International Scholarly Research Network
ISRN Signal Processing
Volume 2011, Article ID 272768, 7 pages
doi:10.5402/2011/272768

Research Article

New Adder-Based RNS-to-Binary Converters for the $\{2^{n+1}+1, 2^{n+1}-1, 2^n\}$ Moduli Set

Kazeem Alagbe Gbolagade

Department of Computer Science, Faculty of Mathematical Sciences, University for Development Studies, P.O. Box 24, Navrongo, Ghana

Correspondence should be addressed to Kazeem Alagbe Gbolagade, gkazy1@yahoo.com

Received 24 March 2011; Accepted 12 April 2011

Academic Editors: C.-C. Hu and P. Szolgay

We investigate Residue Number System (RNS) to binary conversion, which is an important issue concerning the utilization of RNS numbers in Digital Signal Processing (DSP) applications. We propose two new reverse converters for the moduli set $\{2^{n+1}+1, 2^{n+1}-1, 2^n\}$. First, we simplify the Chinese Remainder Theorem (CRT) to obtain a reverse converter that uses mod-$(2^{n+1}-1)$ operations instead of mod-$(2^{n+1}+1)(2^{n+1}-1)$ operations required by other state-of-the-art equivalent converters. Next, we further reduce the hardware complexity by making the resulting reverse converter architecture adder based. Two hybrid Cost-Efficient (CE) and Speed-Efficient (SE) reverse converters are proposed. These two hybrid converters are obtained by combining the best state-of-the-art converter with the newly introduced area-delay efficient scheme. The proposed hybrid CE converter outperforms the best state-of-the-art CE converter in terms of delay with similar area cost. Additionally, the proposed hybrid SE converter requires less area cost with smaller delay when compared to the best state-of-the-art equivalent SE converter.

1. Introduction

The presence of carry chains in conventional weighted number systems such as binary or decimal number systems often limits the efficiency of computer arithmetic operations. Residue Number System (RNS) is a number system having an attractive carry-free property, which has proved to be highly useful in many Digital Signal Processing (DSP) applications requiring high-speed computations [1, 2]. RNS also has the following inherent features due to its carry-free property: modularity, parallelism, and fault tolerance. In order not to offset the speed gained in RNS operations, a fast RNS-to-binary converter is required. The complexity as well as the efficiency of RNS to binary converter is determined by the moduli choice and by the conversion algorithm. Many different choices of moduli sets are available for RNS to represent the binary numbers in a certain range. Three moduli sets have been actively investigated, for example, $\{2^n-1, 2^n, 2^n+1\}$ [3], $\{2^n, 2^n-1, 2^{n-1}-1\}$ [4], $\{2n+1, 2n-1, 2n\}$ [5], and $\{2n+2, 2n+1, 2n\}$ [6]. The dynamic range of these moduli sets is not sufficient for applications requiring larger dynamic range. Thus, the moduli set $\{2^n-1, 2^n, 2^n+1\}$, which is the most popular length three moduli set, has been enhanced to $\{2^{n+1}+1, 2^{n+1}-1, 2^n\}$ [7] and $\{2^{n+1}-1, 2^n, 2^n-1\}$ [8], to mention just a few. Generally speaking, RNS-to-binary conversion is either based on the Chinese Remainder Theorem (CRT) [1, 2, 6] or the Mixed Radix Conversion (MRC) [9].

In this paper, we propose two new hybrid Cost Efficient (CE) and Speed Efficient (SE) reverse converters for the moduli set $\{2^{n+1}+1, 2^{n+1}-1, 2^n\}$. We obtain the two hybrid converters by combining the converters in [7] with the newly introduced area-delay efficient converters. First, we simplify the CRT to obtain a reverse converter that uses mod-$(2^{n+1}-1)$ operations instead of mod-$(2^{n+1}+1)(2^{n+1}-1)$ operations required by the proposal in [7]. Next, we further reduce the hardware complexity by making the resulting reverse converter architecture adder based. Basically, the newly introduced scheme is made up of two Carry-Save Adders (CSAs) and $(n+1)$-bit Carry Propagate Adder (CPA). The proposed CE converter outperforms the one in [7] in terms of delay with similar area cost. Additionally, the

proposed SE converter requires less area cost and smaller conversion delay when compared to the one in [7].

The rest of the paper is organised as follows: Section 2 presents the necessary background information. In Section 3, we describe the proposed algorithm. Section 4 presents the hardware realization of the proposed algorithm, and a comparison with the state-of-the-art reverse converters is provided in Section 5. The paper is concluded in Section 6.

2. Background

RNS is defined in terms of a set of relatively prime moduli set $\{m_i\}_{i=1,k}$ such that $\gcd(m_i, m_j) = 1$ for $i \neq j$, where gcd means the greatest common divisor of m_i and m_j, while $M = \prod_{i=1}^{k} m_i$, is the dynamic range. The residues of a decimal number X can be obtained as $x_i = |X|_{m_i}$, thus X can be represented in RNS as $X = (x_1, x_2, x_3, \ldots, x_k)$, $0 \leq x_i < m_i$. This representation is unique for any integer $X \in [0, M-1]$. We note here that in this paper we use $|X|_{m_i}$ to denote the X mod m_i operation.

For a moduli set $\{m_i\}_{i=1,k}$ with the dynamic range $M = \prod_{i=1}^{k} m_i$, the residue number $(x_1, x_2, x_3, \ldots, x_k)$ can be converted into the decimal number X, according to the Chinese Reminder theorem, as follows [10]:

$$X = \left| \sum_{i=1}^{k} M_i \left| M_i^{-1} x_i \right|_{m_i} \right|_M, \tag{1}$$

where $M = \prod_{i=1}^{k} m_i$, $M_i = M/m_i$, and M_i^{-1} is the multiplicative inverse of M_i with respect to m_i.

This general scheme can be actually simplified when certain moduli sets of interests like $\{2^{n+1}+1, 2^{n+1}-1, 2^n\}$ are utilized. For this moduli set efficient converters have been presented in [7]. In [2], the New CRT proposed in [3] is formulated as follows:

$$X = x_1 + m_1 \left| w_1 x_1 + \sum_{i=2}^{k} w_i \left| N_i^{-1} x_i \right|_{m_i} \right|_{m_2 \cdots m_k}, \tag{2}$$

where $k > 1$, $w_i = N_i/m_1$ and N_i^{-1} is the multiplicative inverse of N_i with respect to m_i. For the moduli set $\{2^{n+1}+1, 2^{n+1}-1, 2^n\}$, the following relation, based on the New CRT represented by (2) has been presented in [7]:

$$X = x_1 + 2^n \left| 2^{n+2}(x_2 - x_1) + 2^{n+1}(2^{n+1}+1)(x_3 - x_2) \right|_{2^{2n+2}-1}. \tag{3}$$

Given that the residues (x_1, x_2, x_3) have the following binary representations:

$$\begin{aligned} x_1 &= (x_{1,n-1} x_{1,n-2} \cdots x_{1,1} x_{1,0}), \\ x_2 &= (x_{2,n+1} x_{2,n} \cdots x_{2,1} x_{2,0}), \\ x_3 &= (x_{3,n} x_{3,n-2} \cdots x_{3,1} x_{3,0}). \end{aligned} \tag{4}$$

Equation (3) was further simplified to obtain

$$X = x_1 + 2^n \left| v_1' + v_{21} + v_3 \right|_{2^{2n+2}-1}, \tag{5}$$

where

$$\begin{aligned} v_1' &= \underbrace{\overline{x}_{1,n-1} \cdots \overline{x}_{1,1} \overline{x}_{1,0}}_{n} \underbrace{\overline{x}_{2,n+1} \cdots \overline{x}_{2,1} \overline{x}_{2,0}}_{n+1}, \\ v_{21} &= \underbrace{x_{2,n} \cdots x_{2,1} x_{2,0}}_{n+1} \underbrace{0 \cdots 00}_{n} x_{2,n+1}, \\ v_3 &= \underbrace{x_{3,n} \cdots x_{3,1} x_{3,0}}_{n+1} \underbrace{x_{3,n} \cdots x_{3,1} x_{3,0}}_{n+1}. \end{aligned} \tag{6}$$

Given that the moduli set $\{2^{n+1}+1, 2^{n+1}-1, 2^n\}$ is desirable, can we obtain a more effective reverse converter when compared to the ones in [7]? In the following section, we present two effective reverse converters for the moduli set $\{2^{n+1}+1, 2^{n+1}-1, 2^n\}$ by first simplifying (1).

3. Proposed Algorithm

Given the RNS number (x_1, x_2, x_3) with respect to the moduli set $\{m_1, m_2, m_3\}$ in the form $\{2^{n+1}+1, 2^{n+1}-1, 2^n\}$, the proposed algorithm computes the decimal equivalent of this RNS number based on a further simplification of the well-known traditional CRT. First, we show that the computation of the multiplicative inverses can be eliminated for this moduli set resulting into memoryless reverse converters. Next, we obtain reverse converters that utilize modulo-$(2^{n+1}-1)$ operations instead of modulo-$(2^{n+1}+1)(2^{n+1}-1)$ operations required by the state-of-the-art equivalent converters. We further reduce the hardware complexity by obtaining adder-based reverse converters.

Theorem 1. *Given the moduli set $\{m_1, m_2, m_3\}$ with $m_1 = 2^{n+1}+1, m_2 = 2^{n+1}-1, m_3 = 2^n$, the following holds true:*

$$\left| (m_1 m_2)^{-1} \right|_{m_3} = 2^n - 1, \tag{7}$$

$$\left| (m_2 m_3)^{-1} \right|_{m_1} = 1, \tag{8}$$

$$\left| (m_1 m_3)^{-1} \right|_{m_2} = 1. \tag{9}$$

Proof. If it can be demonstrated that $|(2^n-1) \times (m_1 m_2)|_{m_3} = 1$, then $(2^n - 1)$ is the multiplicative inverse of $(m_1 m_2)$ with respect to m_3. $|(2^n-1) \times (m_1 m_2)|_{m_3}$ is given by

$$\begin{aligned} &\left| (2^{n+1}+1)(2^{n+1}-1)(2^n-1) \right|_{2^n} \\ &= \left| (2^{2n+2}-1)(2^n-1) \right|_{2^n} \\ &= \left| 2^{3n+2} - 2^n - 2^{2n+2} + 1 \right|_{2^n} \\ &= \left| \left| 2^{3n+2} \right|_{2^n} - |2^n|_{2^n} - \left| 2^{2n+2} \right|_{2^n} + 1 \right|_{2^n} \\ &= |0 - 0 - 0 + 1|_{2^n} = 1, \end{aligned} \tag{10}$$

thus (7) holds true.

In the same way if $|1 \times (m_2 m_3)|_{m_1} = 1$, then 1 is the multiplicative inverse of $(m_2 m_3)$ with respect to m_1. $|1 \times (m_2 m_3)|_{m_1}$ is given by: $|(2^{n+1}-1)(2^n)|_{2^{n+1}+1} = |2^{2n+1} - 2^n|_{2^{n+1}+1} = |1|_{2^{n+1}+1} = 1$, thus (8) holds true.

Again, if $|1 \times (m_1 m_3)|_{m_2} = 1$, then 1 is the multiplicative inverse of $(m_1 m_3)$ with respect to m_2. $|1 \times (m_1 m_3)|_{m_2}$ is given by

$$\left|(2^{n+1}+1)(2^n)\right|_{2^{n+1}-1} = \left|2^{2n+1}+2^n\right|_{2^{n+1}-1} = |1|_{2^{n+1}-1} = 1, \tag{11}$$

thus (9) holds true. □

The following important relations are used in the subsequent theorem: Given the moduli set $\{m_1, m_2, m_3\}$ with $m_1 = 2^{n+1}+1, m_2 = 2^{n+1}-1, m_3 = 2^n$, the following holds true:

$$m_1 = m_2 + 2, \tag{12}$$

$$m_1 = 2m_3 + 1, \tag{13}$$

$$m_2 = 2m_3 - 1. \tag{14}$$

Theorem 2. *The decimal equivalent of the RNS number (x_1, x_2, x_3) with respect to the moduli set $\{m_1, m_2, m_3\}$ in the form $\{2^{n+1}+1, 2^{n+1}-1, 2^n\}$ is computed as follows:*

$$\begin{aligned} X &= 2m_3(x_3 - x_1) + x_3 \\ &\quad + m_3 m_1 |x_1 - 2x_3 + x_2|_{m_2}. \end{aligned} \tag{15}$$

Proof. Equation (1) for $k = 3$ is given by

$$X = \left| \sum_{i=1}^{3} M_i \left| M_i^{-1} x_i \right|_{m_i} \right|_M. \tag{16}$$

By substituting (7), (8), and (9) into (16) we obtain the following:

$$\begin{aligned} X &= \left|(m_2 m_3)x_1 + (m_1 m_3)x_2 + (m_1 m_2)(m_3 - 1)x_3\right|_M, \\ &= \left|(m_2 m_3)x_1 + (m_1 m_3)x_2 + M - m_1 m_2 x_3\right|_M, \\ &= \left|(m_2 m_3)x_1 + (m_1 m_3)x_2 - m_1 m_2 x_3\right|_M, \end{aligned} \tag{17}$$

Substituting (12) in the above equation, we obtain

$$\begin{aligned} X &= |m_3(m_1 - 2)x_1 + m_1 m_3 x_2 - m_1 m_2 x_3|_M, \\ &= \left|m_1 m_3 x_1 - 2m_3 x_1 + m_1 m_3 x_2 - m_1 m_2 x_3\right|_M, \\ &= -2m_3 x_1 + \left|m_1 m_3 x_1 + m_1 m_3 x_2 - m_1 m_2 x_3\right|_{m_1 m_2 m_3}. \end{aligned} \tag{18}$$

Equation (18) can be further simplified by using the following lemma presented in [11]:

$$|a m_1|_{m_1 m_2} = m_1 |a|_{m_2}. \tag{19}$$

Applying (19), (18) becomes

$$X = -2m_3 x_1 + m_1 |m_3 x_1 + m_3 x_2 - m_2 x_3|_{m_2 m_3}. \tag{20}$$

Using (14) in (20), we obtain

$$\begin{aligned} X &= -2m_3 x_1 \\ &\quad + m_1 \left|m_3 x_1 + m_3 x_2 - x_3(2m_3 - 1)\right|_{m_2 m_3}, \\ &= -2m_3 x_1 \\ &\quad + m_1 \left|m_3 x_1 + m_3 x_2 - 2m_3 x_3 + x_3\right|_{m_2 m_3}, \\ &= -2m_3 x_1 + m_1 x_3 \\ &\quad + m_1 \left|m_3(x_1 - 2x_3 + x_2)\right|_{m_2 m_3}. \end{aligned} \tag{21}$$

Applying (19), (21) becomes

$$\begin{aligned} X &= -2m_3 x_1 + m_1 x_3 \\ &\quad + m_1 m_3 \left|x_1 - 2x_3 + x_2\right|_{m_2}. \end{aligned} \tag{22}$$

Using (13) in (22), we obtain

$$\begin{aligned} X &= -2m_3 x_1 + x_3(2m_3 + 1) \\ &\quad + m_1 m_3 \left|x_1 - 2x_3 + x_2\right|_{m_2}, \\ &= 2m_3(x_3 - x_1) + x_3 \\ &\quad + m_1 m_3 \left|x_1 - 2x_3 + x_2\right|_{m_2}, \end{aligned} \tag{23}$$

thus, (15) holds true. □

We reduce the hardware complexity by further simplifying (15) using the following properties [7].

Property 1. Modulo $(2^s - 1)$ multiplication of a residue number by 2^t, where s and t are positive integers, is equivalent to t bit circular left shifting.

Property 2. Modulo $(2^s - 1)$ of a negative number is equivalent to the one's complement of the number, which is obtained by subtracting the number from $(2^s - 1)$.

Assumption 1. The hardware complexity is reduced based on the assumption that $x_1 < 2^{n+1}$ always holds true.

Based on the given assumption, x_1, which is $(n+2)$-bit binary number can now be represented like an $(n+1)$-bit number. Therefore, the residues (x_1, x_2, x_3) have binary representations as follow:

$$\begin{aligned} x_1 &= (x_{1,n} x_{1,n-1} \cdots x_{1,1} x_{1,0}), \\ x_2 &= (x_{2,n} x_{2,n-1} \cdots x_{2,1} x_{2,0}), \\ x_3 &= (x_{3,n-1} x_{3,n-2} \cdots x_{3,1} x_{3,0}). \end{aligned} \tag{24}$$

Then (15) is further simplified using the following theorem.

Theorem 3. *Provided that $x_1 < 2^{n+1}$ holds true, the binary equivalent of the RNS number (x_1, x_2, x_3) with respect to the moduli set $\{m_1, m_2, m_3\}$ in the form $\{2^{n+1}+1, 2^{n+1}-1, 2^n\}$ is computed as follows:*

$$X = A_0 + 2^{2n+1}A_1 + 2^n A_1, \tag{25}$$

where

$$\begin{aligned} A_1 &= |x_1 + x_2 + u_3|_{2^{n+1}-1}, \\ A_0 &= u_1 + u_2, \\ u_1 &= \underbrace{(x_{3,n-1}x_{3,n-2}\cdots x_{3,0})}_{n}0\underbrace{(x_{3,n-1}x_{3,n-2}\cdots x_{3,0})}_{n}, \\ u_2 &= \underbrace{(\overline{x}_{1,n+1}\overline{x}_{1,n}\cdots\overline{x}_{1,0}11\cdots1)}_{2n+2}, \\ u_3 &= \underbrace{(\overline{x}_{3,n-1}\overline{x}_{3,n-2}\cdots\overline{x}_{3,0}1)}_{n+1}. \end{aligned} \tag{26}$$

Proof. We need to show that (15) can be presented as (25), and that the values of $A_0, A_1, \{u_i\}_{i=1,3}$ are valid. It should be noted from (15) that $2m_3(x_3 - x_1) + x_3 = 2^{n+1}x_3 + x_3 - 2^{n+1}x_1 = u_1 + u_2$ and u_1 can be represented as

$$\begin{aligned} u_1 &= 2m_3x_3 + x_3 \\ &= 2^{n+1}x_3 + x_3 \\ &= 2^{n+1}\underbrace{(x_{3,n-1}x_{3,n-2}\cdots x_{3,1}x_{3,0})}_{n} \\ &\quad + \underbrace{(x_{3,n-1}x_{3,n-2}\cdots x_{3,1}x_{3,0})}_{n} \\ &= \underbrace{(x_{3,n-1}x_{3,n-2}\cdots x_{3,1}x_{3,0})}_{n}\underbrace{00\cdots0}_{n+1} \\ &\quad + \underbrace{(x_{3,n-1}x_{3,n-2}\cdots x_{3,1}x_{3,0})}_{n} \\ &= \underbrace{(x_{3,n-1}x_{3,n-2}\cdots x_{3,0})0(x_{3,n-1}x_{3,n-2}\cdots x_{3,0})}_{2n+1}. \end{aligned} \tag{27}$$

Also, we represent u_2 as

$$\begin{aligned} u_2 &= -2^{n+1}x_1 \\ &= -2^{n+1}\underbrace{(x_{1,n+1}x_{1,n}\cdots x_{1,0})}_{n+1} \\ &= -\left(\underbrace{(x_{1,n+1}x_{1,n}\cdots x_{1,0})}_{n+1}\underbrace{(00\cdots0)}_{n+1}\right) \\ &= \left(\underbrace{\overline{x}_{1,n+1}\overline{x}_{1,n}\cdots\overline{x}_{1,0}11\cdots1}_{2n+2}\right). \end{aligned} \tag{28}$$

Next, we convert each term in $|x_1 + x_2 + u_3|_{2^{n+1}-1}$ to an $(n+1)$ bit binary number. No manipulation is required for x_1 since it is treated like an $(n+1)$-bit number and also x_2 is already an $(n+1)$-bit number. The only term to be manipulated is $-2x_3$ and this is carried out as follows:

$$\begin{aligned} u_3 &= -2x_3 \\ &= -2\underbrace{(x_{3,n-1}x_{3,n-2}\cdots x_{3,0})}_{n} \\ &= -\underbrace{(x_{3,n-1}x_{3,n-2}\cdots x_{3,0}0)}_{n+1} \\ &= \underbrace{(\overline{x}_{3,n-1}\overline{x}_{3,n-2}\cdots\overline{x}_{3,0}1)}_{n+1}. \end{aligned} \tag{29}$$

□

However, if the condition $x_1 < 2^{n+1}$ does not hold true, the converter in [7], represented by (5), is utilized.

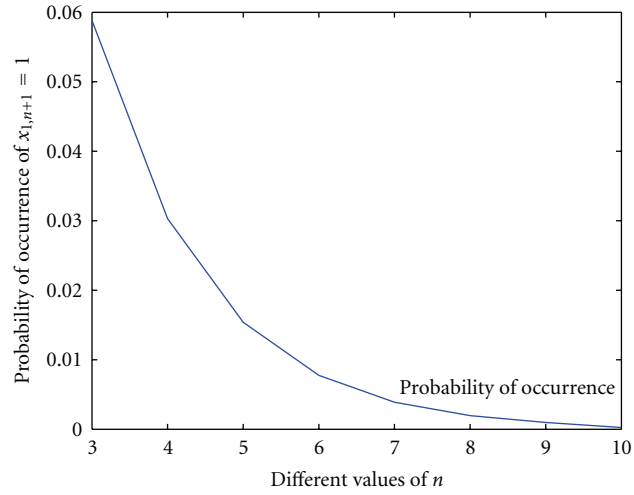

Figure 1: Probability of occurrence of $x_{1,n} = 1$ versus n.

Table 1: Design example.

CSA1	CPA	CSA2
$x_1 : 0010$	$s_1 : 1111$	$B_1 : 00000110011$
$x_2 : 0100$	$c_1 : 0000$	$B_2 : 00000000000$
$u_3 : 1001$	$C_r : \overline{1111}$	$u_2 : 11111011111$
$s_1 : \overline{1111}$	$C_r : 0000$	$s_3 : \overline{11111101100}$
$c_1 : 0000$		$c_3 : 00000100110$
		$R_f : 00000010011$

4. Hardware Realization

The hardware implementations of the proposed reverse converters are based on the combination of (25) and (5). We propose two converters based on this approach, termed "hybrid approach". The hardware realizations of the proposed schemes are depicted by Figures 2 and 3. In Figure 2 (the proposed hybrid cost-efficient converter), x_1 is first input into the system. Given that the Most Significant Bit (MSB) of x_1 is $x_{1,n}$, then $x_{1,n}$ is compared with "1". If $x_{1,n} = 1$, (5) is utilized just as outlined in [7] otherwise the hardware

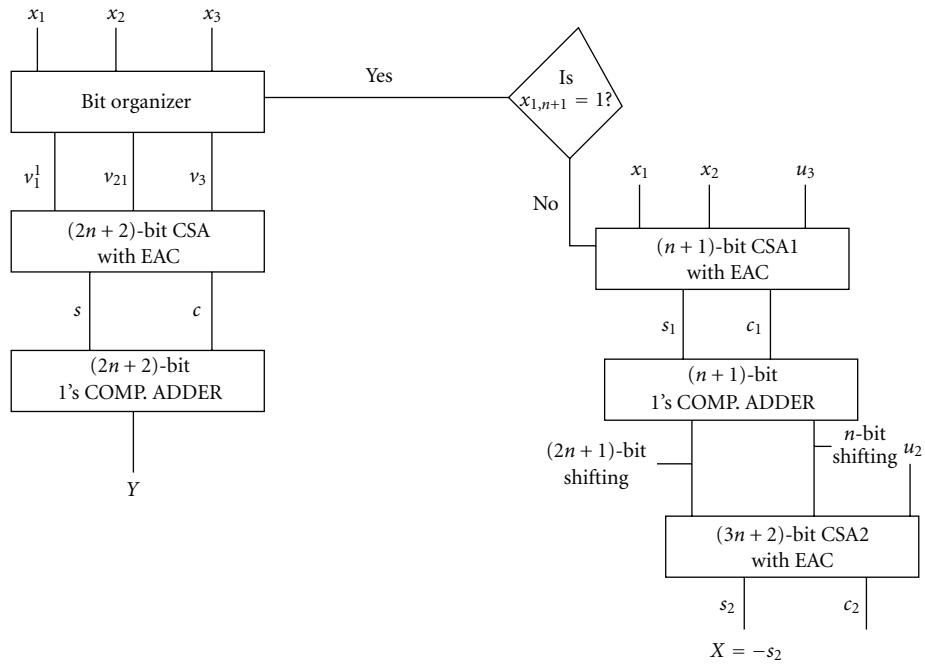

Figure 2: Proposed hybrid cost-efficient RNS to binary converter.

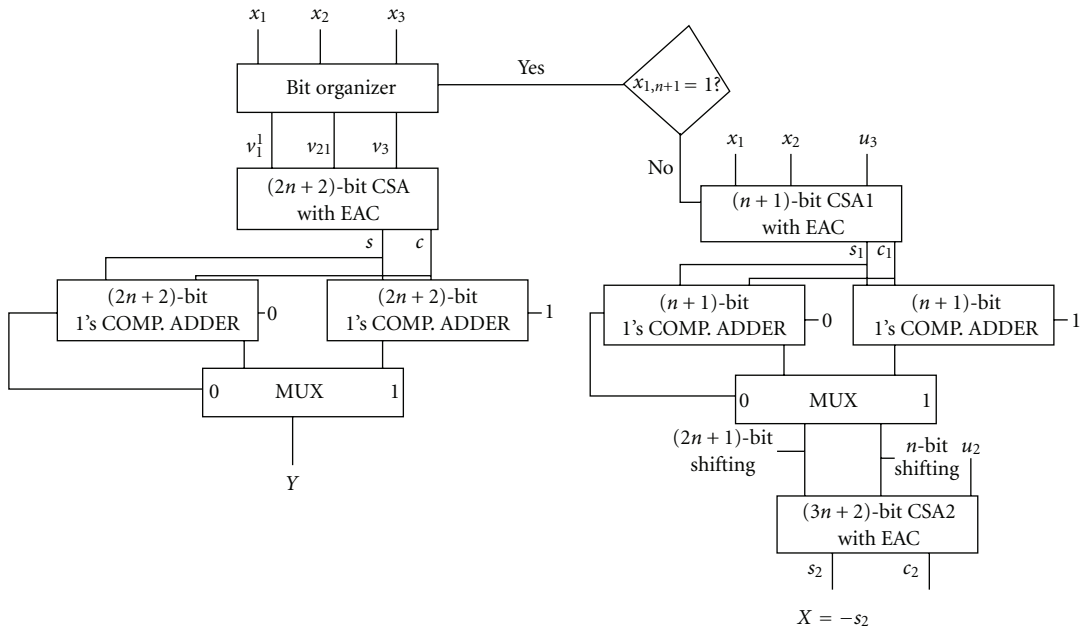

Figure 3: Proposed hybrid speed-efficient RNS to binary converter.

realization follows (25). It should be noted that the overhead for comparison is approximately zero as only the MSB of x_1 needs to be compared with a 1. The condition $x_{1,n} = 1$ holds true only in very few cases. The probability of occurrence of this condition is denoted by p, and it can be seen from Figure 1 that p approaches zero as n increases. This is fully explained in the next section. However, if $x_{1,n} = 0$ (which occurs most of the time) holds true, the hardware realization is as follows. The operands x_1, x_2, and u_3 in (25) are added using CSA1 producing s_1 and c_1, which are in turn added using a one's complement adder (this is equivalent to a CPA with End Around Carry (EAC)). Suppose that B_1 and B_2 are,

Table 2: Area-delay comparison.

Converters	CE [7]	Proposed CE	SE [7]	Proposed SE
FA	$3n+4$	$3n+3$	$5n+6$	$4n+4$
HA	n	$2n$	n	$2n$
OR/NOT	$6n+4$	$3n+3$	$2n+2$	$6n+4$
Multiplexer	—	—	1	1
Delay	$(4n+5)t_{FA}$	$(2n+4)t_{FA}$	$(2n+3)t_{FA}+t_{MUX}$	$(n+3)t_{FA}+t_{MUX}$

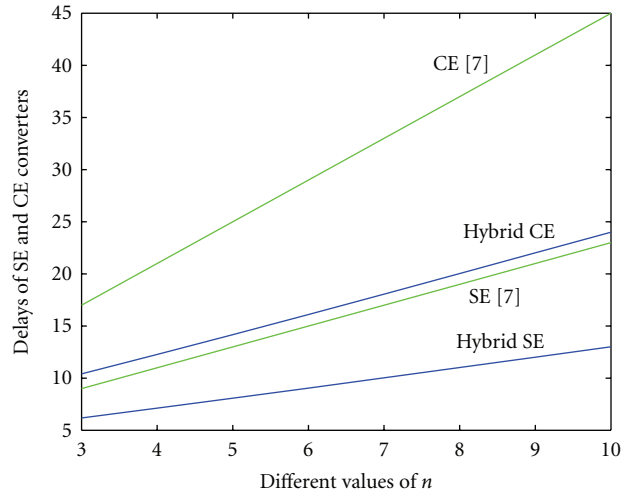

Figure 4: Delay comparison: hybrid converters versus converters [7].

Table 3: Synthesized results: area-delay comparison.

n	Our area	Our delay in (n/s)	Area [7]	Delay [7] in (n/s)
3	14	17.574	26	28.083
4	19	22.808	30	28.289
5	23	24.419	40	32.335
6	25	24.874	47	39.975
7	32	30.166	54	42.598
8	36	30.991	48	46.504
9	37	30.326	66	50.154
10	44	38.375	69	50.700

respectively, used to store the results of the $2n+1$ and n-bit right shifting of the one's complement adder. Since u_1 is a $2n+1$ bit number, it can be concatenated with B_1 with no computational hardware. The second operand B_2 is a $(2n+1)$-bit number with n-bit of zeros. B_2 must be converted to a $(3n+2)$-bit number by appending $(n+1)$-bit of zeros to its MSB part. The third operand u_2, which is a $(2n+2)$-bit number, is also made a $(3n+2)$-bit number by appending n-bit of ones to its MSB part. B_1, B_2, and u_2 are all now $(3n+2)$-bit numbers and are to be added using CSA2 yielding s_2 and c_2. It should be noted that $(2n+1)$ bits of the Full Adders (FAs) in CSA2 are reduced to Half Adders (HAs). The final result is supposed to be obtained by a CPA but the final result (obtained by means of simulation) is always the same as inverting s_2. Thus, the final CPA is eliminated.

On the other hand, Figure 2 depicts the hardware realization of the proposed hybrid speed efficient converter. Just like the proposed hybrid CE converter, hybrid SE converter is also made up of the same level of CSAs, but the only difference is that two CPAs are utilized in parallel instead of the 1's complement adders in Figure 2. Consequently, the conversion time is significantly reduced.

Design Example. Given the moduli set $\{2^{n+1}+1, 2^{n+1}-1, 2^n\}$, where $n = 3$, convert the residue number $(x_1, x_2, x_3) = (2, 4, 3)$ to binary.

In order to obtain the required binary equivalent, (25) is applied as given in Table 1 where CSA1 represents the first CSA in Theorem 3. The CPA then adds s_1 and c_1 producing C_r. We then obtain B_1, B_2, and u_2 as already expalined above. CSA2 is used to add B_1, B_2, and u_2 producing s_2 and c_2. It can be seen from Table 1 that $R_f = -s_2$. Thus, the final CPA can be eliminated, and consequently the conversion delay and the area cost are significantly reduced.

5. Performance Evaluation

The performances of the proposed converters are evaluated in terms of area cost and conversion delay. Two efficient reverse converters have been proposed. The performance comparisons of these converters and the best state-of-the-art converter in [7] are presented in Tables 2, 3, and 4. From Table 2, it can be seen that the proposed hybrid CE outperforms the CE in [7] in terms of delay with slightly lesser or similar area cost (whenever $x_{1,n} = 0$, the same result otherwise). With the same condition, the proposed hybrid SE converter outperforms the SE converter in [7] in terms of both speed and area. Table 4 shows the occurrence probability and other associated parameters. We note here that in Table 4, the following notations are utilized: n is an integer value that determines various dynamic range requirements, c is the number of times $x_{1,n} = 1$ occurs within the dynamic range M of the system, p is the probability of occurrence of the condition $x_{1,n} = 1$ within the dynamic range of the system (p is computed by $p = c/M$), hybrid CE stands for the delay of the hybrid CE converter, CE [7] stands for the delay of the CE converter in [7], and hybrid SE stands for the delay of the hybrid SE converter while SE [7] stands for the delay of the SE converter in [7]. As shown in Table 4, p reduces as the dynamic range increases (i.e., as n increases). For example, when $n = 3$, $p = 0.058824$ whereas $p = 0.000244$ when $n = 10$. This implies, as can also be deduced from Figure 1, that the occurrence probability

Table 4: Simulation results: showing the occurrence probability and other associated parameters.

n	c	M	p	Hybrid CE	CE [7]	Hybrid SE	SE [7]
3	120	2040	0.058824	10.4118	17	6.1765	9
4	496	16368	0.030303	12.2727	21	7.1212	11
5	2016	131040	0.015385	14.1692	25	8.0769	13
6	8128	1048512	0.007752	16.1008	29	9.0465	15
7	32640	8388480	0.003891	18.0584	33	10.0272	17
8	130816	67108608	0.001949	20.0331	37	11.0156	19
9	523776	536870400	0.000976	22.0185	41	12.0088	21
10	1048000	4295000000	0.000244	24.0051	45	13.0024	23

approaches zero as n continues to grow. The delays of hybrid CE and SE converters and that of the CE and SE converters in [7] are depicted by Figure 4. It can be easily seen from Figure 4 that the rate of growth of the conversion delay is comparably very small in the hybrid SE. Another interesting thing to note here is that the proposed CE converter and the SE converter in [7] have nearly equal conversion delay.

Additionally, the proposed hybrid SE converter (whenever $x_{1,n} = 0$) and the SE converter in [7] are implemented using Xilinx92i FPGA technology for varoius dynamic range requirements (different values of n). The target technology is Xillinx (Xa3s200-4vqg100) FPGA. The performance is evaluated in terms of area (measured in terms of the number of slices) and delay (represents the total gate delay, which is measured in nanoseconds). Table 3 shows the synthesized results for various values of n, which show the superiority of our scheme over the one in [7]. Consequently, the RNS-to-binary converters proposed in this paper are better than the ones in [7].

6. Conclusions

In this paper, we proposed two new reverse converters for the moduli set $\{2^{n+1}+1, 2^{n+1}-1, 2^n\}$. First, we simplified the traditional CRT to obtain a reverse converter that uses mod-$(2^{n+1}-1)$ operations instead of mod-$(2^{n+1}+1)(2^{n+1}-1)$ operations required by the proposal in [7]. Next, we further reduced the hardware complexity by making the resulting reverse converter architecture adder based. We proposed two hybrid CE and SE converters. In each of the schemes, the converter in [7] is integrated into a newly proposed area-delay efficient scheme. The path to be followed depends on whether $x_{1,n} = 1$ (which occurs only in few cases) or $x_{1,n} = 0$. In terms of delay, the two proposed hybrid CE and SE converters require $(2n(p+1)+p+4)t_{\mathrm{FA}}$ and $(n(p+1)+3)t_{\mathrm{FA}}+t_{\mathrm{MUX}}$, respectively, while the CE and SE converters in [7], respectively, require $(4n+5)t_{\mathrm{FA}}$ and $(2n+3)t_{\mathrm{FA}}+t_{\mathrm{MUX}}$, where t_{FA} denotes the delay of one full adder and t_{MUX} that of a multiplexer, and p is the occurrence probability of $x_{1,n} = 1$. The proposed hybrid CE converter outperforms the one in [7] in terms of delay with slightly higher or similar area cost. Additionally, with smaller delay, the proposed hybrid SE converter also requires less area cost when compared to the one in [7].

References

[1] A. A. Hiasat, "VLSI implementation of new arithmetic residue to binary decoders," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 13, no. 1, pp. 153–158, 2005.

[2] W. Wang, M. N. S. Swamy, M. O. Ahmad, and Y. Wang, "A study of the residue-to-binary converters for the three-moduli sets," *IEEE Transactions on Circuits and Systems I*, vol. 50, no. 2, pp. 235–243, 2003.

[3] Y. Wang, X. Song, M. Aboulhamid, and H. Shen, "Adder based residue to binary number converters for $\{2^n+1, 2^n, 2^n-1\}$," *IEEE Transactions on Signal Processing*, vol. 50, pp. 1772–1779, 2002.

[4] A. A. Hiasat and H. S. Abdel-Aty-Zohdy, "Residue-to-binary arithmetic converter for the moduli set $\{2^k, 2^k-1, 2^{k-1}-1\}$," *IEEE Transactions on Circuits and Systems II*, vol. 45, no. 2, pp. 204–209, 1998.

[5] K. A. Gbolagade and S. D. Cotofana, "Generalized matrix method for efficient residue to decimal conversion," in *Proceeding of the 10th IEEE Asia-Pacific Conference on Circuits and Systems (APCCAS '08)*, pp. 1414–1417, Macao, China, December 2008.

[6] K. A. Gbolagade and S. D. Cotofana, "Residue number system operands to decimal conversion for 3-moduli sets," in *Proceeding of the 51st IEEE Midwest Symposium on Circuits and Systems*, pp. 791–794, Knoxville, Tenn, USA, August 2008.

[7] A. S. Molahosseini and K. Navi, "New arithmetic residue to binary converters," *International Journal of Computer Sciences and Engineering Systems*, vol. 1, no. 4, pp. 295–299, 2007.

[8] P. V. A. Mohan, "RNS-to-binary converter for a new three-moduli set $\{2^{n+1}-1, 2^n, 2^n-1\}$," *IEEE Transactions on Circuits and Systems II*, vol. 54, no. 9, pp. 775–779, 2007.

[9] D. F. Miller and W. S. McCormick, "An arithmetic free parallel mixed radix conversion algorithm," *IEEE Transactions Circuits and Systems II*, vol. 45, no. 1, pp. 158–162, 1998.

[10] N. Szabo and R. Tanaka, *Residue Arithmetic and Its Application to Computer Technology*, MC-Graw-Hill, New York, NY, USA, 1967.

[11] Y. Wang, "New Chinese remainder theorems," in *Proceeding of the Asilomar Conference*, vol. 1, pp. 165–171, Pacific Grove, Calif, USA, November 1998.