions and multiplications are inherently carry-free [1, 2]. The residue number architecture with the three moduli $(2^n - 1, 2^n, 2^n + 1)$ has been widely used, since the residue addition can be performed by a binary adder [3, 4].

The residue number arithmetic can not be used for some applications, because the RNS does not have weights in the residue digits. Residue-to-binary number conversions are the crucial steps for any successful RNS application. For general moduli sets, residue-to-binary number conversions are based on the Chinese Remainder Theorem or Mixed Radix Conversion.

The three moduli $(2^n - 1, 2^n, 2^n + 1)$ can be performed efficiently with limited amount or even without ROM [5, 6, 7]. The complexity of the conversion has been greatly reduced by using compact forms with the multiplicative inverse and the properties of modular arithmetic. However, since the residue arithmetic mod (2^n+1) requires $(n+1)$ bits to represent (2^n+1) states, it is not easy to perform mod $(2^n + 1)$ arithmetic with a end-around-carry adder. To overcome the drawbacks, several residue-to-binary arithmetic converters for the moduli set $(2^n - 1, 2^n, 2^{n-1} - 1)$ have been proposed [8, 9].

In this paper, we present a novel CRT algorithm using the redundant representation to improve the computation time. The primary advantage is that it is easy to represent $(2^n + 1)$ states for mod $(2^n + 1)$ with a signed-digit (SD) number representation. Wei et al.[10] have shown that it is efficient to use the SD number representation to design the three moduli $(2^n - 1, 2^n, 2^n + 1)$ adders without carry propagation. The multiplicative inverses of three moduli $(2^n - 1, 2^n, 2^n + 1)$ are also used to simplify our conversion algorithm.

2 Preliminaries

2.1 Residue Number System

In this paper, we consider a residue number system which has a set of relatively prime moduli, $\{2^n, 2^n - 1, 2^n + 1\}$. A residue digit with respect to a modulus m_i is represented by the number set:

$$l_{m_i} = \{0, \cdots, (m_i - 1)\}. \qquad (1)$$

Let $M = \prod_{i=1}^{3} m_i = 2^n(2^n - 1)(2^n + 1)$. Szabo et al. [1] proved that if $0 \le A < M$, then the integer A has an one-to-one correspondence to its RNS representation. The integer A is uniquely represented by a 3-tuple (a_1, a_2, a_3), where

$$a_i = |A|_{m_i} = A - [A/m_i] \times m_i, \qquad (2)$$

for $i = 1, 2, 3$. In the above equation, $[A/m_i]$ is the integer part, and each residue digit is defined as the remainder of least magnitude when A is divided by m_i.

Table 1: Rules for adding binary SD numbers

	$abs(x_i) = abs(y_i)$	$abs(x_i) \neq abs(y_i)$	
		$(x_i + y_i) \times (x_{i-1} + y_{i-1}) \leq 0$	$(x_i + y_i) \times (x_{i-1} + y_{i-1}) > 0$
w_i	0	$x_i + y_i$	$-(x_i + y_i)$
c_i	$(x_i + y_i)/2$	0	$x_i + y_i$

2.2 the Extended Dynamic Range for SD Number Representation

A residue number X can be represented by an n-digit radix-two SD number representation as follows:

$$X = x_{n-1}2^{n-1} + x_{n-2}2^{n-2} + \cdots + x_0, \quad (3)$$

where $x_i \in \{-1, 0, 1\}$, and X can be denoted as $(x_{n-1}, x_{n-2}, \cdots, x_0)_{SD}$. To simplify the manipulation of the modular operation in an SD number representation, we apply the definition that each residue digit has the following redundant residue number set:

$$\begin{aligned} L_{m_i} &= \{-(2^n - 1), \cdots, 0, \cdots, (m_i - 1), \\ &\quad \cdots, (2^n - 1)\}, \end{aligned} \quad (4)$$

Thus, X must be in L_{m_i} when it is expressed in an n-digit SD number representation. Obviously,

$$\begin{aligned} -X &= -(x_{n-1}, x_{n-2}, \cdots, x_0)_{SD} \\ &= (-x_{n-1}, -x_{n-2}, \cdots, -x_0)_{SD} \end{aligned}$$

is also in L_{m_i}.

Definition 1 *Let Y be an SD number representation and m be a modulus. Then $y = \langle Y \rangle_m$ is defined as an integer in L_m. When $|Y|_m \neq 0$, y has one of two possible values given by equations*

$$y = \langle Y \rangle_m = |Y|_m, \quad (5)$$

and

$$y = \langle Y \rangle_m = |Y|_m - sign(|Y|_m) \times m, \quad (6)$$

where

$$\text{sign}(s) = \begin{cases} -1 & s < 0 \\ 1 & s \geq 0 \end{cases}.$$

When $|Y|_m = 0$ and $m = 2^n - 1$, there are three possible values for y, that is, $-m$, 0 and m. However, it is difficult to know if Y is in l_m, because of the redundancy of the SD number representation.

The numbers as the intermediate results calculated in L_m are used for fast residue arithmetic. If necessary for a final result, they can be converted into l_m.

2.3 Modulo m Signed-Digit Adder

A novel residue arithmetic hardware algorithm using a radix-two SD number representation has been proposed to implement the modulo m multiplication for the symmetric RNS [10, 11]. It is the key to increase the computation speed of such modular addition.

Figure 1 illustrates a circuit diagram of an n-digit modulo m Signed-Digit adder (MSDA) with n SD full adders (SDFAs), where $m = 2^n + \mu$ and $\mu \in \{-1, 0, 1\}$. One SDFA consists of ADD1 and ADD2. ADD1 generates the intermediate sum and the intermediate carry, and ADD2 sums the low intermediate carry and the intermediate sum. Let c_i and w_i be the carry and the intermediate sum of the ith digit position, respectively. Their values are determinated by Table 1 with respect to the values of $x_i, y_i, x_{i-1}, y_{i-1}$. Thus, the modulo m addition can be performed in parallel without the carry propagation. We use $\otimes$ to mean an 1-by-1 multiplier.

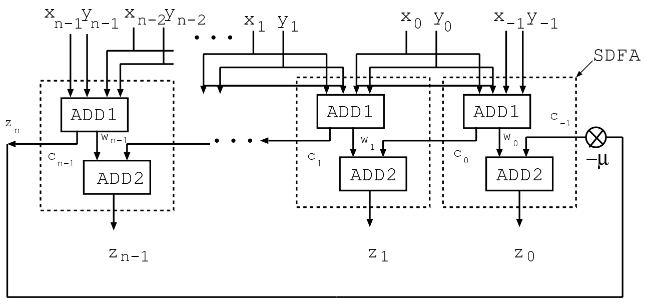

Figure 1: Modulo m Signed-Digit adder (MSDA)

3 Chinese Remainder Theorem

A number X can be generally represented as $X = (x_n, x_{n-1}, \cdots, x_1)$ in an RNS, where $0 \leq x_i < m_i$. To convert $(x_n, x_{n-1}, \cdots, x_1)$ into the binary number representation X_B, the following CRT is generally used.

$$X_B = \left| \sum_{i=1}^{n} \left(\hat{m}_i \left| \frac{1}{\hat{m}_i} \right|_{m_i} x_i \right) \right|_M \quad (7)$$

where

$\hat{m}_i = \frac{M}{m_i}$	
$M = \prod_{i=1}^{n} m_i$	dynamic range
$\{m_1, m_2, \cdots, m_n\}$	relatively prime integers
$\left\|\frac{1}{\hat{m}_i}\right\|_{m_i}$	multiplicative inverse of $\hat{m}_i$

3.1 CRT for Moduli $(2^n-1, 2^n, 2^n+1)$ Using Redundant Binary Number Architecture

The proposed conversion is shown in Fig.2, and the procedure is performed in the following two stages. In the first stage, the main conversion is based on the SD number architecture and we have no carry propagation during addition. In the second stage, for one-to-one correspondence, we convert numbers in L_m to those in l_m and the output is in the binary number representation.

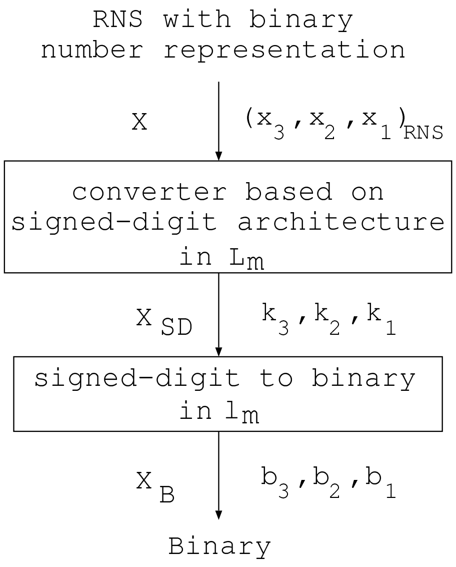
RNS with binary number representation
X
(x3 , x2 , x1)RNS
converter based on signed-digit architecture in Lm
X SD
k3 , k2 , k1
signed-digit to binary in lm
X B
b3 , b2 , b1
Binary

Figure 2: Residue-to-binary number conversions

Let $m_1 = 2^n, m_2 = 2^n - 1, m_3 = 2^n + 1$. Then $\hat{m}_1 = 2^{2n} - 1, \hat{m}_2 = 2^{2n} + 2^n, \hat{m}_3 = 2^{2n} - 2^n$. We use the modulo operation $\langle\cdot\rangle_m$ to deal with $|\cdot|_m$, and then the multiplicative inverse is shown as follows:

$$\left\langle\frac{1}{\hat{m}_1}\right\rangle_{m_1} = -1, \tag{8}$$

$$\left\langle\frac{1}{\hat{m}_2}\right\rangle_{m_2} = 2^{n-1}, \tag{9}$$

$$\left\langle\frac{1}{\hat{m}_3}\right\rangle_{m_3} = 2^{n-1}+1. \tag{10}$$

The proofs of (8), (9), (10) are based on the fact that $\langle\hat{m}_i\langle\frac{1}{\hat{m}_i}\rangle\rangle_{m_i} = 1$. Let x_1, x_2, x_3 be residue numbers for modulo $2^n, 2^n-1, 2^n+1$. Because a binary representation is one in the redundant number representation, we don't need any computation for the conversion from the binary number representation to the redundant one. Thus the calculation of CRT can be written as

$$X_{SD} = \langle A + B + C\rangle_M \tag{11}$$

where $M = \prod_{i=1}^{3} m_i$ and

$$\begin{aligned} A &= (-2^{2n}+1)\langle x_1\rangle_{2^n}, &(12)\\ B &= (2^{2n}+2^n)\langle 2^{n-1}x_2\rangle_{2^n-1}, &(13)\\ C &= (2^{2n}-2^n)\langle\langle 2^{n-1}x_3\rangle_{2^n+1} + x_3\rangle_{2^n+1}. &(14)\end{aligned}$$

where X_{SD} means that X is in the SD number representation. Then the conversion result is $X_{SD} = k_3 2^{2n} + k_2 2^n + k_1$. Let $TA = \langle x_1\rangle_{2^n} = x_1$, $TB = \langle 2^{n-1}x_2\rangle_{2^n-1}$, and $TC = \langle\langle 2^{n-1}x_3\rangle_{2^n+1} + x_3\rangle_{2^n+1}$. X can be written as

$$\begin{aligned} X_{SD} &= \Big\langle(-2^{2n}+1)TA + (2^{2n}+2^n)TB \\ &\quad +(2^{2n}-2^n)TC\Big\rangle_{2^n(2^n+1)(2^n-1)} \\ &= \Big\langle -2^n(2^n)(TA) + 2^n(2^n+1)TB \\ &\quad +2^n(2^n-1)TC\Big\rangle_{2^n(2^n+1)(2^n-1)} + TA \\ &= 2^n\Big\langle\langle(-2^nTA) + (2^n+1)TB\rangle_{2^{2n}-1} \\ &\quad +(2^n-1)TC\Big\rangle_{2^{2n}-1} + TA \\ &= 2^n\Big\langle\langle E+F\rangle_{2^{2n}-1} + GA\Big\rangle_{2^{2n}-1} + TA \end{aligned} \tag{15}$$

where $E = -2^nTA$, $F = (2^n+1)TB$ and $GA = (2^n-1)TC$. Let $GB = \langle E+F\rangle_{2^{2n}-1}$. It is clear that $k_1 = TA = x_1$ since $0 \le x_1 < 2^n$. The calculation of X_{SD} can be considered as how to compute the magnitude of 2^{2n} and 2^n, k_3 and k_2.

Conversion algorithm A
Input: x_1, x_2, x_3 (binary numbers)
Output: k_1, k_2, k_3 (SD numbers)

(1) Procedure for k_1

$k_1 = x_1$;

(2) Procedure for k_2 and k_3

(2A) $TA = x_1$;

```
      $TB = \langle 2^{n-1}x_2 \rangle_{2^n-1}$;
      $TC1 = \langle 2^{n-1}x_3 \rangle_{2^n+1}$;
(2B)  $E = 2^n(-TA)$;
      $F = (2^n+1)TB$
(2C)  $TC = \langle TC1 + x_3 \rangle_{2^n+1}$;
      $GB = \langle E + F \rangle_{2^{2n}-1}$;
(2D)  $GA = (2^n-1)TC$;
(2E)  $k_3 2^n + k_2 = \langle GB + GA \rangle_{2^{2n}-1}$;
```

□

In (2A), TB and $TC1$ are evaluated by performing the end-around-carry shift of $(n-1)$ digit positions to left. $-TA$ is directly derived by changing the signs of nonzero digits. The word length of E, F and GA are twice of n. In (2C), one $2n$-digit and one n-digit SD additions are performed in parallel. In (2E), $k_3 2^n + k_2$ is evaluated by one $2n$-digit MSDA. Therefore, the above algorithm needs two $2n$-digit MSDAs, one n-digit MSDA. Note that the conversion is based on SD number system and the legitimate range of X_{SD} is $[-(M-1),(M-1)]$ by the definition of the SD number for RNS.

3.2 Converting X_{SD} into binary number representation X_B

Then we convert X_{SD} into l_m for one-to-one correspondence. Because $k_1 = x_1 \geq 0$, to convert X_{SD} into l_m can be consider as to convert $k_3 2^{2n} + k_2 2^n$ into l_m. We use the property of mod 2^n-1 arithmetic, and this leads to that we can carry out the conversion through a $2^{2n}-1$ parallel prefix adder.

Definition 2 *Let H and J be two integers in the binary number representation. If $H + J = 2^n - 1$, then J is 1's complement of H.*

Property 1 *Let $R = (r_{n-1}, \cdots, r_0)$ and $r_i \in \{0,-1\}$. Let $T = (t_{n-1}, \cdots, t_0)$. If $R+(2^n-1) = T$, then $t_i \in \{0,1\}$.*

From Property 1, we know $T \geq 0$. It is enough to use the binary number representation to represent T. We use $Q = (q_{n-1}, \cdots, q_0)$ to represent R by the following steps: (1)If $r_i = -1$ then $q_i = 1$. (2)If $r_i = 0$ then $q_i = 0$. Since $J = (2^n-1) + (-H)$, then we can calculate T by using 1's complement representation. For example; when $R = (0,-1,0,-1)$, then $Q = (0,1,0,1)$. By Definition 2, T is 1's complement of Q, and then $T = (1,0,1,0)$. We know $R + (2^4-1) = T$.

Let $D = k_3 2^n + k_2$ and D^+ and D^- be $2n$-digit SD numbers for the positive digits and the negative digits, respectively. For example: if $D = (-1,-1,-1,0,1,0)$, then $D^+ = (0,0,0,0,1,0)$ and $D^- = (-1,-1,-1,0,0,0)$. Thus $k_3 2^n + k_2$ can be written as

$$\begin{aligned} D &= |D^+ + D^-|_{2^{2n}-1} \\ &= |D^+ + D^- + (2^{2n}-1)|_{2^{2n}-1} \\ &= |D^+ + DA|_{2^{2n}-1} \end{aligned} \quad (16)$$

where $DA = D^- + (2^{2n}-1) \geq 0$ and $D^+ \geq 0$. Therefore, $D \geq 0$. We can use 1's complement representation to calculate DA. Since $D^+ \geq 0$ and $DA \geq 0$, it is enough to use the binary number representation to represet D^+ and DA. Then, we use the binary number representation, DP and DN, to express D^+ and DA.

Conversion algorithm B
Input: k_1, k_2, k_3 (SD numbers)
Output: b_1, b_2, b_3 (binary numbers)
Let $X_B = b_3 2^{2n} + b_2 2^n + b_1$,
$D = (d_{2n-1}, \cdots, d_0)$,
$DP = (dp_{2n-1}, \cdots, dp_0)$ and
$DN = (dn_{2n-1}, \cdots, dn_0)$.

```
Procedure for $b_1$

    $b_1 = k_1$;

Procedure for converting $D = k_3 2^n + k_2$ into $l_m$

    (1) for $i = 0$ to $(2n-1)$;
          If $d_i = 0$, then $dp_i = 0$ and $dn_i = 1$;
          else if $d_i = 1$ then $dp_i = 1$ and $dn_i = 1$;
          else $dp_i = 0$ and $dn_i = 0$;

    (2) $b_3 2^n + b_2 = |DP + DN|_{2^{2n}-1}$;
```

□

Consider how to calculate $|DP + DN|_{2^{2n}-1}$. A good idea to perform modulo $2^n - 1$ adder has been proposed [13] by using Kogge-Stone tree structure which uses the associative operator $'o'$ defined in Brent et al. [12] to implement the carry computation and the method is referred to as parallel prefix adder. Brent et al. [12] defined G_i and P_i as "block carry generate" and "block carry propagate", respectively. They have shown that it suffices to compute all the $G_i s, P_i s$ for computing carry propagation. To calculate $|DP+DN|_{2^{2n}-1}$, the above algorithm needs one modulo $(2^{2n}-1)$ parallel prefix adder.

4 Hardware Realization and Performance Evaluation

4.1 Hardware Realization

In hardware implementation, an SD digit x is encoded as a 2-bit binary code defined as $x = [x^s, x^a]$, where x^s is the sign and x^a is the absolute value. This demands more hardware resources, but it also reallocates space, which effects the overall speed. We use a hardware description language, VHDL, to design the residue arithmetic circuits for the implementation of the proposed converters. Then, we performed a simulation under the conditions of 1-μm CMOS gate array technology.

The proposed converter based on conversion algorithms A and B, whose main blocks consist of three MSDAs and one modulo $(2^{2n} - 1)$ prefix adder, is shown in Fig.3. Block E is used to get $2^n(-x_1)$. Block F is used to get $(2^n + 1)TB$ where $TB = \langle 2^{n-1}x_2\rangle_{2^n-1}$. As mentioned before, the $TC1$ and TB are designed as the shift-left operation. GA combines 2^nTC and $(-TC)$. The division block is used to convert L_m to l_m, which divides itself to two positive numbers including '0'. The modulo $(2^{2n} - 1)$ prefix adder[13] is used to sum the two positive numbers, such that $b_32^n + b_2$ is $[0, 2^{2n} - 2]$.

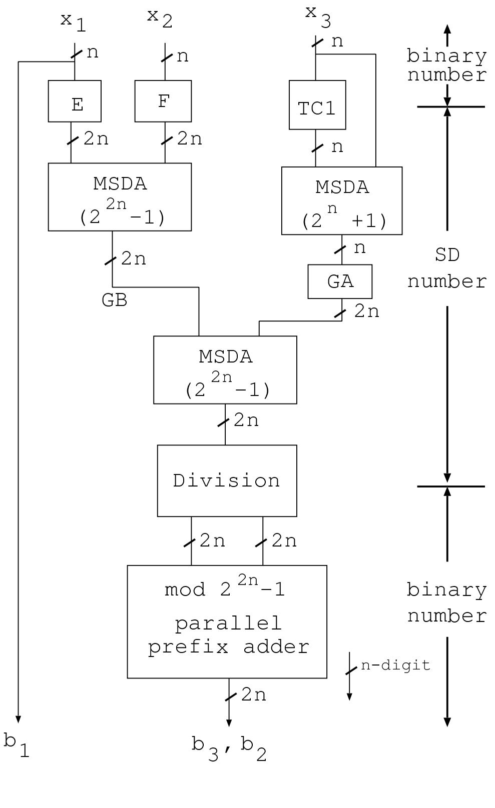

Figure 3: Architecture of the proposed converter

4.2 Comparison between the Proposed Method and Other Methods

The Andraos-Ahmad algorithm [5], which introduces compact forms of multiplicative inverses to simplify CRT, is an efficient technique. The algorithm uses four adders, two of which operate in parallel, to convert the moduli $(2^n + 1, 2^n, 2^n - 1)$ residue number into their binary equivalent. Recently, Piestrak [6] has suggested a simplification of the Andraos-Ahmad technique: the value of $-r_1$ modulo 2^{2n-1} can be easily obtained by the manipulation of r_1. Piestrak proposed two methods. The first method, referred to as the cost-effective (CE) version, uses two $2n$-bit carry save adders (CSAs) and one $2n$ carry propagation adder (CPA) with an end-around-carry to calculate $A + B + C - r_1$ of Andraos et al.[5]. The second method, which is referred to as the high-speed (HS) version, uses two $2n$-bit CSAs and two parallel $2n$-bit CPAs followed by a multiplexer. The CE converter needs $(4n + 1)$ full adder (FA) and the delay time is $(2t_{FA} + 2t_{CPA(2n)})$. The HS converter needs $(6n + 1)FA$ and $2n$-bit multiplexer (MUX), and the delay time is $(2t_{FA} + t_{CPA(2n)} + t_{MUX})$.

As mentioned before, mod $2^n + 1$ requires $(n + 1)$ bits to represent $2^n + 1$ states. Hiasat et al.[8] proposed using mod $2^{n-1} - 1$ to instead of mod $2^n + 1$. The main advantage is that arithmetic mod $2^{n-1} - 1$ is more efficient than that mod $2^n + 1$, especially in the end-around-carry is positive number. Wang et al. [9] improved the method of Hiasat et al. [8], and saved one stage of modulo subtraction for computing Y in Hiasat et al. [8]. The converter [9] needs $(7n - 3)$FA and (3n-7) half adder (HA), and the delay time is $(3n + 2)t_{FA}$.

Now we evaluate our proposed converters. Our aim is to enable **high-speed conversion**, so drawbacks in terms of area are not problems. The main process in Fig.2 is on MSDA and modulo $2^{2n} - 1$ parallel prefix adder. The area of MSDA is the proportion of n and the delay is independent of n [11, 10]. The area of modulo $2^{2n} - 1$ parallel prefix adder [13] is $6n \log 2n + 8n$ additions and the delay is $O(2 \log 2n + 3)$. Therefore, our conversion (Fig.3) is faster than that by Piestrak [6] when n is large.

The area comparison and delay time comparison are shown in Table 2. In Table 2, because our converters are based on the SD number representation, we use two bits to express one-digit SD number. This leads to that we need more hardware than the binary converters. Compared to 16-digit Piestrak's high-speed converters [6], our method is fast and the computation time is shorten by 34%. The proposed converter based on parallel adders (MSDA, parallel prefix adder) is very fast when $n \geq 16$.

Table 2: Performance of the CRT number converters

n	CE[6]		HS[6]		Wang[9]		Fig.3	
	area (gate)	delay (ns)	area (gate)	delay (ns)	area (gate)	delay (ns)	area (gate)	delay (ns)
4	240	20.06	345	15.25	222	24.44	399	21.22
8	480	32.38	689	22.56	468	40.52	849	22.92
16	960	57.02	1377	37.19	1011	94.24	1707	24.6

The moduli $(2^n-1, 2^n, 2^{n-1}-1)$ have more beneficial than the three-moduli $(2^n-1, 2^n, 2^n+1)$ in RNS, because the end-around-carry adder of modulo $2^{n-1}-1$ operation is very simply. However, in the RNS-to-Binary conversion, the CRT based on the moduli $(2^n-1, 2^n, 2^{n-1}-1)$ is more complex than that based on moduli $(2^n-1, 2^n, 2^n+1)$. In fact, the Table 2 shows the computation time of [9] is longer than others.

5 Conclusion

The CRT algorithm based on MSDA and parallel prefix adder has been proposed. The proposed converter has been demonstrated throughout the paper using examples and analysis. The simulations show that the proposed schemes are high-speed architectures. Therefore, the conversion scheme will have less computation time than binary converters.

References:

[1] N.S. Szabo and R.I. Tanaka , Residue Arithmetic and Its Applications to Computer Technology, New York: McGraw-Hill, 1967.

[2] V. Paliouras and T. Stouraitis, Novel high-radix residue number system architectures, *IEEE Trans.on circuits and systems II.,* Vol.47, No.10, Oct. 2000, pp.1059-1073.

[3] A. Skavantzos and P.B. Rao, New multipliers modulo 2^n-1, *IEEE Trans. Comput.,* Vol.41, No.8, Aug. 1992, pp.957-961.

[4] A. Hiasat, New memoryless mod$(2^n\pm1)$ residue multiplier, *Electronics Letters,* Vol.28, No.3, Jan. 1992, pp.314-315.

[5] S. Andraos and H. Ahmad, A new efficient memoryless residue to binary converter, *IEEE Trans. Circuits Syst.,* Vol. CAS-35, Nov. 1988, pp.1441-1444.

[6] S.J. Piestrak, A high-speed realization of a residue to binary number system converter, *IEEE Trans. Circuits Syst.II,* Vol.42, No.10, Oct. 1995, pp.661-663.

[7] Z. Wang, G.A. Jullien and W.C. Miller, An improved residue-to-binary converter, *IEEE Trans. Circuits Syst.I,* Vol.46, No.9, Sept. 2000, pp.1437-1440.

[8] A.A. Hiasat, and H.S. A.-A.-Z., Residue-to-binary arithmetic converter for the moduli set $(2^k, 2^k-1, 2^{k-1}-1)$, *IEEE Trans. Circuits Syst.II,* Vol.45, No.2, Feb. 1998, pp.204-209.

[9] W. Wang, M.N.S Swamy, M.O. 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 Syst.II,* Vol.47, No.12, Dec. 2000, pp.1576-1581.

[10] S. Wei and K. Shimizu, A novel residue arithmetic hardware algorithm using a signed-digit number representation, *IEICE Trans.Inf. & Syst.,* Vol.E83-D, No.12, Dec. 2000, pp.2056-2064.

[11] S. Wei and K. Shimizu, Compact residue arithmetic multiplier based on the radix-4 signed-digit multiple-valued arithmetic circuits, *IEICE Trans. Electron.,* Vol.E82-C,No.9, Sep. 1999, pp.1647-1645.

[12] R.P. Brent and H.T. Kung, A regular layout for parallel adders, *IEEE Trans. on Computers,* Vol.31, No.3, March 1982, pp.260-264.

[13] L. Kalampoukas, D. Nikolos, C. Efstathiou, H.T. Vergos and J. Kalamatianos, High-speed parallel-prefix modulo 2^n-1 adders, *IEEE Trans. on comp.,* Vol.49, No.7, July 2000, pp.673-680.