World Academy of Science, Engineering and Technology
International Journal of Computer and Information Engineering
Vol:3, No:7, 2009

A Parallel Implementation of the Reverse Converter for the Moduli Set $\{2^n, 2^n-1, 2^{n-1}-1\}$

Mehdi Hosseinzadeh, Amir Sabbagh Molahosseini, and Keivan Navi

Abstract**—** In this paper, a new reverse converter for the moduli set $\{2^n, 2^n-1, 2^{n-1}-1\}$ is presented. We improved a previously introduced conversion algorithm for deriving an efficient hardware design for reverse converter. Hardware architecture of the proposed converter is based on carry-save adders and regular binary adders, without the requirement for modular adders. The presented design is faster than the latest introduced reverse converter for moduli set $\{2^n, 2^n-1, 2^{n-1}-1\}$. Also, it has better performance than the reverse converters for the recently introduced moduli set $\{2^{n+1}-1, 2^n, 2^n-1\}$

Keywordss**—** Residue arithmetic; Residue number system; Residue-to-Binary converter; Reverse converter.

I. Introduction

THE basis for each residue number system (RNS) is a moduli set which consists of a set of pairwise relatively prime numbers [1]. Until now, many moduli sets with different dynamic ranges have been introduced for RNS [2]-[8]. Among these, the moduli set $\{2^n-1, 2^n, 2^n+1\}$ is the most well-known. This moduli set can result in simple and efficient designs for reverse converters, but the performance of arithmetic unit of RNS systems based on this moduli set are restricted to the time-performance of the modulo 2^n+1. The modulo 2^n+1 operations are very complex, and are usually the bottleneck for RNS arithmetic units [9]. Hence, the moduli sets $\{2^n, 2^n-1, 2^{n-1}-1\}$ [5],[6] and $\{2^{n+1}-1, 2^n, 2^n-1\}$ [7] have been suggested as alternatives for the moduli set $\{2^n-1, 2^n, 2^n+1\}$. In these moduli sets, the moduli $2^{n-1}-1$ and $2^{n+1}-1$ are used instead of 2^n+1. The arithmetic units of RNS systems based on moduli sets $\{2^n, 2^n-1, 2^{n-1}-1\}$ and $\{2^{n+1}-1, 2^n, 2^n-1\}$ are faster than those based on the moduli set $\{2^n-1, 2^n, 2^n+1\}$, but due to the mathematical properties of these moduli sets, they have more complex reverse conversion than the moduli set $\{2^n-1, 2^n, 2^n+1\}$.

In this paper, we apply some simplifications to the reverse conversion algorithm of [6], and present a new hardware implementation of the reverse converter for the moduli set $\{2^n, 2^n-1, 2^{n-1}-1\}$. The proposed reverse converter has lower conversion delay than the reverse converters of [5] and [6]. Also, it has better performance in comparison to the recently proposed reverse converters for the set $\{2^{n+1}-1, 2^n, 2^n-1\}$ [7].

In the rest of paper, the conversion algorithm of [6] is introduced in section II. In section III we propose the improvements on conversion algorithm of [6]. Also, hardware implementation of the improved conversion algorithm is presented in section IV. Section V evaluates the performance of the proposed reverse converter as well as the other reverse converters, with regard to the conversion delay and hardware complexity, and section VI is conclusion.

M. Hosseinzadeh and A.S. Molahosseini are with Science and Research Branch, Islamic Azad University, Tehran, Iran (e-mails: hosseinzadeh@srbiau.ac.ir; amir.sabbagh@srbiau.ac.ir).

K. Navi is with Department of Electrical and Computer Engineering, Shahid Beheshti University, Tehran, Iran (e-mail: navi@sbu.ac.ir).

II. Wang's Conversion Algorithm

Wang et al. [6] used New CRT-I [10],[11] to derive a high-speed reverse conversion algorithm as follow:

Theorem 1 [6]: In the RNS system based on the moduli set $\{2^n, 2^n-1, 2^{n-1}-1\}$, the residue represented number (x_1, x_2, x_3) can be converted into its equivalent weighted binary number by

$$X = x_1 + 2^n Z \tag{1}$$

where

$$Z = (2^n - 1)Y + \left| x_2 - x_1 \right|_{2^n-1} \tag{2}$$

$$Y = \begin{cases} \left| T_1 + T_2 + T_3 + T_4 + T_5 \right|_{2^{n-1}-1} & x_2 \geq x_1 \\ \left| T_1 + T_2 + T_3 + T_4 + \hat{T}_5 \right|_{2^{n-1}-1} & x_2 < x_1 \end{cases} \tag{3}$$

$$T_1 = \underbrace{x_{3,0} x_{3,n-2} \cdots x_{3,1}}_{n-1 \ bits} \tag{4}$$

$$T_2 = \underbrace{x_{1,0} x_{1,n-2} \cdots x_{1,1}}_{n-1 \ bits} \tag{5}$$

$$T_3 = x_{1,n-1} \underbrace{00 \cdots 0}_{n-2 \ bits} \tag{6}$$

$$T_4 = \underbrace{\bar{x}_{2,n-2} \cdots \bar{x}_{2,0}}_{n-1 \ bits} \tag{7}$$

$$T_5 = \underbrace{11 \cdots 1}_{n-2 \ bits} \bar{x}_{2,n-1} \tag{8}$$

$$\hat{T}_5 = \underbrace{11 \cdots 1}_{n-3 \ bits} \bar{x}_{2,n-1} x_{2,n-1} \tag{9}$$

The proof of this theorem is described in [6].

III. Improved Conversion Algorithm

In this section, we make some simplifications to the Wang et al. [6] conversion algorithm, for achieving a more efficient hardware implementation. First, Theorem 1 can be rewritten as

$$X = x_1 + 2^n Z \tag{10}$$

where

$$Z = (2^n - 1)Y + \left|x_2 - x_1\right|_{2^n-1} = 2^n Y - Y + \left|x_2 - x_1\right|_{2^n-1} \quad (11)$$

$$Y = \begin{cases} \left|T_1 + T_2 + T_3 + T_4 + T_5\right|_{2^{n-1}-1} & x_2 \geq x_1 \\ \left|T_1 + T_2 + T_3 + T_4 + \hat{T}_5\right|_{2^{n-1}-1} & x_2 < x_1 \end{cases} \quad (12)$$

Equation (11) can be parsed as below:

$$Z = 2^n A + B \quad (13)$$

$$B = -Y + \left|x_2 - x_1\right|_{2^n-1} \quad (14)$$

$$A = Y + B_{out} \quad (15)$$

It should be noted that B_{out} is the borrow which is produced by the subtraction of (14). Also, the operands needed for calculating Y, are given in (4)-(9).

First, the relationship between (8) and (9) is remarkable, and based on it, we can write the following equation

$$T_5 = \hat{T}_5 + 1 \quad (16)$$

Therefore, we can add up binary vectors of equations (4)-(7) and (9), and then when $x_2 \geq x_1$, the result should be incremented by one. So,

$$(P_2, P_1) = \left|T_1 + T_2 + T_3 + T_4 + \hat{T}_5\right|_{2^{n-1}-1} \quad (17)$$

The expression (P_2,P_1) denotes two $(n-1)$-bit wide result of end around carry save addition of equations (4)-(7) and (9).

Next, we simplify (14) and (15). The following algorithm can be used for calculating the correct value of Y in (14) and (15).

If $(x_2 < x_1)$ then $Y = \left|P_2 + P_1\right|_{2^{n-1}-1}$

  If $(P_1+P_2 < 2^{n-1}-1)$ then $Y=P_1+P_2$

  If $(P_1+P_2 \geq 2^{n-1}-1)$ then $Y=P_1+P_2-(2^{n-1}-1)$

Else if $(x_2 \geq x_1)$ then $Y = \left|P_2 + P_1 + 1\right|_{2^{n-1}-1}$

  If $(P_1+P_2+1 < 2^{n-1}-1)$ then $Y=P_1+P_2+1$

  If $(P_1+P_2+1 \geq 2^{n-1}-1)$ then $Y=P_1+P_2+1-(2^{n-1}-1)$

End if

Fig. 1 The algorithm for selection of correct value of Y

Based on this algorithm, Y can be calculated by

$$Y = \begin{cases} P_1 + P_2 & x_2 < x_1 \text{ and } P_1 + P_2 < 2^{n-1} - 1 \\ P_1 + P_2 - (2^{n-1} - 1) & x_2 < x_1 \text{ and } P_1 + P_2 \geq 2^{n-1} - 1 \\ P_1 + P_2 + 1 & x_2 \geq x_1 \text{ and } P_1 + P_2 < 2^{n-1} - 1 \\ P_1 + P_2 + 1 - (2^{n-1} - 1), & x_2 \geq x_1 \text{ and } P_1 + P_2 \geq 2^{n-1} - 1 \end{cases} \quad (18)$$

Now, from (14) we have

$$B = \left|x_2 - x_1\right|_{2^n-1} - Y \quad (19)$$

where

$$\left|x_2 - x_1\right|_{2^n-1} = \begin{cases} x_2 - x_1 + (2^n - 1) & x_2 < x_1 \\ x_2 - x_1 & x_2 \geq x_1 \end{cases} \quad (20)$$

Therefore, by substituting the values of (18) and (20) in (19), we have

$$B = \begin{cases} V_1 & x_2 < x_1 \text{ and } P_1 + P_2 < 2^{n-1} - 1 \\ V_2 & x_2 < x_1 \text{ and } P_1 + P_2 \geq 2^{n-1} - 1 \\ V_3 & x_2 \geq x_1 \text{ and } P_1 + P_2 < 2^{n-1} - 1 \\ V_4 & x_2 \geq x_1 \text{ and } P_1 + P_2 \geq 2^{n-1} - 1 \end{cases} \quad (21)$$

where

$$V_1 = x_2 - x_1 + (2^n - 1) - P_1 - P_2 \quad (22)$$

$$V_2 = x_2 - x_1 + (2^n - 1) - P_1 - P_2 + (2^{n-1} - 1) \quad (23)$$

$$V_3 = x_2 - x_1 - P_1 - P_2 - 1 \quad (24)$$

$$V_4 = x_2 - x_1 - P_1 - P_2 - 1 + (2^{n-1} - 1) \quad (25)$$

The equation (22) can be rewritten as follows

$$V_1 = x_2 + ((2^n - 1 - x_1) - (2^n - 1)) + ((2^{n-1} - 1 - P_1) - (2^{n-1} - 1)) + ((2^{n-1} - 1 - P_2) - (2^{n-1} - 1)) + (2^n - 1) \quad (26)$$

So, the above equation becomes:

$$V_1 = x_2 + (2^n - 1 - x_1) + (2^{n-1} - 1 - P_1) + (2^{n-1} - 1 - P_2) + 2 - 2^n \quad (27)$$

with considering the facts that $(2^n - 1 - x_1) = \bar{x}_1$, $(2^{n-1} - 1 - P_1) = \bar{P}_1$ and $(2^{n-1} - 1 - P_2) = \bar{P}_2$, we have

$$V_1 = x_2 + \bar{x}_1 + \bar{P}_1 + \bar{P}_2 + 2 - 2^n \quad (28)$$

Using similar derivation, (23)-(25) can be calculated as

$$V_2 = x_2 + \bar{x}_1 + \bar{P}_1 + \bar{P}_2 + 1 - 2^{n-1} \quad (29)$$

$$V_3 = x_2 + \bar{x}_1 + \bar{P}_1 + \bar{P}_2 + 2 - 2^{n+1} \quad (30)$$

$$V_4 = x_2 + \bar{x}_1 + \bar{P}_1 + \bar{P}_2 + 1 - 2^{n-1} - 2^n \quad (31)$$

The equations (28)-(31) can be rewritten as

$$V_1 = ((2P_6 + P_5 + 1) + 1) - 2^n \quad (32)$$

$$V_2 = (2P_6 + P_5 + 1) - 2^{n-1} \quad (33)$$

$$V_3 = ((2P_6 + P_5 + 1) + 1) - 2^{n+1} \quad (34)$$

$$V_4 = (2P_6 + P_5 + 1) - 2^{n-1} - 2^n \quad (35)$$

where

$$(P_6, P_5) = x_2 + 2P_4 + P_3 \quad (36)$$

$$(P_4, P_3) = \bar{x}_1 + \bar{P}_1 + \bar{P}_2 \quad (37)$$

The terms (P_6,P_5) and (P_4,P_3) denote the result of carry save addition. Also, since P_4 and P_6 are the carry vector results of the carry save additions, they should be shifted by one bit to the left for performing regular addition.

Finally, (15) can be simplified as below

$$A = Y + B_{out} \quad (38)$$

The value of Y from (18) can be substituted in (38) as

$$A = P_1 + P_2 + \alpha + B_{out} \quad (39)$$

where

$$\alpha = \begin{cases} 0 & x_2 < x_1 \text{ and } P_1 + P_2 < 2^{n-1} - 1 \\ 1 & x_2 < x_1 \text{ and } P_1 + P_2 \geq 2^{n-1} - 1 \\ 1 & x_2 \geq x_1 \text{ and } P_1 + P_2 < 2^{n-1} - 1 \\ 2 & x_2 \geq x_1 \text{ and } P_1 + P_2 \geq 2^{n-1} - 1 \end{cases} \quad (40)$$

It should be noted that, -2^{n-1} in (18) only changes the most significant bits of Y, and since the most significant bit of Y for calculation of (38) will be ignored, we don't take into account the -2^{n-1} in computing the value of α.

IV. Hardware Implementation

Hardware architecture of the proposed reverse converter for the moduli set $\{2^n, 2^n-1, 2^{n-1}-1\}$ is based on equations (10) and (13)-(15). Firstly, as shown in Fig.2, by using three (n–1)-bit carry save adders (CSAs) with end around carry (EAC), the modulo carry save addition of (17) is performed. Since, (n–2) bits of (6) are 0's, and (n–3) bits of (9) are 1's, (n–2) and (n–3) of the full adders (FAs) in CSA1 and CSA3 are reduced to (n–2) half adders (HAs) and (n–3) XNOR/OR gates, respectively.

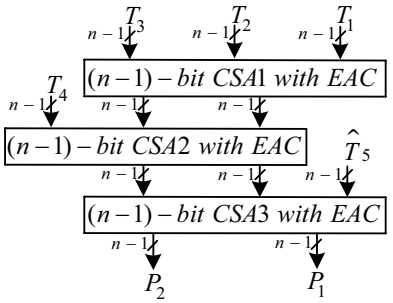

Fig. 2 Realization of (17)

Fig.3 shows the implementation of (21). First, two CSAs is used for realization of (36) and (37). Then, (32)-(35) are implemented by using two (n+1)-bit carry propagate adders (CPAs) followed by a constant subtractor unit (CSU).

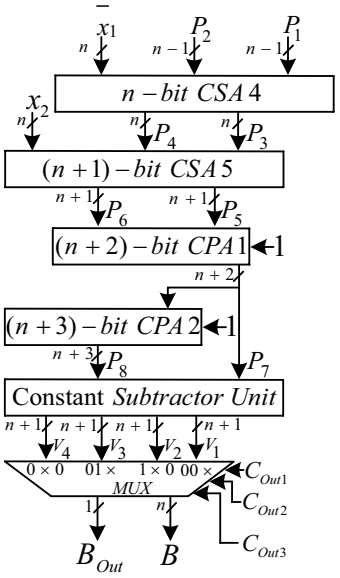

Fig. 3 Calculation of B

It should be noted that the carry-in of these CPAs are 1, and so, CPA2 includes only (n+2) HAs. Hence, CPA1 and CPA2 function in a bit level parallel architecture. Therefore, the total delay of CPA1 plus CPA2 is $(n+2)t_{FA}+t_{HA}$, where t_{FA} and t_{HA} denote the delay of an FA and HA, respectively. Fig. 4 depicts the CPA1 and CPA2 for constant value of three bits. So, It is clear that, the complexity and the delay are $3(FA + HA) + HA$ and $3t_{FA} + t_{HA}$, respectively.

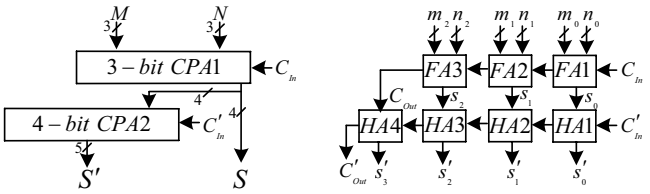

Fig. 4 Implementation of CPA1 and CPA2 for a constant value

The CSU consists of four small subtractors. The first subtractor subtract 2^n from v_1 in (32), and since this subtraction only change the most significant two bits of v_1, it can be simply implemented with two FAs. Similarly, the other subtractors, subtract 2^{n-1}, 2^{n+1} and $2^{n-1}+2^n$ from v_2, v_3 and v_4, respectively. The most significant bit of the result of subtraction will be ignored, and also one of the borrow-out of these four subtractors will be used in the calculation of (38). Totally, we need 9 FAs for CSU, and the total delay of this unit is the delay of three FAs. For example, the subtract part for computing v_4 is shown in Fig.5. So, the CSU produces v_1, v_2, v_3 and v_4 in (32)-(35). One of these four numbers must be chosen for achieving the correct result of B in (21). The correct output between these four numbers will be selected by an n-bit MUX. The detector unit produces the select lines of this MUX. This detector unit (DU) includes two CPAs as shown in Fig. 6.

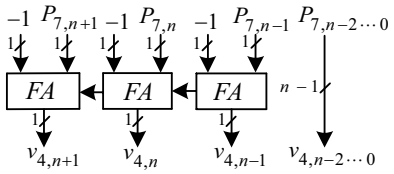

Fig. 5 The subtractor unit for computing v_4

First, for detecting $x_2 \geq x_1$, an n-bit CPA with one carry-in is used. We need only the carry-out of this CPA. Similarly, for finding $P_1+P_2 \geq 2^{n-1}-1$, an $(n-1)$-bit CPA is used. Next, detecting of $P_1+P_2+1 \geq 2^{n-1}-1$ can be simply implemented based on this CPA, and by using n AND gates plus an OR gate. Because, whenever, C_{out2} is 1, consequently C_{out3} is also 1. But where C_{out2} is 0, if all bits of sum vector of CPA8 are 1's, therefore C_{out3} become 1, otherwise C_{out3} is 0. it should be noted that, AND gates work parallel with CPA8. Hence, the total delay of detector unit is delay of an n-bit CPA. Because, C_{out1} and C_{out2} are calculated independently. Also, when C_{out2} is produce, after the delay of an OR gate, C_{out3} will be obtained.

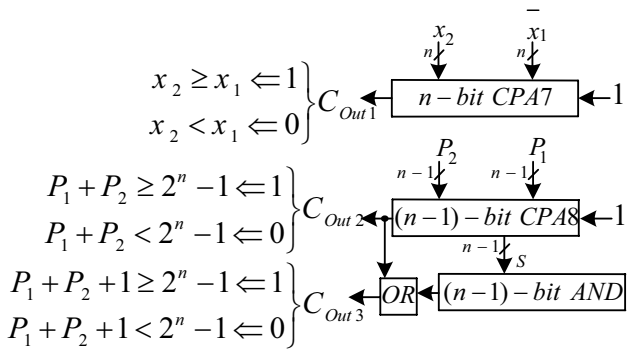

Fig. 6 The detector unit

Table I presents different cases of the outputs of detector unit, and the correct result that should be select.

TABLE I. CORRECT SELECTION THE VALUE OF B

C_{Out1}	C_{Out2}	C_{Out3}	Correct output	$\alpha = \alpha_1\alpha_0$
0	0	0	V_1	0=00
0	0	1	V_1	1=01
0	1	0	V_2	1=01
0	1	1	V_2	1=01
1	0	0	V_3	1=01
1	0	1	V_4	2=11
1	1	0	V_3	2=11
1	1	1	V_4	2=11

Now, we investigate the implementation of A in (39). First, as shown in Table I, the value of α can be computed by

$$\alpha_0 = C_{out1} + C_{out2} + C_{out3} \quad (41)$$

$$\alpha_1 = C_{out1}(C_{out2} + C_{out3}) \quad (42)$$

Therefore, α can be simply prepared by using some logic gates. Next, with considering the facts that B_{out}=–1,0 and α=0,1,2, and substituting in (39), we have

$$A = P_1 + P_2 + (-1, 0, 1, 2) \quad (43)$$

Hence, we do these four additions by using four CPAs, and then the correct result will be selected by a multiplexer. Fig. 7 shows the hardware architecture for calculation of A. The CPAs of Fig.7 work in a bit level parallel fashion like those in Fig.4. Hence, the total delay of Fig.7 is $(n-1)t_{FA}+2t_{HA}+t_{MUX}$.

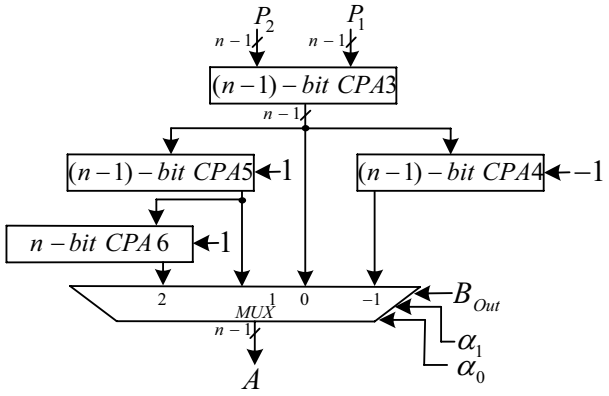

Fig. 7 Calculation of A

Finally, since x_1 and B are n-bit numbers, equation (10) can be implemented by concatenating x_1, B and A in (13) and (10), without using any computational hardware. Table II presents the characteristics of each part of the proposed reverse converter in terms of FA and HA. It should be noted that, we used the same assumptions used in [6], such as ignoring the cost and the delay of required NOT gates, considering the complexity of XNOR/OR gate and MUX as HA and FA, respectively.

TABLE II. HARDWARE REQUIREMENTS OF THE PROPOSED CONVERTER

Components	Full Adder	Half Adder
CSA1	1	n–2
CSA2	n–1	0
CSA3	2	n–3
CSA4	n	0
CSA5	n+1	0
CPA1	n+2	0
CPA2	0	n+3
CSU	9	0
DU	2n–1	0
CPA3	n–1	0
CPA4,5	0	2(n–1)
CPA6	0	n
MUX	2n	0
Total	9n+12	6n–4

V. PERFORMANCE EVALUATION

The critical delay path of the proposed reverse converter composed of the delay of CSA1 to CSA5, CPA1, CPA2, CSU and MUX. The delay of a CSA is the same as that of an FA. Also, CPA2 adds the result of CPA1 with one. So, CPA2 can work in a bit level parallel fashion with CPA1, and therefore it adds only the delay of an HA to the total delay. The hardware architecture of the presented reverse converter consists of (9n+12) FA's, (6n–4) HA's and (n–1) two-input AND gates. Similar to [6], for the proposed converter as well as for the converters of [7], we consider the complexity of an FA is twice

that of an HA and the two-input AND gate. Hence, the total hardware complexity of the proposed reverse converter can be calculated as

$2\times(9n+12)$ HA + $(9n+12)$AND + $(6n-4)$HA + $(n-1)$AND
$=(24n+20)$HA + $(10n+11)$AND

There exist two other reverse converters for the moduli set $\{2^n, 2^n-1, 2^{n-1}-1\}$ which have been introduced in [5] and [6]. The hardware architecture of the reverse converter of [6] is based on Theorem 1, and composed of four $(n-1)$-bit CSAs with EAC for performing modulo $(2^{n-1}-1)$ carry save addition, two $(n-1)$-bit modulo $(2^{n-1}-1)$ adder that work in parallel, a $(n-1)$-bit modulo $(2^{n-1}-1)$ subtractor, a $(2n-1)$-bit regular binary subtractor, and a $(n-1)$-bit 2×1 multiplexer (MUX). The critical delay path of the reverse converter of [6] consists of three CSAs, a $(n-1)$-bit modulo $(2^{n-1}-1)$ adder and $(2n-1)$-bit subtractor. The use of two CSAs with EAC, and two modulo adders that work in parallel, resulted in reducing the conversion delay, but the hardware cost increased. Also, recently, the new three-moduli set $\{2^{n+1}-1, 2^n, 2^n-1\}$ has been proposed by Mohan [7]. He introduced three reverse converters for this moduli set by using CRT and MRC algorithms.

TABLE III. PERFORMANCE COMPARISON

Converters	Complexity	Delay	Time-complexity
[5]	$(12n-8)$HA, $(6n-4)$ AND	$(5n-4)t_{FA}$	$60n^2$
[6]	$(17n-13)$HA, $(7n-3)$ AND	$(3n+2)t_{FA}$	$51n^2$
[7]-CI	$(9n+3)$HA, $(8n+3)$AND	$(6n+5)t_{FA}$	$54n^2$
[7]-CII	$(30n+45)$HA, $(14n+21)$ AND	$(2n+7)t_{FA}$	$60n^2$
[7]-CIII	$(26n+40)$HA, $(12n+19)$ AND	$(2n+7)t_{FA}$	$46n^2$
Proposed	$(24n+20)$HA, $(10n+11)$ AND	$(n+11)t_{FA}$	$24n^2$

Table III compares the performance of the different reverse converters for the moduli set $\{2^n, 2^n-1, 2^{n-1}-1\}$ as well as the reverse converters for moduli set $\{2^{n+1}-1, 2^n, 2^n-1\}$. As seen from Table III, the proposed converter is the fastest between the other existing methods for moduli set $\{2^n, 2^n-1, 2^{n-1}-1\}$. However, the hardware cost of the presented converter is much. But it is essential to remark to the point that, the reverse converters of [5] and [6], both use modular adders. They used the method of [12] for the implementation of the needed modular adders. While, we present the full design of the reverse converter without using modular adders. Because of the complex structure of the modular adder of [12], the authors of [6], assumed the cost and the delay of the modular adder of [12] are nFA and nt_{FA}, respectively. But these estimations are not exact, and the real cost and delay of the adder of [12] are much more. With considering these points, the proposed converter has a much better area-time complexity than converter of [6]. Also, it can be seen that the proposed converter is faster than the reverse converters of [7], and also it has better hardware complexity than converters [7]-CII and [7]-CIII. Even if proposed converter consume more hardware but it demonstrated significant improvement in terms of speed in comparison to [7]-CI.

VI. CONCLUSION

This paper presents an efficient reverse converter for the well-known RNS moduli set $\{2^n, 2^n-1, 2^{n-1}-1\}$. The hardware architecture of the proposed converter consists of regular binary adders and logic gates, without the need for using modular adders. Also, the presented reverse converter results in significant improvement in terms of conversion delay and time-complexity, compared to the last works.

REFERENCES

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

[2] Y. Wang, X. Song, M. Aboulhamid, and H. Shen, "Adder based residue to binary numbers converters for $(2^n-1, 2^n, 2^n+1)$," IEEE Trans. Signal Processing, vol. 50, no. 7, pp. 1772-1779, 2002.

[3] Z. Wang, G. Jullien, and W. Miller, "An Improved Residue to Binary Converter," IEEE Trans. Circuits Syst.-I, vol. 45, no. 9, pp. 998-1002, 2000.

[4] A. Hariri, K. Navi, and R. Rastegar, "A new high dynamic range moduli set with efficient reverse converter," Elsevier Journal of Computers and Mathematics with Applications, vol. 55, no. 4, pp. 660-668, 2008.

[5] 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 Trans. Circuits Syst.-II, vol. 45, no. 2, pp. 204–208, 1998.

[6] W.Wang, M. N. S. Swamy, M. O. Ahmad, and Y.Wang, "A high-speed residue-to-binary converter and a scheme of its VLSI implementation," IEEE Trans. Circuits Syst.-II, vol. 47, no. 12, pp. 1576–1581, 2000.

[7] P.V.A. Mohan,. "RNS-To-Binary converter for a new three-moduli set $\{2^{n+1}-1, 2^n, 2^n-1\}$," IEEE Trans. Circuits Syst.-II, vol. 54, pp. 775-779, 2007.

[8] M. Hosseinzadeh, A.S. Molahosseini, and K. Navi, "An improved reverse converter for the moduli set $\{2^n-1, 2^n, 2^n+1, 2^{n+1}-1\}$," IEICE Electronics Express, In Press, 2008.

[9] C.H. Chang, S. Menon, B. Cao, and T. Srikanthan , "A configurable dual moduli multi-operand modulo adder," Proc. IEEE Symp. Circuits Syst., vol. 2, pp. 1630-1633, 2005.

[10] Y. Wang, "Residue-to-Binary Converters Based on New Chinese remainder theorems," IEEE Trans. Circuits Syst.-II, vol. 47, no. 3, pp. 197-205, 2000.

[11] A.S. Molahosseini, K. Navi, O. Hashemipour, and A. Jalali, "An efficient architecture for designing reverse converters based on a general three-moduli set, Elsevier Journal of Systems Architecture, In Press, 2008.

[12] C. Efstathiou, D. Nikolos, and J. Kalamatianos, "Area-time efficient modulo 2^n-1 adder design," IEEE Trans. Circuits Syst.-II, vol. 41, pp. 463–467, 1994.