A Novel Binary-to-Residue Conversion Algorithm for Moduli $(2^n - 1, 2^n, 2^n + 2^\alpha)$

Makoto Syuto, Eriko Satake, Koichi Tanno and Okihiko Ishizuka
Department of Electrical and Electronic Engineering, Faculty of Engineering
Miyazaki University, Miyazaki, Japan
Tel. +81-985-58-7409, E-mail: syuto@esl.miyazaki-u.ac.jp

Abstract: This paper describes a novel converter to implement high-speed binary-to-residue conversion for moduli $2^n - 1, 2^n, 2^n + 2^\alpha$ ($\alpha \in \{0, 1, \cdots, n-1\}$) without using look-up table. In our implementation, the high-speed converter can be achieved, because of the modulo addition time is independent of the word length of operands by using the Signed-Digit (SD) adders inside the modulo adders. For a LSI implementation of residue SD number system with ordinary binary system, the proposed binary-to-residue converter is the efficient circuit.

1. Introduction

The residue number system (RNS) has certain advantages of high-speed DSP applications, which require the modular RNS operations of additions and multiplications. On the other hand, it has the disadvantages of non-modular RNS operations, such as scaling, sign-detection and converting into and out of the RNS. With regard to the disadvantages, the hardware is complex and the operating speed is significant slow.

A novel residue arithmetic hardware algorithm using a signed-digit (SD) number representation has been proposed by S. Wei [1]. In the residue arithmetic hardware algorithm, since the modulo adders are designed based on the SD number representation system, the carry propagation will not arise during modulo additions Consequently, the computation time of the modulo addition is independent of the word length of operands.

For LSI implementation of the residue arithmetic hardware using the SD number representation with conventional binary number system, a binary-to-residue converter must be required. A number of the binary-to-residue converters have been proposed [2]–[4]. However the converters involve additions with carry propagation [2]. Since the carry propagation chain dominates the operating speed, it is a significant problem for a set of large integer modulus. For this problem, we proposed the high-speed binary-to-residue converter using the SD number representation [5]. But, the converter is limited to implement the conversion for three moduli $2^n, 2^n \pm 1$. On the other hand, the converters based on look-up tables were proposed. The disadvantage of the look-up table approach is that the size of the look-up table increases disproportionately with increasing magnitudes of the moduli [3], [4].

In this paper, we propose a novel binary-to-residue conversion algorithm for the moduli $2^n - 1, 2^n, 2^n + 2^\alpha$ ($\alpha \in \{0, 1, \cdots, n-1\}$). The conversion algorithm is extended from the Guan's conversion algorithm. And, we propose a binary-to-residue converter based on our conversion algorithm without using look-up table. In the proposed converter, the SD adders that can perform carry free addition are used inside the modulo adder. Since the modulo addition time is independent of the word length of operands, the high-speed converter can be achieved for set of large integer modulus without using look-up table.

2. Novel Binary-to-Residue Conversion Algorithm

2.1 Guan's Conversion Algorithm

The our proposed binary-to-residue conversion algorithm is based on the Guan's conversion algorithm [2]. The Guan's algorithm is briefly presented for completeness.

Let N be original integer ($3n$ bits). It can be represented as Eq. (1).

$$N = K_2 2^{2n} + K_1 2^n + K_0 \tag{1}$$

Where the range of N is $0 \leq N < (2^n - 1)2^n(2^n + 1)$, and the range of each coefficients in Eq. (1) are $0 \leq K_2, K_1, K_0 \leq 2^n - 1$.

The each residue numbers are converted from the original binary number as follows.

$$|N|_{2^n-1} = |K_0 + K_1 + K_2|_{2^n-1} \tag{2}$$
$$|N|_{2^n} = K_0 \tag{3}$$
$$|N|_{2^n+1} = |K_0 - K_1 + K_2|_{2^n+1} \tag{4}$$

2.2 Basic Algorithm

The Guan's algorithm limits that the modulus set is $2^n, 2^n \pm 1$.

Let N be original integer that is formed as $m \cdot n$ bits, and it can be represented as Eq. (5).

$$\begin{aligned} N &= (k_{m-1\ n-1}, \cdots, k_{m-1\ 1}, k_{m-1\ 0}, \cdots, \\ &\quad k_{1\ n-1}, \cdots, k_{1\ 1}, k_{1\ 0}, k_{0\ n-1}, \cdots, k_{0\ 1}, k_{0\ 0})_2 \\ &= K_{m-1} 2^{(m-1)\cdot n} + \cdots + K_2 2^{2n} + K_1 2^n + K_0 \quad (5) \\ &= \sum_{l=0}^{m-1} K_l \cdot 2^{l\cdot n}, \end{aligned}$$

where the range of N is $0 \leq N < (2^n - 1)2^n(2^n + 1)\cdots(2^n + m - 2) = \prod_{l=0}^{m-1}(2^n + l - 1)$, the range of each coefficients, K_l, in Eq. (5) are $0 \leq K_l \leq 2^n - 1$, and $K_l = (k_{l\ n-1}, \cdots, k_{l\ 1}, k_{l\ 0})_2$, ($l \in \{0, 1, \cdots, m-1\}$).

The each residue numbers are converted from the original binary number as follows.

$$\begin{aligned}|N|_{2^n-1} &= |K_0+K_1+K_2+\cdots+K_{m-1}|_{2^n-1} \\ &= \left|\sum_{l=0}^{m-1} K_l\right|_{2^n-1} \qquad (6)\\ |N|_{2^n} &= K_0 \qquad (7)\\ |N|_{2^n+1} &= |K_0-K_1+K_2-\cdots+K_{m-1}|_{2^n+1} \\ &= \left|\sum_{l=0}^{m-1}(-1)^l\cdot K_l\right|_{2^n+1} \qquad (8)\\ |N|_{2^n+2} &= \left|\sum_{l=0}^{m-1}(-1)^l\cdot 2^l\cdot K_l\right|_{2^n+2}\\ &\vdots\\ |N|_{2^n+m-2} &= \left|\sum_{l=0}^{m-1}(-1)^l\cdot(m-2)^l\cdot K_l\right|_{2^n+m-2} \qquad (9)\end{aligned}$$

Example 1: Let $m = 5$, $n = 3$ and $N = (110\ 101\ 100\ 011\ 010)_2 = 27418$. Thus $K_4 = (110)_2 = 6$, $K_3 = (101)_2 = 5$, $K_2 = (100)_2 = 4$, $K_1 = (011)_2 = 3$ and $K_0 = (010)_2 = 2$. Hence $|27418|_{2^3} = K_0 = (010)_2 = 2$, and,

$$\begin{aligned}&|27418|_{2^3-1} = |2+3+4+5+6|_{2^3-1} = 6\\ &|27418|_{2^3+1} = |2-3+4-5+6|_{2^3+1} = 4\\ &|27418|_{2^3+2} = |2-2\cdot 3+2^2\cdot 4-2^3\cdot 5+2^4\cdot 6|_{2^3+2} = 8\\ &|27418|_{2^3+3} = |2-3\cdot 3+3^2\cdot 4-3^3\cdot 5+3^4\cdot 6|_{2^3+3} = 6.\end{aligned}$$

2.3 VLSI Oriented Algorithm

Let the modulus be $2^n+\mu$. For the implementation of the above basic algorithm, the coefficient K_l must need to be multiplied by constant number. The operation needs a lot of hardware resource and increases the computation time. But, by the assumption that $\mu = 2^\alpha$ ($\alpha \in \{0,1,2,\cdots,n-1\}$), the one bit shifting means the multiplication by 2. This operation can save the hardware resource and speed up the conversion time. The μ is defined as 2^α in this paper.

The conversion algorithms for moduli $2^n, 2^n \pm 1$ are Eqs. (6)–(8). And the conversion algorithm for modulus 2^n+2^α is the equation below.

$$\begin{aligned}|N|_{2^n+2^\alpha} &= \left|\sum_{l=0}^{m-1}(-1)^l\cdot 2^{\alpha\cdot l}\cdot K_l\right|_{2^n+2^\alpha} \qquad (10)\\ &= \left|\sum_{l=0}^{m-1}(-1)^l\cdot K'_l\right|_{2^n+2^\alpha} \qquad (11)\end{aligned}$$

Where

$$|K'_l|_{2^n+2^\alpha} = \left|\sum_{i=\alpha}^{n-1} k'_{l\ i}\cdot 2^i\right|_{2^n+2^\alpha} \qquad (12)$$

Let be K''_l as Eq. (13).

$$2^{\alpha\cdot l}\cdot K_l = K''_l = \sum_{i=0}^{\alpha\cdot l+n-1} k''_{l\ i}\cdot 2^i \qquad (13)$$

Then, $k'_{l\ i}$ can be calculated using the following recurrence formula.

$$\begin{aligned}k'_{l\ \alpha\cdot l+n-1-x} &= k''_{l\ \alpha\cdot l+n-1-x} - k'_{l\ \alpha\cdot l+2n-\alpha-1-x}\\ x &= (0,1,\cdots,\alpha\cdot l+n-\alpha-1) \qquad (14)\end{aligned}$$

We call the calculation of $|K'_l|_{2^n+2^\alpha}$ as "pre-conversion process" in this paper. After the pre-conversion process, the binary-to-residue conversion can be implemented by the steps that is similar to the conversion algorithm for moduli $2^n, 2^n \pm 1$.

3. Modulo Addition Algorithm using SD Number Representation

The modulo addition algorithm using SD number representation is briefly shown in this section. Let the modulus be $2^n+\mu$. The algorithm is classified into an algorithm for $\mu \in \{-1,0,1\}$ and an algorithm for $1 < \mu < 2^{n-1}$.

For our proposed binary-to-residue conversion algorithm in this paper, the modulo addition for $-1 \le \mu \le 2^{n-1}$ is needed. So we modify the algorithm for $1 < \mu < 2^{n-1}$ to extend for implementation of the modulo 2^n+2^{n-1} addition.

z formed as $n+1$-bits is the addition result of two variables a and b (n-bits).

$$a+b = z = z_n 2^n + z_{n-1}2^{n-1} + \cdots + z_0 = z_n 2^n + Z_0 \qquad (15)$$

3.1 Addition Algorithm for $\mu \in \{-1,0,1\}$

Eq. (16) can be satisfied for $\mu \in \{-1,0,1\}$.

$$|z|_{2^n+\mu} = Z_0 - z_n\mu \qquad (16)$$

Thus, modulo addition for $\mu \in \{-1,0,1\}$ can be implemented by the manner which the most significant bit of z is connected to the carry input of the least significant bit (end-around-carry technique).

3.2 Addition Algorithm for $1 < \mu < 2^{n-1}$

Eq. (16) is also satisfied for $1 < \mu < 2^{n-1}$. When $1 < \mu < 2^{n-1}$, the calculation method of $|z|_{2^n+\mu}$ is classified into three conditions according to the value of z_n, z_{n-1}.

(case 1) when $z_n = 0$

$$|z|_{2^n+\mu} = Z_0 \qquad (17)$$

(case 2) when $z_n \neq 0$ and $z_n = -z_{n-1}$

$$\begin{aligned}|z|_{2^n+\mu} &= -z_{n-1}2^{n-1} + z_{n-2}2^{n-2} + \cdots + z_0\\ &= (-z_{n-1}, z_{n-2}, \cdots, z_0)_2 \qquad (18)\end{aligned}$$

(case 3) when $z_n \neq 0$ and $z_n \neq -z_{n-1}$

$$\begin{aligned}|z|_{2^n+\mu} &= Z_0 - z_n\mu\\ &= (z_{n-1}, z_{n-2}, \cdots, z_0)_2\\ &\quad + (0, -z_n, -z_n\mu_{n-3}, \cdots, -z_n\mu_0)_2 \qquad (19)\end{aligned}$$

Note that the condition of μ is always $1 < \mu < 2^{n-1}$. The most significant bit of μ is always zero, Thus, the μ can be expressed as $\mu = (0, 1, \mu_{n-3}, \cdots, \mu_0)_2$.

3.3 Addition Algorithm for $\mu = 2^{n-1}$

In Wei's paper, the authors cautioned that new carry is generated from the second SD adder for $\mu = 2^{n-1}$. But using the property of $2^{n-1} = (2^{n-2} + 2^{n-3} + \cdots + 2^0) + 1$, Eq. (19) in (case 3) can be reformed as Eq. (20).

$$\begin{aligned} |z|_{2^n+2^{n-1}} &= (z_{n-1}, z_{n-2}, \cdots, z_0)_2 \\ &\quad + (0, -z_n, -z_n, \cdots, -z_n)_2 - z_n \quad (20) \end{aligned}$$

Namely, z_n as the carry output from the first SD adder is connected to the carry input of least significant bit of the second SD adder, so as not to generate new carry from the second SD adder. Using this method, the modulo adder for $\mu = 2^{n-1}$ can be structured similarly to the modulo adder for $1 < \mu < 2^{n-1}$, except for the carry input of the second SD adder. The computation time of the modulo adder is independent of the bit width of the operands.

Figure. 1 shows the circuit diagram of modulo SD adder (MSDA) for modulus $2^n + 2^{n-1}$.

4. Hardware Implementation and Evaluation

In this section, we show the architecture of the binary-to-residue converter for moduli $2^n - 1, 2^n, 2^n + 2^\alpha$ ($\alpha \in \{n-1, \cdots, 1, 0\}$) based on our proposed algorithm.

To the best of our knowledge, the SD full adder (SDFA) proposed by H. Makino is the fastest one in the voltage-mode CMOS implementation [6]. Thus, in our implementation of the modulo SD adders (MSDA), we design them based on the Makino's SDFA. The MSDAs are described using Verilog-HDL. we apply the design optimization to the MSDAs using Synopsys Design Compiler. Since the input at second SDA is constant number, the logic synthesis tool can produce more optimum gate-level netlists. Table 1 summarizes the performance of the MSDAs. The delay time is obtained by HDL simulation before place-and-routing, under the condition of 0.6μm CMOS technology. Our MSDA designs are much faster than Wei's MSDA designs.

Table 1. Performances of MSDAs.

Modulus	Transistors	Delay (ns)
1023 ($2^{10} - 1$)	740	0.97
1025 ($2^{10} + 1$)	744	1.03
1032 ($2^{10} + 2^3$)	1378	2.63
1056 ($2^{10} + 2^5$)	1348	2.51
1152 ($2^{10} + 2^7$)	1294	2.18
1536 ($2^{10} + 2^9$)	1474	2.71

As shown in Fig. 2, the proposed binary-to-residue converter can be constructed using MSDAs as components. Figure 2(a) and (b) show the converters for modulus $2^n - 1$ and $2^n + 1$, respectively. As shown in Fig. 2(a) and (b), a binary tree of modulo $2^n - 1$ ($2^n + 1$) SD adders can be constructed for the modulo $2^n - 1$ ($2^n + 1$) sum of the K_l in Eq. (6) (Eq. (8)). Therefore, the delay time of the proposed converter for modulus $2^n - 1$ ($2^n + 1$) is time proportional to $\log_2 m$.

The converter for modulus $2^n + 2^\alpha$ is shown in Fig. 2(c). The pre-converter (written as "pre-CONV" in Fig. 2(c)) is the circuit to calculate $|K_l'|_{2^n+2^\alpha}$ in Eq. (12). Eq. (14) shows that the modulo addition of ($\left\lceil \frac{n}{n-\alpha} \right\rceil$) is needed for the pre-conversion process. Thus, the delay time of the pre-conversion process is time proportional to $\log_2 \left\lceil \frac{n}{n-\alpha} \right\rceil$ by using the MSDA tree. After pre-conversion process, as shown in Fig. 2(c), the binary tree of the MSDAs for modulo $2^n + 2^\alpha$ can be constructed for the modulo $2^n + 2^\alpha$ sum of the K_l' in Eq. (11). Therefore, the proposed converter for modulus $2^n + 2^\alpha$ can perform the conversion in time proportional $\log_2 \left(\left\lceil \frac{n}{n-\alpha} \right\rceil \cdot m \right)$.

The performances of the proposed converters in terms of the delay time and the number of transistors are summarized in table 2 and 3. Table 2 shows the comparison between the proposed converters and the ordinary converters using CPAs (Carry Propagation Adders), for modulus set $2^n, 2^n \pm 1$ ($m = 3$). From the simulation results summarized in table 2, the proposed binary-to-residue converter can achieve high speed conversion compared with ordinary converter using CPAs. Table 3 shows the simulation results of the proposed converters for modulus $2^n + 2^\alpha$. They are based on the delay time expected by our algorithm. On the other hand, in order to implement the binary-to-residue conversion using look-up table, $2^{m \cdot n} \cdot n$ bits ROM is needed for one modulus. When the bit length of the modulus is large, the implementation seems not to be real. And the conversion using look-up table is not said to be high speed conversion.

5. Conclusion

This paper described the novel converter to implement high-speed binary-to-residue conversion for moduli $2^n - 1, 2^n, 2^n + 2^\alpha$ ($\alpha \in \{0, 1, \cdots, n-1\}$) without using look-up table. The conversion delay time of the proposed converter for moduli $2^n \pm 1$ is proportional to $\log_2 m$. And the delay time of the converter for moduli $2^n + 2^\alpha$ is proportional to $\log_2 \left(\left\lceil \frac{n}{n-\alpha} \right\rceil \cdot m \right)$. Thus, the proposed binary-to-residue converter can achieve high speed conversion compared with ordinary converter using CPAs for moduli $2^n \pm 1$. In the implementation of the converter for moduli $2^n + 2^\alpha$, when the bit length of the modulus is large, it can be implemented using less hardware resource. Consequently, the proposed converter is high-speed and efficient circuit to convert into the residue SD systems proposed by S. Wei.

References

[1] 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, pp. 2056–2064, Dec. 2000.

[2] B. Guan and E. V. Jones, "Fast conversion between

Table 2. Performances of binary-to-residue converters (modulus set $2^n \pm 1$).

Modulus	Proposed converter		The converter using CPAs	
	Transistors	Delay (ns)	Transistors	Delay (ns)
$511(2^9-1)$	792	1.78	572	5.17
$1023(2^{10}-1)$	880	1.78	638	5.55
$2047(2^{11}-1)$	968	1.78	698	5.94
$513(2^9+1)$	794	1.73	540	6.94
$1025(2^{10}+1)$	882	1.73	602	7.70
$2049(2^{11}+1)$	970	1.73	662	8.47

Table 3. Performances of binary-to-residue converters (modulus set $2^n \pm 1, 2^n + 2^\alpha$).

m	Modulus	Transistors	Delay (ns)
4	$1023(2^{10}-1)$	1280	1.71
	$1025(2^{10}+1)$	1938	2.01
	$1032(2^{10}+2^3)$	2916	4.96
	$1056(2^{10}+2^5)$	2467	4.83
	$1152(2^{10}+2^7)$	2792	7.11
	$1536(2^{10}+2^9)$	2342	8.30
8	$1023(2^{10}-1)$	3312	2.78
	$1025(2^{10}+1)$	4592	3.25
	$1032(2^{10}+2^3)$	6697	7.39
	$1056(2^{10}+2^5)$	5465	8.18
	$1152(2^{10}+2^7)$	5926	9.50
	$1536(2^{10}+2^9)$	4870	10.78

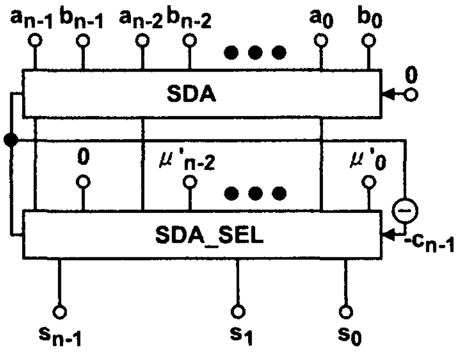
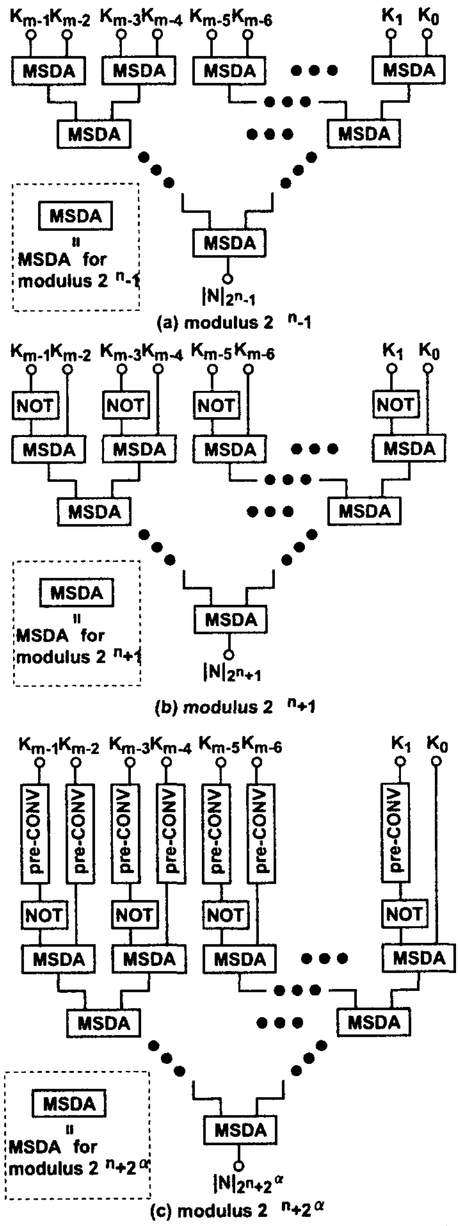


Figure 1. Modulo SD adders (MSDAs).



$K_{m-1}K_{m-2}$ $K_{m-3}K_{m-4}$ $K_{m-5}K_{m-6}$ K_1 K_0
MSDA
MSDA = MSDA for modulus 2^n-1
$|N|_{2^n-1}$
(a) modulus 2^n-1
NOT
MSDA = MSDA for modulus 2^n+1
$|N|_{2^n+1}$
(b) modulus 2^n+1
pre-CONV
MSDA = MSDA for modulus 2^n+2^α
$|N|_{2^n+2^\alpha}$
(c) modulus 2^n+2^α

Figure 2. Binary-to-residue converters for moduli ($2^n \pm 1, 2^n + 2^\alpha$).

binary and residue numbers," Electron. Lett., vol. 24, no. 19, pp. 1195–1197, Sept. 1988.

[3] P. V. Ananda Mohan, "Novel design for binary to RNS converters," Proc. IEEE ISCAS, pp. 357–360, 1994.

[4] S. Perumal and R. Siferd, "Pipelined 50 MHz CMOS ASIC for 32 bit binary to residue conversion and residue to binary conversion," IEEE Int. ASIC Conference & Exhibit (ASIC'94), Sept. 1994.

[5] M. Syuto, E. Satake, K. Tanno and O. Ishizuka, "A high-speed binary to residue converter using a signed-digit number representation," IEICE Trans. Inf. and Syst. vol. E85-D, no. 5, pp. 903–905, May 2002.

[6] H. Makino, Y. Nakase, H. Suzuki, H. Morinaka, H. Shinohara and K. Mashiko, "An 8.8-ns 54 × 54-bit multiplier with high speed redundant binary architecture," IEEE J. Solid-State Circuits, vol. 31, pp. 773–782, June 1996.