J. Basic. Appl. Sci. Res., 2(6)6167-6171, 2012

ISSN 2090-4304
Journal of Basic and Applied Scientific Research
www.textroad.com

Design and FPGA Implementation of an Improved RNS Converter

Amir Sabbagh Molahosseini[1], SomayyehJafarali Jassbi[2] and Saeid Sorouri[3]

[1] Department of Computer Engineering, Kerman Branch, Islamic Azad University, Kerman, Iran
[2] Faculty Member of Computer Department, Science and Research Branch, Islamic Azad University, Tehran, Iran
[3] Science and Research Branch, Islamic Azad University, Kerman, Iran

ABSTRACT

The recently introduced four-moduli set $\{2^n-1, 2^n, 2^n+1, 2^{2n+1}-1\}$ have the capability to provide high-performance residue number systems (RNS) due to its large dynamic range and well-formed moduli. There have been two RNS-to-binary converters for this moduli set up to now. The first one uses new Chinese remainder theorem 2 (CRT-II) to derive highly area-efficient converter but in expense of degrading the speed. The second one which is recently introduced tries to design highly speed-efficient converter using CRT-I but with significantly larger area than CRT-II-based converter. In this paper, we simplify the existing CRT-I-based converter for the moduli set $\{2^n-1, 2^n, 2^n+1, 2^{2n+1}-1\}$ to achieve a new RNS-to-binary converter which is faster than CRT-II-based converter while requires less area than CRT-I-based converter. The presented converter removes one $2n$-bit carry-save adder and one $2n$-bit carry-propagate adder from the architecture of the CRT-I-based converter.
KEYWORDS: Residue Number System, Digital Arithmetic, Residue to Binary Converter.

1. INTRODUCTION

Nowadays, with the extensive use of mobile and portable devices, the necessity of low-power and high-speed VLSI design is evident. The residue number system (RNS) has been known as an unconventional number system. RNS can be utilized to provide power-dissipation savings in the design of computation systems where addition, subtraction and multiplication are needed [1], [2]. However, binary-to-RNS and RNS-to-binary converters are required to act as interfaces between RNS and digital systems. The overhead of these converters can decrease the speed efficiency of RNS, and due to this a lot of research has been done to design efficient RNS converters [3].

In this work, we consider the design of RNS-to-binary converter for the recently suggested four-moduli set $\{2^n-1, 2^n, 2^n+1, 2^{2n+1}-1\}$. This moduli set has been named as arithmetic-friendly moduli set in [4], because of its largest modulo is in the form of 2^k-1. Furthermore, only one modulo of the type 2^k+1 is used. However, the moduli set $\{2^n-1, 2^n, 2^n+1, 2^{2n+1}-1\}$ relies on more complex RNS-to-binary conversion than the conversion-friendly moduli sets such as $\{2^n-1, 2^n, 2^n+1, 2^{2n}+1\}$ [5]. The design of first RNS-to-binary converter for the moduli set $\{2^n-1, 2^n, 2^n+1, 2^{2n+1}-1\}$ was mentioned in [4]. It uses CRT-II to construct a low-cost RNS-to-binary converter. Although, the pipelined use of this converter, [4], can provide high speed, using it directly results in larger delay compared with the converters designed for the conversion friendly four-moduli sets. Recently, a new RNS-to-binary converter for the moduli set $\{2^n-1, 2^n, 2^n+1, 2^{2n+1}-1\}$ is proposed in [6]. This converter has less delay and on the other hand a larger area compared to [4]. This paper proposes a new RNS-to-binary converter for the moduli set $\{2^n-1, 2^n, 2^n+1, 2^{2n+1}-1\}$ where $n\geq 4$, based on modification of some of the [6]'s conversion formulas. Moreover, FPGA implementations indicate that the resulted converter requires less hardware than [6], while it is faster than [4].

2. The Previous CRT-I Based Converter

This section briefly reviews the final conversion equations of the converter of [6]. It should be mentioned that the proof of the following equations has been described in [6]. Consider the moduli set $\{2^n, 2^{2n+1}-1, 2^n+1, 2^n-1\}$ with corresponding residues (x_1, x_2, x_3, x_4). The weighted number X can be computed using the following equation:

$$X = x_1 + 2^n(2^{2n+1}-1)Y + 2^n \times 2^{n+1}T \qquad (1)$$

Where

***Corresponding Author :**Amir SabbaghMolahosseini, Department of Computer Engineering, Kerman Branch, Islamic Azad University, Kerman, Iran. E-mail address:sabbagh@iauk.ac.ir.

$$Y = \begin{cases} \left|R_1+R_{22}+R_{31}+R'_{32}+R_4\right|_{2^{2n}-1} & if \quad x_2-x_1 \geq 0 \\ \left|R_0+R_1+R_{22}+R_{31}+R'_{32}+R_4\right|_{2^{2n}-1} & if \quad x_2-x_1 < 0 \end{cases} \tag{2}$$

$$T = \left|x_2 - x_1\right|_{2^{2n+1}-1} = \left|x_2 + \overline{x}_1\right|_{2^{2n+1}-1} \tag{3}$$

$$R_0 = \underbrace{1\cdots110}_{n-2}\underbrace{1\cdots11}_{n+1} \tag{4}$$

$$R_1 = \underbrace{x_{1,n-1}\cdots x_{1,1}x_{1,0}}_{n}\underbrace{0\cdots00}_{n} \tag{5}$$

$$R_{22} = \underbrace{\overline{x}_{2,n-2}\cdots\overline{x}_{2,1}\overline{x}_{2,0}}_{n-1}\underbrace{\overline{x}_{2,2n-1}\cdots\overline{x}_{2,n}\overline{x}_{2,n-1}}_{n+1} \tag{6}$$

$$R_{31} = \underbrace{x_{3,n}\cdots x_{3,1}x_{3,0}}_{n+1}\underbrace{0\cdots00}_{n-1} \tag{7}$$

$$R'_{32} = \overline{x}_{3,0}\underbrace{1\cdots11}_{n-3}\overline{x}_{2,2n}1\underbrace{\overline{x}_{3,n}\cdots\overline{x}_{3,2}\overline{x}_{3,1}}_{n} \tag{8}$$

$$R_4 = x_{4,0}\underbrace{x_{4,n-1}\cdots x_{4,1}x_{4,0}}_{n}\underbrace{x_{4,n-1}\cdots x_{4,2}x_{4,1}}_{n-1} \tag{9}$$

The equation (1) has been realized using the following equation

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

$$Z = Z_1 + Z_2 + Z_3 + 1 \tag{11}$$

$$Z_1 = \underbrace{0\cdots00}_{n-1}\underbrace{T_{2n}\cdots T_1T_0}_{2n+1}\underbrace{0\cdots00}_{n+1} \tag{12}$$

$$Z_2 = \underbrace{Y_{2n-1}\cdots Y_1Y_0}_{2n}\underbrace{0\cdots00}_{2n+1} \tag{13}$$

$$Z_3 = \underbrace{1\cdots11}_{2n+1}\underbrace{\overline{Y}_{2n-1}\cdots\overline{Y}_1\overline{Y}_0}_{2n} \tag{14}$$

The hardware architecture of the RNS-to-binary converter of [6] is based on equations (2) and (11). Implementation of (2) requires four $2n$-bit carry-save adders (CSAs) with end-around carry (EAC) and two $2n$-bit carry-propagate adders (CPAs) with EAC followed by a multiplexer. This is the main reason for the large area of the RNS-to-binary converter of [6]. Also, the select line of the multiplexer is connected to the carry-out of the CPA1 with EAC. This carry is available after the first round of addition. Realization of (11) has been done using a regular CSA followed by a CPA. Note that the carry vector of CSA5 is shifted one bit to the left, and its most significant bit should be ignored.

3. The Proposed Converter

In this section, we perform some modifications to the [6]'s conversion algorithm to attain an area-efficient RNS-to-binary converter than [6]. Our target is to reduce the number of the operands of the multi-operand modular

adder which is needed to realize (2). Hence, consider the operands (5) and (8). We can substitute the n least significant bits of these two binary vectors with each other as follows.

$$R_1' = \underbrace{x_{1,n-1}\cdots x_{1,1}x_{1,0}}_{n}\ \underbrace{\overline{x}_{3,n}\cdots\overline{x}_{3,2}\overline{x}_{3,1}}_{n} \tag{15}$$

$$R_{32}'' = \overline{x}_{3,0}\ \underbrace{1\cdots11}_{n-3}\overline{x}_{2,2n}1\underbrace{0\cdots00}_{n} \tag{16}$$

Next, with combination of (16) and (4), we achieve

$$R_{32}''' = \left|R_{32}'' + R_0\right|_{2^{2n}-1} = \overline{x}_{3,0}\ \underbrace{1\cdots11}_{n-4}\overline{x}_{2,2n}x_{2,2n}1\underbrace{0\cdots00}_{n} \tag{17}$$

This equation can be easily verified using Table 1. Note that end-around carry is considered to perform modulo 2^{2n}–1 addition. Therefore, we achieve

$$R_5 = \begin{cases} R_{32}''=\overline{x}_{3,0}\underbrace{1\cdots11}_{n-3}\overline{x}_{2,2n}1\underbrace{0\cdots00}_{n} & if\ \ x_2-x_1\geq 0 \\ R_{32}'''=\overline{x}_{3,0}\underbrace{1\cdots11}_{n-4}\overline{x}_{2,2n}x_{2,2n}1\underbrace{0\cdots00}_{n} & if\ \ x_2-x_1<0 \end{cases} \tag{18}$$

Consequently, we have the following equation for computing Y instead of (2):

$$Y = \left|R_1' + R_{22} + R_{31} + R_4 + R_5\right|_{2^{2n}-1} \tag{19}$$

These operands are described in (15), (6), (7), (9) and (18). It can be seen that six operands in converter of [6] are reduced to five operands which results in significant hardware savings. Note that the other parts of the conversion algorithm are the same as the converter of [6] (i.e. (10)-(14)). The proposed converter is shown in Fig. 1. Moreover, Table 2 presents details of the converter.

Table 1. Verification of equation (17)

$x_{3,0}$	$x_{2,2n}$	R_{32}''	$R_{32}''' = \left\|R_{32}'' + R_0\right\|_{2^{2n}-1}$
0	0	$11\cdots11110\cdots00$ ($n-3$, n)	$11\cdots11010\cdots00$ ($n-3$, n)
0	1	$11\cdots11010\cdots00$ ($n-3$, n)	$11\cdots11110\cdots00$ ($n-3$, n)
1	0	$01\cdots11110\cdots00$ ($n-3$, n)	$01\cdots11010\cdots00$ ($n-3$, n)
1	1	$01\cdots11010\cdots00$ ($n-3$, n)	$01\cdots11110\cdots00$ ($n-3$, n)

4. Performance Evaluation

The theoretical formula of the critical delay path of the proposed converter can be obtained as

$$\text{Delay}=t_{NOT}+(2n+1)t_{FA}+t_{MUX}+(4n+1)t_{FA}+t_{NOT}+(4n+2)t_{FA}=(10n+4)\ t_{FA}+2t_{NOT}+t_{MUX}$$

Note that t_{FA} and t_{NOT} denote the delay of one full adder (FA) and one NOT gate, respectively.

Table 2. Details of each part of the proposed converter

Part	NOT	FA	XNOR/OR pairs	XOR/AND pairs	Delay
OPU1	$4n+2$	–	–	–	t_{NOT}
CPA1	–	n	$n+1$	–	$(4n+2)t_{FA}$
CSA1	–	$2n$	–	–	t_{FA}
CSA2	–	$n+1$	–	$n-1$	t_{FA}
CSA3	–	4	$n-4$	n	t_{FA}
CPA2	–	$2n$	–	–	$(4n)t_{FA}$
OPU2	$2n$	–	–	–	t_{NOT}
CSA4	–	–	$2n$	$2n+1$	t_{FA}
CPA3	–	$4n+1$	–	–	$(4n+1)t_{FA}$

Table 3 compares the hardware requirements and conversion delays of the proposed converter with previous designs [4] and [6] in terms of logic gates and FAs. It can be seen that, although the proposed converter has more delay than [6], it needs less hardware requirements. Moreover, our converter relies on lower delay than the converter of [4]. Therefore, it can be summarized that the proposed converter results in a faster RNS-to-binary conversion than [4] while it demands fewer hardware requirements than [6].

Table 3. Hardware requirements and conversion delays

Converter	Hardware Requirements	Conversion Delay
[4]	$(8n+2)A_{FA} + (n-1)A_{XOR} + (n-1)A_{AND} + (4n+1)A_{XNOR} + (4n+1)A_{OR} + (7n+1)A_{NOT} + (n)A_{MUX2\times1}$	$(12n+5)t_{FA}+3t_{NOT}+t_{MUX}$
[6]	$(12n+4)A_{FA} + (6n-1)A_{XOR} + (6n-1)A_{AND} + (4n)A_{XNOR} + (4n)A_{OR} + (6n+2)A_{NOT} + (2n)A_{MUX2\times1}$	$(8n+6)t_{FA}+2t_{NOT}+t_{MUX}$
Proposed	$(10n+6)A_{FA} + (4n)A_{XOR} + (4n)A_{AND} + (4n-3)A_{XNOR} + (4n-3)A_{OR} + (6n+2)A_{NOT} + (2n)A_{MUX2\times1}$	$(10n+4)t_{FA}+2t_{NOT}+t_{MUX}$

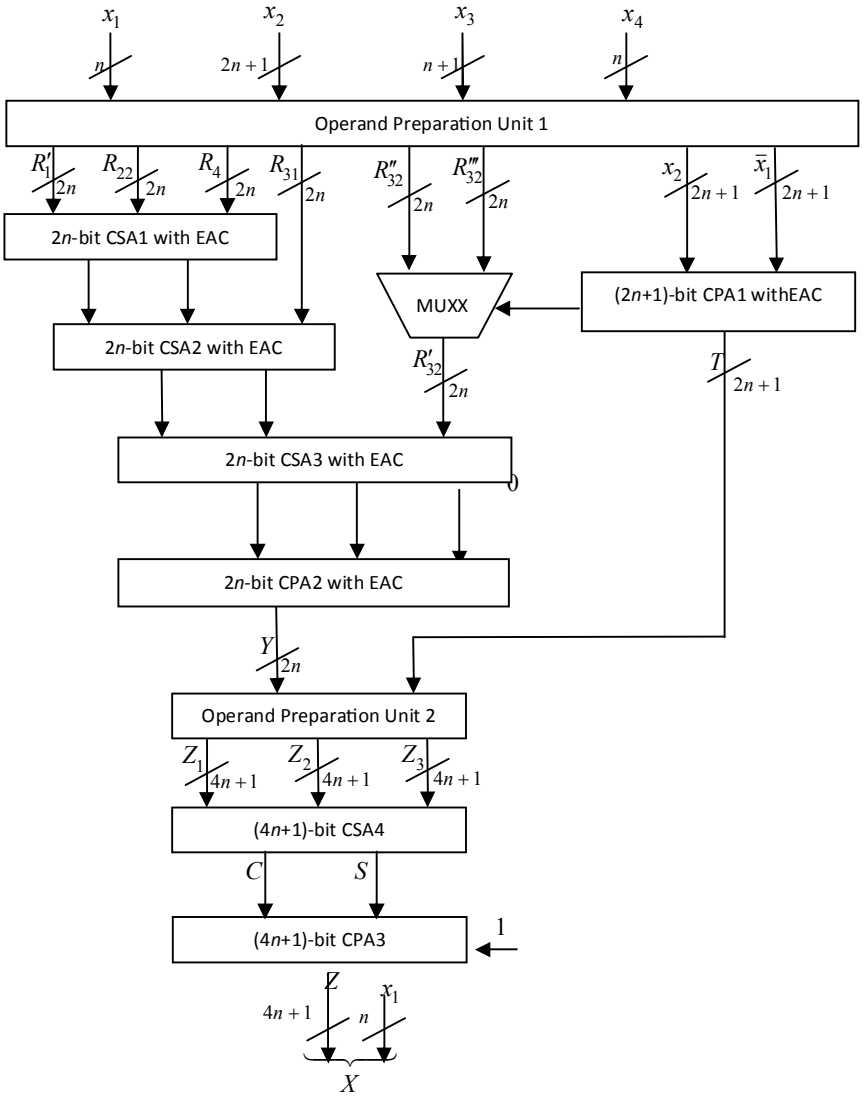

Fig. 1: The proposed converter.

In order to perform exact comparisons, we have described the proposed converter as well as designs of [4] and [6] in VHDL codes. Next, the codes were implemented on a Spartan-3 FPGA using Xilinx ISE v9.1. Table 4 presents the resulted data, i.e. delay (ns), area (in terms of number of LUTs) for different values of n. As expected,

the converters of [4] and [6] have the lowest and the highest area, respectively, and vice versa for delay. However the proposed converter stands in the middle.

Table 4. FPGA implementation results

N	Converter	Area (LUTs)	Delay (ns)
4	Proposed	132	60.805
	[6]	126	63.198
	[4]	159	51.955
8	Proposed	262	99.545
	[6]	243	106.538
	[4]	331	81.258
12	Proposed	380	139.436
	[6]	354	152.295
	[4]	485	111.856
16	Proposed	506	181.231
	[6]	479	192.505
	[4]	645	145.483

5. Conclusion

It is expected that the moduli set $\{2^n-1, 2^n, 2^n+1, 2^{2n+1}-1\}$ plays an important role in efficient realization of high-performance RNS-based computation systems where large dynamic range and high parallelism are required. We have presented a new RNS-to-binary converter for the moduli set $\{2^n-1, 2^n, 2^n+1, 2^{2n+1}-1\}$ by improving an existing previously design. Comparisons based on FPGA implementation have shown that the presented RNS-to-binary converter has the average performance among the earlier designs which are introduced in [4] and [6]. Not only it has less delay rather than [6], but also it consumes lower area than [4].

6. Acknowledgements

This work is supported by Kerman Branch, Islamic Azad University, Kerman, Iran.

7. REFERENCES

1. Stouraitis T. and V. Paliouras, 2001. Considering the alternatives in lowpower design. IEEE Circuits and Devices. **7**: 23-29.

2. Cardarilli G. C., A. D. Re, A. Nannarelli, and M. Re, 2007. Low power and low leakage implementation of RNS FIR filters. In the Proceedings of the Asilomar Conference on Signals, Systems and Computer. pp. 1620-1624.

3. Navi K., A.S. Molahosseini, and M. Esmaeildoust, 2011. How to Teach Residue Number System to Computer Scientists and Engineers. IEEE Transactions on Education, 54(1): 156-163.

4. Molahosseini A.S., K. Navi, C. Dadkhah, O. Kavehei, and S. Timarchi, 2010. Efficient Reverse Converter Designs for the New 4-Moduli Sets $\{2^n-1, 2^n, 2^n+1, 2^{2n+1}-1\}$ and $\{2^n-1, 2^n+1, 2^{2n}, 2^{2n}+1\}$ Based on New CRTs. IEEE Transactions on Circuits and Systems-I, 57(4):823-835.

5. Cao B., C. H. Chang and T. Srikanthan, 2003. An Efficient Reverse Converter for the 4-Moduli Set $\{2^n-1, 2^n, 2^n+1, 2^{2n}+1\}$ Based on the New Chinese Remainder Theorem. IEEE Transactions on Circuits and Systems-I, 50(10): 1296-1303.

6. Molahosseini A.S., 2011. Improving the Delay of Residue-to-Binary Converter for a Four-Moduli Set. Advances in Electrical and Computer Engineering, 11(2): 37-42.