A High Speed Residue-to-Binary Converter for Balanced 4-Moduli Set

MohammadReza Taheri, Nasim Shafiee, Mohammad Esmaeildoust, Zhale Amirjamshidi, Reza Sabbaghi-nadooshan, Keivan Navi

A R T I C L E   I N F O.
Article history:
Received: 6 January 2014
Revised: 7 November 2015
Accepted: 7 December 2015
Published Online: 7 February 2016
Keywords:
Mixed Radix Conversion, Residue Arithmetic, Residue Number System, Residue-to-Binary Converter

A B S T R A C T
The moduli set \( \{2^{n-1} - 1, 2^{n+1} - 1, 2^n, 2^n - 1\} \) has been recently proposed in literature for class of \( 4n \)-bit dynamic range in residue number system. Due to only utilizing modulus in the form of \( 2^k - 1 \) besides modulo \( 2^n \), this moduli set enjoys the efficient Arithmetic Unit (AU) in its architecture. Not only does the efficiency of a RNS system depend on the residue arithmetic unit but it also is limited to the residue to binary converter. In this paper, a new two level residue-to-binary converter architecture based on Mixed Radix Conversion (MRC) is presented for the aforementioned moduli set. The proposed converter includes two levels of design based on MRC properties. Firstly, the 3-moduli subset \( \{2^{n-1} - 1, 2^{n+1} - 1, 2^n - 1\} \) is properly organized and as it does not calculate several values, it results in some cost modifications. Eventually, a two-moduli set \( \{(2^{n-1} - 1)(2^{n+1} - 1)(2^n - 1), 2^n\} \) is formed to compute the binary of RNS counterpart. The proposed architecture is shown to be more efficient both in terms of hardware cost and conversion delay in comparison with the related state-of-the-art works.

1 Introduction
The carry-free nature of the residue number system (RNS) makes it suitable to be used in the arithmetic level in VLSI design to achieve parallelism [1], [2]. In RNS, a weighted number is decomposed into a set of residues. Since arithmetic operations on residues can be performed without carry propagation between them, RNS results in high-speed addition, subtraction, and multiplication [3], [4], which is appropriate for digital signal processing (DSP) [5], [6], image processing [7], cryptography [8], [9] and communication systems [10]. However, arithmetic operations like division, sign detection and comparison are difficult in RNS.

RNS includes three main parts: the binary-to-residue (forward) converter, arithmetic operator and residue-to-binary (reverse) converter. The forward converter transforms a weighted binary number into residue numbers, based on the moduli set. The arithmetic unit generally contains modular adder, subtractor and multiplier. The reverse converter transforms residue numbers into a weighted binary number [11]. An appropriate choice of moduli set determines the efficiency of forward conversion, arithmetic operation...
and reverse conversion. A reverse converter has more complex architecture and its complexity will grow depending on the number of modules. Therefore, an effective design of reverse converter is needed in order to get the benefit of the RNS [12].

Many works have been reported on balanced 4-moduli sets such as \( \{2^n - 1, 2^n, 2^n+1 - 1\} \) [13–15], \( \{2^n - 1, 2^n, 2^n+1, 2^n+1 - 1\} \) [13, 15], \( \{2^n - 3, 2^n - 1, 2^n + 1, 2^n + 3\} \) [16] and \( \{2^n, 2^n+1 - 1, 2^n - 1, 2^{n-1} - 1\} \) [17]. Efficiency of arithmetic operations is restricted to critical modulus. The critical moduli in [13–17] are shown in Table 1. The unit gate delays of the parallel prefix adders \( 2^k - 1, 2^k + 1 \) and \( 2^k + 3 \) are \( 2\log_2 n + 3, 2\log_2 n + 6 \) and \( 2\log_2 (n + 1) + 6 \), respectively [18–20]. Therefore, as it is shown in Table 1, moduli set \( \{2^n, 2^n+1 - 1, 2^n - 1, 2^{n-1} - 1\} \) [17] provides more efficient arithmetic unit. However, more efficient reverse converter for the moduli set \( \{2^n, 2^n+1 - 1, 2^n - 1, 2^{n-1} - 1\} \) with less hardware requirements and delay, compared to [17] and other moduli sets in literature, is needed. Therefore, in this paper, a new design of the reverse converter for the 4-moduli set is presented. The proposed converter has achieved less delay and more desirable hardware requirements compared to the state-of-the-art converters.

This paper consists of a background about RNS in Section 2, design of the proposed RNS to binary converter in Section 3, evaluation of hardware requirements and critical path delay of the proposed reverse converter in Section 4, comparison of the performance of the proposed RNS to binary converter with other moduli sets in Section 5 and finally the conclusions of the paper in Section 6.

### Table 1. Comparison of arithmetic operation for different moduli sets for high dynamic range applications

<table>
<thead>
<tr>
<th>Moduli set</th>
<th>Design</th>
<th>Critical modulus</th>
<th>Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>( {2^n - 1, 2^n, 2^n+1 - 1} )</td>
<td>[13, 14]</td>
<td>( 2^n + 1 )</td>
<td>( 2\log_2 n + 6 )</td>
</tr>
<tr>
<td>( {2^n - 1, 2^n, 2^n+1, 2^n+1 - 1} )</td>
<td>[13, 15]</td>
<td>( 2^n+1 + 1 )</td>
<td>( 2\log_2 (n + 1) + 6 )</td>
</tr>
<tr>
<td>( {2^n - 3, 2^n - 1, 2^n + 1, 2^n + 3} )</td>
<td>[16]</td>
<td>( 2^n + 3 )</td>
<td>( 2\log_2 (n + 1) + 7 )</td>
</tr>
<tr>
<td>( {2^n, 2^n+1 - 1, 2^n - 1, 2^{n-1} - 1} )</td>
<td>[17]</td>
<td>( 2^{n+1} - 1 )</td>
<td>( 2\log_2 (n + 1) + 3 )</td>
</tr>
</tbody>
</table>

Reverse conversion algorithms are principally based on the Chinese remainder theorem (CRT), Mixed-radix conversion (MRC) and new Chinese remainder theorems (New CRTs) [11]. Through the MRC, the number \( X \) can be calculated using

\[
X = v_0 \prod_{i=1}^{n-1} P_i + \cdots + v_3 P_2 P_1 + v_2 P_1 + v_1 \tag{1}
\]

The coefficient \( \{v_1, v_2, \ldots, v_n\} \) can be obtained from the following formulas:

\[
v_1 = x_1 \tag{2}
\]

\[
v_2 = \left( x_2 - v_1 \right) \left| P_1^{-1} \right| P_2 \tag{3}
\]

\[
v_3 = \left( \left( x_3 - v_1 \right) \left| P_1^{-1} \right| P_3 - v_2 \right) \left| P_2^{-1} \right| P_3 \tag{4}
\]

In general

\[
v_n = \left( \left( x_n - v_1 \right) \left| P_1^{-1} \right| P_n - v_{n-1} \right) \left| P_2^{-1} \right| P_n - \cdots - v_{n-1} \left| P_{n-1}^{-1} \right| P_n \tag{5}
\]

Where \( P_i^{-1} \left| P_j \right. \) is the multiplicative inverse of \( P_i \) modulu \( P_j \) [11].

Three types of adders are used to realize the hardware architecture of the reverse converter, Carry Save Adder (CSA) for operations in modulo \( 2^n \), CSA with End Around Carry (EAC) for operations in modulo \( 2^k - 1 \), Carry Propagate Adder (CPA) and Modular Adder (MA). For MA in modulo \( 2^k - 1 \), CPA with end around carry (EAC) is used, which has the similar area and double delay in comparison with a regular CPA [22]. These are explained more in Section 4.

### 3 Proposed RNS to Binary Converter

The two-level architecture, realized by the MRC method, can lead to an efficient implementation of RNS to binary converter of moduli set \( \Psi = \{2^{n-1} - 1, 2^n+1 - 1, 2^n - 1\} \). In the first step, number \( Y \) is calculated from the residues in the subset \( \Gamma = \{2^{n-1} - 1, 2^n+1 - 1, 2^n - 1\} \) by
using MRC in a parallel manner. In the second step, the MRC method is applied to the superset \( \Lambda = \{2^{n-1} - 1, 2^{n+1} - 1, 2^n, 1\} \) and the final result is realized. The proposed reverse converter scheme is composed of two parts, as shown in Figure 1. The details are presented in the next subsections.

3.1 First Step Design

In the first step, the reverse converter of the subset \( \Gamma \) is designed. In order to decrease the delay generated by the serial attribute of the MRC method, the proposed approach in [23] is utilized. Using this approach, more parallelism is obtained without noteworthy hardware redundancy. Also to reduce the total architecture delay, in the first step, all moduli which are in the form of \( 2^k - 1 \) and modulo \( 2^n \) will be included in the next step. Utilizing modulo \( 2^n \) in the second step leads to significant improvement in terms of delay because this modulo has better speed compared to modulus in the forms of \( 2^k - 1 \). The first step of design is described as follows. The weighted number \( Y \) can be calculated as

\[
Y = Z_1 + Z_2 P_1 + Z_3 P_1 P_2 \tag{6}
\]

where

\[
Z_1 = x_1 \tag{7}
\]

\[
Z_2 = \left| (x_2 - x_1) | P_1^{-1} | P_2 \right| P_3 \tag{8}
\]

\[
Z_3 = \left| \left( (x_3 - x_1) | P_1^{-1} | P_3 - Z_2 \right) | P_2^{-1} | P_3 \right| P_4 \tag{9}
\]

and \( P_1 = 2^n - 1 \), \( P_2 = 2^{n+1} - 1 \) and \( P_3 = 2^n - 1 \).

**Proposition 1.** The multiplicative inverse of \( P_1 \) in modulo \( P_2 \) is \( |P_1^{-1}| P_2 = -2 \).

**Proof.** By considering multiplicative inverse definition we have:

\[
|2^n - 1| \times |P_1^{-1}| P_2 |_{2^{n+1} - 1} = 1 \rightarrow |2^n - 1| \times (-2) |_{2^{n+1} - 1} = |2 - 2^{n+1}| |_{2^{n+1} - 1} = 1 - (2^{n+1} - 1) |_{2^{n+1} - 1} = 1
\]

\[
\square
\]

**Proposition 2.** The multiplicative inverse of \( P_1 \) in modulo \( P_3 \) is \( |P_1^{-1}| P_3 = 1 \).

**Proof.** Based on multiplicative inverse definition, it’s clear that:

\[
|2^n - 1| \times |P_1^{-1}| P_3 |_{2^n - 1} = 1 \rightarrow |(2^n - 1)| |_{2^n - 1} = |2 \times (2^n - 1) + 1| |_{2^n - 1} = 1
\]

\[
\square
\]

**Proposition 3.** The multiplicative inverse of \( P_2 \) in modulo \( P_3 \) is \( |P_2^{-1}| P_3 = \sum_{i=0}^{2^n - 1} 2^i \).

**Proof.** Based on multiplicative inverse definition, it’s obvious:

\[
|2^{n+1} - 1| \times |P_2^{-1}| P_3 |_{2^{n+1} - 1} = 1 \rightarrow |(2^{n+1} - 1) \times \sum_{i=0}^{2^n - 1} 2^i |_{2^{n+1} - 1} = \left\lfloor \frac{1 - 2^n}{-3} \right\rfloor |_{2^{n+1} - 1} = \left\lfloor (4 \times (2^{n-1} - 1) + 3) \times \frac{2^n - 1}{3} \right\rfloor |_{2^{n+1} - 1} = |2^n - 1| |_{2^{n+1} - 1} = |2 \times (2^n - 1) + 1| |_{2^{n+1} - 1} = 1
\]

\[
\square
\]

After realizing multiplicative inverses, \( Z_2 \) can be calculated as follows

\[
Z_2 = |(x_2 - x_1) \times (-2)| |_{2^{n+1} - 1} \tag{10}
\]

**Lemma 1.** If \( V \) is an \( n \)-bit number in the interval \([0, 2^n - 1]\), the residue of \((-V)\) in modulo \( 2^n - 1 \) equals to one’s complement of \( V \) [24].

**Lemma 2.** If \( V \) is an \( n \)-bit number in the interval \([0, 2^n - 1]\), the multiplication of \( V \) by \( 2^p \) in modulo \( 2^n - 1 \), equals to its \( p \)-bit circular left shift counterpart [24].

By multiplying \( x_2 - x_1 \) by \(-2\), based on lemma 2, \( Z_2 \) is resulted as:

\[
Z_2 = |L_1 - L_2| |_{2^{n+1} - 1} \tag{11}
\]

where

\[
L_1 = x_{1,n-1} \cdots x_{1,0} \tag{12}
\]

\[
L_2 = x_{2,n-1} \cdots x_{2,0} x_{2,n} \tag{13}
\]
\[ Z_2 = \begin{cases} 
L_1 - L_2 & \text{if } L_1 - L_2 \geq 0 \\
L_1 - L_2 + (2^{n+1} - 1) & \text{if } L_1 - L_2 < 0 
\end{cases} \]

(14)

To calculate \( Z_3 \), after calculating \(|P_1^{-1}|_{P_3}\) and \(|P_2^{-1}|_{P_3}\), the results are replaced in Equation (9) as follow:

\[
Z_3 = \left| \frac{(x_3 - x_1) \times 1 - Z_2}{2^{n-1-1}} \cdot (2^0 + 2^2 + \cdots + 2^{n-2}) \right|
\]

(15)

To eliminate the computation of \( Z_2 \) in modulo \( 2^{n+1-1} \), in computing \( Z_3 \), the following method can be utilized.

The result of subtracting \( L_2 \) from \( L_1 \) will be either a positive number smaller than \( 2^{n+1-1} \) or a negative number greater than \( 1 - 2^{n+1} \). By default the first case has a result in modulo \( 2^{n+1-1} \); however, adding \( 2^{n+1-1} \) to the result of \( e_1 - e_2 \) is required when \( e_1 - e_2 \) is negative. The outgoing carry of the adder utilized for \( L_1 \) and \( L_2 \) subtraction, can distinguish the two cases indicated in Equation (14).

If \( L_1 > L_2 \), \( Z_3 \) can be obtained as Equation (16):

\[
Z_3 = \left| \frac{(x_3 - x_1 - L_1 + L_2)}{2^{n-1-1}} \cdot (2^0 + 2^2 + \cdots + 2^{n-2}) \right|
\]

(16)

For more simplicity \( x_3 - x_1 - L_1 + L_2 \) is rewritten in the bit-level representation and then segregated in numbers with the length of \( n - 1 \) bit to ease applying its coefficient, \( (2^0 + 2^2 + \cdots + 2^{n-2}) \).

\[
Z_3 = \left| \frac{x_{3,n-2} \cdots x_{3,0} - 0 \cdots 0}{2^{n-2}} x_{1,n-1} - x_{1,0} - 0 \cdots 0 L_{1,n-1} + \cdots + 0 \cdots 0 L_{2,n-1}}{2^{n-1-1}} \times (2^0 + 2^2 + \cdots + 2^{n-2}) \right|
\]

(17)

and using Lemma 1:

\[
Z_3 = \left| \frac{x_{3,n-2} \cdots x_{3,0} + 1 \cdots 1 - x_{1,n-1} + 1 \cdots 1 L_{1,n-1} + \cdots + 1 \cdots 1 L_{2,n-1}}{2^{n-1-1}} \times (2^0 + 2^2 + \cdots + 2^{n-2}) \right|
\]

(18)

Equation (18) can be simplified as following:

\[
Z_3 = \left| \frac{Z_{3,1} + Z_{3,2} + Z_{3,3} + Z_{3,4}}{2^{n-1-1}} + Z_{3,5} + Z_{3,6} + Z_{3,7} \right| \times (2^0 + 2^2 + \cdots + 2^{n-2})
\]

(19)

where

\[
Z_{3,1} = x_{3,n-2} \cdots x_{3,0}
Z_{3,2} = 1 \cdots 1 \bar{x}_{1,n-1}
Z_{3,3} = \bar{x}_{1,n-2} \cdots \bar{x}_{1,0}
Z_{3,4} = 1 \cdots 1 \bar{L}_{1,n-1} \bar{L}_{1,n-1}
Z_{3,5} = L_{1,n-2} \cdots L_{1,0}
Z_{3,6} = 0 \cdots 0 L_{2,n-2} L_{2,n-1}
Z_{3,7} = L_{2,n-2} \cdots L_{2,0}
\]

In the other case when \( L_1 < L_2 \),

\[
Z_3 = \left| \frac{(x_3 - x_1 - L_1 + L_2 - (2^{n+1} - 1))}{2^{n-1-1}} \times (2^0 + 2^2 + \cdots + 2^{n-2}) \right|
\]

and since \( (2^{n+1} - 1) \) \( \equiv \) \(-1\)\( \mod 2^{n-1-1} \), the following expression is resulted:

\[
(-1)^{2^{n-1-1}} = 1 \cdots 1 0^{n-2}
\]

therefore, \( Z_3 \) can be rewritten as

\[
Z_3 = \left| \frac{Z_{3,1} + Z_{3,2} + Z_{3,3} + Z_{3,4}}{2^{n-1-1}} + Z_{3,5} + Z_{3,6} + Z_{3,7} \right| \times (2^0 + 2^2 + \cdots + 2^{n-2}) + Z_{3,8}
\]

(21)

where

\[
Z_{3,8} = 1 \cdots 1 0^{n-2}
\]

In Figure 2, \( Z_3 \) is generated by Operand Preparation Unit1 (OPU1) with \( x_1, x_2 \) and \( x_3 \) as its inputs. The values of \( Z_{3,1}, Z_{3,2}, Z_{3,3}, Z_{3,4}, Z_{3,5}, Z_{3,6} \) and \( Z_{3,7} \) are also reduced to \( S_1 \) and \( C_1 \) by CSA1, CSA2, CSA3, CSA4 and CSA5.

\[
Z_3 \text{ is then obtained as}
\]

\[
Z_3 = \left| \frac{(S_1 + C_1)}{2^{n-1-1}} \times (2^0 + 2^2 + \cdots + 2^{n-2}) \right|
\]

if \( L_1 - L_2 \geq 0 \)

\[
Z_3 = \left| \frac{(S_1 + C_1)}{2^{n-1-1}} \times (2^0 + 2^2 + \cdots + 2^{n-2}) \right|
\]

if \( L_1 - L_2 < 0 \)
Based on Lemma 2

\[
Z_3 = \begin{cases} 
|\varphi + \theta|_{2^n-1} & \text{if } L_1 - L_2 \geq 0 \\
|\varphi + \theta + Z_{3,8}|_{2^n-1} & \text{if } L_1 - L_2 < 0
\end{cases}
\] (23)

\[
\varphi = \sum_{i=0}^{n-2} \text{CLS}(S_1, 2i)
\] (24)

\[
\theta = \sum_{i=0}^{n-2} \text{CLS}(C_1, 2i)
\] (25)

where \(\text{CLS}(x, y)\) equals to \(y\)-bit circular left shift of \(x\).

Operand Preparation Unit 2 (OPU2) is used to implement \(Z_3\) with \(S_1\) and \(C_1\) as its inputs. Also OPU2 generates multiple \((n - 1)\)-bit outputs. The outputs are \(\text{CLS}(C_1, n - 2), \ldots, \text{CLS}(C_1, 0)\) and \(\text{CLS}(S_1, n - 2), \ldots, \text{CLS}(S_1, 0)\). To obtain a compact final output, all of the outputs must be reduced by a CSA tree. So the output of the mentioned CSA tree and the output of the MUX connect to the CSA block and later to the modulo \(2^{n-1} - 1\) adder. The final result of the MA2 is \(Z_3\) signal which will be used in the next step.

\(Z_2\) is the other signal which should be prepared for the next step. To decrease the delay of the proposed design, it is preferred to compute \(Z_2\) in parallel with calculating \(Z_3\). Based on the Equation (11), \(Z_2\) should be obtained by a modular adder in modulo \(2^{n+1} - 1\) named MA1. Also, \(Z_2\) is needed in the second step of design; therefore, the output of the MA1 goes to Operand Preparation Unit 3 (OPU3), including \(n + 1\) inverters, to produce \(Z_2\). Only in the first step, all of the above mentioned calculations are done in parallel with computing \(Z_3\). Thus the delay of computing \(Z_2\) is not considered in the critical path delay. Hardware implementations of \(Z_2\) and \(Z_3\) are shown in Figure 2.

After the calculation of \(Z_2\) and \(Z_3\), \(Y\) can be obtained from its residues in 3-moduli set \(\Gamma\) as

\[
Y = Z_1 + Z_2 P_1 + Z_3 P_1 P_2
\] (26)

\[
Y = Z_1 + Z_2 (2^n - 1) + Z_3 (2^n - 1) + (2^{n+1} - 1)
\] (27)

At the next level of design, \(Z_1\), \(Z_2\), and \(Z_3\) are used with consideration of the value of \(Y\). Furthermore, \(Z_2\) and \(Z_3\) are computed at the first stage for more parallel architecture. There is no need to compute the final value of \(Y\) at the first stage. Only some arrangements of \(Z_1\), \(Z_2\) and \(Z_3\) which are needed in computing \(Y\) and indicated by \(y_i, i = 1, \ldots, 7\), are utilized at the next level of design. Therefore the delay of computing \(Y\) is omitted. The \(y_i\) signals are expressed by the following expression:

\[
Y = Z_1 + Z_2 (2^n - 1) + Z_3 (2^n - 1) (2^{n+1} - 1)
\]

\[
= Z_1 + Z_2 0 \cdot \cdots 0 - Z_2 + Z_3 0 \cdot \cdots 0 - Z_3 0 \cdot \cdots 0
\]

\[
= Z_1 + Z_2 0 \cdot \cdots 0 + Z_3
\] (28)

\[
Y = y_1 + y_2 + y_3 + y_4 + y_5 + y_6 + y_7
\] (29)

where \(y_1 = Z_1\), \(y_2 = Z_2 0 \cdot \cdots 0\), \(y_3 = -Z_2\), \(y_4 = Z_3 0 \cdot \cdots 0\), \(y_5 = -Z_3 0 \cdot \cdots 0\), \(y_6 = -(Z_3 0 \cdot \cdots 0)\) and \(y_7 = Z_3\).

3.2 Second Step Design

After the computation of \(y_i, i = 1, \ldots, 7\), the two modulus superset \(\Lambda\) is considered for obtaining weighted number \(X\). The residue of weighted number \(X\) in modulo \(P_{123}\) and \(P_4\) is equal to \(Y\) and \(x_4\), respectively, where \(P_{123} = (2^n - 1) \times (2^{n+1} - 1) \times (2^{n-1} - 1)\) and \(P_4 = 2^n\). The MRC method for moduli set with two modulus is utilized to calculate \(X\) as follows:

\[
X = v_1 + v_2 P_{123}
\] (30)

where

\[
v_1 = Y
\] (31)

\[
v_2 = |(x_4 - Y)| P_{123}^{-1} P_4
\] (32)

Proposition 4. The multiplicative inverse of \(P_{123}\) in modulo \(P_4\) is equal to \(-2^{n-1} - 1\).

Proof. According to multiplicative inverses definition, we have:

\[
|2^n - 1\times (2^{n+1} - 1)\times (2^{n-1} - 1)|_{2^n} = 1
\]

\[
\Rightarrow |(-1)\times (-1)\times (-2^{n-1} + 1)|_{2^{n+1}} = 1
\]

\[
\Rightarrow |(-1)\times (-1)|_{2^{n+1}} = 1
\]

thus,

\[
v_2 = |(x_4 - Y)| (2^{n-1} + 1)|_{2^n} = |Y - x_4|_{2^n}
\] (33)

By replacing \(Y\) based on Equation (29), \(V_2\) could be rewritten as:

\[
v_2 = |(y_1 + y_2 + y_3 + y_4 + y_5 + y_6 + y_7 + x_4 + 1)\times (2^{n-1} + 1)|_{2^n}
\] (34)

The digits which weigh more than or equal \(2^n\) is not considered in the operations of modulo \(2^n\). Accord-
moderately, only the lowest weighted \( n \) bits are used in the operations. Therefore, \( v_2 \) can be expressed as:

\[
v_2 = \left| Z_1,0,0 \cdots 0 + \tilde{Z}_2,0,0 \cdots 0 + Z_{3,0}Z_3 \right|_{2^n} + \left| \tilde{x}_4,0,0 \cdots 0 + \tilde{x}_4 + Z_1 + Z_2 + \mu \right|_{2^n}
\]

where

\[
k_1 = XOR(\tilde{x}_4,0, \tilde{x}_4,1,0) \tilde{x}_4,0,1,0 \cdots \tilde{x}_4,1,0 \cdots \tilde{x}_4,1,0
\]

\[
k_2 = XOR(\tilde{Z}_2,0, \tilde{Z}_2,0, Z_3,0) \tilde{Z}_3
\]

\[
\mu = 0 \cdots 00 \text{10}^{n-2}
\]

The structure of the OPU4 is implemented based on the above equation for \( v_2 \). The inputs are \( Z_1, Z_2, Z_3, \) and \( x_4 \) and its outputs are \( k_1 \) and \( k_2 \). CSA7 neglects the \( n^{th} \) bit of its outputs, CSA7 is also put a 0 in the least significant bit of the carry. It also omits the most significant bit of \( \tilde{Z}_2 \) by considering Equation (29) based on the previous subsection. This procedure is also done by CSA8 and CSA9. Finally, the outputs of the CSAs go to the Modular adder 3 (MA3) as its inputs to compute \( v_2 \). Hardware implementation of \( v_2 \) is shown in Figure 3.

The value of weighted number \( X \), based on the computed value of \( v_2 \), is calculated by the Equation (40).
\[ X = v_1 + v_2 \times (2^n - 1) \times (2^{n+1} - 1) \times (2^{n-1} - 1) \]  
(40)

Since \( v_1 = Y \), \( v_1 \) is replaced by \( Y \) as follow:
\[ X = Y + v_2 \times (2^n - 1) \times (2^{n+1} - 1) \times (2^{n-1} - 1) \]  
(41)

Equation (41) can be simplified as
\[ X = v_2 Z_3Z_2Z_1 - 0 \cdots 0 v_2 0 \cdots v_2 Z_3Z_2 + 0 \cdots 0 v_2Z_3v_2 + v_2 0 \cdots 0 + 0 + v_2 Z_3 \]  
(42)

\[ X = v_2 Z_3Z_2Z_1 + \frac{1}{n} \sum_{k=1}^{n} v_2 \frac{1}{n} + 1 \frac{1}{n} v_2 \frac{3}{2} \frac{2}{n} + 1 \frac{1}{n} v_2 Z_3 Z_2 + 0 \cdots 0 + v_2 Z_3 \]  
(43)

\( X \) is the summation of seven values, \( \sum_{k=1}^{n} X_k \),
where \( X_1 = v_2 Z_3Z_2Z_1 \), \( X_2 = 1 \frac{1}{n} \sum_{k=1}^{n} v_2 \frac{1}{n} + 1 \frac{1}{n} v_2 \frac{3}{2} \frac{2}{n} \),
\( X_3 = 1 \frac{1}{n} \sum_{k=1}^{n} v_2 \frac{1}{n} \), \( X_4 = 1 \frac{1}{n} \sum_{k=1}^{n} v_2 \frac{3}{2} \frac{2}{n} \), \( X_5 = v_2 0 \cdots 0 + v_2 Z_3 \), \( X_6 = v_2 0 \cdots 0 \) and \( X_7 = v_2 Z_3 \). \( X_k \) with the bit-length of \( 4n \)-bit enters to the carry save adder tree and the outcomes of carry save adder connect to the input of a \( 4n \)-bit CPA to compute weighted number \( X \). Figure 4 depicts the architecture of this scenario.

3.3 Numerical Example

Considering moduli set \( \{63, 127, 31, 64\} \), which is derived from moduli set \( \Psi \) when \( n = 6 \), the RNS number \( (33, 7, 5, 63) \) can be converted to its equivalent in weighted number \( X \) as:

**First stage:**
\[ x_1 = 33_{10} = 100001_2 \]
\[ x_2 = 7_{10} = 0000111_2 \]
\[ x_3 = 5_{10} = 00101_2 \]

By substituting these values in Equation (6), (11), (19) and (21), the following results will be obtained:
\[ Z_1 = 33_{10} = 100001_2 \]
\[ L_1 = 100001_2 \]
\[ L_2 = 0001110_2 \]
\[ Z_2 = 0110100_2 \]
\[ Z_3 = 11001_2 \]

**Second stage:** by considering Equation (31), (36) and (43), the desired values in second step are obtained:
\[ v_1 = Y = 107322_{10} = 1101000110011101010_2 \]
\[ k_1 = 0 \]
\[ k_2 = 57_{10} = 111001_2 \]
\[ v_2 = 39_{10} = 100111_2 \]
\[ X = 107322 + 39 \times 63 \times 127 \times 31 = 9876543 \]

Thus \( X = 9876543 \), and the verification can be simply done as
\[ x_1 = [9876543]_{63} = 33 \]
\[ x_2 = [9876543]_{27} = 7 \]
\[ x_3 = [9876543]_{31} = 5 \]
\[ x_4 = [9876543]_{61} = 63 \]

4 Hardware Cost and the Delay of Proposed Converter

The hardware requirements were indicated briefly in Section 2. In this section the evaluation of the hardware cost and the critical path delay of the proposed reverse converter are done in detail. In such an evaluation process, the wire loads are usually assumed to be negligible. The hardware costs is based on the number of primitive logic components utilized in the reverse converter of moduli set \( \Psi \). To calculate the delay of the whole circuit, the critical path, which is shown in Figures 2, 3 and 4 with the red dash line, should be determined. In the reverse converter evaluation process, the delay of \( n \)-bit CPA equals to the time in
which two numbers are aggregated in a ripple structure. The logic function of a full adder is described by the following equations:

\[
\text{Sum} = \text{XOR}(x, y, z) = xy\overline{z} + \overline{x}yz + \overline{x}\overline{y}z + \overline{x}\overline{y}\overline{z} \quad (44)
\]

\[
\text{Carry} = xy + xz + zy \quad (45)
\]

According to Equations (44) and (45), if one of the inputs of the full adder equals to 1 (for instance \( z = 1 \)), the Sum and Carry are equivalent to \( xy + \overline{xz} \) and \( x + y \) respectively. If one of the inputs equals to 0 (for instance \( z = 0 \)), the Sum and Carry are equivalent to \( \overline{xy} + xz + zy \) respectively. In these two cases only two input gates are used. If two inputs have constant values, the simplification process is the same as above. Table 2 shows different conditions of a full adder cell according to constant inputs.

In the first step of the design of the reverse converter, the modular adders (MA1, MA2) are implemented by CPAs with EAC (Figure 2). The \( 2^k - 1 \) modular adder has the similar area and double delay compared to the \( k \)-bit CPA. The latter modular adder (MA3) is implemented by a regular \( n \)-bit CPA neglecting its carryout (Figure 3). CSAs used in the design of the reverse converter are divided to the regular CSA and CSA with EAC [13].

1. In the first level of design, CSAs of the reduction tree are CSAs with EAC. The output of a final CSA enters to the modulo \( 2^{n-1} - 1 \) adder (MA2).
2. The summation of five operands in modulo \( 2^n \) computes \( V_2 \) in the initiation of the second step of design. The structure of the reduction tree used for computing \( V_2 \) employs three regular CSAs. Due to the fact that the output of the reduction tree must be in modulo \( 2^n \), for achieving the hardware cost efficiency, the carryout signal of every CSA in the architecture shown in Figure 3 is neglected.
3. In the last part of the second step of design, for computing \( X \), CSAs which their inputs are signals with different bit numbers, are used.

The only difference between regular CSA and CSA with EAC is the generated result of \( c \) from CSA. Figure 5 demonstrates basic architecture of a CSA and a CSA with EAC. The delay of \( n \)-bit CSA denotes additional time of a full adder cell. In CSA like CPA, hardware cost can be reduced according to the constant input values. In general, the hardware cost of \( n \)-bit CSA is equal to the hardware cost of a full adder cell. Table 3 shows the hardware cost and the delay of various components in the proposed reverse converter.

### 5 Comparison

This section presents the comparison of the proposed reverse converter architecture for the moduli set \( \Psi \) with other balanced 4-moduli sets with the same dynamic range class, such as the 4-moduli sets \( \{2^n-1, 2^n, 2^n+1, 2^{n+1}+1\} \) [13, 14], \( \{2^n-1, 2^n, 2^n+1, 2^n+1\} \) [13, 15], \( \{2^n-3, 2^n-1, 2^n+1, 2^n+3\} \) [16] and \( \{2^n, 2^{n+1}-1, 2^n-1, 2^n-1\} \) [17]. The comparisons are done in terms of the delay and the area of the reverse converter. Table 4 shows the comparison between the proposed reverse converter and its other state-of-the-art counterpart. In order to achieve fair comparison, the delay and the area of the modulus adders and carry save adders are considered the same as [24]. As shown in Table 2, the proposed reverse converter for the moduli set \( \Psi \) has achieved the highest speed of the reverse converter compared to \( \{2^n-1, 2^n, 2^n+1, 2^{n+1}+1\} \) [13, 14], \( \{2^n-1, 2^n, 2^n+1, 2^n+1\} \) [13, 15], \( \{2^n-3, 2^n-1, 2^n+1, 2^n+3\} \) [16] and \( \{2^n, 2^{n+1}-1, 2^n-1, 2^n-1\} \) [17]. It is worth mentioning that, the proposed reverse converter is the fastest adder based reverse converter in the balanced 4-moduli class [17].

The unit gate delay and the unit gate area are models for evaluation of the hardware requirement and the critical path delay between the different adder based reverse converters. In this model, FA gates have the area of seven gates and the delay of four gates. XOR/XNOR gates have two gates area and delay, and each two-input monotonic gates have one area and delay [22]. For more fair comparison, the unit gate delay and the unit gate area of the different adder based reverse converters are included in the Table 5.

<table>
<thead>
<tr>
<th>Number of constant value</th>
<th>Constant value</th>
<th>Reduced gates</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>a pair of two input XNOR and OR gate</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>a pair of two input XOR and AND gate</td>
</tr>
<tr>
<td>2</td>
<td>Same value ( both 0 or both 1)</td>
<td>Wire</td>
</tr>
<tr>
<td>2</td>
<td>Same value ( one input 0 another 1)</td>
<td>Inverter gate</td>
</tr>
</tbody>
</table>
Figure 5. Basic Architecture for (a) a Carry Save Adder and (b) a CSA with End Around Carry.

Table 3. Hardware and delay of various components in the proposed reverse converter.

<table>
<thead>
<tr>
<th>Component</th>
<th>Area</th>
<th>Delay</th>
<th>Component</th>
<th>Area</th>
<th>Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>OPU1</td>
<td>(2(n + 1)A_{Inw})</td>
<td>(1D_{Inw})</td>
<td>CSA7</td>
<td>(nA_{FA})</td>
<td>(1D_{FA})</td>
</tr>
<tr>
<td>Comparator</td>
<td>(3nA_{AND} + nA_{OR3})</td>
<td>(nD_{AND}) + (nD_{OR3})</td>
<td>CSA8</td>
<td>((n - 1)A_{FA} + A_{XOR}) + (A_{AND})</td>
<td>(1D_{FA})</td>
</tr>
<tr>
<td>CSA1</td>
<td>(A_{FA})</td>
<td>(1D_{FA})</td>
<td>CSA9</td>
<td>((n - 2)A_{XOR} + 1A_{XOR}) + ((n - 2)A_{AND} + 1A_{OR})</td>
<td>(1D_{Inw})</td>
</tr>
<tr>
<td>CSA2</td>
<td>(2A_{FA})</td>
<td>(1D_{FA})</td>
<td>MA3</td>
<td>((n - 3)A_{FA} + 1A_{XOR} + 1A_{AND})</td>
<td>((n - 3)D_{FA}) + (1D_{FA})</td>
</tr>
<tr>
<td>CSA3</td>
<td>((n - 1)A_{FA})</td>
<td>(1D_{FA})</td>
<td>OPU5</td>
<td>((2n - 1)A_{Inw})</td>
<td>(1D_{Inw})</td>
</tr>
<tr>
<td>CSA4</td>
<td>((n - 1)A_{FA})</td>
<td>(1D_{FA})</td>
<td>CSA10</td>
<td>((n - 2)A_{FA} + 2A_{XOR} + 2A_{AND})</td>
<td>(1D_{FA})</td>
</tr>
<tr>
<td>CSA5</td>
<td>((n - 1)A_{FA})</td>
<td>(1D_{FA})</td>
<td>CSA11</td>
<td>((n - 2)A_{FA} + (2n + 2)(A_{XOR} + A_{OR}))</td>
<td>(1D_{FA})</td>
</tr>
<tr>
<td>OPU2</td>
<td>0</td>
<td>0</td>
<td>CSA12</td>
<td>(2nA_{FA} + (n + 1)(A_{XOR} + A_{AND}))</td>
<td>(1D_{FA})</td>
</tr>
<tr>
<td>CSA Tree</td>
<td>((n^2 - n)A_{FA})</td>
<td>(pD_{FA})</td>
<td>CSA13</td>
<td>(2nA_{FA} + (n + 1)(A_{XOR} + A_{AND}))</td>
<td>(1D_{FA})</td>
</tr>
<tr>
<td>CSA6</td>
<td>((n - 2)A_{FA} + A_{XOR} + A_{AND})</td>
<td>(1D_{FA})</td>
<td>CSA14</td>
<td>((3n + 1)A_{FA} + 2(A_{XOR} + A_{AND}))</td>
<td>(1D_{FA})</td>
</tr>
<tr>
<td>MA1</td>
<td>((n + 1)A_{FA})</td>
<td>((2n + 2)D_{FA})</td>
<td>CPA</td>
<td>((4n - 2)A_{FA} + 2(A_{XOR} + A_{AND}))</td>
<td>((4n - 2)D_{FA}) + (1D_{HA})</td>
</tr>
<tr>
<td>MA2</td>
<td>((n - 1)A_{FA})</td>
<td>((2n - 2)D_{FA})</td>
<td>MUX 2:1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>OPU3</td>
<td>((n + 1)A_{Inw})</td>
<td>(1D_{Inw})</td>
<td>OPU4</td>
<td>(2A_{XOR} + nA_{Inw})</td>
<td>(1A_{XOR3})</td>
</tr>
</tbody>
</table>

Table 4. Hardware requirements and delay of reverse converters.

<table>
<thead>
<tr>
<th>Moduli Set</th>
<th>Design</th>
<th>Hardware requirements</th>
<th>Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>{2^n - 1, 2^n, 2^n + 1, 2^{n+1} - 1 }</td>
<td>[13]−1</td>
<td>(9n + 5 + ((n - 4)(n + 1)/2)) A_{FA} + 2nA_{XNOR} + 2nA_{OR} + (6n + 1) A_{INV}</td>
<td>(23n + 12)/2D_{FA}</td>
</tr>
<tr>
<td>{2^n - 1, 2^n, 2^n + 1, 2^{n+1} + 1 }</td>
<td>[13]</td>
<td>2n^2 + 11n + 3</td>
<td>11.5nD_{FA}</td>
</tr>
<tr>
<td>{2^n - 1, 2^n, 2^n + 1, 2^{n+1} + 1 }</td>
<td>[13]−2</td>
<td>(6n + 7) A_{INV} + (n^2 + 12n + 12) A_{FA} + 2n(A_{XNOR} + A_{OR}) + (4n + 8) A_{21MUX}</td>
<td>(16n + 22) D_{FA}</td>
</tr>
<tr>
<td>{2^n - 1, 2^n, 2^n + 1, 2^{n+1} + 1 }</td>
<td>[15]</td>
<td>(58n + 23 + \log_2 (c + 1)) A_{FA}</td>
<td>(24n + 17 + \log_2^{c+1}) D_{FA}</td>
</tr>
<tr>
<td>{2^n - 3, 2^n, 1, 2^n + 1, 2^n + 3 }</td>
<td>[16]−C1 CE</td>
<td>(25.5n + 12 + 2.5n^2) A_{FA} + 5nA_{HA} + 3n(A_{XNOR} + A_{OR})</td>
<td>(18n + 23) D_{FA}</td>
</tr>
<tr>
<td>{2^n - 3, 2^n, 1, 2^n + 1, 2^n + 3 }</td>
<td>[16]−C2 CE</td>
<td>(20n + 17) A_{FA} + (3n - 4) A_{HA} + 2^n(5n + 2) A_{ROM}</td>
<td>(13n + 22) D_{FA}</td>
</tr>
<tr>
<td>{2^n - 3, 2^n, 1, 2^n + 1, 2^n + 3 }</td>
<td>[16]−C3 CE</td>
<td>(23n + 11)A_{FA} + (2n - 2)A_{HA} + (6n + 4)^2A_{ROM}</td>
<td>(16n + 14) D_{FA}</td>
</tr>
<tr>
<td>{2^n - 1, 2^n, 2^n + 1, 2^{n+1} - 1 }</td>
<td>[14]−3stage-CE</td>
<td>(n^2 + 10n + 3)A_{FA} + A_{HA} + (3n + 2)A_{INV} + 2A_{21MUX}</td>
<td>(9n + 6 + m) D_{FA}\ast</td>
</tr>
<tr>
<td>{2^n, 2^{n+1} - 1, 2^n - 1, 2^{n-1} - 1 }</td>
<td>[17]−D1-C-I</td>
<td>(n^2 + 16n + 6) A_{FA} + 4nA_{INV} + (n + 2)(A_{XNOR} + A_{OR}) + (3n - 5)(A_{XOR} + A_{AND})</td>
<td>(12n + 9 + q) D_{FA}\ast</td>
</tr>
<tr>
<td>{2^n, 2^{n+1} - 1, 2^n - 1, 2^{n-1} - 1 }</td>
<td>[17]−D1-C-II</td>
<td>(n^2 + 24n + 24) A_{FA} + (2n + 3)A_{HA} + (2n - 5)(A_{XOR} + A_{AND}) + (2n + 1) A_{3MUX} + 4nA_{INV}</td>
<td>(8n + 11 + q) D_{FA}\ast</td>
</tr>
<tr>
<td>{2^n, 2^{n+1} - 1, 2^n - 1, 2^{n-1} - 1 }</td>
<td>[17]−D1-C-III</td>
<td>(n^2 + 22n + 22) A_{FA} + (2n + 2)A_{HA} + 10(2n + 1)A_{ROM} + 2(A_{XNOR} + A_{OR}) + (2n - 5)(A_{XOR} + A_{AND}) + (2n + 1) A_{21MUX} + 4nA_{INV}</td>
<td>(8n + 11 + q) D_{FA}\ast</td>
</tr>
<tr>
<td>{2^n, 2^{n+1} - 1, 2^n - 1, 2^{n-1} - 1 }</td>
<td>Proposed</td>
<td>(n^2 + 21n - 11) A_{FA} + (3n + 1)(A_{XNOR} + A_{OR}) + (6n + 9) A_{AND} + (3n + 9) A_{XOR} + nA_{OR} + (5n + 1) A_{INV}</td>
<td>(7n + 1) D_{FA} + n(D_{OR} + D_{AND}) + 2D_{HA} + 4D_{INV}</td>
</tr>
</tbody>
</table>

* m and q are the number of levels in CSA tree with \(n+2\), \(n+1\) inputs, respectively.

which confirms the remarkable improvement in terms of speed of the reverse converter. Also degraded hardware resources are achieved compared to [13–17].

6 Conclusion

In this paper, the quadruple moduli set \(\Psi\) was the focus of study in reducing the computational intensity of the reverse converter design. \(\Psi\) has the dynamic range of \(4n\) and utilizes modulos only in the form of \(\{2^n - 1\}\) beside modulo \(2^n\), which provides efficient arithmetic operations in RNS channels. The new reverse converter eliminates the extra intermediate calculations. For each level of design, the moduli subsets are selected to make the design more efficient in both delay and the hardware cost. To put everything in the nutshell, the overall area and time complexity analysis indicates that the proposed reverse converters are more efficient than the converters for the 4-moduli set \(\{2^{n-1} - 1, 2^{n+1} - 1, 2^n, 2^n - 1\}\).
Acknowledgements

The authors are grateful to the anonymous reviewers’ valuable comments and suggestions that improved the quality of manuscript. Also the authors would like to thank Dr. B. Yoberd and Ms. F. Shaker for their literature contributions.

References

[13] PV Ananda Mohan and AB Premkumar. Rns-to-binary converters for two four-moduli sets \( \{2^n - 1, 2n, 2n+1, 2n+1+1\} \) and \( \{2n-1, 2n, 2n+1, 2n+1+1\} \). Circuits and Systems I: Regular Papers, IEEE Transactions on, 54(6):1245–1254, 2007.
[14] B Cao, T Srikanth, and CH Chang. Efficient reverse converters for four-moduli sets \( \{2n-1, 2n, 2n+1, 2n+1-1\}\) and \( \{2n-1, 2n, 2n+1, 2n+1-1\}\). IEE Proceedings-Computers and Digital Techniques, 152(5):687–696, 2005.

Table 5. Unit gate area and delay of reverse converters.

<table>
<thead>
<tr>
<th>Moduli Set</th>
<th>Design</th>
<th>Unit gate area</th>
<th>Unit gate delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>( {2^n - 1, 2^n, 2^n + 1} )</td>
<td>[13]-1</td>
<td>( 3.5n^2 + 72.5n + 23 )</td>
<td>( 46n + 24 )</td>
</tr>
<tr>
<td>( {2^n - 1, 2^n, 2^n + 1} )</td>
<td>[13]-2</td>
<td>( 7n^2 + 128n + 146 )</td>
<td>( 64n + 88 )</td>
</tr>
<tr>
<td>( {2^n3, 2^n1, 2^n + 1} )</td>
<td>[16] C1 CE</td>
<td>( 17.5n^2 + 210.5n + 84 )</td>
<td>( 72n + 92 )</td>
</tr>
<tr>
<td>( {2^n - 1, 2^n, 2^n + 1} )</td>
<td>[14]-3Stage-CE</td>
<td>( 7n^2 + 76n + 41 )</td>
<td>( 36n + 24 + 4m^* )</td>
</tr>
<tr>
<td>( {2^n, 2^{n+1} - 1, 2^n - 1, 2^n - 1} )</td>
<td>[17] D1-C-I</td>
<td>( 7n^2 + 136n + 30 )</td>
<td>( 48n + 36 + 4q^* )</td>
</tr>
<tr>
<td>( {2^n, 2^{n+1} - 1, 2^n - 1, 2^n - 1} )</td>
<td>[17] D1-C-II</td>
<td>( 7n^2 + 204n + 174 )</td>
<td>( 32n + 44 + 4q^* )</td>
</tr>
<tr>
<td>( {2^n, 2^{n+1} - 1, 2^n - 1, 2^n - 1} )</td>
<td>Proposed</td>
<td>( 7n^2 + 169n + 47 )</td>
<td>( 30n + 4 )</td>
</tr>
</tbody>
</table>

* \( m \) and \( q \) are the number of levels in CSA tree with \((n + 2), (n + 1)\) inputs, respectively.


[22] Mohammad Esmaeildoust, Keivan Navi, and MohammadReza Taher. High speed reverse converter for new five-moduli set \{2n, 2n+ 1-1, 2n-2-1, 2n/2+ 1, 2n+ 1\}. *IEICE Electronics Express*, 7(3):118–125, 2010.


MohammadReza Taheri received his B.Sc. in Computer Hardware Engineering from Isfahan University, Isfahan, Iran. He obtained his M.Sc. degree in Computer System Architecture from Science and Research Branch of Islamic Azad University, Tehran, Iran. He is currently pursuing his Ph.D. Degree in Computer Architecture at Shahid Beheshti University, Tehran, Iran. He is also a member of the Nanotechnology and Quantum Computing Laboratory of Shahid Beheshti University since 2009. His current research interests include residue number system, low power arithmetic, approximate computing, and circuit techniques for emerging technologies.

Nasim Shafighi received the B.Sc degree in computer hardware engineering from Shahid Beheshti University, Tehran, Iran in 2014. She is a member of the Nanotechnology and Quantum Computing Laboratory of Shahid Beheshti University since 2013. Her research interests include low power computer arithmetic, approximate computing, and robotic.

Mohammad Esmaeildoust received his M.Sc. degree in Computer architecture at Shahid Beheshti University of Technology, Tehran, Iran, in 2008. He also received the Ph.D. degree in computer architecture from Shahid Beheshti University of Technology, Tehran, Iran, in 2012. He is currently Assistant Professor in faculty of Marine Engineering, Khorramshahr University of Marine Science and Technology. His research interests include VLSI design, Cryptography, Network security, computer arithmetic.

Zhale Amirjamshidi earned her M.Sc. in electronic engineering from Central Branch of Islamic Azad University, Tehran, Iran. She is currently pursuing the Ph.D. degree in electronic engineering at Iran University of Science and Technology, Tehran, Iran. Her research interests are mainly focus on low power digital arithmetic and renewable energy.

Reza Sabbagh-nadooshan received his B.Sc. and M.Sc. in electrical engineering from the Iran University of Science and Technology, Tehran, Iran, in 1991 and 1994 and Ph.D. in Electrical Engineering from the Science and Research Branch, Islamic Azad University, Tehran, Iran in 2010. From 1998, he became faculty member of Department of Electronics in Central Tehran branch, Islamic Azad University, Tehran, Iran. His current research interests include nanocomputing and networks-on-chips. He is a member of IEEE.

Keivan Navi received the B.Sc. and M.Sc. degrees in computer hardware engineering from Beheshti University, Tehran, Iran, in 1987 and Sharif University of Technology, Tehran, Iran, in 1990, respectively. He also received the Ph.D. degree in computer architecture from Paris XI University, Paris, France, in 1995. He is currently Professor in faculty of electrical and computer engineering of Beheshti University. His research interests include VLSI design, single electron transistors (SET), carbon nanotube, computer arithmetic, interconnection network and quantum computing.