Residue Arithmetic Circuits Based on Signed-Digit Number Representation and the VHDL Implementation

Shugang Wei Kensuke Shimizu
Department of Computer Science, Gunma University, Kiryu 376, Japan
wei@ja4.cs.gunma-u.ac.jp

Abstract

Residue arithmetic circuits based on radix-2 signed-digit(SD) number representation, using integers 2^p and $2^p \pm 1$ as moduli of residue number system(RNS), are presented. The modulo m addition, $m = 2^p$ or $m = 2^p \pm 1$, is performed by a carry-free SD adder and the modulo m multiplier is constructed using a binary modulo m SD adder tree. The implementation for the residue arithmetic circuits with VHDL description is proposed. The modulo m adders and multipliers have about 530 and 5000 gates, respectively, in cases of $m = 2^{16} \pm 1$.

1. Introduction

Integers 2^p and $2^p \pm 1$ are commonly used as moduli for the residue number system(RNS), because the addition modulo 2^p or $2^p \pm 1$ can be implemented by p-bit binary adders[1, 2]. Some modulo $2^p \pm 1$ multipliers have been proposed[3, 4]. However, since these modulo $2^p \pm 1$ adders and multipliers are designed based on the ordinary binary arithmetic systems, the carry propagation will arise during additions and limits the speed of arithmetic operations in residue modules.

It is known that carry propagation is limited to one position during additions of signed-digit (SD) numbers[5]. In this paper, we apply the SD number system to the residue number system with moduli $\{2^p \pm 1, 2^p\}$, and present the implementation with VHDL. We give a binary representation for the radix-2 signed-digit and modified algorithms. Then describing the algorithms by VHDL, 8-digit and 16-digit residue arithmetic circuits have been designed. The addition time is about 7.5ns, and the multiplication times are about 30ns and 38ns for the 8-digit and for the 16-digit multipliers, respectively, by simulation using 1μ CMOS design rule. The modulo m adders and multipliers have about 530 and 5000 gates, respectively, in cases of $m = 2^{16} \pm 1$.

2. Residue Number System

A set of positive integers in the forms $2^p - 1, 2^p$ and $2^p + 1$ is used as the moduli of an RNS, where p is a positive integer. Thus, a modulus m is written as follows:

$$m = 2^p \pm h, h \in \{-1, 0, 1\}. \quad (1)$$

Different values of p are selected by satisfying the condition that the moduli are relatively prime in pairs. For example, $\{2^7 - 1, 2^7 + 1, 2^8, 2^8 + 1, 2^9 - 1\}$ can be used as the moduli in an RNS.

Conventionally, a nonnegative integer A is represented by the n-tuple $(A_1, A_2, \cdots, A_n)$ in an RNS with moduli $m_1, m_2, \cdots, m_n$, where A_i is in the range $[0, m_i)$ by

$$A_i = A \bmod m_i = |A|_{m_i}, (i = 1, 2, \cdots, n). \quad (2)$$

In this paper, we replace the binary number by the radix-2 SD number for high-speed residue arithmetic.

Definition 1: Let X be an integer and m be a positive integer. Then $x = \langle X \rangle_m$ is defined as an integer in the range $[-m, m]$ as follows:

$$x = \langle X \rangle_m = sign(X) \times |abs(X)|_m, \quad (3)$$

or,

$$x = \langle X \rangle_m = sign(X) \times |abs(X)|_m - sign(X) \times m, \quad (4)$$

where

$$sign(X) = \begin{cases} -1 & X \le 0 \\ 1 & X \ge 0 \end{cases}$$

and

$$abs(X) = \begin{cases} -X & X \le 0 \\ X & X \ge 0 \end{cases}.$$

□

For example, $\langle 17 \rangle_7 = 3$ by Eq.(3) or -4 by Eq.(4). Thus, the addition, subtraction and multiplication of n-tuples $A = (A_1, A_2, \cdots, A_n)$ and $B = (B_1, B_2, \cdots, B_n)$ in an RNS can be represented as follows:

$$A \pm B = (\langle A_1 \pm B_1 \rangle_{m_1}, \cdots, \langle A_n \pm B_n \rangle_{m_n}), \quad (5)$$
$$A \times B = (\langle A_1 \times B_1 \rangle_{m_1}, \cdots, \langle A_n \times B_n \rangle_{m_n}). \quad (6)$$

We have the following properties which are very useful for the high-speed residue arithmetic.

Property 1: Let k be a positive integer. Then

$$\langle 2^k \rangle_{2^p+h} = \begin{cases} 2^k & k < p \\ (-h)^{(k \ div \ p)} \times 2^{|k|_p} & k \ge p \end{cases}, \quad (7)$$

where $h \in \{-1, 0, 1\}$, and $(k \ div \ p)$ is the integer part of the division result. □

For example, $\langle 2^4 \rangle_{2^3+1} = (-1)^{(4 \ div \ 3)} \times 2^{|4|_3} = -2$.

Property 2: Let a and b be integers. Then
(a) $abs(\langle a \rangle_m) \le m$,

0-7695-0104-4/99 $10.00 © 1999 IEEE

(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. □

3. Residue Arithmetic Based on Signed-Digit Number Representation

A. Modulo m Addition Algorithm

Integer x is represented by the q-digit radix-2 SD number representation as follows:

$$x = x_{q-1}2^{q-1} + x_{q-2}2^{q-2} + \cdots + x_0, \quad x_i \in \{-1,0,1\}\ (i=0,1,\cdots,q-1), \quad (8)$$

which can be rewritten as $x = (x_{q-1}, x_{q-2}, \cdots, x_0)_{SD}$. Obviously, $-x = -(x_{q-1}, x_{q-2}, \cdots, x_0)_{SD} = (-x_{q-1}, -x_{q-2}, \cdots, -x_0)_{SD}$. The SD number representation has redundancy; for example, 7 may be represented by $(0,1,1,1)_{SD}$, $(1,-1,1,1)_{SD}$ or $(1,0,0,-1)_{SD}$ for $q=4$.

Using the properties mentioned above, modulo m SD addition, where $m = 2^p + h, h \in \{-1,0,1\}$, can be performed by the following algorithm.

[Algorithm 1 (Calculation of $\langle a+b\rangle_m$)] Let a and b be two integers in the p-digit SD number representation, and c_i, z_i and s_i be the intermediate carry, intermediate sum and sum at the ith SD digit ($i = 0,1,\cdots,p-1$), respectively. For each digit, the following two steps are performed.

1) When $abs(a_i) = abs(b_i)$,

$$z_i = 0$$

and

$$c_i = (a_i + b_i)\ div\ 2;$$

when $abs(a_i) \neq abs(b_i)$,

$$z_i = \begin{cases} -(a_i+b_i) & \text{if } a_i+b_i \text{ and } a_{i-1}+b_{i-1} \text{ for } i\neq 0 \text{ or} \\ & a_0+b_0 \text{ and } -h(a_{p-1}+b_{p-1}) \text{ have the same signs} \\ a_i+b_i & \text{otherwise} \end{cases}$$

and

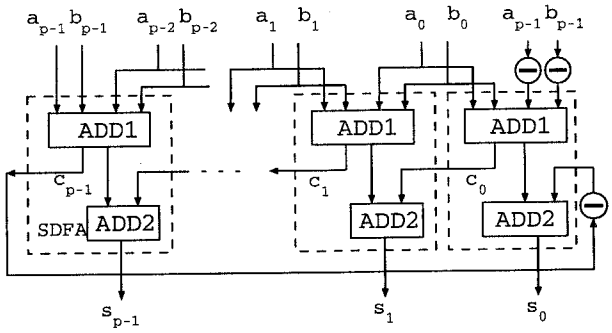
$$c_i = \begin{cases} a_i+b_i & \text{if } a_i+b_i \text{ and } a_{i-1}+b_{i-1} \text{ for } i\neq 0 \text{ or} \\ & a_0+b_0 \text{ and } -h(a_{p-1}+b_{p-1}) \text{ have the same signs} \\ 0 & \text{otherwise} \end{cases}.$$

Since $a_i, b_i \in \{-1,0,1\}$, $z_i, c_i \in \{-1,0,1\}$.

2) $s_i = z_i + c_{i-1}$ for $i \neq 0$ and $s_0 = z_0 + (-h)\times c_{p-1}$. Thus,

$$\langle a+b\rangle_m = s = s_{p-1}2^{p-1} + s_{p-2}2^{p-2} + \cdots + s_0.$$

□



⊖ :Sign inveter.

Figure 1: Modulo m SD adder (MSDA).

```
        i : 4  3  2  1  0
        a : 1  0 -1  1  1
        b : 1  1 -1  0  0
1)  ------------------------
    z :    0  1  0 -1  1
    c : 1  0 -1  1  0 [-1]
2)  ------------------------
    s :    0  0  1 -1  0
```

Figure 2: Example of modulo m addition.

In the above algorithm, it is always true that $2c_i + z_i = a_i + b_i$, z_i and c_{i-1} do not have the same signs. Therefore $s_i \in \{-1,0,1\}$. Thus the carry propagation is limited to one digit. A modulo m SD adder(MSDA) could be constructed using p SD full adders (SDFAs) as shown in Fig.1, where $m = 2^p + 1$. The blocks ADD1 and ADD2 perform step 1) and step 2) of Algorithm 1, respectively. The corresponding inputs to the SDFA at the least significant digit are wired with $-a_{p-1}$,$-b_{p-1}$ and $-c_{p-1}$,

The calculation of $\langle a-b\rangle_m$ can be realized by replacing b with $-b$ in the above algorithm.

Example 1 : Let $p=5$, $a = (1,0,-1,1,1)_{SD}$ and $b = (1,1,-1,0,0)_{SD}$, so that $m = 2^p+1 = 33$, $a = 15$ and $b = 20$. Fig.2 illustrates the calculations of $\langle a+b\rangle_m$ using the above algorithm. The result is $\langle 15+20\rangle_{33} = 2$. □

B. Modulo m Multiplication Algorithm

To calculate $\langle a\times b\rangle_m$, where a and b are integers in the p-digit radix-2 SD number representation, $a\times b$ can be extended as follows:

$$\begin{aligned} a\times b &= (a_{p-1}2^{p-1} + a_{p-2}2^{p-2} + \cdots + a_0) \\ &\quad \times (b_{p-1}2^{p-1} + b_{p-2}2^{p-2} + \cdots + b_0) \\ &= \sum_{i=0}^{p-1} b_i 2^i \times (a_{p-1}2^{p-1} + a_{p-2}2^{p-2} \\ &\quad + \cdots + a_0). \end{aligned}$$

Thus,

$$\begin{aligned}\langle a \times b\rangle_m &= \langle\sum_{i=0}^{p-1}\langle b_i 2^i \times (a_{p-1}2^{p-1} + a_{p-2}2^{p-2} \\ &\qquad + \cdots + a_0)\rangle_m\rangle_m \\ &= \langle\sum_{i=0}^{p-1} pp_i\rangle_m,\end{aligned}$$

where pp_i denotes as a partial product. Since $m = 2^p + h, h \in \{-1, 0, 1\}$, by Property 1 we have

$$\begin{aligned}pp_i &= \langle b_i 2^i (a_{p-1}2^{p-1} + a_{p-2}2^{p-2} + \cdots + a_0)\rangle_m \\ &= b_i(a_{p-i-1}2^{p-1} + a_{p-i-2}2^{p-2} + \cdots + a_0 2^i \\ &\quad -h(a_{p-1}2^{i-1} + \cdots + a_{p-i+1}2 + a_{p-i})).\end{aligned}$$

Therefore, $\langle a \times b\rangle_m$ can be represented by a modulo m sum of p partial products described above.

[Algorithm 2 (Calculation of $\langle a \times b\rangle_m$)] Let a and b be two p-digit radix-2 SD numbers.

1) Calculate partial products, pp_i $(i = 0, 1, \cdots, p-1)$, as follows:

$$\begin{aligned}pp_i &= b_i(a_{p-i-1}, a_{p-i-2}, \cdots, a_0, \\ &\quad -ha_{p-1}, \cdots, -ha_{p-i+1}, -ha_{p-i})_{SD} \\ &= (b_i a_{p-i-1}, b_i a_{p-i-2}, \cdots, b_i a_0, \\ &\quad -hb_i a_{p-1}, \cdots, -hb_i a_{p-i+1}, -hb_i a_{p-i})_{SD},\end{aligned}$$

where $a_i b_j \in \{-1, 0, 1\} (j = 0, 1, \cdots, p-1)$ and the subscripts are in the range $[0, p-1]$.

2) Calculate the modulo m sum of these partial products by performing the steps in Algorithm 1 repeatedly.

$$\langle a \times b\rangle_m = \langle pp_0 + pp_1 + \cdots + pp_{p-1}\rangle_m$$

□

4. VHDL Implementation

We specify a binary representation for a radix-2 signed-digit a_i, as shown in Table 1, 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.

$$\begin{aligned}a &= (a_{p-1}, a_{p-2}, \cdots, a_0)_{SD} \\ &= [a_{p-1}(1)a_{p-1}(0)\ \ a_{p-2}(1)(a_{p-2}(0) \\ &\qquad \cdots a_0(1)(a_0(0)] \qquad (9)\end{aligned}$$

For example, $(1, 0, 0, -1)_{SD} = [01000011]$. Using the binary representation, for two signed digits x_i and y_j, $p_{ij} = x_i y_j$ and $nx_i = -x_i$, can be expressed by the following equations.

$$\begin{aligned}p_{ij}(0) &= x_i(0)\ and\ y_j(0), \qquad (10)\\ p_{ij}(1) &= (x_i(1)\ xor\ y_j(1))\ and\ p_{ij}(0), \qquad (11)\end{aligned}$$

and

$$\begin{aligned}nx_i(0) &= x_i(0), \qquad (12)\\ nx_i(1) &= (not\ x_i(1))\ and\ nx_i(0), \qquad (13)\end{aligned}$$

Table 1: Binary representation for a radix-2 signed digit.

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

where *and*, *xor* and *not* are AND, EXCLUSIVE-OR and NOT operations, respectively.

As shown in Fig.1, there are 10 binary input signals to every SD full adder(SDFA) in a modulo m adder(MSDA). We modify the step 1 of Algorithm 1 for $abs(a_i) \neq abs(b_i)$ as follows:

$$z_i = \begin{cases} -(a_i + b_i) & if\ T = TL \\ a_i + b_i & \text{otherwise} \end{cases} \qquad (14)$$

and

$$c_i = \begin{cases} a_i + b_i & if\ T = TL \\ 0 & \text{otherwise} \end{cases}, \qquad (15)$$

where

$$\begin{aligned}T &= a_i(1)\ or\ b_i(1), \qquad (16)\\ TL &= a_{i-1}(1)\ or\ b_{i-1}(1), \qquad (17)\end{aligned}$$

or is OR operation. Since $abs(a_i) \neq abs(b_i)$, $T = 1$ means $(a_i + b_i) = -1$. $TL = 1$ means $(a_{i-1} + b_{i-1}) \in \{-2, -1, 0\}$ and $c_{i-1} \in \{-1, 0\}$. Therefore the modification does not change the fact that z_i and c_{i-1} do not have the same signs and $s_i \in \{-1, 0, 1\}$. Thus, the input signals of SDFA are reduced to 8, and the complexity of the logic circuit is simplified.

Based on the above discussion, SDFA is specified with VHDL and Fig.3 shows the logic circuit after design optimization. By using SDFAs as components, an MSDA can be constructed as shown in Fig.1. For a p-digit MSDA, two vectors,a and b, with $2p$-bit length are used as input signals and a vector, s, as output signals.

For the VHDL implementation of a modulo m multiplier, a binary tree of MSDAs is constructed, which implements step 2) of Algorithm 2. Fig.4 illustrates the VHDL description of a modulo m multiplier for $m = 2^8 + 1$. The partial product generator(PPG) is implemented by a VHDL process, which performs step 1 of Algorithm 2. MSDAs are used as components in a binary adder tree having 3 stages. Obviously, there are $p - 1$ MSDAs in the adder tree so that $p(p-1)$ SDFAs are needed. The modulo m additions are performed $\lceil \log_2(p) \rceil$ times in the longest addition path.

In Table 2, we show the performance of the residue arithmetic circuits by synthesis and design optimization. For $h = 0$, since the partial products have many ZERO signals, the optimized MSDM has much fewer gates. The delay time is obtained by simulation under the condition of 1μ CMOS technology. The delay time of a multiplier based on the architecture proposed by A.Hiasat[4] is about 95ns for $m = 16$. Therefore,

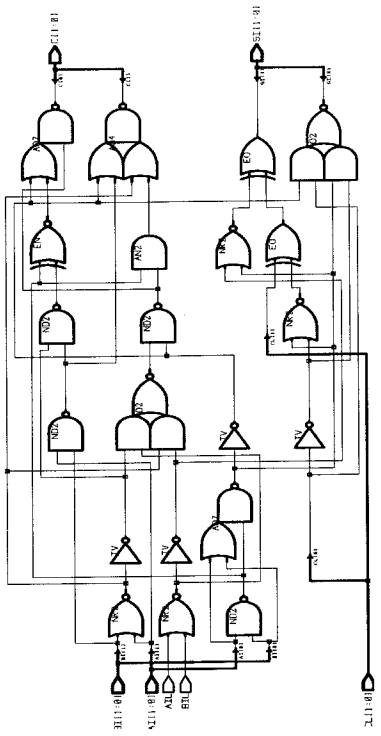


Figure 3: Logic circuit of SDFA.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity MSDMLY is
  generic(P : integer := 8);
  port(x,y:in std_logic_vector(2*P-1 downto 0);
       mly:out std_logic_vector(2*P-1 downto 0));
end MSDMLY;
architecture RTL of MSDMLY is
  component MSDA
   port(a,b:in std_logic_vector(2*P-1 downto 0);
        s :out std_logic_vector(2*P-1 downto 0));
  end component;
  constant H : std_logic_vector(1 downto 0):="01";
  subtype WORD is std_logic_vector(2*P-1 downto 0)
  type PROD is array (0 to P-1) of WORD;
  signal PP : PROD;
  signal SS : PROD;
begin
 PPG: process(x,y)
  variable T : WORD;
  begin
  for i in 0 to P-1 loop
    T(2*P-1 downto 2*i):=x(2*(P-i)-1 downto 0);
    for j in 0 to i-1 loop
      T(2*j):=x(2*(P-i+j)) and H(0);
      T(2*j+1):=(x(2*(P-i+j)+1)xor H(1))and T(2*j)
    end loop;
    for j in 0 to P-1 loop
      T(2*j):=T(2*j)and y(2*i);
      T(2*j+1):=T(2*j+1) and y(2*i_1) and T(2*j);
    end loop;
    PP(i)<=T;
  end loop;
  GEN1: for i in 0 to P/2-1 generate
    MU1:MSDA port map(PP(2*i),PP(2*i+1),SS(i));
  end generate GEN1;
  GEN2: for i in 0 to P/4-1 generate
    MU2:MSDA port map(SS(2*i),SS(2*i+1),SS(P/2+i))
  end generate MU2;
  MU3: MSDF port map(SS(P/2),SS(P/2+1),mly);
end RTL;
```

Figure 4: VHDL description of modulo m multiplier.

Table 2: Performance of the presented residue arithmetic circuits.

Modulus	Number of Gates		Delay Time(ns)	
m	MSDA	MSDM	MSDA	MSDM
2^8-1	264	2024	7.47	29.74
2^8+1	265	2084	7.51	30.13
$2^{16}-1$	528	4981	7.47	38.23
$2^{16}+1$	530	5037	7.51	39.03

high-speed residue circuits can been implemented. For the design of RNS chips, the MSDAs and the multipliers may be pre-designed with VHDL and are used as functional cells.

6. Conclusion

Residue arithmetic circuits using signed-digit number representation have been presented. Using integers $m \in \{2^p, 2^p \pm 1\}$ as moduli, the modulo m multiplier can be constructed with a binary SD adder tree. Thus the modulo m multiplication is performed in a time proportional to $\log_2 p$.

VHDL implementation for the presented circuits has been also proposed. The simulation results using 1μ CMOS technology show that high-speed residue arithmetic circuits can be obtained. Our studies also focus on the application of the presented residue arithmetic circuits 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] D.P. Agrawal and T.R.N.Rao,"Modulo (2^n+1) arithmetic logic,"IEE J. Electronic Circuits and Systems, Vol.2, pp. 186-188, Nov. 1978.

[3] F.J.Taylor,"A VLSI residue arithmetic multiplier," IEEE Trans. Comput., Vol.C-31, pp.540-546,June 1982.

[4] A.Hiasat,"New memoryless, mod $(2^n \pm 1)$ residue multiplier," Electron. Lett., Vol.28, No.3, pp.314-315,Jan. 1992.

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