Modular Multipliers Using a Modified Residue Addition Algorithm with Signed-Digit Number Representation

Shugang Wei*

Abstract— **In this paper, we present multipliers using a modified addition algorithm modulo m with a signed-digit(SD) number representation where $m = 2^n - 1, 2^n$ or $2^n + 1$. To simplify an SD modular adder, new addition rules are proposed for generating the intermediate sum and carry with a binary number representation. By using the new codes for intermediate sum and carry and the end-around carry architecture, the proposed modulo m addition requires less hardware and short delay time for the residue addition than previous methods. A modulo m multiplier can be implemented by a binary modulo m adder tree. Compared to previous work, the circuit area and delay time of the multiplier are improved by 21% and 30%, respectively.**

Keywords: Residue number system, Signed-Digit Number, Residue arithmetic, modular addition, modular multiplier

1 Introduction

A residue number system(RNS) features ith residue digit of sum, difference and product is exclusively dependent on the ith digits of the operands [1],[2]. In general, to compute a remainder, it is usually to use read-only memories to do residue arithmetic [3]. However, to store all residue arithmetic tables, many read-only memories are required. Moduli $2^n, 2^n - 1, 2^n + 1$ have been widely used to simplify the residue arithmetic without memory. When a binary number system is used to perform the residue arithmetic, the carry propagation will arise inside the residue digits and the speed of arithmetic operation will be limited. For example, the residue addition time is proportional to $\log(n)$ even for the improved adder architectures[4].

Signed-digit(SD) number system [5] offers a carry-free addition. We have presented a novel residue arithmetic hardware algorithm using the SD number representation to implement the residue addition in a constant time and the residue multiplication using a residue SD adder tree architecture for the symmetric RNS [6]. High speed conversion algorithms between weighted and residue numbers also have been proposed by using the presented residue SD additions [7]. For moduli $2^n, 2^n-1, 2^n+1$, the modular addition can be performed with an end-around-carry SD adder[6]. However, in the residue SD adder, a 2-bit code was used to represent an SD number and this leads to that the circuits is very larger than the binary architecture. To simplify an SD adder, in this paper, a binary number representation is used to express the intermediate sum and carry of SD addition, such that the performance of residue SD adder can be improved. We propose a modified modulo m signed-digit addition algorithm which is designed by generating the residue intermediate sum and carry with the binary number representation. By using the modified modulo m SD addition algorithm, a high speed multiplier with a residue adder tree can be implemented. Compared with our previous work [6], the evaluation results of hardware performance show that the area and delay time of the multipliers are improved by 21% and 30%, respectively.

*Supported by Grant-in-Aid Research(C)(19500039) from Japan Society for the Promotion of Science(JSPS). Manuscript submitted November 27 2008. The author is with the faculty of Engineering, Gunma University, Japan 376-8515. Tel/Fax:81-277-30-1824/1826 Email:wei@ja4.cs.gunma-u.ac.jp

2 Preliminaries

A residue number with respect to an odd modulus m is represented by the symmetric set:

$$l_m = \{-(m-1)/2, \cdots, 0, \cdots, (m-1)/2\}. \quad (1)$$

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

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

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 the SD number representation, we enlarge l_m to the following redundant residue number set:

$$\begin{aligned} L_m &= \{-(2^n-1), \cdots, -(m-1)/2, \cdots 0, \\ &\quad \cdots, (m-1), \cdots, (2^n-1)\}. \end{aligned} \quad (3)$$

Thus, x must be in L_m when it is expressed in an n-digit SD number representation. Obviously,

$$-x = -(x_{n-1}, x_{n-2}, \cdots, x_0)_{SD}$$

Table 1: Rules for adding SD numbers

	$abs(x_i) = abs(y_i)$	$abs(x_i) \neq abs(y_i)$	
		$xpy_i \times xpy_{i-1} < 0$ or $(xpy_{i-1} = 0)$ and $(xpy_i < 0)$	$xpy_i \times xpy_{i-1} > 0$ or $((xpy_{i-1} = 0)$ and $(xpy_i > 0))$
w_i	0	xpy_i	$-xpy_i$
c_i	$xpy_i/2$	0	xpy_i

Note: $xpy_i = x_i + y_i$.

Table 2: New rules for adding SD numbers

	$abs(x_i) = abs(y_i)$	$abs(x_i) \neq abs(y_i)$	
		$xpy_i \times xpy_{i-1} < 0$ or $(xpy_{i-1} = 0)$ and $(xpy_i < 0)$	$xpy_i \times xpy_{i-1} > 0$ or $((xpy_{i-1} = 0)$ and $(xpy_i > 0))$
u_i	t_{i-1}	$t_{i-1} - xpy_i$	$t_{i-1} + xpy_i$
v_i	$t_i + xpy_i/2$	t_i	$t_i + xpy_i$

Note: $xpy_i = x_i + y_i$.

$$= \quad (-x_{n-1}, -x_{n-2}, \cdots, -x_0)_{SD}$$

is also in L_m.

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

$$x = \langle Y \rangle_m = |X|_m, \tag{4}$$

and

$$x = \langle X \rangle_m = |X|_m - sign(|X|_m) \times m, \tag{5}$$

where

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

When $|X|_m = 0$ and $m = 2^n - 1$, there are three possible values for x, that is, $-m$, 0 and m.

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.

3 Modulo m Signed-Digit Addition

3.1 Previous Study

Consider addition $x + y$, where x, y are the SD numbers in the n-digit SD representation shown in (2), can be performed as follows: Let w_i and c_i be the intermediate sum and the carry of ith digit position, respectively. The values of them are determined by Table 1 with respect to the values of $x_i, y_i, x_{i-1}, y_{i-1}$. In Table 1, $abs(x_i)$ is the absolute value of x_i and $xpy_i = x_i + y_i$. Thus the addition at each digit can be implemented by the following two steps:

ADD1: Determine c_i and w_i using Table 1, and

$$2 \times c_i + w_i = x_i + y_i. \tag{6}$$

ADD2:

$$s_i = w_i + c_{i-1}, \tag{7}$$

where $c_{-1} = 0$ and $x_{-1} = y_{-1} = 0$. Then

$$S = x + y \quad = \quad (c_{n-1}, s_{n-1}, s_{n-2}, \cdots, s_0). \tag{8}$$

Let μ be a residue parameter and defined as

$$\mu = m - 2^n. \tag{9}$$

In this cases of $\mu \in \{0, 1, -1\}$, an end-around-carry SD adder is constructed for the modulo m signed-digit addition:

$$c_{-1} = \langle c_{n-1} 2^n \rangle_m = -c_{n-1} \times \mu \tag{10}$$

and

$$x_{-1} \quad = \quad -\mu \times x_{n-1}, \tag{11}$$

$$y_{-1} \quad = \quad -\mu \times y_{n-1}. \tag{12}$$

Using the above values of c_{-1},x_{-1} and $= y_{-1}$ instead of $c_{-1} = 0$ and $x_{-1} = y_{-1} = 0$, we have

$$s = \langle x + y \rangle_m \quad = \quad (s_{n-1}, s_{n-2}, \cdots, s_0). \tag{13}$$

Figure 1 illustrates a circuit diagram of an n-digit Modulo m Signed-Digit adder (MSDA) with n SD full adders (SDFAs). One SDFA consists of one ADD1 and one ADD2. ADD1 generates the intermediate sum and the intermediate carry, and ADD2 sums the low intermediate carry and the intermediate sum. We use $\otimes$ to denote an 1-by-1 multiplier. The MSDA can be performed in parallel without the carry propagation.

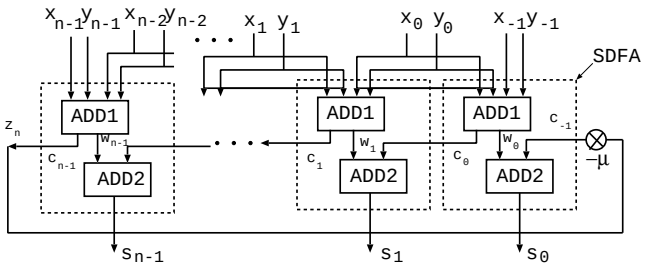


Figure 1: Modulo m Signed-Digit adder

3.2 Improved Signed-Digit Adder

For the circuit design, an SD digit $d \in \{-1, 0, 1\}$ is encoded as a 2-bit binary code $d = [d(s), d(a)]$, where $d(s)$ is the sign and $d(a)$ is the absolute value. Thus, the area of the addition circuit is larger than the binary number architecture. To modify the area cost, a method using the binary number representation for the intermediate carry and sum for the SD addition can be considered.

We use $t_i \in \{0, 1\}$ to express the sign of $x_i + y_i$ as

$$t_i = \begin{cases} 1 & x_i + y_i < 0 \\ 0 & x_i + y_i \geq 0 \end{cases}.$$

Then, we modify Table 1 by

$$u_i = t_{i-1} - w_i \quad (14)$$
$$v_i = t_i + c_i \quad (15)$$

and new rules for the SD addition are shown in Table 2. We can show that $u_i \in \{0, 1\}$ and $v_i \in \{0, 1\}$ as follows: In the case of $abs(x_i) = abs(y_i)$, $u_i = t_{i-1}$ and $v_i = t_i + (x_i + y_i)/2 \in \{0, 1\}$, because $t_i = 1$ while $x_i + y_i = -2$ and $t_i = 0$ while $x_i + y_i = 2$. When $abs(x_i) \neq abs(y_i)$ and $(x_i + y_i) \times (x_{i-1} + y_{i-1}) < 0$, we have $u_i = t_{i-1} - (x_i + y_i) \in \{0, 1\}$ from the facts that $t_{i-1} = 1$ while $x_i + y_i = 1$ and $t_{i-1} = 0$ while $x_i + y_i = -1$. When $abs(x_i) \neq abs(y_i)$ and $(x_i + y_i) \times (x_{i-1} + y_{i-1}) > 0$, we also have $u_i = t_{i-1} + (x_i + y_i) \in \{0, 1\}$ from the facts that $t_{i-1} = 1$ while $x_i + y_i = -1$ and $t_{i-1} = 0$ while $x_i + y_i = 0$. When $x_{i-1} + y_{i-1} = 0$ and $x_i + y_i = -1$, $u_i = t_{i-1} - (x_i + y_i) = 1$, and when $x_{i-1} + y_{i-1} = 0$ and $x_i + y_i = 1$, $u_i = t_{i-1} + (x_i + y_i) = 0$. Thus, it is always true that $u_i \in \{0, 1\}$. For $v_i = t_i + c_i$, when $(x_i + y_i)/2 = 1$ or $x_i + y_i = 1$, $t_i = 0$ and $v_i = 1$. When $(x_i + y_i)/2 = -1$ or $x_i + y_i = -1$, $t_i = 1$ and $v_i \in \{0, 1\}$. When $x_i + y_i = 0$, $t_i = 0$ and $v_i = 0$. Thus, $u_i \in \{0, 1\}$.

Therefore, we can modify the signed-digit addition and perform the following two steps at each SD digit position:

ADD1*: Calculate u_i and v_i by Table 2 meeting

$$u_i = t_{i-1} - w_i \quad (16)$$
$$v_{i-1} = t_{i-1} + c_{i-1}. \quad (17)$$

		x_{n-1}	x_{n-2}	$\cdots$	x_1	x_0
	+	y_{n-1}	y_{n-2}	$\cdots$	y_1	y_0
U:	t_{n-1}	t_{n-2}-w_{n-1}	t_{n-3}-w_{n-2}	$\cdots$	t_0-w_1	t_{-1}-w_0
V:	t_{n-1}+c_{n-1}	t_{n-2}+c_{n-2}	t_{n-3}+c_{n-3}	$\cdots$	t_0+c_0	t_{-1}
V-U:	c_{n-1}	c_{n-2}+w_{n-1}	c_{n-3}+w_{n-2}	$\cdots$	c_0+w_1	w_0

Figure 2: Improved signed-digit addition

ADD2*: Add v_{i-1} to $-u_i$ that meets

$$s_i = c_{i-1} + w_i = v_{i-1} - u_i, \quad (18)$$

Then we have

$$\begin{aligned} S &= V - U \\ &= (v_{n-1}, v_{n-2}, \cdots, v_0, v_{-1}) \\ &\quad -(u_n, u_{n-1}, \cdots, u_1, u_0) \\ &= (t_{n-1} + c_{n-1}, t_{n-2} + c_{n-2}, \cdots, t_0 + c_0, t_{-1}) \\ &\quad -(t_{n-1}, t_{n-2} - w_{n-1}, \cdots, t_0 - w_1, t_{-1} - w_0) \\ &= (c_{n-1}, c_{n-2} + w_{n-1}, \cdots, c_0 + w_1, w_0) \\ &= (s_n, s_{n-1}, s_{n-2}, \cdots, s_0) \\ &= x + y, \end{aligned} \quad (19)$$

where $v_{-1} = t_{-1} = 0$ and the calculating diagram is shown in Fig. 2. Note that a new carry is not generated in ADD2* similar as ADD2, and the values of u_i and v_i are decided directly from the values of x_i, $+y_i$,x_{i-1} and y_{i-1}.

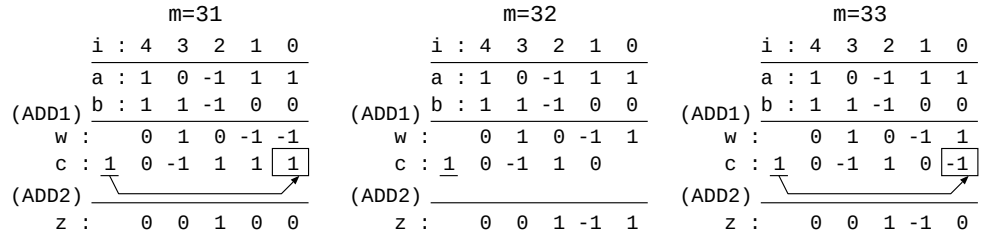
In Table 1, w_i and c_i are in the SD number representation such that we have to use 2-bit binary codes to represent them. However, in Table 2, we use the binary number representation to represent u_i and v_i, so that the performance of the circuits implementing ADD1* and ADD2* can be improved.

3.3 New Modulo m Signed-Digit Adder

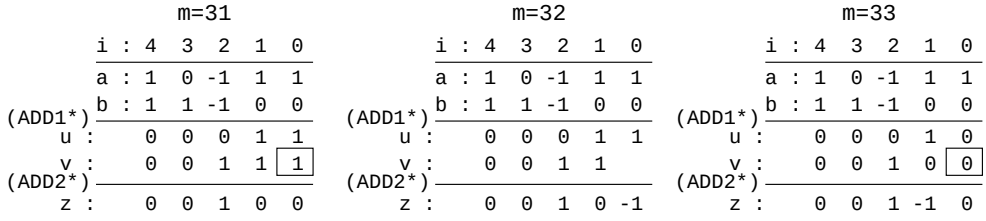
We now consider how to deal with the residue operation modulo m, where $m = 2^n + \mu$ and $\mu \in \{-1, 0, 1\}$. By using the relationships of $x_{-1} = -\mu \times x_{n-1}$, $y_{-1} = -\mu \times y_{n-1}$ and t_{n-1}, the value of t_{-1} is obtained from the sign of $-\mu \times (x_{-1} + y_{-1})$. Then, the modulo m signed-digit addition can be expressed as follows.

$$\begin{aligned} \langle S \rangle_m &= \langle X + Y \rangle_m = \langle \langle V \rangle_m - \langle U \rangle_m \rangle_m \\ &= (v_{n-2} - u_{n-1}, \cdots, v_0 - u_1, v_{-1} - u_0). \end{aligned} \quad (20)$$

In equation (20), the digits of U and V including v_0 and u_{-1} at 0-th digit position can be determinated by the rules of Table 2. However, u_0 and v_{-1} on the end-around-carry path are generated by the following steps



(a) Addition using prevous algorithm



(b) Addition using new algorithm

Figure 3: Example of Modulo m signed-digit addition. algorithm.

ADD1**: Determine u_0 and v_{-1} by Table 2, meeting

$$u_0 = t_{-1} - w_0 \quad (21)$$
$$v_{-1} = t_{-1} - \mu \times c_{n-1}, \quad (22)$$

where

$$\mathrm{t}_{-1} = \begin{cases} 1 & -\mu(x_{n-1} + y_{n-1}) < 0 \\ 0 & -\mu(x_{n-1} + y_{n-1}) \geq 0 \end{cases}.$$

Because we use the binary number representation instead of the signed-digit number representation for the intermediate sums and carries, the new modulo m signed-digit addition can achieve high speed and small circuits.

Example 1 : Let $n = 5$, $a = (1, 0, -1, 1, 1)_{SD}$ and $b = (1, 1, -1, 0, 0)_{SD}$. Thus, $a = 15$ and $b = 20$, and in the case of $\mu = 1$ $m = 2^5 + 1 = 33$. We have $t_0 = 0, t_1 = 0, t_2 = 1, t_3 = 0, t_4 = 0$ and $t_{-1} = 1$. By the rules of Table 2, u_i and v_i are determined. Fig.3(b) illustrates the calculation of $\langle x + y \rangle_m$ using the new algorithm for $m = 31, 32$ and $m = 33$. The results are $\langle 15 + 20 \rangle_{31} = 4$, $\langle 15 + 20 \rangle_{32} = 3$ and $\langle 15 + 20 \rangle_{33} = 2$. In Fig.3(a), the residue addition process is also shown by the previous algorithm.

4 Modulo m SD Multipliers

To calculate $\langle x \times y \rangle_m$, where x and y are integers in the n-digit radix-2 SD number representation, $x \times y$ is expanded as follows:

$$\begin{aligned} x \times y &= (x_{n-1}2^{n-1} + x_{n-2}2^{n-2} + \cdots + x_0) \\ &\quad \times (y_{n-1}2^{n-1} + y_{n-2}2^{n-2} + \cdots + y_0) \\ &= \sum_{i=0}^{n-1} y_i 2^i \times (x_{n-1}2^{n-1} + x_{n-2}2^{n-2} \\ &\quad + \cdots + x_0). \end{aligned}$$

We have

$$\begin{aligned} \langle x \times y \rangle_m &= \langle \sum_{i=0}^{n-1} \langle y_i 2^i \times (x_{n-1}2^{n-1} + x_{n-2}2^{n-2} \\ &\quad + \cdots + x_0) \rangle_m \rangle_m \\ &= \langle \sum_{i=0}^{n-1} pp_i \rangle_m, \end{aligned} \quad (23)$$

where

$$pp_i = \langle y_i 2^i \times (x_{n-1}2^{n-1} + x_{n-2}2^{n-2} + \cdots + x_0) \rangle_m \quad (24)$$

denotes as a partial product. Since $y_i \in \{-1, 0, 1\}$,

$$\begin{aligned} pp_i &= y_i \langle 2^i \times (x_{n-1}2^{n-1} + x_{n-2}2^{n-2} + \cdots + x_0) \rangle_m \\ &= y_i \times sx_i, \end{aligned} \quad (25)$$

where

$$sx_i = \langle 2^i (x_{n-1}2^{n-1} + x_{n-2}2^{n-2} + \cdots + x_0) \rangle_m. \quad (26)$$

Therefore, a modulo m multiplication can be implemented by calculating Eqs.(26).(25) and (23) to obtain partial products and the sum of the partial products.

In the cases of $\mu = 0$ and $\mu = \pm 1$,

$$\begin{aligned} sx_i &= (x_{n-i-1}, x_{n-i-2}, \cdots, x_0, \\ &\quad -\mu x_{n-1}, \cdots, -\mu x_{n-i+1}, -\mu x_{n-i})_{SD}. \end{aligned}$$

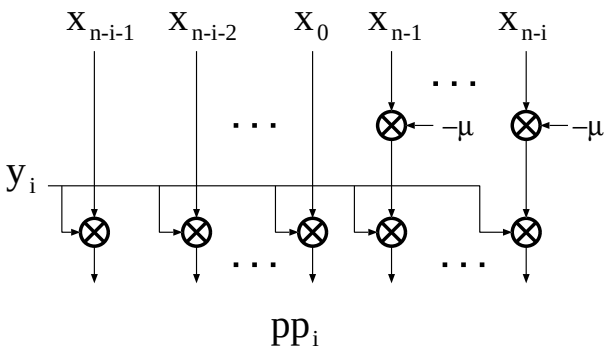


Figure 4: A partial product generating unit

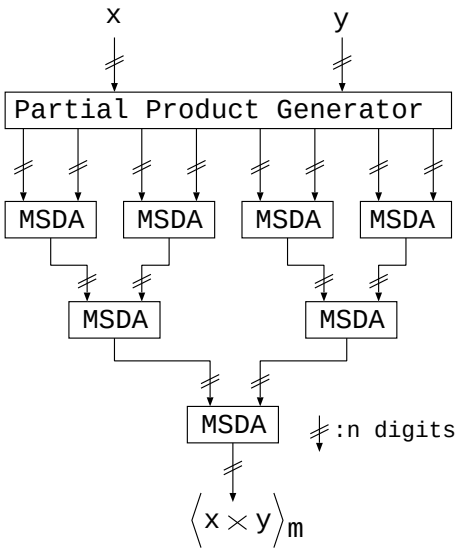


Figure 5: Modulo m SD multiplier with an MSDA tree.

Thus, a partial product is simply obtained by an i-digit end-around-shift and an n-by-1 digit multiplication. The i-digit end-around-shift by directly wiring correspond input and output signals can performed in a constant time.

A binary tree of the modulo m SD adders can be constructed for the modulo m sum of the partial products as shown in Fig.5. The modulo m circuit for the shifted number may be constructed by the AND-OR two-stage logical network which has a constant delay time. However, the circuit may be very complicated for a large n. the modulo m multiplications can be performed in a time proportional to $\log_2 n$.

5 Hardware Realization and Performance Evaluation

We use a hardware description language, VHDL, to design the residue arithmetic circuits, for the implementation of the proposed algorithm.

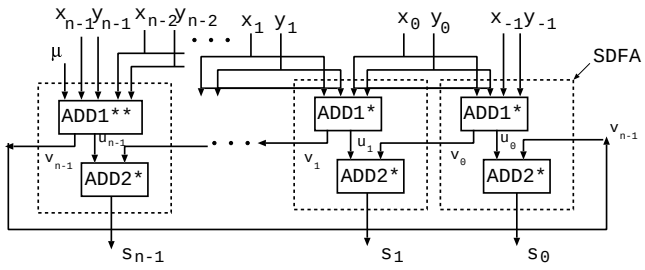


Figure 6: New Modulo m Signed-Digit adder

Table 3: Binary representation for a radix-two signed digit.

a_i	$a_i(1)$	$a_i(0)$
-1	1	1
0	0	0
1	0	1

We specify a binary representation for a radix-two signed-digit a_i as Table 3, where $a_i(1)$ is the sign and $a_i(0)$ is the absolute value of a_i. Thus, a p-digit radix-2 SD number a is represented by a vector with $2p$-bit length.

$$\begin{aligned} a &= (a_{n-1}, a_{n-2}, \cdots, a_0)_{SD} \\ &= [a_{n-1}(1)a_{n-1}(0) \;\; a_{n-2}(1)a_{n-2}(0) \\ &\qquad \cdots a_0(1)a_0(0)] \end{aligned} \tag{27}$$

For example, $(1,0,0,-1)_{SD} = [01000011]$. Figure 6 illustrates a circuit diagram of the new MSDA, where $\mu \in \{-1,0,1\}$. One SDFA consists of sub-circuits, ADD1* and ADD2*, and in the most significant position ADD1** is designed for the end-around-carry generation. u_i and v_i generated by ADD1* or ADD1** are in the binary number representation. $s_i = v_{i-1} - u_i$ is performed in ADD2* and $s_i \in \{-1,0,1\}$. The whole design has been verified by 1-μm CMOS gate level simulation. The delay times and gates of the design results are aummarized in Tables 4 and 5. Because the proposed MSDA is designed with binary number representation except input and output, we have small circuits and speed-up of computation time.

Compared to our previous work, the circuits are improved by 21% and the computation times are shorten by 30%. The new residue SD adder can be applied to a residue multiplier and the delay time and hardware cost of the multiplier can be also improved by about 30% and 20%, respectively. We also designed fast binary residue adders[8] for the performance comparison with the presented SD adders. The proposed addition is high-speed for large wordlength residue additions.

The delay time for a 16-bit binary modulo m multiplier is about 95ns, so that the proposed modulo m SD multiplier constructed with a binary tree of the improved modulo

Table 4: Performance of modulo $2^n + 1$ adders

	area (gates)			delay (ns)		
n	[6]	[8]	this study	[6]	[8]	this study
8	274	213	215	7.47	6.34	5.22
16	530	612	415	7.47	9.03	5.22
32	1,016	1,744	823	7.47	12.86	5.22

Table 5: Performance of modulo $2^n + 1$ multipliers

	area (gates)		delay (ns)	
n	[6]	this study	[6]	this study
8	2,052	1,653	31.88	21.63
16	4,781	3,562	39.03	27.85
32	9,853	7,164	46.64	33.03

m SD adders is very high speed. For the design of RNS chips, the modulo m SD adders and multipliers may be pre-designed with VHDL and are used as functional cells.

6 Conclusion

A new algorithm of modulo m SD adder has been presented. The binary number coding is used for achieving the high speed residue SD addition. A high speed residue multiplier can be constructed with a binary MSDA tree for the high speed calculations or only with one MSDA for a compact structure. Thus the modulo m multiplication is performed in a time proportional to $\log_2 p$ by using the binary adder tree.

High-speed computations can be performed based on the assumption that input and output data of the residue arithmetic circuits are in the residue SD number form, because some computing system applications, such as digital filtering, require repeated calculations of sums of products before the final results are obtained. For integration with conventional binary systems, efficient circuits are required to convert into and out of the residue SD systems. Our studies also focus on the evaluation of the presented residue arithmetic circuits, and the application to the computation systems, such as digital signal processing and digital control systems.

References

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

[2] Vassilis Paliouras and Thanos Stouraitis, "Novel High-Radix Residue Number System Architectures," *IEEE Tran.on circuits and systems II.*, vol.47,no.10, pp.1059-1073, Oct. 2000.

[3] G.A. Jullien, "Residue number scaling and other operations using ROM arrays," *IEEE Trans. Comput.*, vol.C-27, no.4, pp.325-336, April 1978.

[4] L.Kalampoukas, D.nikolos, C.Efstathiou, H.T.Vegos and J. Kakamatianos, "High-Speed Parallel-Prefix Modulo $2^2 - 1$ Adders," IEEE Trans. Comp., vol.49, no.7, pp.673-680, 2000.

[5] A.Avizienis, "Signed-digit number representations for fast parallel arithmetic," *IRE Trans. Elect. Comput.*, EC-10, pp.389-400, 1961.

[6] S.Wei and K.Shimizu," Residue Arithmetic Circuits Using a Signed-Digit Number Representation", Proceedings of the IEEE 2000 International Symposium on Circuits and Systems, Vol.-I, pp.24-27, 2000.

[7] S. Wei and K. Shimizu: A New RNS to Mixed-Radix Number Converter Using Modulo $(2^p - 1)$ Signed-Digit Arithmetic, Proceedings of 2004 IEEE Asia-Pacific Conference On Circuits and Systems, vol. 1, pp.377-380, 2004.

[8] C.Efstathiou, H.T. Vergos and D.Nikolos, " Modulo $2^n \pm 1$ Adder Design Using Select Prefix Blocks", IEEE Trans. on comput. vol.52, no.11, pp.1399-1406, 2003.