An improved reverse converter for the moduli set $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} - 1\}$

Mehdi Hosseinzadeh[1a)], **Amir Sabbagh Molahosseini**[1], **and Keivan Navi**[2]

[1] *Ph.D Student, Science & Research Branch, Islamic Azad University (IAU), Tehran, Iran*

[2] *Department of Electrical and Computer Engineering, Shahid Beheshti University, GC, Tehran, Iran*

a) *hosseinzadeh@sr.iau.ac.ir*

Abstract: In this paper, a new reverse converter for the moduli set $\{2^n-1, 2^n, 2^n+1, 2^{n+1}-1\}$ is presented. We improved a previously introduced reverse converter architecture for deriving a high-speed hardware design. Hardware architecture of the proposed converter is based on adders, without the need for ROM or Multiplier. The presented design resulted in a significant reduction in conversion delay in comparison to the last reverse converter for the moduli set $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} - 1\}$.

Keywords: residue number system, reverse converter, VLSI architectures

Classification: Integrated circuits

References

[1] M. A. Bayoumi and P. Srinivasan, "Parallel arithmetic: from algebra to architecture," *Proc. IEEE International Symposium on Circuits and Systems*, pp. 2630–2633, 1990.

[2] T. Stouratitis and V. Paliouras, "Considering the alternatives in lowpower design," *IEEE Circuits and Devices*, vol. 7, pp. 23–29, 2001.

[3] 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.

[4] 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.

[5] 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.

[6] P. V. A. Mohan and A. B. Premkumar, "RNS-to-Binary Converters for Two Four-Moduli Set $\{2^n-1, 2^n, 2^n+1, 2^{n+1}-1\}$ and $\{2^n-1, 2^n, 2^n+1, 2^{n+1}+1\}$," *IEEE Trans. Circuits Syst. I*, vol. 54, no. 6, pp. 1245–1254, 2007.

DOI: 10.1587/elex.5.672
Received June 30, 2008
Accepted July 11, 2008
Published September 10, 2008

1 Introduction

One of the most effective ways for achieving parallelism on arithmetic level in VLSI systems design is the use of residue number system (RNS) [1]. In RNS, a weighted number is converted into a set of small residues. Since arithmetic operations on residues can be performed without carry propagation between residues, RNS leads to high-speed addition, subtraction and multiplication. Today, RNS is used for reducing the power dissipation in high-performance systems. RNS can tolerate a large reduction in the supply voltage comparing to the corresponding binary architecture while achieving a particular delay specification [2].

Each RNS system is based on a moduli set which consists of a set of relatively prime integers. The dynamic range of an RNS system is defined in terms of the product of the moduli, and it denotes the interval of integers uniquely represented in RNS. Until now, many moduli sets with different dynamic ranges have been proposed for RNS such as $\{2^n,\ 2^{2n}-1,\ 2^{2n}+1\}$ [3], $\{2^n-1,\ 2^n,\ 2^n+1\}$ [4], $\{2^{2n-1},\ 2^{2n+1}-1,\ 2^{2n+1}+1\}$ [5]. But, the parallelism provided by these moduli sets is not appropriate for applications which require large dynamic range with high parallelism. Hence, some four-moduli sets such as $\{2^n-1,\ 2^n,\ 2^n+1,\ 2^{n+1}+1\}$ and $\{2^n-1,\ 2^n,\ 2^n+1,\ 2^{n+1}-1\}$ [6] have been introduced. Since, the largest modulo of the moduli set $\{2^n-1,\ 2^n,\ 2^n+1,\ 2^{n+1}-1\}$ is in the form 2^k-1, it can results in faster RNS arithmetic unit than $\{2^n-1,\ 2^n,\ 2^n+1,\ 2^{n+1}+1\}$. From the another side, some of the multiplicative inverses of these four-moduli sets have an inelegant forms, and this resulted in long conversion delay of the reverse converters for these moduli sets. In this paper, we make some modifications to the reverse converter of [6], and present a new hardware implementation of the reverse converter for the moduli set $\{2^n-1,\ 2^n,\ 2^n+1,\ 2^{n+1}-1\}$. The proposed reverse converter has much lower conversion delay than the converter of [6].

In the rest of paper, the reverse converter of [6] is introduced in section 2. In section 3, we propose the improvements on reverse converter of [6]. Section 5 evaluates the performance of the proposed reverse converter as well as the other reverse converter, with regard to the conversion delay and hardware complexity.

2 Mohan et al. reverse converter

Mohan et al. [6] used mixed-radix conversion (MRC) to derive a reverse conversion algorithm for the moduli set $\{2^n-1,\ 2^n,\ 2^n+1,\ 2^{n+1}-1\}$. But, MRC is a sequential process, and because of this their adder-based reverse converter for the moduli set $\{2^n-1,\ 2^n,\ 2^n+1,\ 2^{n+1}-1\}$ has long conversion delay. As shown in Fig. 1, their converter architecture consists of many modulo adders which implemented by using carry-propagate adders (CPAs) with end-around carry (EAC). For reducing the conversion delay, they substituted one part of converter with ROM, and derived a ROM-based reverse converter. Although, the ROM-based converter of [6] has less conversion de-

© IEICE 2008
DOI: 10.1587/elex.5.672
Received June 30, 2008
Accepted July 11, 2008
Published September 10, 2008

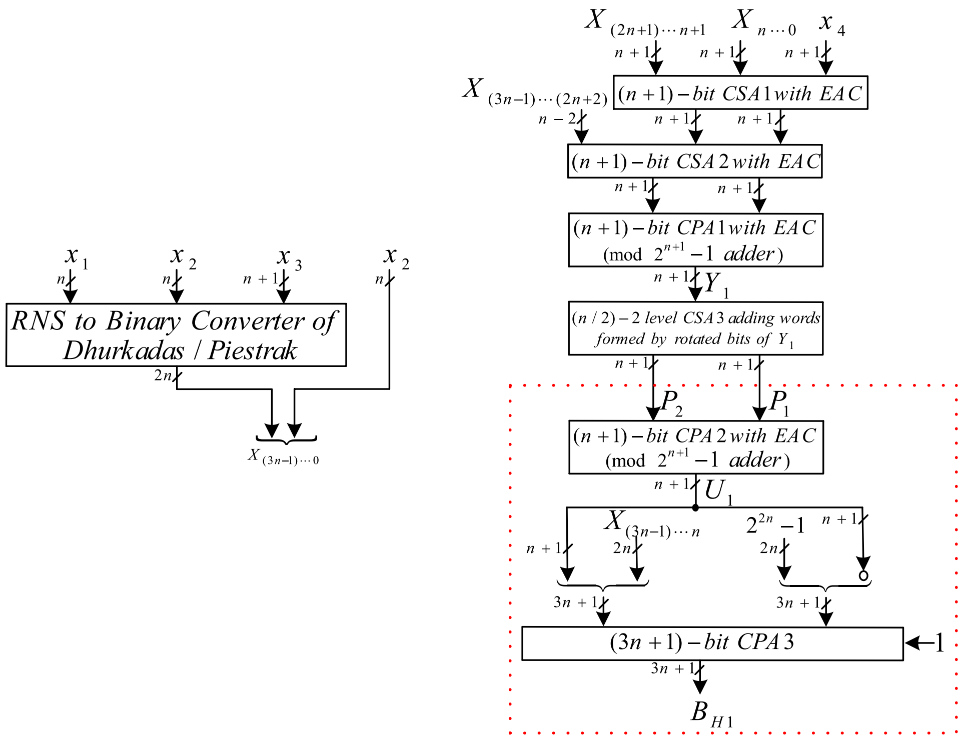
$X_{(2n+1)\cdots n+1}$
$X_{n\cdots 0}$
x_4
$X_{(3n-1)\cdots(2n+2)}$
$(n+1)$ – bit CSA1 with EAC
$(n+1)$ – bit CSA2 with EAC
$(n+1)$ – bit CPA1 with EAC (mod $2^{n+1}-1$ adder)
Y_1
$(n/2)-2$ level CSA3 adding words formed by rotated bits of Y_1
P_2
P_1
$(n+1)$ – bit CPA2 with EAC (mod $2^{n+1}-1$ adder)
U_1
$X_{(3n-1)\cdots n}$
$2^{2n}-1$
$(3n+1)$ – bit CPA3
1
B_{H1}
x_1
x_2
x_3
x_2
RNS to Binary Converter of Dhurkadas / Piestrak
$X_{(3n-1)\cdots 0}$
$n+1$
$n-2$
n
$2n$
$3n+1$

Fig. 1. The reverse converter architecture of [6]

lay than the corresponding adder-based architecture but with increasing the value of n, the ROM size grows exponentially, and this leads to increasing the cost of reverse converter.

3 The proposed improvements

The main reason for long conversion delay of the adder-based reverse converter of [6] is the modulo $(2^{n+1}-1)$ adder that followed by a $(3n+1)$-bit CPA (demonstrated within the dotted lines in Fig. 1). Hence, we are motivated to substitute this part with other hardware components for decreasing the total conversion delay of the reverse converter. The authors of [6] calculated the most significant bits (MSBs) of the final weighted number as:

$$B_{H1} = U_1{}^*(2^{2n}-1) + X_{(3n-1)\cdots n}. \tag{1}$$

In [6], an $(n+1)$-bit CPA2 with EAC is used for performing the modulo $(2^{n+1}-1)$ addition of the binary vectors P_1 and P_2 (as shown in Fig. 1, we named the inputs of CPA2 as P_1 and P_2). Next, a $(3n+1)$-bit CPA3 used for calculation of Eq. (1). The main idea is based on the fact that it is not essential to calculate B_{H1} directly by a $(3n+1)$-bit CPA3. But, we can consider B_{H1} as three independent binary vectors P_3, P_4 and P_5. Then, we calculate these numbers in a parallel architecture.

First, Eq. (1) can be computed by adding the following binary vectors

$$\begin{array}{lcccccccccccccc}
X_{(3n-1)\cdots n}: & 0 & 0 & \cdots 0 & 0 & X_{3n-1} & X_{3n-2} & X_{3n-3} & \cdots & X_{2n+1} & X_{2n} & X_{2n-1} & \cdots & X_{n+1} & X_n \\
U_1 2^{2n}: & u_{1n} & u_{1(n-1)} & \cdots u_{11} & u_{10} & 0 & 0 & 0 & \cdots 0 & 0 & 0 & & \cdots 0 & 0 & \\
-U_1: & 0 & 0 & \cdots 0 & 0 & 0 & 0 & 0 & \cdots 0 & -u_{1n} & -u_{1(n-1)} & \cdots & -u_{11} & -u_{10} &
\end{array} \tag{2}$$

© IEICE 2008
DOI: 10.1587/elex.5.672
Received June 30, 2008
Accepted July 11, 2008
Published September 10, 2008

These can be compressed as:

$$\begin{array}{cccccccccccccc} u_{1n} & u_{1(n-1)} & \cdots u_{11} & u_{10} & X_{3n-1} & X_{3n-2} & X_{3n-3} & \cdots X_{2n+1} & X_{2n} & X_{2n-1} & \cdots X_{n+1} & X_n \\ 0 & 0 & \cdots 0 & 0 & 0 & 0 & 0 & \cdots 0 & -u_{1n} & -u_{1(n-1)} & \cdots -u_{11} & -u_{10} \end{array} \tag{3}$$

U_1 is the result of modulo $(2^{n+1}-1)$ addition of P_1 and P_2, and it can be obtained as:

$$U_1 = \begin{cases} P_1 + P_2 & if \quad P_1 + P_2 < 2^{n+1} - 1 \\ P_1 + P - 2^{n+1} + 1 & if \quad P_1 + P_2 \geq 2^{n+1} - 1 \end{cases} \tag{4}$$

Hence, instead of direct performing modulo addition, we can substitute the value of (4) in (3). With performing this, the three parts of B_{H1} can be formulated as

$$P_3 = \begin{cases} X_{2n \cdots n} + \bar{P}_1 + \bar{P}_2 + 2 & if \quad P_1 + P_1 < 2^n - 1 \\ X_{2n \cdots n} + \bar{P}_1 + \bar{P}_2 + 2 + (2^{n+1} - 1) & if \quad P_1 + P_1 \geq 2^n - 1 \end{cases} \tag{5}$$

P_3 is an $(n+3)$-bit number. The first $(n+1)$ bits of P_3 forms the least significant $(n+1)$ bits of B_{H1}, and the last two bits of P_3 will be used in the next stage for computing P_4 as

$$P_4 = X_{3n-2 \cdots 2n+1} - P_{3(n+2)} P_{3(n+1)} \tag{6}$$

Since $P_{3(n+2)} P_{3(n+1)}$ can only have one of the values 0,1 and 2, we can rewrite the Eq. (6) as below

$$P_4 = \begin{cases} X_{3n-2 \cdots 2n+1} & if \quad P_{3(n+2)} P_{3(n+1)} = 0 \\ X_{3n-2 \cdots 2n+1} - 1 & if \quad P_{3(n+2)} P_{3(n+1)} = 1 \\ X_{3n-2 \cdots 2n+1} - 2 & if \quad P_{3(n+2)} P_{3(n+1)} = 2 \end{cases} \tag{7}$$

The MSB of P_4 will be used in the next stage for calculating P_5, and the remaining $(n-1)$ bits of P_4 forms the middle bits of B_{H1}. Finally, P_5 (the most significant $(n+1)$ bits of B_{H1}) can be evaluated as follow

$$P_5 = \Big((P_1 + P_2) \bmod 2^{n+1} - 1 \Big) - P_{4n} \tag{8}$$

$$P_5 = \begin{cases} P_1 + P_2 - P_{4n} & if \quad P_1 + P_2 < 2^n - 1 \\ P_1 + P_2 - P_{4n} - (2^{n+1} - 1) & if \quad P_1 + P_2 \geq 2^n - 1 \end{cases} \tag{9}$$

$$P_5 = \begin{cases} P_1 + P_2 \begin{cases} P_1 + P_2 & if \quad P_1 + P_2 < 2^{n+1} - 1 \quad and \quad P_{4n} = 0 \\ P_1 + P_2 - 1 & if \quad P_1 + P_2 < 2^{n+1} - 1 \quad and \quad P_{4n} = 1 \end{cases} \\ P_1 + P_2 + 1 - 2^{n+1} \begin{cases} P_1 + P_2 + 1 - 2^{n+1} & if \quad P_1 + P_2 \geq 2^{n+1} - 1 \quad and \quad P_{4n} = 0 \\ P_1 + P_2 + 1 - 2^{n+1} - 1 & if \quad P_1 + P_2 \geq 2^{n+1} - 1 \quad and \quad P_{4n} = 1 \end{cases} \end{cases} \tag{10}$$

It should be noted that, -2^{n+1} in (10) only changes the most significant bits of P_5, and since the most significant bit of P_5 will be ignored, we don't take into account the -2^{n+1} in computing the value of P_5. Hence, Eq. (10) can be rewritten as

$$P_5 = \begin{cases} P_1 + P_2 & if \quad (P_1 + P_2 < 2^{n+1} - 1 \quad and \quad P_{4n} = 0) \quad OR \quad (P_1 + P_2 \geq 2^{n+1} - 1 \quad and \quad P_{4n} = 1) \\ P_1 + P_2 - 1 & if \quad P_1 + P_2 < 2^{n+1} - 1 \quad and \quad P_{4n} = 1 \\ P_1 + P_2 + 1 & if \quad P_1 + P_2 \geq 2^{n+1} - 1 \quad and \quad P_{4n} = 0 \end{cases} \tag{11}$$

© IEICE 2008
DOI: 10.1587/elex.5.672
Received June 30, 2008
Accepted July 11, 2008
Published September 10, 2008

Example: Consider the following values

$$n = 3, \quad P_1 = 0101, \quad P_2 = 1000 \text{ and } X_{(3n-1)\cdots n} = 000111$$

First, using Mohan et al. method [6], we have

$$U_1 = 1101$$

$$B_{H1} = U_1{}^*(2^{2n} - 1) + X_{(3n-1)\cdots n}$$

$$B_{H1} = 1101^*(111111) + 000111 = 826$$

Now, by using the proposed method B_{H1} can be efficiently obtained as follow:

$$\begin{cases} P_1 + P_2 = 01101 \\ P_1 + P_2 + 1 = 01110 \end{cases} \Rightarrow C_{Out=0}$$

$$C_{Out=0} \Rightarrow P_3 = X_{2n\cdots n} + \bar{P}_1 + \bar{P}_2 + 2 = 0111 + 1010 + 0111 + 10 = 11010$$

$$(P_{3(n+1)} = 1, P_{3(n+2)} = 0, C_{Out} = 0) \Rightarrow P_4 = x_{(3n-1)\cdots(2n+1)} - 1 = 00 - 1 = 011$$

$$(P_{3(2n+1)} = 0, C_{Out} = 0) \Rightarrow P_5 = P_1 + P_2 - 1 = 0101 + 1000 - 1 = 1100$$

Fig. 2 shows hardware implementation of the proposed improvements. The architecture is based on Eqs. (5), (7) and (11). Implementation of (5) relies on an $(n+1)$-bit carry-save adder (CSA) followed by two CPAs. It should be noted that, CPA4 includes only $(n+3)$ half adders (HAs). Also, CPA3 and CPA4 function in a bit level parallel architecture. Therefore, the total delay of CPA3 plus CPA4 is $(n+2)t_{\text{FA}} + t_{\text{HA}}$, where t_{FA} and t_{HA} denote the delay of a full adder (FA) and HA, respectively.

Next, for realization of (7) two $(n-1)$-bit CPAs are used. Finally, Implementation of (10) has been done by three $(n+1)$-bit CPAs. Also, the carry-outs of CPA7 and CPA8 are used for detecting $P_1 + P_2 \geq 2^{n+1} - 1$. If C_{out} is zero, the value of P_1 plus P_2 is smaller than $2^{n+1} - 1$, and whenever C_{out} is one, we have $P_1 + P_2 \geq 2^{n+1} - 1$. The C_{out} used for selecting the correct values of multiplexers (MUXs).

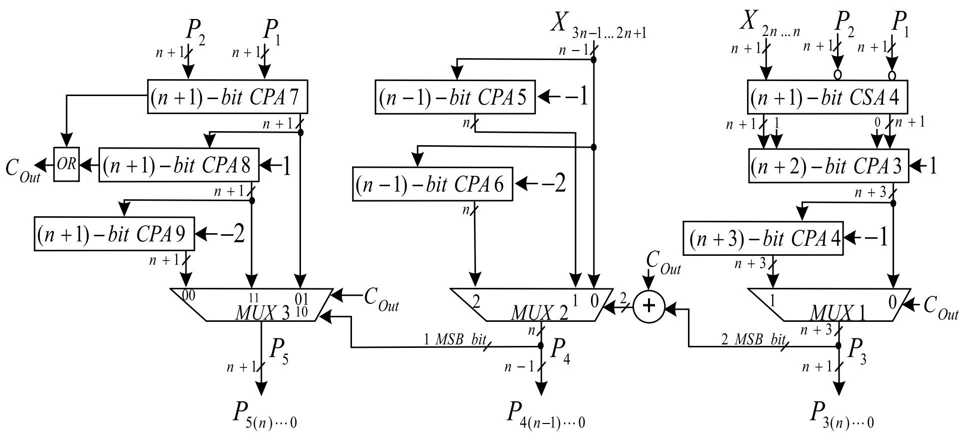

Fig. 2. Hardware implementation of the proposed improvements

© IEICE 2008
DOI: 10.1587/elex.5.672
Received June 30, 2008
Accepted July 11, 2008
Published September 10, 2008

4 Performance Evaluation

The proposed reverse converter consists of two level. The first level is the same as the first part of the converter of [6], and composed of a three-modulus reverse converter for calculating $X_{(3n-1)\dots 0}$, two $(n+1)$-bit CSAs, an $(n+1)$-bit CPA1 with EAC and a CSA tree (these components showed in Fig. 1, outside the dotted lines). The second level of the presented reverse converter substituted the components which showed within the dotted lines in Fig. 1, with those showed in Fig. 2.

The total conversion delay of the presented reverse converter is the delay of first level plus the delay of second level. The delay of the first level is the same with the first level of the converter of [6]. The critical delay path of the second level (Fig. 2) consists of an $(n + 1)$-bit CSA4, CPA2, CPA3, an FA and MUX 1, 2, 3.

Tab. I presents the total conversion delay and hardware complexity of the reverse converters. In this table, the unit gate model [3, 5] used to derive total area and delay estimations. In this model, each two-input monotonic gate, an XOR/XNOR gate and an FA counts as one, two and seven gates in area, respectively. Moreover, each FA has the delay of four unit gates. Also, the time complexity is the product of the total unit gate area and delay. As seen from Tab. I, the proposed adder-based reverse converter for moduli set $\{2^n - 1,\ 2^n,\ 2^n + 1,\ 2^{n+1} - 1\}$ is faster than original work. However, the hardware cost of the presented converter is more. But it is essential to remark to the point that, although the proposed converter consume more hardware but it demonstrated significant improvement in terms of speed and time complexity.

Table I. Performance Comparison

Converter	Area	Unit Gate Area	Delay	Unit Gate Delay	Time Complexity
[6]-CI	$(9n+5+(n–4)(n+1)/2)A_{FA}$ $+(2n)A_{XNOR}+(2n)A_{OR}$ $+(6n+1)A_{NOT}$	$(129n+7n^2)/2$ $+4$	$((23n+12)/2)t_{FA}$	$46n$	$161n^3$ $+2976n^2$
Proposed	$(10n+6+(n-4)(n+1)/2)A_{FA}$ $+(6n+2)A_{XNOR}+(6n+2)A_{OR}$ $+(7n+2)A_{NOT}+(n+3)A_{MUX2-1}$ $+(2n+1)A_{MUX3-1}$	$(193n+7n^2)/2$ $+50$	$((15n+22)/2)t_{FA}$	$30n$	$105n^3$ $+2895n^2$

5 Conclusion

This paper presents an improved reverse converter for the well-known RNS moduli set $\{2^n - 1,\ 2^n,\ 2^n + 1,\ 2^{n+1} - 1\}$. The hardware architecture of the proposed converter consists of two levels. The first level is the same as the first level of the original reverse converter. The second level composed of regular binary adders, without the need for using modular adders. Hence, the presented reverse converter results in significant reduction in the conversion delay and time complexity, compared to the original work.

DOI: 10.1587/elex.5.672
Received June 30, 2008
Accepted July 11, 2008
Published September 10, 2008