High Speed VLSI Architecture for $2^n$ Scaling of Signed Integer in RNS

Steffi Keran Rani J$^{1*}$, Dr. J.M.Mathana$^2$

$^{1,2}$Department of Electronics and Communication Engineering
S.A.Engineering College, Chennai (India)

Email: steffikeranj12@gmail.com$^1$, jm.mathana@gmail.com$^2$

Abstract—Residue Number System (RNS) is a non-weighted robust number system that is capable of mapping large numbers to smaller residues. This allows arithmetic operations of long integers to be carried out without any need for carry propagation. Scaling of signed integers in RNS is a problematic and relatively difficult operation which leads to performance bottlenecks while implementing signal processing algorithms. But when scaling of an integer is performed by one of its moduli, the complexity of intermodulo operation can be greatly reduced. This paper proposes an efficient architecture signed integer RNS scaler for the moduli set $\{2^{-1}, 2^1, 2^3+1\}$. The architecture is implemented by employing the number theoretic properties of the moduli set $\{2^{-1}, 2^1, 2^3+1\}$. The proposed RNS scaler possesses zero scaling error because of the additional merged sign detection and correction circuit which eliminates the ambiguities present in scaling the signed integers and is also faster compared to the existing works.

Index Terms—Chinese Remainder Theorem, residue number system, scaling, moduli set, signed integer.

1. INTRODUCTION

Sophisticated real-time application specific digital signal processing (DSP) algorithms are widely used in modern electronic gadgets such as digital cameras, smart phones, camcorders etc. Residue number system (RNS) plays a very crucial role in this domain for mapping complex signal processing algorithms into application-specific hardware accelerators. This is due to the fact that RNS is capable of decomposing any long integer arithmetic operations such as addition, subtraction, and multiplication into smaller carry-free modulo operations for parallel processing [1]. Thus RNS proves to be a promising alternative for the conventional two’s complement number system for computation and data representation purposes. Since RNS has greater advantages compared to two’s complement system and other conventional full adder implementations, RNS has been extensively used to implement the filter components in DSP and Very Large Scale Integration (VLSI) systems [2] and for solving precise systems using a low resolution multimoduli system [3].

Though RNS has many advantages such as modularity, parallelism, and localized carry propagation arithmetic operations, it has some drawbacks such as RNS to binary conversion, magnitude scaling, overflow detection, sign detection, and parity detection [3]. The most erroneous ambiguities occur during reverse conversion of an integer from its residue number representation, and while performing scaling of an integer by a known constant [3]-[5].

This paper presents an architecture that can perform scaling for signed integers. This design performs error correction of residues for negative numbers and when simulated in the Xilinx environment, the results show that the proposed architecture is faster and occupies less area compared to the existing works.

2. BACKGROUND MATERIAL

2.1 Residue Number System (RNS)

Residue Number System (RNS) has its origination in the ancient Chinese mathematical book “Sun Zi Suan Jing”. As mentioned earlier, RNS decomposes any long integers into a set of small residues, thereby reducing the carry chain length of adders and the size of multipliers. A residue number is characterized by k-tuple of integer set $\mathbb{P} = \{p_1, p_2, p_3, ..., p_k\}$. Each of this $p_i$ is called modulus and $\mathbb{P}$ is called moduli set. Let $M$ be product of all elements of set $\mathbb{P}$. A number $X$ in decimal number system can be represented as $\{x_1, x_2, x_3, ..., x_k\}$ where $x_i = X \mod p_i$. This unique representation is applicable if $X \cdot M$ and if all elements of $\mathbb{P}$ are relatively prime to each other [7]. Residue arithmetic operation of two numbers $X_1$ and $X_2$ is defined by:

$$(Y_1, Y_2, ..., Y_k) = (X_{11}, X_{12}, ..., X_{1n}) \# (X_{21}, X_{22}, ..., X_{2n})$$

(1)

Suppose $X_1$ and $X_2$ are two different numbers with the same moduli set $\mathbb{P}$. Then, $|X_1|_{p_i} = |X_2|_{p_i}$, and so $|X_1 - X_2|_{p_i} = 0$. Hence, $X_1$ and $X_2$ are the least common multiple (LCM) of $p_i$. But if the elements of set $\mathbb{P}$ are relatively prime, then their LCM is $M$, and it must be that $X_1$ and $X_2$ is a multiple of $M$. So it cannot be that $X_1 < M$ and $X_2 < M$. Therefore, the set $\{X_{min} \leq i \leq N\}$ is unique and can be a mathematical representation of $X$.

In RNS, arithmetic operations such as addition, subtraction, and multiplication are inherently carry-free [4]-[5]. Since each digit of the result is a function of only one digit of each operand, the computational complexity is reduced to a greater extent and the desired operation is performed in minimal time. Whenever the input integer $X$ is a positive number, its range is $[0, M-1]$. On the other hand, when the integer $X$ is a non-positive number, its dynamic range is $[-M/2, M/2-1]$. 

E-ISSN: 2348-5523

International Journal of Advent Research in Computer and Electronics (IJARCE)
Vol. 2, No. 8, August 2015
2.2 Chinese Remainder Theorem

Assume two positive numbers, \( x \) (the dividend) and \( y \) (the divisor). Now \( x \) modulo \( y \) (abbreviated as \( x \mod y \)) can be thought of as the remainder, on division of \( x \) by \( y \). Suppose we have the system of equations:

\[
X \equiv x_1 \mod m_1 \\
X \equiv x_2 \mod m_2 \\
\ldots \\
X \equiv x_n \mod m_n, \text{ where } M=\{m_1, m_2, \ldots, m_n\}
\]

(2)

This can be constructed from the formula:

\[
X = \sum_{i}^{n} M_i [x_i/m_i] M_i^{-1}
\]

Where, \( M_i = m_i^{-1} \) for \( 1 \leq i \leq n \) and \( |M_i|^{-1} \) represents the inverse of \( M_i \) modulo \( m_i \).

The dynamic range of the chosen moduli set \( \{m_1, m_2, \ldots, m_n\} \) is given by the following relation:

\[
M = \prod_{i}^{n} m_i
\]

(4)

**CRT states:** If each pair of moduli \( m_i \) and \( m_j \) are relatively prime, \( i \neq j \), then the equations have a solution and any two solutions are congruent i.e. the solution is unique when \( m_i \) and \( m_j \) are co-prime and their greatest common divisor (GCD) is 1. [2] Thus according to CRT, the integer \( X \) is related to its residues by the following relation:

\[
X = \sum_{i}^{n} M_i [x_i/m_i] M_i^{-1}
\]

(5)

where \( M_i = M/m_i \). The Chinese Remainder Theorem (CRT) may be stated as one of the most important fundamental results in the theory of RNS as it solves the residue-to-binary conversion problem [5].

3. CHOICE OF MODULI SET

There are many moduli sets such as \( \{2^n-1, 2^n, 2^n+1\}, \{2^n-1, 2^n, 2^n+1, 2^n+1\}, \{2^n-1, 2^n, 2^n+1, 2^n+1\}, \{2^n-1, 2^n, 2^n+1\}, \ldots \), etc. The proposed system uses \( \{2^n-1, 2^n, 2^n+1\} \) as the desired moduli set. The moduli set \( \{2^n-2^n+1\} \) is superior to other moduli sets because it offers simple reverse conversion method from residues-to-binary number representation, thereby reducing the hardware complexity at the latter stages of the scaler [11]. Another important feature of the moduli set \( \{2^n-1, 2^n, 2^n+1\} \) is that the delay is determined only by the modulo \( 2^n+1 \) channel because of the operand length of these moduli. In other words, if we reduce the time required for \( 2^n+1 \) addition, we can easily cut down the overall delay of the RNS system [16]. The moduli set \( \{2^n-1, 2^n, 2^n+1\} \) allows easy conversion because of the presence of very efficient combinational converters from/to the binary system. Modulo \( 2^n+1 \) channel handles \( n+1 \) bit input because of the Diminished-One modulo arithmetic representation of the input, whereas modulo \( 2^n-1 \) and \( 2^n \) channels handle only \( n \) bit input operands [7]-[10]. The complexity faced in implementing modulo \( 2^n+1 \) channel is reduced using Diminished-One modulo arithmetic which can be implemented by either Circular Carry Selection scheme or the Parallel Prefix scheme.

4. UNSIGNED AND SIGNED INTEGARS

An RNS is defined by a set of \( N \) pairwise relatively prime integers \( \{m_1, m_2, \ldots, m_n\} \), where \( m_i \) is called a modulus. An unsigned integer \( X \) within the range of \([0, M-1]\) can be uniquely represented by an \( N \)-tuple \( \{x_1, x_2, \ldots, x_N\} \). The residue \( x_i \) is the least positive remainder of the division of \( X \) by \( m_i \), and is usually represented as \( X \mod m_i \). Let \( \hat{X} \) be a signed integer in the range \([-M/2, M/2 - 1]\) if \( M \) is even, or in the range \([-M-1/2, M-1/2]\) if \( M \) is odd. \( X \) can also be uniquely represented by an \( N \)-tuple \( \{\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_N\} \) in signed RNS representation. The relationship between the residue representations of \( X \) in unsigned RNS and \( \hat{X} \) in signed RNS under the same moduli set is

\[
\hat{X} = X \text{ if } X \geq 0 \text{ and } X - M \text{ if } X < 0
\]

(6)

When \( \hat{X} = 0 \), the residue representation of \( X \) can be mapped to that of \( \hat{X} \) in the range \([0, M/2 - 1]\) if \( M \) is even, or in the range \([0, (M-1)/2]\) if \( M \) is odd. When \( \hat{X} \leq 0 \), the residue representation of \( X \) can be mapped to that of \( \hat{X} \) in the range \([M/2, M-1]\) if \( M \) is even, or in the range \([(M + 1)/2, M-1]\) if \( M \) is odd. After further research into the mathematical background of RNS, the valid RNS scaler was designed by utilizing the moduli set \( \{2^n-1, 2^n, 2^n+1\} \) and the scaling factor used was \( 2^n \).

5. PROPOSED SIGNED INTEGER SCALING ALGORITHM

5.1 Scaling of Signed Integers

**Fig.1 Proposed architecture for signed integral RNS scaler**

From the definition of RNS, any integer \( X \) is decomposed into a set of residues \( \{x_1, x_2, x_3, \ldots, x_n\} \) with the help of the moduli set \( \{m_1, m_2, m_3, \ldots, m_n\} \). As mentioned earlier, the moduli set chosen for the proposed architecture is \( \{2^n-1, 2^n, 2^n+1\} \). When the residues \( x_1, x_2, x_3 \) are scaled by the scaling factor \( k = 2^n \), the resulting residues are \( \{y_1, y_2, y_3\} \). Following are the expressions for the scaled residues \( \{y_1, y_2, y_3\} \):
\[ y_1 = x_1 \cdot x_2_{m_1} \]  
\[ y_2 = (2^{n+1} + 2^{n-1}) \cdot x_1 - 2^n \cdot x_2 + \binom{2^{n+1} + 2^{n-1}}{m_1} \cdot x_3 \cdot l_{m_1} \cdot m_{l_2} \]  
\[ y_3 = x_2 \cdot 2^n \cdot x_3 \cdot l_{m_1} \]  

Let \( Y = \{ y_1, y_2, y_3 \} \) be the residues obtained by scaling \( X = \{ x_1, x_2, x_3 \} \) by \( k \) in the residue domain of RNS \( \{ 2^n-1, 2^n, 2^n+1 \} \). For \( m_1 = 2^n-1, m_2 = 2^n, \) and \( m_3 = 2^n+1 \), \( |M_i|_{m_i} = 2^n-1, |M_2|_{m_2} = 2^n, \) and \( |M_3|_{m_3} = 2^n+1 \), respectively, where \( M_i = M_{i,n} \) and \( |M_i|_{m_i} \) is the multiplicative inverse of \( M_i \) in \( \mathbb{Z} \). The scaled residues obtained after scaling for the following example can be seen from Table I.

Example: Consider the same integer \( X = 3316100 \) from Table I with the RNS representation of \( (80, 132, 29) \) for the moduli set \( \{255, 256, 257\} \). The scaled output in RNS representation is calculated in Table II. It can be verified that the binary representation of 12953 is equal to the integer, 12953 computed directly from [3316100/256]. Note that the binary representation of 12953 = 0011001010011012 is a byproduct in the computation of the scaled residue before the final modulo istaken. Its eight least significant bits are 10011001.

There is no correction of the residues \( y_1, y_2, \) and \( y_3 \) when the signed integer \( X < 0 \). When the signed integer \( X > 0 \), the residue \( y_3 \) alone needs to be corrected using the sign detection and correction circuit.

This is understood from the following expressions:

\[ \bar{y}_1 = |y_1|_{m_1} = y_1 \]  
\[ \bar{y}_2 = |y_2|_{m_2} = y_2 \]  
\[ \bar{y}_3 = |y_3|_{m_3} = y_3 \]  

The additional sign detection and correction circuit using modulo adders is employed in our proposed design as shown in Fig. 1. Since sign detection is a complex operation in RNS, the challenge of implementing a signed version of an unsigned integer RNS scaler is the overheads required for sign detection in order to correctly perform the mapping of the signed scaled results into the legitimate dynamic ranges. Ambiguities often occur when a residue lies in the boundary of the dynamic ranges for positive and negative numbers.

Besides, it may not be possible to access the already overflowed digits because the modulo operators in RNS are usually designed such that modulo reductions are performed to keep the final results and even their intermediate computations within the legitimate ranges of the respective moduli. The modulo \( 2^{n+1} \) adder of the magnitude scaler and the correction circuit are integrated into a “merged sign detection and correction” module. It comprises modified mod \( 2^n \) adder with cin, as simplified mod \( 2^n \) adder, modified control signal generation block, and an AND gate array.

The proposed architecture with merged sign detection and correction circuit is depicted in Fig. 1. The blocks present inside the dashed line box comprises the merged sign detection and correction circuit. The control signal generator present in the correction detects whenever \( Y < (2^{n-1} - 1) \) and \( \bar{Y} \) are in negative range.

| Table I Scaling of \((80, 132, 29)\) by \(k = 256\) in RNS |
|---------------------------|---------------------------|
| \( [X/k] \) \( m_1 \) | \( [80-132] \) \( 255 = 203 \) |
| \( [X/k] \) \( m_2 \) | \( [32896\times80-256\times132+32895\times29]_{255} = 12953 \) \( 256 = 153 \) |
| \( [X/k] \) \( m_3 \) | \( [132+256\times29]_{255} = 103 \) |

In this case, “1” will be chosen as the LSB of the correction factor. Otherwise, \((Y)_{2n+1}\) will be considered.

5.2 The Need for the Signed RNS Scalerm

As mentioned earlier, the ranges of the scaled unsigned integer \( Y \) differ for positive and negative values of signed integer \( X \) which is determined as follows:

Case 1: \( \bar{X} < 0 \)

\[ 0 \leq X \leq M \]  
\[ 0 \leq Y \leq \frac{M+2}{2k} \]  
\[ 0 \leq Y \leq \frac{2^{n-1}-1}{2} \]  

Case 2: \( \bar{X} > 0 \)

\[ \frac{M}{2} \leq X \leq M-1 \]  
\[ \frac{M}{2} \leq Y \leq \frac{M-1}{2k} \]  
\[ 2^{n-1}-1 \leq Y \leq 2^{2n}-2 \]  

The above cases imply that \( \bar{Y} < 0 \), when \( Y > 2^{n-1}-1 \) and \( \bar{Y} > 0 \), when \( Y < 2^{n-1}-1 \). This condition is detected by checking whether \( (2\cdot1)^{th} \) bit of \( Y \) is “1” or “0”, that is, \( \bar{Y} \) is negative when \((Y)_{2n+1}\) is “1”. In order to eliminate the sign detection ambiguity, the resultant error correction is performed by the sign detection and correction circuit. The sign detection circuit adds the \((Y)_{2n+1}\) to the LSB of \( y_2 \), whereby we get \( \bar{y}_2 = |y_2 + (Y)_{2n-1}|_{2^n} \), where \( \bar{y}_2 \) is the resultant scaled signed residue associated with \( y_2 \). Though this implementation simplifies the error, it can result in a very high fan-in multilevel logic structure, which could make the system very slow and costlier.
when \( n \) becomes large. Therefore, the fan-in of the multilevel logic gate circuit for the condition \( Y = 2^{n-1} - 1 \) can be reduced from 2n bits to n+2 bits. That is, when the condition \( Y = 2^{n-1} - 1 \) is encountered, \( Y_{2n-1} = "0" \) and \( y_1 = |2^{n-1} - 1|_{2n-1} = [(2^n)(2^{n-1}) - 1]_{2n-1} = 2^{n-1} - 1 \). Using this relation, \( y_1 \) can be computed as \( 2^{n-1} - 1 \) when \( Y = 2^{n-1} - 1 \). As there are \( 2^n + 1 \) different integers that possess \( y_1 = 2^n - 1 \) in their residue representation, \( 2^n + 1 \) of those integers will have values less than or equal to \( 2^{n-1} - 1 \). When \( y_1 = 2^n - 1 \) and \( Y \leq 2^{n-1} - 1 \), the values of \( Y \) will be \( 2^{n-1} - 1 \), \( 2^{n-1} + 2^n - 2 \), \( 2^{n-1} - 2 \), \( 2^{n-1} - 2 \), \( 2^{n-1} - 2 \), \( 2^{n-1} - 2 \), \( 2^{n-1} - 1 \). Thus, \( y_2 = 2^n - 1 \), \( 2^n - 1 \), \( 2^n - 1 \), \( 2^n - 1 \). As a result, \( (Y)_{2n-1}, y_1, \) and \( (y)_{2n-1} \) are sufficient to determine if \( Y = 2^{2n-1} - 1 \).

5.3 Overall Architecture of the Proposed Design

As seen in Fig. 1, the scaled signed residue \( y_1 \) is obtained by complementing \( x_2 \) and then by performing modulo addition between the residues \( x_1 \) and the complemented \( x_2 \). This is known as modulo \((2^n - 1)\) addition, or one’s complement addition. The modulo \((2^n - 1)\) addition of two numbers \( x \) and \( y \) is defined as follows:

\[
(x+y)\mod(2^n-1) = \begin{cases} 
(x+y+1)\mod 2^n, & x+y+1 \geq 2^n \\
 x+y, & x+y+1 < 2^n 
\end{cases}
\]

(15)

Designing a modulo \((2^n)\) adder is a challenging task as the number has to be represented in Diminished-One number system.

Any number in Diminished-One system is represented by \((n+1)\) bit in which \((n+1)\) bit is generally assigned as zero. In this system, if the most significant bit of any one addend is found to be ‘1’, it is mandatory to inhibit addition, and the other addend is taken as sum. If the most significant bits of both addends are zero, it is necessary to ignore the sum, add ‘n’ lsb’s, complement carry and then add it to the n lsb’s of the sum. This operation is done by the Carry Save Adder with Complementary End-Around Carry.

The additional bit is always assumed to be ‘1’ when the number to be represented is ‘0’ in order to avoid additions and multiplications involving the additional bit. The highlight of Diminished-One representation is that zero is uniquely identified by MSB=1, for which it is necessary to inhibit all other arithmetic operations. In \( 2^n \) channel, we incorporate a circuitry for performing bit rewiring. Bit rewiring takes the unscaled residues as inputs, arranges them in a particular mathematical format using concatenation, and then produces three vectors \( Q_1, Q_2, \) and \( Q_3 \). These vectors are then used to yield the desired scaled residue after performing the required modulo operations.

The control signal generation block used in the computation of \( y_2 \) can be built by using \( n+2 \) two-input AND gates with \( y_2, (\bar{x}_2)_{2n-1}, (Y)_{2n-1}, \) and \( (y)_{2n-1} \) as the inputs. The proposed modified mod \( 2^n \) adder with \( c_0 \) performs the computation of \( y_2 \). As seen from the figure, the \( 2n \)-bit carry-save adder (CSA) with end-around carry block produces the sum and carry vectors \( A \) and \( B \), respectively. The modified mod \( 2^n \) adder adds the two \( n \)-bit operands taken from the \( n \) LSBs of the sum and carry vectors \( A \) and \( B \).

The simplified mod \( 2^n \) adder performs the mapping by adding \((Y)_{2n-1}\) to \( y_1 \), such that \( y_2 = y_2 + (Y)_{2n-1} \). In other words, when \( Y = 2^{2n-1} - 1 \), and \( \bar{Y} \) is in the negative range

\[
y_2 = y_2 + (Y)_{2n-1} = 2^{n-1} + 1 + 1 = 2^n \text{ and } \bar{Y} = 0
\]

For other cases, \((Y)_{2n-1}\) must be added to \( y_2 \), whereby we arrive at the following relation:

\[
y_2 = y_2 + (Y)_{2n-1} \text{ and } \bar{Y} = 1
\]

Thus the corrected output can be either “0” or \(|y_2 + (Y)_{2n-1}| \). In order to obtain the corrected output, an AND gate array comprising \( n \) two-input AND gates is employed instead of an \( n \)-way multiplexer.

6. RESULTS

The RTL code for the proposed signed integer RNS scaler for the moduli set \{2, 2^n, 2^{n+1}\} is written using Verilog. The Verilog coding is then synthesized and simulated to check the correctness of the design using Xilinx ISE 9.11.

<table>
<thead>
<tr>
<th>Table II</th>
<th>Comparison of hardware usage by various modules</th>
</tr>
</thead>
<tbody>
<tr>
<td>MODULE</td>
<td>RNS SCALER</td>
</tr>
<tr>
<td>Adder</td>
<td>118</td>
</tr>
<tr>
<td>1-bit XOR</td>
<td>1</td>
</tr>
<tr>
<td>LUT</td>
<td>105</td>
</tr>
<tr>
<td>Muxcy</td>
<td>7</td>
</tr>
<tr>
<td>Input Buffers</td>
<td>25</td>
</tr>
<tr>
<td>Output Buffers</td>
<td>24</td>
</tr>
</tbody>
</table>

Table II shows the comparison of hardware usage by various modules. From the table, it is observed that RNS scaler requires 118 additions, 1 ex-or operation, 105 look-up tables, 7 multiplexers and 49 buffers. Further it shows that the computation of scaled residue \( y_3 \) requires only 32 buffers, 63 additions and 61 look-up tables. The RTL schematic of the proposed signed integer RNS scaler is given in figure 2. The design aims to achieve optimization in terms of speed.
Each channel for the computation of the scaled residues from unscaled residues is specified at structural level using Verilog and synthesized.

Figure 3 provides a chart comparing the hardware usage of various modules of the architecture. From the chart, it is understood that the RNS scaler module occupies more slices compared to other basic modules used for the independent computations of the scaled residues.

Table III Memory consumption by various modules

<table>
<thead>
<tr>
<th>MODULES</th>
<th>MEMORY CONSUMPTION (IN KiloBytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>RNS scaler</td>
<td>105656</td>
</tr>
<tr>
<td>Scaled residue $y_1$</td>
<td>105528</td>
</tr>
<tr>
<td>Scaled residue $y_2$</td>
<td>106104</td>
</tr>
<tr>
<td>Scaled residue $y_3$</td>
<td>105656</td>
</tr>
</tbody>
</table>
consumes more memory compared to
Garcia et al. (1999) Two lookup table
works. From the
International Journal of Advent Research in Computer and Electronics
design with its contender
Fig. 6
s a comparison of the unit
chart, it is understood that the proposed RNS scaler module
50
15
HARDWARE USAGE
its existing systems, thereby making the proposed design
15
89196
Proposed Design
Fig. 5
modules in a chart format in which the module used for
finding scaled residue $y$ consumes more memory compared to
other modules.

Table IV Comparison of number of logic elements in the
proposed design with its contenders

<table>
<thead>
<tr>
<th>IMPLEMENTATION</th>
<th>CONCEPT</th>
<th>NO. OF LOGIC ELEMENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Griffin et al. (1988)</td>
<td>L-CRT</td>
<td>89196</td>
</tr>
<tr>
<td>Shenoy et al. (1989)</td>
<td>CRT Decomposition</td>
<td>1216816</td>
</tr>
<tr>
<td>Ulman et al. (1993)</td>
<td>CRT based</td>
<td>323438</td>
</tr>
<tr>
<td>Garcia et al. (1999)</td>
<td>Two lookup table</td>
<td>918252</td>
</tr>
<tr>
<td>Dasygenis et al. (2008)</td>
<td>FA based</td>
<td>11940</td>
</tr>
<tr>
<td>Thian Fatt Tay et al. (2013)</td>
<td>ROM based</td>
<td>2648</td>
</tr>
<tr>
<td>Proposed Design</td>
<td>CSA based</td>
<td>1890</td>
</tr>
</tbody>
</table>

Table IV gives the comparison between the number of logic
elements required in the proposed design with its existing works. From the table, it is understood that the
proposed work requires less number of logic elements, thereby
minimizing the area requirement. Figure 5 provides a chart
comparing the number of logic elements present in the
proposed design as well as in the existing works. From the chart, it is understood that the proposed RNS scaler module
requires less number of logic elements compared to its
existing works.

Table V gives the comparison of unit-gate delay of the
proposed design with its existing works. From the table, it is
observed that the proposed design has less delay compared to
its existing systems, thereby making the proposed design
morefaster than the existing works.

Table V Comparison of unit-gate delay of the
proposed design with its contender

<table>
<thead>
<tr>
<th>IMPLEMENTATION</th>
<th>DELAY (in ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shenoy et al. (1989)</td>
<td>131</td>
</tr>
<tr>
<td>Ulman et al. (1993)</td>
<td>51</td>
</tr>
<tr>
<td>Garcia et al. (1999)</td>
<td>50</td>
</tr>
<tr>
<td>Dasygenis et al. (2008)</td>
<td>30</td>
</tr>
<tr>
<td>Thian Fatt Tay et al. (2013)</td>
<td>27</td>
</tr>
<tr>
<td>Proposed Design</td>
<td>15</td>
</tr>
</tbody>
</table>

Fig. 6 Comparison of logic units of the proposed
design with its contender
Figure 6 provides a chart comparing the unit-gate delay of proposed work with the existing works. From the chart, it is observed that the proposed design is faster compared to other existing works as it has less unit-gate delay.

7. CONCLUSION

A fast and low complex $2^n$ signed integer RNS scaler for moduli set $\{2^n - 1, 2^n, 2^{n+1}\}$ has been proposed in this work. By simplifying and merging the expensive sign detection and correction circuits, the complexity of implementing the signed integer RNS scaler was reduced substantially. The proposed design has been benchmarked against two latest signed integer RNS scalers using the unit-gate analysis and logic synthesized results. The reduction in the number of logic gates is achieved in this design along with the reduction in the delay. In continuation to this work, further research can be done in designing interpolation and folding techniques to increase the speed of the system. The pipelining technique can also be used to improve the throughput in the future works.

REFERENCES


