The CSI Journal on
Computer Science and Engineering
Vol. 16, No. 2, 2019
Pages 38-43
**Regular Paper**

# An RNS Comparator via Dynamic Range Partitioning: The Case of $\{2^n - 1, 2^n, 2^{n+1} - 1\}$

Zeinab Torabi [1] Armin Belghadr [2]

[1] Department of Computer Engineering, Shahid Rajaee Teacher Training University, Tehran, Iran
[2] Department of Computer Science and Engineering, Shahid Beheshti University, Tehran, Iran

## Abstract

It is common to speed-up addition, subtraction and multiplication via Residue Number Systems (RNS). The key advantage of the RNS is its limited carry propagation scheme, which is due to breaking down the long-word operations into a number of independent small-word parallel operations. Applications with repeated use of addition and multiplication have mostly benefited from RNS, while lack of efficient realizations for other operations does not allow for wider range of applications. Comparison, as a basic building block for other difficult RNS operations such as division, has been subject of numerous studies. Comparison via dynamic range partitioning has been shown to be the most successful implementation for the 3-moduli set $\{2^n, 2^n \pm 1\}$. In this paper, we proposed an efficient RNS comparator for the moduli set $\{2^n - 1, 2^n, 2^{n+1} - 1\}$ via dynamic range partitioning technique. Synthesis results reveal 27.5% delay, 64% area, 59.7% power dissipation and 71% energy consumption reduction for the proposed design against straightforward comparator (i.e. reverse conversion followed by binary comparator)

**Keywords:** Computer Arithmetic, Residue Number System, Difficult Operations, Dynamic Range Partitioning, Comparison.

## 1. Introduction

Nowadays many applications demand for high speed of operations, less power consumption and smaller area footprint. Residue number system (RNS) considered as a vehicle for these requirements in special-purpose systems.
Advancements of the RNS in developing adders and multipliers has made its way through finite impulse response digital filters [1], data transmission [2], cryptography [3], image processing [4] and many other applications that frequently employ additions and multiplications in their instructions. However, lack of appropriate implementations for division, comparison, sign and overflow detection (called difficult operations) has been an obstacle to its popularity among other applications. Despite complications in the implementations of the called difficult operations, high speed and low power consumption in RNS still make it an appropriate case of study. For example, one of the Intel's largest research projects in the year 2012 was adoption of the RNS as a solution to reduce power consumption of today computers [5].

Regarding prominent role of the comparison in the development of sign and overflow detection and division units in RNS, it is anticipate that a cost-effective and high-speed implementation of the comparison will be also profitable to other difficult operations. Indeed, via introduction of appropriate implementations for the difficult operations, RNS finds suitability in broader ranges of applications.

Comparison in RNS considered as a complex operation chiefly because RNS is not a weighted number system. For example, in an RNS with moduli set $\{2, 3, 5, 7\}$, $36 = <$

$0,0,1,1 >$ is almost twice $19 =< 1,1,4,5 >$, while it is not perceptible in their modular representations.

Whenever an application necessitates denote both positive and negative values, sign and overflow detection operations are inevitable. In order to detect overflow in such systems one can use a comparator to compare any value against a specific number in the domain. In case the value is larger than the specific value, overflow has happened. For example, if we select $\frac{M}{2}$ as an indicator point and consider larger (smaller) numbers as positive (negative) values, we can simply detect sign of any number through a comparison against $\frac{M}{2}$. Division is another difficult operation in RNS that requires comparison operation in its implementation.

Moduli set $\varphi = \{2^n - 1, 2^n, 2^{n+1} - 1\}$ has been introduced in [ŕ] which inherits two of its moduli $2^n - 1$ and $2^n$ from the popular moduli set $\{2^n - 1, 2^n, 2^n + 1\}$ while it substitutes the modulo $(2^n + 1)$ with a faster choice modulo $2^{n+1} - 1$. As a result, larger dynamic range is acquired and the RNS-to-binary conversion is slightly lengthened. Nevertheless, since the conversion process is not frequent, burden of the lengthier reverse conversion is bearable.

A recent study reveals that dynamic range partitioning (DRP) [7] technique gains the best performance for the modular comparators of the well-known moduli set $\{2^n - 1, 2^n, 2^n + 1\}$. In this paper, we take advantage of the DRP and propose a new comparator for the moduli set $\varphi$. Since no modular comparator for the aforementioned modulo set found in the literature, we opt on the comparison with a non-modular comparator, which preceded by RNS-to-binary conversion process. Merits of the proposed design against reference work studied and presented through analytical and synthesis-based experiments.

In the rest of this paper, Section 2 reviews different RNS comparison methods. In Section 3, new comparator for $\varphi$ based on DRP proposes, whereas its detailed implementation provides in Section 4. Evaluations and experimental results found in Section 5 and Finally Section 6 presents our concluding remarks.

## 2. Background Material

A number $X$ in an RNS with $k$ mutually prime moduli $\{m_1, m_2, \dots, m_k\}$ is represented with its remainders corresponding to each moduli as $\{x_1, x_2, \dots, x_k\}$, where $x_i = |X|_{m_i}$ and $M = m_1 \times m_2 \dots \times m_k$. As a result, any long word operation is split into $k$ small word parallel operation corresponding to $k$ moduli.

There are several techniques for RNS comparison that can be categorized as follows [7]:

1) Conversion-Based:

   A straightforward solution is to convert modular numbers into binary format and then perform comparison using conventional binary comparators. This scheme demands for undesirable reverse conversions prior to its final binary comparison. However this can be improved via starting comparison earlier than the conversions being finished [8, 9]. The incomplete reverse conversion is carried out via one of the variants of the Chinese remainder theorem (CRT) [10], the mixed radix conversion (MRC) [10], a combination of both, or via special converters.

2) Subtraction-Based:

   This solution compares RNS operands based on the sign of their difference in binary format [11, 12].

3) Parity Checking [13]:

   In this method, instead of converting numbers (or their difference) into binary format, parity of the numbers and their difference is used to compare them [14, 15]. However, such methods require all the modulus to be odd, where in practice virtually all the RNS moduli sets include a power-of-two modulo, corresponding to their most efficient arithmetic channel.

4) Diagonal Mapping [16]:

   It is shown that numbers in a $k$-moduli set RNS can be mapped into a $k$-dimension space, so that numbers are arranged as diagonal lines (for k = 2), diagonal surfaces (for k = 3), and so on. Every number $X$ in the dynamic range is assigned to a specific diagonal $D(X)$. Comparison of the numbers is based on their diagonals values. A concrete description of this method on specific moduli set is offered in [16].

5) Dynamic Range Partitioning:

   This method is based on ordering the whole dynamic range into successive partitions, and detecting the partitions that own the corresponding comparison operands [7]. For this purposed it requires to compute $p_1$ and $p_2$ for both of operands as shown in equation set 1. This equation set describes DRP components for any 3 moduli set $\{m_1, m_2, m_3\}$, wherein $x_{23} = |X|_{m_2 m_3}$, $x_2 = |x_{23}|_{m_2}$ and $x_3 = |x_{23}|_{m_3}$.

$$\begin{cases} p_1(X) = \left| \left| {M_1}^{-1} \right|_{m_1} (x_1 - x_{23}) \right|_{m_1} \\ p_2(X) = \left| |{m_3}^{-1}|_{m_2} (x_2 - x_3) \right|_{m_2} \end{cases} \quad (1)$$

   It is also worth mentioning that this technique implies no restriction on the value of the moduli (such as being odd, prime, etc.).

## 3. $\varphi$ −Comparator Via DRP

DRP makes comparison much easier via sorting numbers in ascending order. With partitioning of an arbitrary moduli set, such as $\{m_1, m_2, m_3\}$, daynamic range is divided into $m_1$ partitions of size $m_2 \times m_3$ that are assigned with integer primary identifiers ($p_1$) in $[0, m_1)$ and each partition is divided to $m_2$ sections of size $m_3$, with the corresponding integer identifiers ($p_2$) in $[0, m_2)$.

Comparison of two numbers $X = (x_1, x_2, x_3)$ and $Y = (y_1, y_2, y_3)$ can be reduced to the comparison of their partition numbers (represented by $p_1(x)$ and $p_1(y)$), although $p_1(x) =$

$p_1(y)$ does not necessarily mean $X = Y$. In that case, section numbers (represented by $p_2(x)$ and $p_2(y)$) are required. In case $p_2(x) = p_2(y)$, comparison of the $x_3$ and $y_3$ designates comparison result for the $X$ and $Y$.
For example, in the moduli set $\{3,4,7\}$ three partitions with size $7 \times 4 = 28$ exist so that $p_1(13) = 0 < p_1(47) = 1$ and $p_1(29) = 1 = p_1(39)$ but $p_2(29) = 1 < 2 = p_2(39)$.
Let $m_1 = 2^n - 1, m_2 = 2^n$ and $m_3 = 2^{n+1} - 1$, $p_1(X)$ and $p_2(X)$, can be computed via following equations:

$$x_{23} = x_3 + (2^{n+1}-1)||(2^{n+1}-1)^{-1}|_{2^n}(x_2 - x_3)|_{2^n}$$
$$p_1(X) = \left| \begin{matrix} \left|(2^n(2^{n+1}-1))^{-1}\right|_{2^n-1} \\ (x_1 - x_3 - (2^{n+1}-1)||(2^{n+1}-1)^{-1}|_{2^n}(x_2 - x_3)|_{2^n}) \end{matrix} \right|_{2^n-1} \quad (2)$$
$$p_2(X) = ||(2^{n+1}-1)^{-1}|_{2^n}(x_2 - x_3)|_{2^n}$$

Replacing on multiplicative inverses (see properties 1 and 2) in $p_1$ and $p_2$ leads us to equation set 3.

**Property 1:** $\left|(2^n(2^{n+1}-1))^{-1}\right|_{2^n-1} = 1$

**Property 2:** $|(2^{n+1}-1)^{-1}|_{2^n} = -1$

$$p_1(X) = \left| \begin{matrix} x_1 - x_3 \\ -(2^{n+1}-1)|x_3 - x_2|_{2^n} \end{matrix} \right|_{2^n-1} \quad (3)$$
$$p_2(X) = |x_3 - x_2|_{2^n}$$

## 4. Implementation

Here, we provide implementation details for the proposed comparator in $\varphi$.
Seeing that $\overline{x_2} = 2^n - 1 - x_2$, we assume $n$-bit $p_2(X) = w$ as in equation 4.

$$p_2(X) = |x_3 - x_2|_{2^n} = |x_3 + \overline{x_2} + 1|_{2^n} = w = w_{n-1} \dots w_0 \quad (4)$$

Since $\overline{x_3} = 2^{n+1} - 1 - x_3$, we can simplify $p_1(X)$ as in equation 5.

$$p_1(X) = \left| \begin{matrix} x_1 - x_3 \\ -(2^{n+1}-1)|x_3 - x_2|_{2^n} \end{matrix} \right|_{2^n-1}$$
$$= \left| \begin{matrix} x_1 + \overline{x_3} - 2^{n+1} + 1 \\ -(2^{n+1}-1)|x_3 - x_2|_{2^n} \end{matrix} \right|_{2^n-1}$$
$$= |x_1 + \overline{x_3} - 1 - |x_3 - x_2|_{2^n}|_{2^n-1}$$
$$= |x_1 + \overline{x_3} - 1 - w|_{2^n-1}$$
$$= |x_1 + \overline{x_3} - 1 + \overline{w} - 2^n + 1|_{2^n-1}$$
$$= |x_1 + \overline{x_3} + \overline{w} + 2^n - 2|_{2^n-1} \quad (5)$$

Figure 1 illustrates weighted bit arrangements for the 4 terms of $p_1(X)$ as shown in equation 5 where $a_{n-1} \dots a_0$ and $c_n \dots c_0$ represent $x_1$ and $x_3$, respectively.

| $2^{n-1}$ | $2^{n-2}$ | $2^{n-3}$ | ... | $2^2$ | $2^1$ | $2^0$ |
|---|---|---|---|---|---|---|
| $a_{n-1}$ | $a_{n-2}$ | $a_{n-3}$ | ... | $a_2$ | $a_1$ | $a_0$ |
| $\overline{c_{n-1}}$ | $\overline{c_{n-2}}$ | $\overline{c_{n-3}}$ | ... | $\overline{c_2}$ | $\overline{c_1}$ | $\overline{c_0}$ |
| 1 | 1 | 1 | ... | 1 | 1 | $\overline{c_n}$ |
| $\overline{w_{n-1}}$ | $\overline{w_{n-2}}$ | $\overline{w_{n-3}}$ | ... | $\overline{w_2}$ | $\overline{w_1}$ | $\overline{w_0}$ |

Figure 1. Bit arrangements for the terms in $p_1(X)$

The four rows of Figure 1 can be reduced to three as is demonstrated in Figure 2 wherein $\oplus$ and $\odot$ symbolize XOR and XNOR functions, respectively.
Block diagram of Figure 3 shows how $p_2(X)$ is produced as a byproduct of $p_1(X)$.

| $2^{n-1}$ | $2^{n-2}$ | ... | $2^2$ | $2^1$ | $2^0$ |
|---|---|---|---|---|---|
| $a_{n-1} \odot \overline{c_{n-1}}$ | $a_{n-2} \odot \overline{c_{n-2}}$ | ... | $a_2 \odot \overline{c_2}$ | $a_1 \odot \overline{c_1}$ | $a_0 \oplus \overline{c_0} \oplus \overline{c_n}$ |
| $a_{n-2} \vee \overline{c_{n-2}}$ | $a_{n-3} \vee \overline{c_{n-3}}$ | ... | $a_1 \vee \overline{c_1}$ | $a_0\overline{c_0} \vee \overline{c_n}(a_0 + \overline{c_0})$ | $a_{n-1} \vee \overline{c_{n-1}}$ |
| $\overline{w_{n-1}}$ | $\overline{w_{n-2}}$ | ... | $\overline{w_2}$ | $\overline{w_1}$ | $\overline{w_0}$ |

Figure 2. Reduced $p_1(X)$

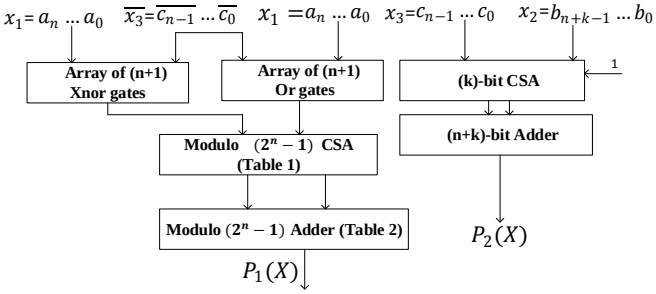


Figure 3. Block diagram for the $p_1(X)$ and $p_2(X)$ generator

After computation of partition and section numbers of both operands, Block diagram of Figure 4 summarizes how complex comparison of two RNS numbers is accomplished using $n$-bit comparators in the proposed method.

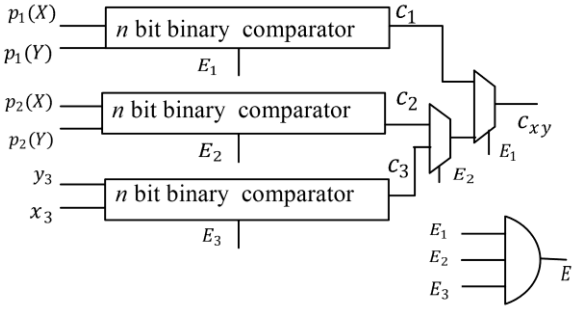


Figure 4. High level block diagram for the proposed comparator [7]

## 5. Evaluation and Comparison

Since no modular comparator for the moduli set $\{2^n - 1, 2^n, 2^{n+1} - 1\}$ has been presented so far, we opt on the evaluation of the proposed design against a long-word non-modular comparator that is preceded by an RNS-to-binary converter [6]. Tables 1 and 2 show analytical evaluations for the delay and area of the proposed and reference design, based on the unit gate model, respectively. In our analytical evaluations we consider area and delay of one simple 2-input logic gate (e.g., AND, OR, NAND, NOR) as 1 unit of area ($A_G$) and delay ($\Delta_G$), except for the XOR and XNOR gates that claim 2 and 3 times of the delay ($2\Delta_G$) and area ($3A_G$) of one unit gate.

In order to find better insight into merits of the proposed design, all the required hardware descriptions for the proposed and reference design for $n = 8$ were implemented and simulated to verify their correct functioning. The corresponding HDL codes were then mapped to the CMOS standard cell library of the 90nm technology node of the TSMC, using the Synopsys Design Compiler. As it was anticipated from our analytical evaluations in Tables 1 and 2, Table 3 confirms superiority of the proposed comparator in comparison with the reference design, in terms of delay, area, power and the power-delay product (PDP), based on the experimental result.

Figures 5 and 6 provide better visibility for the power and area measures, wherein the labels in the horizontal axis display different timing constraints applied in the syntheses iterations.

Table 1. Analytical delay comparison

| Method | CSA | Adder | | Comparator | | Total Delay |
|---|---|---|---|---|---|---|
| | | $n$ -bit | $(2n+3)$-bit | $n$ –bit | $3n$ –bit | |
| **[6]** | 4 | - | 1 | - | 1 | $(10n+22)\Delta_G$ |
| **proposed** | 1 | 2 | - | 1 | - | $(6n+4)\Delta_G$ |

Table 2. Analytical area comparison

| Method | CSA | Adder | | MUX | XOR/ XNOR | AND /OR | Comparator | | Total Area |
|---|---|---|---|---|---|---|---|---|---|
| | | $n$ -bit | $(2n+3)$-bit | | | | $n$ –bit | $3n$ –bit | |
| **[6]** | $12n+18$ | - | 3 | $2n+1$ | - | - | - | 1 | $(153n+192)A_G$ |
| **proposed** | $n$ | 2 | - | 2 | $n+1$ | $n+1$ | 3 | - | $(46n+4)A_G$ |

Table 3. Synthesis based comparison results

| Design | Delay ($ns$) | Ratio | Area ($\mu m^2$) | Ratio | Power ($mW$) | Ratio | PDP (pJ) | Ratio |
|---|---|---|---|---|---|---|---|---|
| **[6]** | 2.5 | 1.38 | 7298.02 | 2.78 | 2.11 | 2.48 | 5.27 | 3.45 |
| **Propposed** | 1.8 | 1.00 | 2624.12 | 1.00 | 0.85 | 1.00 | 1.53 | 1.00 |

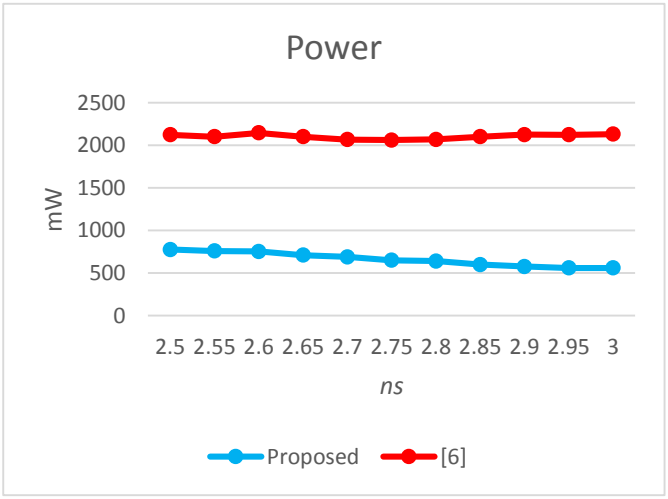


Figure 5. Power comparison for the proposed and reference comparator

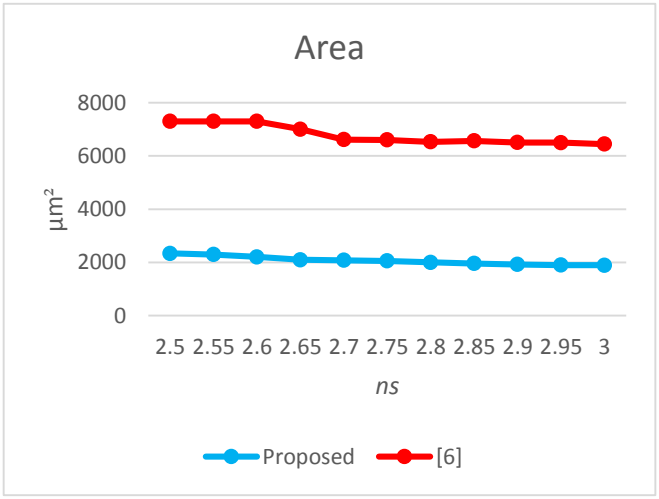


Figure 6. Area comparison for the proposed and reference comparator

# 6. Conclusion

Despite advantages of the addition, subtraction and multiplication, various studies consider division, comparison, sign and overflow detection as difficult operations in RNSs. However, an efficient comparator enables RNSs effectively implement difficult operations. In this paper, a new $\varphi-$comparator via dynamic range partitioning is proposed. Synthesis-based experiments showed 27.5%, 64%, 59.7% and 71% delay, area, power and energy improvements, respectively, for the proposed design in comparison with the straightforward comparator. For the future work, we plan to study impact of the newly proposed comparator in the implementation of other difficult operations in moduli set $\varphi$.

# References


[1] G. L. Bernocchi, G. C. Cardarilli, A. Del Re, A. Nannarelli, and M. Re, "Low-power adaptive filter based on RNS components," *Proc. IEEE Int. Symp. Circuits Syst.*, pp. 3211–3214, 2007.

[2] A. S. Madhukumar and F. Chin, "Enhanced architecture for residue number system-based CDMA for high-rate data transmission," *IEEE Trans. Wireless Communications*, vol. 3, no. 5, pp. 1363-1368, 2004.

[3] D. M. Schinianakis, A. P. Fournaris, H. E. Michail, A. P. Kakarountas, and T. Stouraitis, "An RNS implementation of an *Fp* elliptic curve point multiplier," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 56, no. 6, pp. 1202–1213, 2009.

[4] W. Wang, M. N. S. Swamy, and M. O. Ahmad, "RNS application for digital image processing," *Proc. 4th IEEE Int. Workshop System-Chip Real-Time Appl.*, pp. 77–80, 2004.

[5] Elise King, "kumar investigates residue number system to reduce power usage," https://www.ece.illinois.edu/newsroom/article/1578. 2012.

[6] 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: Express Briefs* 54, no. 9, pp. 775-779, 2007.

[7] Z. Torabi and G. Jaberipur. "Low-power/cost RNS comparison via partitioning the dynamic range." *IEEE Transactions on Very Large Scale Integration (VLSI) Systems vol.* 24, no. 5, pp. 1849-1857, 2016.

[8] S. Bi and W. J. Gross, "The mixed-radix Chinese remainder theorem and its applications to residue comparison," *IEEE Trans. Comput.*, vol. 57, no. 12, pp. 1624–1632, Dec. 2008.

[9] Y. Wang, X. Song, M. Aboulhamid, and H. Shen, "A new algorithm for RNS magnitude comparison based on new Chinese remainder theorem II," *Proc. 9th Great Lakes Symp. VLSI*, pp. 362–365, 1999.

[10] N. S. Szabó and R. I. Tanaka, *Residue Arithmetic and Its Applications to Computer Technology*. New York, NY, USA: McGraw-Hill, 1967.

[11] E. Gholami, R. Farshidi, M. Hosseinzadeh, and K. Navi, "High speed residue number system comparison for the moduli set $\{2^n-1, 2^n, 2^n+1\}$," *J. Commun. Comput*, vol. 6, no. 3, pp. 40–46, 2009.

[12] S. T. Eivazi, M. Hosseinzadeh, and O. Mirmotahari, "Fully parallel comparator for the moduli set $\{2^n-1, 2^n, 2^n+1\}$," *IEICE Electron. Exp.*, vol. 8, no. 12, pp. 897–901, 2011.

[13] M. Lu and J.-S. Chiang, "A novel division algorithm for the residue number system," *IEEE Trans. Comput.*, vol. 41, no. 8, pp. 1026–1032, 1992.

[14] L. Sousa, "Efficient method for magnitude comparison in RNS based on two pairs of conjugate moduli", Proceeding of *IEEE Symposium on Computer Arithmetic, ARITH*, pp. 240–250, 2002.

[15] B. Zarei, , M. Askarzadeh, , N. Derakhshanfard, and M. Hosseinzadeh, "A high-speed Residue Number comparator For the 3-Moduli Set $\{2^n-1, 2^n+1, 2^n+3\}$ ", In *Proceedings of International Symposium on Signals Systems and Electronics (ISSSE)*, Vol. 1, pp. 1–4, 2010.

[16] G. Dimauro, S. Impedovo, G. Pirlo, and A. Salzo, "RNS architectures for the implementation of the 'diagonal function," *Inf. Process. Lett*, vol. 73, nos. 5–6, pp. 189–198, 2000.



**Zeinab Torabi** received her B.Sc. degree in Computer Science and Ph.D. degree in Computer System Architecture from Shahid Beheshti University, Tehran, Iran, in 2005 and 2016 respectively. She received her M.Sc. degree in Computer Engineering (Algorithms and Computations) from University of Tehran, Tehran, Iran, in 2009.
She is currently an Assistant Professor of Computer Engineering in Department of Computer Engineering, Shahid Rajaee Teacher Training University, Tehran, Iran. Her research interests include computer arithmetic, residue number system, design and analysis of algorithms.
**Email:** z.torabi@sru.ac.ir



**Armin Belghadr** received the B.S. degree in Computer Hardware Engineering and the M.S. degree in Computer Architecture from Shahid Beheshti University, Tehran, Iran, in 2011 and 2013, respectively.
He is currently a Ph.D. Candidate in Computer Architecture at the Department of Computer Science and Engineering of Shahid Beheshti University, Tehran, Iran. He has been pursuing teaching and research in the mainstreams of

computer-aided design (CAD), computer arithmetic, and three-dimensional field programmable gate arrays (FPGA). His research interests include computer arithmetic, particularly residue and redundant number systems.
**Email:** a_belghadr@sbu.ac.ir

**Paper Handling Data:**

Submitted: 01.06.2019
Received in revised form: 05.15.2019
Accepted: 05.28.2019
Corresponding author: Zeinab Torabi
Affiliation of the corresponding author: Shahid Rajaee Teacher Training University