Regular Paper

New Booth Modulo m Multipliers with Signed-Digit Number Arithmetic

Shuangching Chen† and Shugang Wei†

New modulo m multipliers with a radix-two signed-digit (SD) number arithmetic is presented by using a modified Booth recoding method. To implement a modulo m multiplication, we usually generate modulo m partial products, then perform modulo m sum of them. In this paper, a new Booth recoding method is proposed to convert a radix-two SD number into a recoded SD (RSD) number in parallel. In the RSD number representation, there are no $(1, 1)$ and $(-1, -1)$ at any two-digit position. Thus, by using the RSD converted, the modulo m partial products can be cut from n into $n/2$ for an $n \times n$ modulo m multiplication. Parallel and serial modulo m multipliers have been designed by using the SD number arithmetic and the proposed Booth recoding. Compared to the former work, the area for VLSI implementation of the parallel modulo m multiplier is reduced to 80% from the original design, and the speed performance of the serial multiplier is improved up to twice by using the Booth recoding. The implementation method of the proposed Booth modulo m multipliers has been verified by a gate level simulation.

1. Introduction

A Residue Number System (RNS) features highly parallel carry-free addition and multiplication and borrow-free subtraction between residue digits [1),2)]. Various methods of applications of RNS in digital signal processing have been presented [3)]. To compute a remainder, it is usually to use read-only memories for residue arithmetic. Some modulo m multipliers based on the use of lookup tables were proposed [4)]. However, to store the residue arithmetic tables, many read-only memories (ROMs) are required and an exponential growth of ROM's size is concerned in the size of the modulus m. Several adders moduli $2^n - 1$ and $2^n + 1$ have been proposed by using the conventional binary number system without memory [5)], but the carry propagation arises and the modulo m addition time is proportional to $\log(n)$, even for the improved adder architectures [6)].

It is well known that a signed-digit (SD) number system [7),8)] takes advantages in an arithmetic circuit implementation without the carry propagation. 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 [9),10)]. For moduli 2^n, $2^n - 1$ and $2^n + 1$, the modulo m addition can be performed by an SD adder. When a modulus m satisfies $2^n + 1 < m \leq 2^n + 2^{n-1} - 1$, the modulo m addition is implemented with two SD adders: the first one is for the addition and the second one is for the residue operation. Thus, the delay time of the modulo m adders are independent of the word length of the operands. By using the modulo m SD adders, a parallel modulo m multiplier with a binary adder tree structure and a serial multiplier using one modulo m SD adder have been presented.

The performance of a modulo m multiplier is concerned in how many modulo m partial products to be added and the efficiency of the modulo m addition. Usually, a Booth algorithm is considered to generate fewer partial products for a binary multiplication [11)], and several modified Booth recoding methods have been proposed for high speed multiplications [12)~15)]. A modified Booth method for the modulo $2^n - 1$ binary multiplication has been proposed [17)]. However, these Booth recoding techniques can not be applied for the modulo m SD multiplication presented. A recoding method based on a two-stage radix-four SD arithmetic for radix-four modular multiplication has been proposed [18)]. The first stage is for generating an intermediate carry and an intermediate recoded digit with the radix-four representation. The second stage is for summing the intermediate recoded digit and the intermediate carry. In this paper, we present a new Booth recoding method which converts the multiplier into another SD representation by eliminating the strings of 1's and -1's. The proposed Booth recoding is faster than that proposed in Ref. 18), and results in a smaller

† Department of Computer Science, Gunma University

recoding circuit. The proposed method can reduce the modulo m partial products from n to $n/2$, such that the time needed for the modulo m sum of the products can be improved.

In the following section, we give the definition and several properties of a redundant modular representation for RNS, by which an efficient arithmetic algorithm with SD numbers can be constructed [9]. We also mention the modulo m addition algorithm based on the SD number arithmetic. In Section 3 a Booth recoding method is proposed, by which a radix-two SD number used as the multiplier of the modulo m multiplication is converted efficiently into a recoded SD (RSD) number representation without $(1,1)$ or $(-1,-1)$ in any two-digit position. Thus, the modulo m partial products can be cut from n into $n/2$ for an $n \times n$-digit modulo m SD multiplication. By using the Booth recoding, several architectures of the modulo m multipliers are presented, and the modulo m multipliers with the SD arithmetic are designed by using a hardware description language,VHDL. The design and simulation results show that it is possible to implement a modulo m multiplier which is high-speed, especially in comparison with the original one and a binary one.

2. Symmetric RNS and Residue Arithmetic with SD Number Representation

2.1 Symmetric Residue Number System

A symmetric residue number system (RNS) has normally a set of relatively prime odd-numbered moduli, $\{m_1, m_2, \cdots, m_k\}$, and the residue digit with respect to a modulus m_i is represented by the symmetric number set:

$$l_{m_i} = \{-(m_i-1)/2, \cdots, 0, \cdots, (m_i-1)/2\}. \tag{1}$$

An integer A in a value range $[-(M-1)/2, (M-1)/2]$ $(M = \prod_{i=0}^{k} m_i)$ is uniquely represented by a k-tuple $(A_1, A_2, \cdots, A_k)$, where

$$A_i = |A|_{m_i} = A - [A/m_i] \times m_i, \tag{2}$$

for $i = 1, 2, \cdots, k$. In the above equation, $[A/m_i]$ is a closet integer to A/m_i in the symmetric RNS, and each residue digit is defined to be the remainder of least magnitude when A is divided by m_i. When $m_i = 3$ and $l_{m_i} = \{-1, 0, 1\}$, for example, $|-7|_3 = -7 - [-7/3] \times 3 = -7 - (-2) \times 3 = -1$ and $|-8|_3 = -8 - [-8/3] \times 3 = -8 - (-3) \times 3 = 1$.

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

$$x = x_{n-1}2^{n-1} + x_{n-2}2^{n-2} + \cdots + x_0, \tag{3}$$

$x_i \in \{-1, 0, 1\}$, which can be denoted as $x = (x_{n-1}, x_{n-2}, \cdots, x_0)_{SD}$. In the SD number representation, x has a value in the range of $[-(2^n-1), 2^n-1]$. However, it is difficult to know if x is in l_m, because of the redundancy of the SD number representation (The subscript i is omitted.)

To simplify the manipulation of the modular operation in an SD number representation, we define that each residue digit has the following redundant residue number set:

$$L_m = \{-(2^n-1), \cdots, -(m-1)/2, \cdots, 0, \cdots, (m-1)/2, \cdots, 2^n-1\}, \tag{4}$$

Thus, x must be in L_m 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.

[Definition 1] Let X be an integer 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

$$x = \langle X \rangle_m = |X|_m, \tag{5}$$

and

$$x = \langle X \rangle_m = |X|_m - \mathrm{sign}(|X|_m) \times m, \tag{6}$$

respectively, where

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

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

The integer set l_m in Eq. (1) is a partial set of L_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. Thus, the addition, subtraction and multiplication of k-tuples $A = (A_1, A_2, \cdots, A_k)$ and $B = (B_1, B_2, \cdots, B_k)$ in an RNS can be represented as follows:

$$A \pm B = (\langle A_1 \pm B_1 \rangle_{m_1}, \cdots, \langle A_k \pm B_k \rangle_{m_k}), \tag{7}$$

$$A \times B = (\langle A_1 \times B_1 \rangle_{m_1}, \cdots, \langle A_k \times B_k \rangle_{m_k}). \tag{8}$$

Table 1 The 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	$x_i + y_i$

Obviously, the following properties exist in the redundant modular representation.

[Property 1]

Let a and b be integers. Then

(a) $\text{abs}(\langle a \rangle_m) \leq 2^n - 1$,

(b) $\langle a + b \rangle_m \equiv \langle \langle a \rangle_m + \langle b \rangle_m \rangle_m$,

(c) $\langle a \times b \rangle_m \equiv \langle \langle a \rangle_m \times \langle b \rangle_m \rangle_m$,

and

(d) $\langle -a \rangle_m \equiv -\langle a \rangle_m$,

where $\equiv$ indicates a binary congruent relation with modulo m. □

[Example 1] When $n = 4$ and $m_i = 17$,

$$l_{m_i} = \{-8, -7, -6, \cdots, 0, \cdots, 6, 7, 8\}$$

and

$$L_{m_i} = \{-15, -14, \cdots, 0, \cdots, 14, 15\}.$$

Thus, $|29|_{17} = -5$ in l_{m_i}. By the definition, $\langle 29 \rangle_{17} = -5$ or $\langle 29 \rangle_{17} = -5 - (-1) \times 17 = 12$. □

2.2 Signed-Digit Addition

An addition $Z = X + Y$, where X, Y are SD numbers in the n-digit SD representation shown in Eq. (3), can be performed as follows: Let c_i and w_i be the carry and the intermediate sum of ith digit position, respectively. The values of them are decided 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. Thus the addition at each digit can be implemented by the following two steps:

ADD1:

$$2 \times c_i + w_i = x_i + y_i \tag{9}$$

ADD2:

$$z_i = w_i + c_{i-1}, \tag{10}$$

where $c_{-1} = 0$. Then

$$\begin{aligned} Z &= X + Y \\ &= (c_{n-1}, z_{n-1}, z_{n-2}, \cdots, z_0) \\ &= (z_n, z_{n-1}, z_{n-2}, \cdots, z_0). \end{aligned} \tag{11}$$

In the above equations, $x_i, y_i, w_i, c_i, z_i \in \{-1, 0, 1\}$. **Figure 1** illustrates a circuit diagram of the n-digit SD adder (SDA) with n SD full adders (SDFAs), by which the addition can be performed in parallel without the carry propagation.

2.3 Modulo m Addition

Let μ be a residue parameter defined as

$$\mu = m - 2^n. \tag{12}$$

In the case of $\mu \in \{-1, 0, 1\}$, an end-around-carry SD adder is implemented for the modulo m SD addition (MSDA) by

$$c_{-1} = \langle c_{n-1} 2^n \rangle_m = -c_{n-1} \times \mu \tag{13}$$

and

$$x_{-1} = -\mu \times x_{n-1}, \tag{14}$$

$$y_{-1} = -\mu \times y_{n-1}. \tag{15}$$

In the case of $1 < \mu < 2^{n-1} - 1$, the MSDA can be implemented with two SDAs [9].

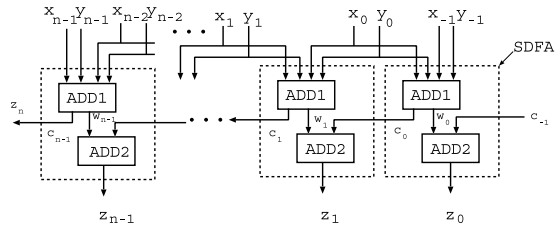

Fig. 1 Signed-Digit Adder (SDA).

3. Booth Modulo m Multipliers

3.1 Modulo m Multiplication with SD Number Representation

A modulo m multiplication, $X \times Y$, with the n-digit radix-two SD number representation can be expressed as

$$\langle X \times Y \rangle_m = \left\langle \sum_{i=0}^{n-1} \langle y_i 2^i \times X \rangle_m \right\rangle_m \tag{16}$$

$$= \left\langle \sum_{i=0}^{n-1} pp_i \right\rangle_m, \tag{17}$$

where $pp_i = \langle y_i 2^i \times (X) \rangle_m$ is a modulo m partial product. The modulo m partial product is generated with y_i, ith digit position of the multiplier Y, and $\langle 2^i X \rangle_m$. When $m = 2^n \pm 1$, an i-digit end-around-shift is performed for the partial product generation. In this case, n modulo m partial products are generated for the modulo m sum. The performance of the modulo m SD multiplication is concerned in how many modulo m partial products to be added and the efficiency of the modulo m SD addition. A fast modulo m SD adder has been proposed and the addition time is independent of the word length of operands [9]. Then, in this paper, a suitable Booth recoding method is presented for generating fewer modulo m partial products.

Table 2 Recoding to eliminate strings of 1's.

y_i	y_{i-1}	p_i
-1	-1	-1
-1	0	-1
-1	1	0
0	-1	0
0	0	0
0	1	1
1	-1	-1
1	0	-1
1	1	0

3.2 Booth Recoding

To generate fewer modulo m partial products for a modulo m multiplication with the radix-two SD numbers, an idea is to convert the multiplier, Y, in the SD number representation into a recoded SD (RSD) number representation, in which there are no two consecutive negative one's (-1s) and one's (1s). Therefore, all two digits in the RSD number representation have a value in $\{-2,-1,0,1,2\}$ and the modulo m products can be obtained by operations such as modulo m shift and inverse of the value of X. To obtain such an SD number representation, we first recode Y to eliminate strings '11' in any two-digit position as shown in **Table 2**. In the following representation, cf_i and wf_i are the carry and intermediate sum of ith digit position, respectively. The rule is based on the fact that if $y_i = 1$, then $cf_i = 1$, $wf_i = -1$; if $y_{i-1} = 1$, then $cf_{i-1} = 1$, $wf_{i-1} = -1$; otherwise, $cf_i = 0$, $wf_i = y_i$ and $cf_{i-1} = 0$, $wf_{i-1} = y_{i-1}$. These yield the following equations:

$$2cf_i + wf_i = y_i, \tag{18}$$
$$2cf_{i-1} + wf_{i-1} = y_{i-1}, \tag{19}$$
$$p_i = wf_i + cf_{i-1}. \tag{20}$$

Note that we do not have the string '11' in P. Then, we recode p_i into ws_i and cs_i, where cs_i and ws_i are the carry and the intermediate sum of ith digit position of P respectively, for the elimination of the string -1's, and it does not recreate new adjacent 1's by the equations below.

$$2cs_{i-1} + ws_{i-1} = p_{i-1}, \tag{21}$$
$$2cs_i + ws_i = p_i, \tag{22}$$

where $cs_i, ws_i, p_i \in \{-1,0,1\}$. ws_i and cs_i are based on **Table 3**.

Finally, t_i is the sum of ws_i and cs_{i-1} representing as follows:

$$t_i = ws_i + cs_{i-1}. \tag{23}$$

The step of recoding p_i into t_i can be combined into the single-step recoding of **Table 4**. In Table 4, * means don't care. Therefore, we have

$$T = \sum_{i=0}^{n} t_i 2^i = \sum_{i=0}^{n-1} y_i 2^i, \tag{24}$$

where $y_i, t_i \in \{-1,0,1\}$. Then T is in the recoded SD(RSD) number representation and has the same numerical value as Y does. In every two digits of the RSD number representation, there are no the strings of '11' or '$-1-1$'.

[Theorem 1] If $Y = (y_{n-1}\cdots y_1 y_0)$ is recoded to $T = (t_n \cdots t_1 t_0)$ using Tables 2 and 4, where $y_i, t_i \in \{-1,0,1\}$, then $t_{i+1}t_i \neq 1$ for $i = 0,1,\cdots,n-1$, and $\sum_{i=0}^{n} t_i 2^i = \sum_{i=0}^{n-1} y_i 2^i$.

(proof) It is convenient to write $\bar{1} = -1$. If $t_i = \bar{1}$, $(p_i p_{i-1} p_{i-2})$ must be in $\{(\bar{1}1*), (0\bar{1}0), (0\bar{1}\bar{1})\}$ from Table 4. To recode p_{i+1} to t_{i+1}, p_i and p_{i-1} are used as the reference digits. If $t_{i+1} = \bar{1}$, $(p_{i+1}p_i p_{i-1})$ also have to belong to one of $\{(\bar{1}1*), (0\bar{1}0), (0\bar{1}\bar{1})\}$, that is, $(p_i p_{i-1}) \in \{(1*), (\bar{1}0), (\bar{1}\bar{1})\}$ is not in $\{(\bar{1}1), (0\bar{1})\}$. Therefore, we don't have the string of $\bar{1}\bar{1}$. Analogously, if $t_i = 1$, $(p_i p_{i-1} p_{i-2})$ must be in $\{(\bar{1}0*), (\bar{1}\bar{1}1), (10*), (1\bar{1}1)\}$ from Table 4. If $t_{i+1} = 1$, we can also see that $(p_i p_{i-1}) \in \{(0*), (\bar{1}1)\}$ is not in $\{(\bar{1}0), (\bar{1}\bar{1}), (10), (1\bar{1})\}$ from Table 4. Therefore, we don't have the string of 11 and $\bar{1}\bar{1}$. In other words $t_{i+1}t_i \neq 1$.

Table 3 The intermediate carry and the intermediate sum.

	$p_i \geq 0$	$p_i < 0$	
		$p_{i-1} > 0$	$p_{i-1} \leq 0$
ws_i	p_i	p_i	$-p_i$
cs_i	0	0	p_i

Table 4 Recoding to eliminate strings of $\bar{1}$'s.

p_i	p_{i-1}	p_{i-2}	t_i
-1	1	*	-1
-1	0	*	1
-1	-1	1	1
-1	-1	0	0
-1	-1	-1	0
0	1	*	0
0	0	*	0
0	-1	1	0
0	-1	0	-1
0	-1	-1	-1
1	0	*	1
1	-1	1	1
1	-1	0	0
1	-1	-1	0

In the next, we proof $\sum_{i=0}^{n} t_i 2^i = \sum_{i=0}^{n-1} y_i 2^i$. From Eqs. (18)–(20),

$$\begin{aligned} Y &= \sum_{i=0}^{n-1} y_i 2^i \\ &= y_{n-1}2^{n-1} + \cdots + y_1 2^1 + y_0 2^0 \\ &= (2cf_{n-1} + wf_{n-1})2^{n-1} + \cdots \\ &\quad + (2cf_0 + wf_0)2^0 \\ &= (cf_{n-1})2^n + (wf_{n-1} + cf_{n-2})2^{n-1} + \\ &\quad \cdots + (wf_0 + 0)2^0 \\ &= p_n 2^n + p_{n-1}2^{n-1} + \cdots + p_1 2^1 + p_0 2^0, \end{aligned}$$

where $p_n = cf_{n-1}$, $cf_{-1} = 0$ and $p_i = wf_i + cf_{i-1}$ for $0 \le i \le n-1$. According to Eq. (23), $T = \sum_{i=0}^{n} t_i 2^i$ can be expressed as follows:

$$\begin{aligned} T &= cs_n 2^{n+1} + (ws_n + cs_{n-1})2^n \\ &\quad + (ws_{n-1} + cs_{n-2})2^{n-1} \\ &\quad + \cdots + (ws_0 + 0)2^0 \\ &= (2cs_n + ws_n)2^n \\ &\quad + (2cs_{n-1} + ws_{n-1})2^{n-1} \\ &\quad + \cdots + (2cs_0 + ws_0)2^0 \end{aligned}$$

where $cs_n = 0$. From Table 3, when $p_i \ge 0$, then $2cs_i + ws_i = 0 + p_i = p_i$. When $p_i < 0$ and $p_{i-1} > 0$, then $2cs_i + ws_i = 0 + p_i = p_i$. When $p_i < 0$ and $p_{i-1} \le 0$, then $2cs_i + ws_i = 2p_i - p_i = p_i$. Therefor, $2cs_i + ws_i = p_i$ for $i = 1, \cdots, n$. Thus T can be rewritten as follows:

$$\begin{aligned} T &= p_n 2^n + p_{n-1}2^{n-1} + \cdots + p_1 2^1 + p_0 2^0 \\ &= Y \end{aligned}$$

□

Let B_i be a Booth-code, where $B_i = 2b_{2i+1} + b_{2i} = 2t_{2i+1} + t_{2i}$ and $b_{2i+1} \cdot b_{2i} = 0$. Then the multiplier Y can be expressed as

$$Y = \sum_{i=0}^{n} t_i 2^i = \sum_{i=0}^{n} b_i 2^i = \sum_{i=0}^{\lfloor n/2 \rfloor} B_i 2^{2i}. \tag{25}$$

Substituting above equation to Eq. (16) and this will imply

$$\langle X \times Y \rangle_m = \left\langle \sum_{i=0}^{\lfloor n/2 \rfloor} \langle B_i 2^{2i} \times (X) \rangle_m \right\rangle_m . \tag{26}$$

[Example 2] $Y = (110\bar{1}\bar{1}110)$, and $y_8 = y_{-1} = y_{-2} = y_{-3} = 0$. Then $P = (10\bar{1}0\bar{1}00\bar{1}0)$ by Table 2, and $T = (1\bar{1}1\bar{1}10\bar{1}10)$ by Table 4. Then $B = (10\bar{1}0\bar{1}0\bar{1}10)$. □

According to Tables 4 and Table 2, t_i is dependent on p_i, p_{i-1}, p_{i-2}, and p_i, p_{i-1}, p_{i-2} are dependent on $y_i, y_{i-1}, y_{i-2}, y_{i-3}$. It means that $y_i, y_{i-1}, y_{i-2}, y_{i-3}$ are used for getting t_i. In **Fig. 2** the functional block BR implements Tables 2 and 4 and block NR is for transforming $(1, -1)$ and $(-1, 1)$ into $(0, 1)$ and $(0, -1)$ respectively. If the word length n is odd, the number of Booth-codes is $\frac{n+1}{2}$. If n is even, we need to add an extra digit 0 to the left of B and the number of Booth-codes is $\frac{n}{2} + 1$.

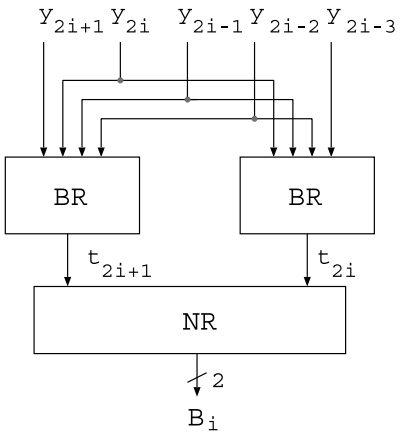

Fig. 2 Two-digit Booth Recoding (TBR).

Table 5 The rule for generating modulo m partial products.

b_{2i+1}	b_{2i}	PP_i
1	0	$\langle 2 \times 2^{2i} X \rangle_m$
0	1	$\langle 2^{2i} X \rangle_m$
0	0	$\langle 0 \rangle_m$
0	−1	$-\langle 2^{2i} X \rangle_m$
−1	0	$-\langle 2 \times 2^{2i} X \rangle_m$

3.3 Modulo m Partial Product Generation

The advantage of the Booth-code lies in the reduced number of iterations required, the reduction being from n steps to $\frac{n+1}{2}$ steps when n is odd and $\frac{n}{2}+1$ steps when n is even. A Booth-code consists of a pair of (b_{2i+1}, b_{2i}) which is used to reduce the number of modulo m partial products. When $(t_{2i+1}, t_{2i}) = (1, -1)$ or $(t_{2i+1}, t_{2i}) = (-1, 1)$, we transform them into $(b_{2i+1}, b_{2i}) = (0, 1)$ and $(0, -1)$ respectively. Thus, only five kinds of products can be generated by using the Booth-code (b_{2i+1}, b_{2i}) as shown in **Table 5**. In the partial product generation with the SD number representation, $-X$ means that the signs of all digits of X are inversed.

Let $SX = \langle 2^{2i} X \rangle_m$. Then $\langle 2 \times 2^{2i} X \rangle_m = \langle 2 \times SX \rangle_m$. In the case of $\mu \in \{-1, 0, 1\}$, $\langle 2SX \rangle_m$

can be calculated by shifting and/or inversing some digits of SX. In the case of $1 < \mu \leq 2^{n-1} - 1$, $\langle 2 \times SX \rangle_m$ is implemented with one SD addition. Since $b_{2i} \times b_{2i+1} = 0$, PP_i can be generated with an OR operation of pp_{2i+1} and pp_{2i}:

$$PP_i = pp_{2i+1} + pp_{2i} = pp_{2i+1}\ OR\ pp_{2i}$$

where pp_{2i} and pp_{2i+1} are two modulo m partial products as shown in Eq. (17).

In the case of $\mu \in \{-1, 0, 1\}$, since $\langle 2^{n-1} \times y_{n-1} \rangle_{2^n+\mu} = \langle 2^{-1} \times (-\mu y_{n-1}) \rangle_{2^n+\mu}$, we have one less modulo m partial product by using

$$y_{-1} = -\mu y_{n-1} \quad (27)$$

$$y_{-2} = -\mu y_{n-2} \quad (28)$$

$$y_{-3} = -\mu y_{n-3}. \quad (29)$$

[Example 3] Let $Y = (\bar{1}\bar{1}0011)$, modulus $m = 65$ and $\mu = 1$. From Eqs. (27), (28) and (29), $y_{-1} = 1$, $y_{-2} = 1$ and $y_{-3} = 0$. Then $P = (\bar{1}\bar{1}0100)$ and $p_{-1} = 0, p_{-2} = \bar{1}$. From Table 4, $T = (010100) = 20$. By Definition 1, $\langle Y \rangle_m = \langle -45 \rangle_{65} = -45 = 20 = T$. □

By using the above modulo m product generation method, a modulo m multiplication can be performed by the following algorithm.

Algorithm A ($\langle X \times Y \rangle_m$):

If $\mu \in \{-1, 0, 1\}$, let $y_{-1} = -\mu y_{n-1}$, $y_{-2} = -\mu y_{n-2}$, and $y_{-3} = -\mu y_{n-3}$; if $1 < \mu \leq 2^{n-1} - 1$, let $y_n = y_{-1} = y_{-2} = y_{-3} = 0$. Let $SX = X$, $sum = 0$ and $i = 0$.

(1) Generate Booth-code
  1a) Use Tables 2 and 4 to recode (y_{2i+1}, y_{2i}, y_{2i-1}, y_{2i-2}, y_{2i-3}) to (t_{2i+1}, t_{2i});
  2b) If $abs(t_{2i+1}) = abs(t_{2i})$, then $(b_{2i+1}, b_{2i}) = (0, -t_{2i})$, else $(b_{2i+1}, b_{2i}) = (t_{2i+1}, t_{2i})$.

(2) Generate modulo m partial products
  2a) $pp_{2i} = b_{2i} \times SX$;
  2b) $SX = \langle 2SX \rangle_m$;
  2c) $pp_{2i+1} = b_{2i+1} \times SX$;
  2d) $SX = \langle 2SX \rangle_m$;
  2e) $PP_i = pp_{2i}\ OR\ pp_{2i+1}$;

(3) Sum the modulo m partial products
  3a) $sum = \langle sum + PP_i \rangle_m$;
  3b) $i = i + 1$, and return to (1) until $i = (n/2) - 1$

Note that the number of modulo m partial products is $\frac{n}{2}$ in the case of $\mu \in \{-1, 0, 1\}$ and $\frac{n}{2} + 1$ in the case of $1 < \mu \leq 2^{n-1} - 1$ when n is even. $\frac{n}{2}$ times of the modulo m additions are performed serially. □

A serial Booth modulo m SD multiplier can be designed to implement the above algorithm. By Theorem 1, the Booth-code generation in step (1) can be performed in parallel. Thus, if step (2) is performed in parallel, the modulo m sum in step (3) can be implemented by a binary modulo m adder tree [9),10)].

X=6		00	00	10	$\bar{1}0$	
Y=170		10	11	$0\bar{1}$	$\bar{1}0$	
B		$0\bar{1}$	$0\bar{1}$	$\bar{1}0$	01	B_i
$i = 0$						
sum		00	00	00	00	
PP_0	+	00	00	10	$\bar{1}0$	01
w_i		00	00	10	$\bar{1}0$	
c_i	+	00	00	00	00	
sum		00	00	10	10	
$i = 1$						
sum		00	00	10	$\bar{1}0$	
PP_1	+	$0\bar{1}$	01	00	00	$\bar{1}0$
w_i		01	01	10	10	
c_i	+	00	10	00	00	
sum		01	11	10	10	
$i = 2$						
sum		$0\bar{1}$	$1\bar{1}$	10	$\bar{1}0$	
PP_2	+	$\bar{1}0$	10	00	00	$0\bar{1}$
w_i		$1\bar{1}$	$0\bar{1}$	10	$\bar{1}0$	
c_i	+	01	00	00	01	
sum		10	01	10	11	
$i = 3$						
sum		10	$0\bar{1}$	10	$\bar{1}1$	
PP_3	+	10	00	00	10	$0\bar{1}$
w_i		00	01	10	01	
c_i	+	00	00	00	$0\bar{1}$	
$\langle X \times Y \rangle_{257}$		00	01	10	00	

Fig. 3 Example of serial Booth modulo m SD multiplication.

[Example 4] Let $m = 257$, $X = 6 = (00001\bar{1}0)$, and $Y = 170 = (10110\bar{1}\bar{1}0)$. From (27), (28) and (29), $y_{-1} = \bar{1}, y_{-2} = 0, y_{-3} = \bar{1}$. Then $P = (\bar{1}10\bar{1}0\bar{1}\bar{1}0)$ and $p_{-1} = \bar{1}, p_{-2} = 0$. From Table 4, $T = (\bar{1}\bar{1}\bar{1}\bar{1}01\bar{1})$. Then $B = (0\bar{1}0\bar{1}\bar{1}001)$. **Figure 3** illustrates the calculation process by Algorithm A. The result of $\langle 6 \times 170 \rangle_{257}$ is equal to -8. □

4. On VLSI Implementation and Performance Evaluation

For the circuit design, an SD digit $a \in \{-1, 0, 1\}$ is encoded as a two-bit binary code as follows

$$a = [a_s, a_0],$$

where a_s is the sign and a_0 is the absolute value. We use VHDL to design the residue arithmetic circuits for the implementation of the proposed Booth modulo m SD multiplication. Then a simulation is performed under the condition of $1\,\mu$m CMOS gate array technology.

Table 6 Performance of parallel modulo m multipliers.

	SD representation				Binary representation			
Modulus	normal [9]		Fig. 4		normal		Ref. 17)	
(2^n-1) n	delay (ns)	area (gates)	delay (ns)	area (gates)	delay (ns)	area (gates)	delay (ns)	area (gates)
4	16.94	474	14.75	364	20.89	139	19.72	136
8	22.92	2106	21.28	1422	35.62	582	37.03	456
16	29.94	8826	29.4	7138	71.79	2585	73.75	2103

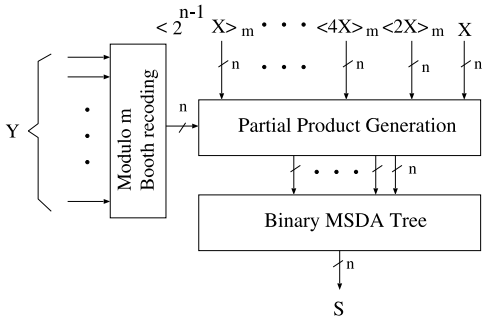

Fig. 4 Parallel Booth modulo m multiplier.

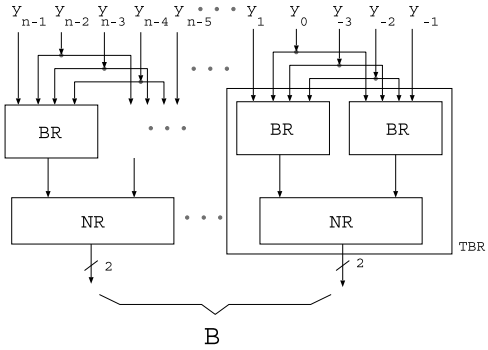

Fig. 5 Modulo m Booth recoding.

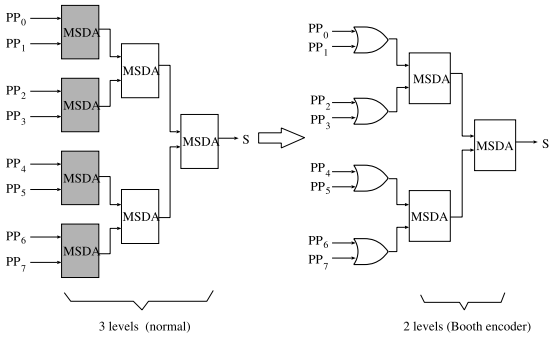

Fig. 6 MSDAs binary tree for $n = 8$.

4.1 Comparison of Parallel Modulo m Multipliers

In the case of $\mu \in \{-1, 0, 1\}$, a parallel Booth modulo m multiplier with a binary MSDA tree is constructed as shown in **Fig. 4**. The modulo m Booth recoding block consists of n identical sub-blocks BRs as shown in **Fig. 5**. In Fig. 5, five consecutive digits of Y are sent to a TBR block to generate a Booth-code B_i. In Partial Product Generation block, $\frac{n}{2}$ modulo m partial products are generated by using the Booth-codes and summed up in the MSDAs. Since the MSDAs are organized in a binary tree structure, the stages of the modulo m additions is usually $\log_2(n)$. The number of MSDAs required in the binary tree is equal to $n-1$. Since MSDAs of the first stage can be replaced by OR gates by the proposed Booth recoding, the number of MSDAs required in the binary tree is reduced to $\frac{n}{2}-1$ as shown in **Fig. 6**. As mentioned before, when n is even, in the case of $1 < \mu \leq 2^{n-1}-1$, an extra one digit 0 will be added to the left of B and it leads that one more MSDA is necessary.

The area of a parallel modulo m multiplier is mainly concerned with how many MSDAs are used. Compared the proposed Booth modulo m multiplier with Ref. 9), $(\frac{n}{2})$ MSDAs are saved. The performances of four kinds of modulo m multipliers are summarized in **Table 6**, where $m = 2^n - 1$. The multiplier with the Booth recoding has the higher performance than that the original SD one [9] has. The area of the modulo m SD multiplier using the Booth recoding is reduced to 80% from the normal one. Two modulo 2^n-1 multipliers with a binary number representation have been also implemented by the same design tool. Both designs used the carry save addition (CSA) and a wallace tree structure, but one of them used a Booth recoding method [17]. The delay time of the proposed SD multiplier is about 40% of the delay time of the binary ones.

In the case of $\mu = 0$, the modulo m multipliers can be designed with the same performances as that the multipliers have. In the case of $\mu = 1$, our proposed multipliers can also have the same performance, because of the end-around-carry SD addition. However, a modulo

$2^n + 1$ multiplier with the binary number representation can not be designed by using the similar method proposed in Ref. 17).

4.2 Comparison of Serial Modulo m Multipliers

Figure 7 illustrates the block diagram of a serial Booth modulo m multiplier using one MSDA for the repeated modulo m sum. X, Y and $S = \langle X \times Y \rangle_m$ represent the multiplicand, multiplier and multiplication result with the n-digit SD number representation, respectively. The multiplier consists of modulo m signed-digit adder (MSDA), some registers, a two-digit Booth Recoding (TBR) shown in Fig. 2, a modulo m partial product generation (PPG), a modulo m shifter ($\langle 2SX \rangle_m$). In the cases of $m = 2^n + 1, 2^n, 2^n - 1$, we set $y_{-1} = -\mu y_{n-1}$, $y_{-2} = -\mu y_{n-2}$ and $y_{-3} = -\mu y_{n-3}$. In the case of $2^n + 1 < m \leq 2^n + 2^{n-1} - 1$, we set $y_n = y_{-1} = y_{-2} = y_{-3} = 0$. Every five digits of Y are used to generate one Booth-code B_i used for PPG. Reg-SX and $\langle 2SX \rangle_m$ are inputed to PPG. When $b_{2i+1} = 1$ or -1, $\langle 2SX \rangle_m$ will be selected; when $b_{2i} = 1$ or -1, Reg-SX will be selected for generating the modulo m partial product. In the case of $(b_{2i+1}b_{2i}) = (00)$, PPG is equal to n-digit 0. An MSDA is used to add modulo m partial products by PPG and Reg-S.

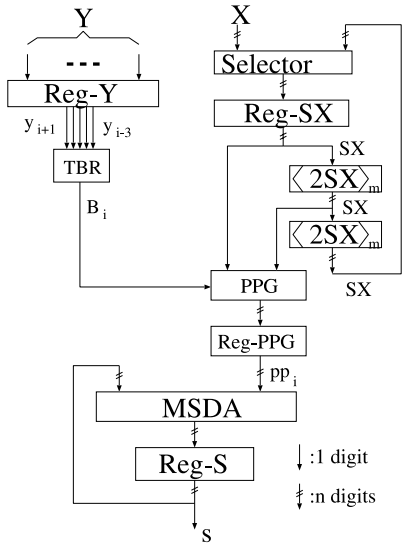

Fig. 7 Serial Booth modulo m multiplier.

To evaluate the proposed recoding method, we also implement the recoding algorithm of Ref. 18) as two components. The first component is for generating an intermediate carry and an intermediate recoded digit. The second component is for generating Booth-code by summing the intermediate recoded digit and a low intermediate carry. The performances of the Booth recoding circuits are illustrated in **Table 7**, and our Booth recoding is more efficient.

In **Table 8**, the performance comparison of two serial modulo m SD multipliers with the architectures as shown in Fig. 7 and presented in Ref. 9) is illustrated. Since the longest delay path is in the modulo m SD adder, the working clock is dependent on it. Thus, both the multipliers using the Booth recoding or not can work at the same clock, and they are much faster than that using a binary adder. Because the modulo m partial products using the proposed Booth recoding are half, the speed of the presented modulo m multiplier is twice as fast as that of the original one [9]. In Table 8, the hardware overhead for Booth recoding is needed. However the high speed multipliers are implemented and the small drawback in terms of the area is not a problem.

5. Conclusion

A Booth recoding method has been newly introduced to reduce the modulo m partial products for high speed modulo m multiplication with the SD number representation. In a parallel Booth modulo m multiplier, a smaller binary MSDA tree can be designed and the hardware cost can be reduced to 80% from the orig-

Table 7 Performance of Booth recoding.

n	area(gates)		delay(ns)	
	Ref. 18)	this paper	Ref. 18)	this paper
8	192	168	8.44	4.06
16	384	336	8.44	4.06

Table 8 Performance of Serial Modulo m Multipliers with SD number.

Modulus	normal [9]			Fig. 7		
	clock (MHz)	total delay (ns)	area (gates)	clock (MHz)	total delay (ns)	area (gates)
257	130.4	60.8	620	130.4	30.4	804
259	61.65	129.76	1387	61.65	81.1	1927
65537	130.4	121.6	1294	130.4	60.8	1657
65539	61.65	259.52	2975	61.65	145.98	4128

inal one. A serial Booth modulo m multiplier has the same clock rate compared to the original one [9], so that the speed performance of the modulo m multiplication can be up to twice as fast.

For VLSI implementation of the modular SD arithmetic in RNS, the logic circuit descriptions by VHDL are also designed.. The design and simulation results under the condition of 1 μm CMOS gate array technology show that high-speed modulo m multipliers based on SD arithmetic can be obtained. The proposed modulo m SD multipliers have high performances comparing to that with the binary number arithmetic.

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 conversion between the arithmetic number systems, and the application to the computation systems, such as digital signal processing and digital control systems.

References

1) Szabo, N.S. and Tanaka, R.I.: *Residue Arithmetic and Its Applications to Computer Technology*, New York: McGraw-Hill (1967).
2) Paliouras, V. and Stouraitis, T.: Novel High-Radix Residue Number System Architectures, *IEEE Trans.on circuits and systems II.*, Vol.47, No.10 (2000).
3) Sonderstrand, M.A., Jendins, W.K., Junllien, G.A. and Taylor, F.J.: *Residue Number System Arithmetic: Modern Applications in Digital Signal Processing*, IEEE Press, New York (1986).
4) Skavantzos, A. and Rao, P.B.: New Multipliers Modulo $2^n - 1$, *IEEE Trans. Comput.*, Vol.41, No.8, pp.957–961 (1992).
5) Hiasat, A.: New memoryless, mod $(2^n \pm 1)$ residue multiplier, *Electron. Lett.*, Vol.28, No.3, pp.314–315 (Jan. 1992).
6) Kalampoukas, L., Nikolos, D., Efstathiou, C., Vergos, H.T. and Kalamatianos, J.: High-Speed Parallel-Prefix Modulo $2^n - 1$ Adders, *IEEE Trans. Compute.*, Vol.49, No.7, pp.673–680 (2000).
7) Avizienis, A.: Signed-digit number representations for fast parallel arithmetic, *IRE Trans. Elect. Comput.*, EC-10, pp.389–400 (Sep. 1961).
8) Parhami, B.: Carry-Free Addition of Recoding Binary Signed-Digit Numbers, *IEEE Trans. comput.*, Vol.37, No.11 (1988).
9) Wei, S. and Shimizu, K.: A Novel Residue Arithmetic Hardware Algorithm Using a Signed-Digit Number Representation, *IEICE Trans. Inf.& Syst.*, Vol.E83-D, No.12, pp.2056–2064 (2000).
10) Wei, S. and Shimizu, K.: Compact Residue Arithmetic Multiplier Based on the Radix-4 Signed-Digit Multiple-Valued Arithmetic Circuits, *IEICE Trans. Electron.*, Vol.E82-C, No.9, pp.1647–1645 (1999).
11) Booth, A.D.: A signed binary multiplication technique, *Quart. J. Mech. Appl. Math.*, Vol.4, pp.236–240 (1951).
12) MacSorley, O.L.: High speed arithmetic in binary computers, *Proc. IRE*, Vol.49, pp.67–91 (1961).
13) Lyu, C.N. and Matula, D.W.: Redundant binary booth recoding, *Proc. IEEE 12th Symp. Cumput. Arith.*, pp.50–57 (1995).
14) Yeh, W.-C. and Jen, C.-W.: High-Speed Booth Encoded Parallel Multiplier Design, *IEEE Trans. Comput.*, Vol.49, No.7, pp.692–701 (2000).
15) Inoue, T., Tamura, A., Ochi, H. and Tsuda, T.: On the Circuit for Booth Recoder in Multiplier, *Technical report of IEICE, CAS2002-23, VLD2002-37, DSP2002-63*, pp.131–136 (June 2002).
16) Chen, S., Wei, S. and Shimizu, K.: A Booth Recoding Method for Serial Modular Multipliers with Signed-Digit Number Representation, *ICF 2002*, pp.2-10–2-15 (Mar. 2002).
17) Efstathiou, C., Vergos, H.T. and Nikolos, D.: Modified Booth Modulo $2^n - 1$ Multipliers, *IEEE Trans. Compute.*, Vol.53, No.3, pp.370–374 (2004).
18) Takagi, N.: A radix-4 Modular Multiplication Hardware Algorithm for Modular Exponentiation, *IEEE Trans. Compute.*, Vol.41, No.8 (1992).

(Received March 1, 2005)
(Accepted September 2, 2005)
(Online version of this article can be found in the IPSJ Digital Courier, Vol.1, pp.616–625.)

Shuangching Chen was born in Kaohsiung, Taiwan on July 18, 1972. He received the B.E. degree in Applied Mathematic from Feng Chia University, Taiwan, Republic of China in 1994, and the M.E. degree in Computer Science from Gunma University, Kiryu, Japan in 2002. He is currently a doctoral student at the Department of Computer Science, Gunma University. His research interests include parallel computer architecture, residue architecture, VLSI design and digital signal processing.

Shugang Wei was born in Harbin, China on September 19, 1957. He received the B.E. degree in Radio Engineering from Harbin Institute of Technology, Harbin, China, the M.E. degree in Computer Science from Gunma University, Kiryu, Japan and the Dr. Eng. degree in Electronic Engineering from Tohoku University, Sendai, Japan, in 1982, 1987, and 1990, respectively. He was an Assistant Professor with the Department of Radio Engineering, Harbin Institute of Technology from 1982 to 1984. In 1990 he joined Matsushita Communication Industrial Co., Ltd., Yokohama, Japan. At present he is an Associate Professor in the Department of Computer Science, Faculty of Engineering, Gunma University. His research interests include logic design, high-speed arithmetic circuits, VLSI systems and digital audio signal processing. Dr. Wei is a member of the Acoustical Society of Japan and IEEE.