RNS Scaling Algorithm for a New Moduli Set $\{2^{2n+1}+1, 2^{2n+1}, 2^{2n+1}-1\}$

Khalid Shahabdeen Mustapha
University for Development Studies
Navrongo Campus

Edem Kwedzo Bankas
University for Development Studies,
Faculty of Mathematical Sciences,
Department of Computer Science.

ABSTRACT
Scaling is one of the difficult operations in Residue Number System (RNS) and also one of the most important units and a necessary operation used to avoid overflow in RNS based systems. In this paper, a scaling algorithm for a new moduli set $\{2^{2n+1}+1, 2^{2n+1}, 2^{2n+1}-1\}$ using the Chinese Remainder Theorem (CRT) is presented. In the design of digital systems, the goal of designers is to increase performance and decrease the amount of hardware resources. In order to achieve this, a new moduli sets is proposed to obtain a larger dynamic range and less complex hardware architecture. The CRT is further simplified for the selected moduli set to reduce the hardware complexity of the scaling algorithm. The scaling algorithm does not introduce any scaling errors and thus is efficient. When compared with the state of the art scaling algorithm using the Unit- Gate model, the results show that, the proposed scaling algorithm outperforms the state of the art scaling algorithm in terms of dynamic range (DR), area consumption, and delay by 98%, 18.4% and 21.7% respectively.

General Terms
Moduli Set, Scaling Algorithm, Residue Number System, Chinese Remainder Theorem.

Keywords
Scaler, RNS, Chinese Remainder Theorem, Moduli Set, Dynamic Range.

1. INTRODUCTION
The advent of technology in the fields of embedded systems which has found wide spread uses in mobile telephony and tablet personal computers has generated the interest of many researchers in the areas of improving power consumption, speed and hardware resource requirements [1]. Through the quest of these researchers, Residue Number System (RNS) has been discovered to be the torch bearer in these fields [2]. As a result, RNS has over the years played an important role in championing the campaign in designing digital systems that require computational intensive arithmetic operations such as addition, subtraction, and multiplication.

This advantage is as a result of the carry propagation problem in binary number systems [3]. Performing calculations in an RNS based system leads to less delay in processing time (clock cycles), reduction in the cost of hardware resources and power consumption [4]. In RNS the carry propagation is limited in modular channels, while in other number systems distribution chain of the carry is long, which makes the process slow. Consequently in the RNS addition, subtraction and multiplication operations are very fast. This makes RNS to be applied in the areas of communications, digital signal processing, computer security, image processing, speech processing and transformation, error detection and correction algorithms, and encoding [1], [4], [5]. Also, RNS is used in designing Inner Product Step (IPS) processors.

However, RNS has not found wide spread usage in general purpose computing due to the following difficult and costly to implement arithmetic operations: sign detection, division, reverse division, magnitude comparison, overflow detection, moduli selection and scaling. The two latter operations are more urgent to be solved because they are the gateway to designing a circuit for converting numbers from residue to binary systems. Reverse conversion is a costly and complicated operation, and scaling circuits are used for preventing overflow after each level of processing [6]. Complex operations are usually performed by employing algorithms that are used in reverse conversion such as Chinese Remainder Theorem (CRT) and Mixed Radix Conversion (MRC). The primary and most important parameter in designing an RNS based system is moduli set selection, in which the numbers are relatively prime Scaling therefore plays an important role in achieving universal application of RNS in general purpose computing but scaling within RNS is less efficient and this problem has long prevented wider adoption of RNS [7]. In this paper, a scaling algorithm for a new moduli set $\{2^{2n+1}+1, 2^{2n+1}, 2^{2n+1}-1\}$ is presented.

1.1 Fundamental Principles of RNS
RNS is defined in terms of a set of relatively prime integer set $\{m_i\}_{i=1,k}$ called the moduli set, such that the $\gcd(m_i, m_j) = 1$ for $i \neq j$, where gcd means the greatest common divisor of m_i, and m_j, while

$M = \prod_{i=1}^{k} m_i$, is the dynamic range. The residues of a decimal number can be obtained as $x_i = |X|_{m_i}$, thus X can be represented in RNS as $X = (x_1, x_2, x_3, \ldots, x_k)$. This representation is unique for any integer $X \in [0, M-1]$ [8].

2. RELATED WORKS
The first scaling scheme ever to have been proposed is the one proposed in 1967 by Szabo and Tanaka [9]. They proposed a scalar that needed n clock cycles for n-bit moduli set. Although the scaled residues had errors, and the scheme did not provide correct scaled residues, it was a significant stage in the development of RNS-based systems. In another major study in 1973, O'keefe and Wright proposed a faster and more efficient scalar than Szabo. Again the results were not error-free but their approach provided results closer to the correct scaled integers [10]. In 1978 Jullien was successful in designing an algorithm that needed fewer clock cycles, but provided faulty results [11]. In 1981, Taylor and Huang proposed a design based on the MRC. It was the first time a scaler based on the MRC was proposed [12]. Until then, all designs were based on CRT or base-extension. The CRT-based algorithms generally generated fractional errors due to

inherent assumptions, while the latter approach was error-free but computationally intensive.

One year later, Taylor and Huang presented a scaler that used a special moduli set and LUTs. However, their design required n clock cycles to generate the scaled residues [13]. Miller and Polky proposed a design that needed (n+1) clock cycles but the scaled residues were closer to the correct results [14]. In other words, their design provided more accurate scaled residues at the cost of more clock cycles. Two scaling algorithms for the moduli set (2^n-1, 2^n, 2^n + 1) using CRT were proposed in 1989 [15]. The first algorithm was based on the L-CRT algorithm for L- tuple RNS. The second algorithm was based on 2^{2n} - q CRT using an approximation of M = 2^{3n} and M1= 2^q, q < 2^n. Both algorithms came with assumptions and errors. Later, Shenoy and Kumaresan proposed two scaling techniques (approximate and exact), where residues were scaled by the product of a subset of the moduli set [16]. The approximate technique used a redundant residue to eliminate modulo-(M) operation, while the exact technique used a modified version of CRT. The scaling error in the approximate technique was bounded by (i=2), where i is the number of moduli in the moduli set. Their design saved a considerable amount of delay and generated results in only log n clock cycles. The exact technique, however, generated an error of at most unity, and used a redundant channel to keep track of odd or even residues.

Ulman published a modified version of the Szabo scaler in 1993 [17], and since then, the results of all scalers have errors less than 1.5. Another CRT- based scaling scheme was presented in 1995 [18]. It used LUTs and $\log_2$ n clock cycles to generate scaled residues. The aim of the design was to achieve a precise result without using any redundant representation of numbers. The disadvantage was its worst case delay of n clock cycles. Two stages of look-up-cycle scaling, namely look-up calculation and look-up generation, were presented in [19]. The design was recommended for 5-bit input and three moduli sets. It was cascadable to other algorithms for larger sets of moduli and reduced the bulk memory requirement for small moduli sets. In 2003, an alternative CRT- based scaler for up to 16-bit dynamic range was proposed [20]. The proposed scheme used only RNS operations within small-word-length channels. It was suitable for small-word-length applications and performed scaling directly on the residue digits rather than relying on residue-to-binary conversion. From the implementation point of view, scaling algorithms are implemented either in LUT (look up table) based approaches [7], [15], [16], [17], [21], [22], [23], or adder-based approaches [5]. Generally, all the LUT-based designs in the literature are subject to poor pipeline-ability and high hardware complexity when the number of moduli increases. Adder-based designs are faster and provide huge savings in storage space. There are also other scaling circuits that benefit from both LUTs and full adders [6]. Most scaling schemes reported in the literature are based on LUTs, and none of the papers discussed the order of generation of scaled residues until 2007 [24]. Almost all publications have agreed that LUTs are more efficient for small inputs, as in image processing applications, while a FA-based structure is well suited for long inputs [3]. Extensive measurements in area, delay and hardware utilization for full-adder-based designs have been proposed.

2.1 Mathematical Basis for Scalers

Scaling in RNS is a special type of division in which a number is divided by a constant factor followed by truncation or rounding. It corresponds to the division of an integer (X) by a constant (K). It can be shown as:

$$Y = \left\lfloor \frac{X}{K} \right\rfloor \quad (1)$$

Where K is called the scaling factor, Y is the result of scaling X by K and $\lfloor . \rfloor$ is the floor function.

2.2 The Chinese Remainder Theorem (CRT)

The Chinese Remainder Theorem is a very useful theorem used in the reverse conversion process and other operations in RNS [25]. With well selected moduli set, the CRT guarantees that a number within the legitimate range will have unique representation in RNS. The unique number represented in RNS can be derived through the use of the CRT.

Given a set of pair wise relatively prime moduli set, $m_1, m_2, \ldots m_n$ and a number X whose residue representation is $(x_1, x_2, \ldots\ldots, x_n)$ in the system, where $x_i = |X|_{m_i}$, the number X and its residue are related by the equation below;

$$X = \left| \sum_{i=1}^{n} M_i \left| M_i^{-1} \right|_{m_i} x_i \right|_M \quad (2)$$

Where $M_i = \frac{M}{m_i}$ and $\left|M_i^{-1}\right|_{m_i}$ is the multiplicative inverse of M_i with respect to m_i, $x_i = |X|_{m_i}$, M= $\prod_{i=1}^{N} m_i$ is the DR and m_i is the moduli set.

Equation (2) is the Chinese Remainder Theorem (CRT) [26], [27], and [28].

2.3 Moduli Set Selection

The choice of moduli set affects both the representational efficiency and the complexity of the arithmetic of the algorithm. It is therefore critical to choose the moduli set carefully and strategically to ensure that efficiency is guaranteed [29]. First we demonstrate that the moduli set chosen are pairwise relatively prime.

Theorem 1:

The moduli set $\{2^{2n+1}+1, 2^{2n+1}, 2^{2n+1}-1\}$ contains pair wise relatively prime numbers..

Proof:

From Euclidean theorem, we have gcd (a,b) = (b, $|a|_b$), where gcd refers to greatest common divisor. Applying this to the moduli set pair wisely will yield the following;

gcd(2^{2n+1} , $|2^{2n+1}+1|_{2^{2n+1}}$) = gcd (2^{2n+1},1) = 1

gcd($2^{2n+1}-1$, $|2^{2n+1}+1|_{2^{2n+1}-1}$) = gcd($2^{2n+1}-1$, 2)= 1

gcd($2^{2n+1}-1$, $|2^{2n+1}|_{2^{2n+1}-1}$) =gcd ($2^{2n+1}-1, -1$)= 1

Hence the moduli are relatively prime.

Example 1: Given n =1, the moduli set are {9, 8, 7}. gcd (9, 8) = 1, gcd (9, 7) = 1 and gcd (8, 7) = 1

3. PROPOSED SCALING ALGORITHM

In RNS an integer X is represented by an N-tuple $(x_1, x_2, \ldots., x_N)$ with respect to a set of pairwise relatively.

Hence the moduli set are relatively prime numbers $\{m_1, m_2, \ldots., m_N\}$, where $x_i = |X|_{m_i}$, i= 1, 2, . . . ,N and $|X|_{m_i}$ is defined as X mod m_i. The dynamic range M is the number of representable numbers of a selected moduli set {m, m, ….. , m} is given by:

$$M = \prod_{i=1}^{N} m_i \quad (3)$$

Based on the CRT, X is related to its residue digits by:

$$X = \left| \sum_{i=1}^{N} M_i \left| M_i^{-1} \right|_{m_i} x_i \right|_M \quad (4)$$

Where $M_i = \frac{M}{m_i}$ and $\left|M_i^{-1}\right|_{m_i}$ is the multiplicative inverse of M_i with respect to m_i.

3.1 Formulation of Scaling Equation

The proposed scaling algorithm is designed for the - moduli set $\{2^{2n+1}+1, 2^{2n+1}, 2^{2n+1}-1\}$. using the above moduli set, equation (2) can be expressed as follows:

$$X = \left| m_2 m_3 |M_1^{-1}|_{m_1} x_1 + m_1 m_3 |M_2^{-1}|_{m_2} x_2 + m_1 m_2 |M_3^{-1}|_{m_3} x_3 \right|_M \quad (5)$$

The following axioms are used for the derivation of the scaling equation.

Axiom 1: $A\ |X|_B = |AX|_{AB}$

Axiom 2: $|X \pm Y|_m = ||X|_m \pm |Y|_m|_m$

Axiom 3: $|X \,.\, Y|_m = ||X|_m . |Y|_m|_m$

Lemma 1: Given $p = K \times q$, where K is an integer, $\left||X|_p\right|_q = |X|_q$

Proof: Based on the definition of modulo operation;

$$|X|_p = X - \varepsilon p,\ \varepsilon = 0, 1, 2, \ldots \quad (6)$$

Using Axioms 2 and 3,

$$\left||X|_p\right|_q = |X - \varepsilon p|_q = \left||X|_q - |\varepsilon|_q \,.\, |p|_q\right|_q$$

Since p is divisible by q, $|p|_q = 0$

$$\left||X|_p\right|_q = |X|_q \quad (7)$$

3.2 The Scaling Process

Scaling is a special type of division. By definition, scaling an integer variable X by a constant K can be obtained by dividing both sides of (5) by constant K and taking the floor value. Let Y be the integer results of the scaling operation, we shall have:

$$Y = \left\lfloor \frac{X}{K} \right\rfloor \quad (8)$$

$$Y = \left\lfloor \frac{1}{K} \left| m_2 m_3 |M_1^{-1}|_{m_1} x_1 + m_1 m_3 |M_2^{-1}|_{m_2} x_2 + m_1 m_2 |M_3^{-1}|_{m_3} x_3 \right|_M \right\rfloor \quad (9)$$

$$Y = \left\lfloor \left| \frac{m_2 m_3}{K} |M_1^{-1}|_{m_1} x_1 + \frac{m_1 m_3}{K} |M_2^{-1}|_{m_2} x_2 + \frac{m_1 m_3}{K} |M_3^{-1}|_{m_3} x_3 \right|_{\frac{M}{K}} \right\rfloor \quad (10)$$

Let K be equal to $m_2 = 2^{2n+1}$ (This choice of K is crucial because it ensures that the truncation error becomes negligible). Substituting $K = m_2$ into equation (10), we shall have the exact scaling equation below:

$$\left\lfloor \frac{X}{K} \right\rfloor = \left\lfloor \left| m_3 |M_1^{-1}|_{m_1} x_1 + \frac{m_1 m_3}{m_2} |M_2^{-1}|_{m_2} x_2 + m_1 |M_3^{-1}|_{m_3} x_3 \right|_{m_1 m_3} \right\rfloor \quad (11)$$

Using (11), the scaled integer can be computed directly from the RNS representation of X.

3.3 Formulation of RNS Scaling Algorithm

From the above deductions, the complexity of the modulus channels can be reduced with the proposed theorem.

Theorem 2:

Given the moduli set $\{2^{2n+1}+1, 2^{2n+1}, 2^{2n+1}-1\}$, where $m_1 = 2^{2n+1}+1$, $m_2 = 2^{2n+1}$, $k = m_2$

$m_3 = 2^{2n+1} - 1$, the following holds true

$$\left|\left|\frac{X}{k}\right|\right|_{m_1} = |-x_1 + x_2|_{2^{2n+1}+1} \quad (12)$$

$$\left|\left|\frac{X}{k}\right|\right|_{m_2} = ||(2^{4n+1} + 2^{2n} - 1)(x_1) + (x_2)(2^{4n+2} - 2^{2n+1} - 1) + (2^{4n+1} + 2^{2n})(x_3)|_{2^{4n+2}-1}|_{2^{2n+1}} \quad (13)$$

$$\left|\left|\frac{X}{k}\right|\right|_{m_3} = |-x_2 + x_3\ |_{2^{2n+1}-1} \quad (14)$$

The proof and formulation of this theorem is presented as follows.

Using the exact scaling equation (9), RNS scaling can be performed in each channel independently by performing modulo reduction in each channel. This Results in the equation below.

$$\left|\left|\frac{X}{k}\right|\right|_{m_i} = \left| \left| m_3 |M_1^{-1}|_{m_1} x_1 + \frac{m_1 m_3}{m_2} |M_2^{-1}|_{m_2} x_2 + m_1 |M_3^{-1}|_{m_3} x_3 \ \right|_{m_1 m_3} \right|_{m_i} \quad (12)$$

Where i= 1, 2 and 3.

For m_1 and m_3 channels, where i = 1 and 3 using lemma 1 on equation (12) we obtain:

$$\left|\left|\frac{X}{k}\right|\right|_{m_i} = \left| m_3 |M_1^{-1}|_{m_1} x_1 + \frac{m_1 m_3}{m_2} |M_2^{-1}|_{m_2} x_2 + m_1 |M_3^{-1}|_{m_3} x_3 \ \right|_{m_i} \quad (13)$$

Each independently scaled residue of equation (13) can further be reduced respectively for m_1 and m_3 to;

$$\left|\left|\frac{X}{k}\right|\right|_{m_1} = \left| m_3 |M_1^{-1}|_{m_1} x_1 + \frac{m_1 m_3}{m_2} |M_2^{-1}|_{m_2} x_2 \right|_{m_1} \quad (14)$$

$$\left|\left|\frac{X}{k}\right|\right|_{m_3} = \left| \frac{m_1 m_3}{m_2} |M_2^{-1}|_{m_2} x_2 + m_1 |M_3^{-1}|_{m_3} x_3 \ \right|_{m_3} \quad (15)$$

For the m_2 channel, we have the following;

$$\left|\left|\frac{X}{k}\right|\right|_{m_2} = \left| \left| m_3 |M_1^{-1}|_{m_1} x_1 + \frac{m_1 m_3}{m_2} |M_2^{-1}|_{m_2} x_2 + m_1 |M_3^{-1}|_{m_3} x_3 \ \right|_{m_1 m_3} \right|_{m_2} \quad (16)$$

From equations (14) through to (16), it can be seen that there exists a common term

$\frac{m_1 m_3}{m_2}|M_2^{-1}|_{m_2}$. To simplify the common term, the following theorem is proposed.

Theorem 3:

For the moduli set $\{2^{2n+1}+1, 2^{2n+1}, 2^{2n+1}-1\}$ where;

$M_1 = (2^{2n+1})(2^{2n+1}-1)$,

$M_2 = (2^{2n+1}+1)(2^{2n+1}-1)$,

$M_3 = (2^{2n+1}+1)(2^{2n+1})$

The following holds true:

$$|M_1^{-1}|_{m_1} = 2^{2n}+1 \quad (17)$$

$$|M_2^{-1}|_{m_2} = 2^{2n+1}-1 \quad (18)$$

$$|M_3^{-1}|_{m_3} = 2^{2n} \quad (19)$$

Substituting the parameters $m_1 = 2^{2n+1}+1,\ m_2 = 2^{2n+1}$ „

$m_3 = 2^{2n+1}-1 \ \ and\ |M_2^{-1}|_{m_2} = 2^{2n+1}-1$

into the common term we simplify it as;

$$\frac{m_1 m_3}{m_2}|M_2^{-1}|_{m_2} = \left\lfloor \frac{(2^{2n+1}-1)(2^{2n+1}-1)(2^{2n+1}-1)}{2^{2n+1}} \right\rfloor$$

$$\frac{m_1 m_3}{m_2}|M_2^{-1}|_{m_2} \cong 2^{4n+2} - 2^{2n+1} - 1 \quad (20)$$

The least integer function$\lfloor . \rfloor$, justifies the truncation in (20). By substituting (20) into (14), (15) and (16) where $m_1 = 2^{2n+1}+1$, $m_2 = 2^{2n+1}$, and $m_3 = 2^{2n+1}-1$ we obtain

$$y_1 = \left|\left|\frac{X}{k}\right|\right|_{2^{2n+1}+1} = \Big|\underbrace{(2^{2n+1}-1)(2^{2n}+1)x_1}_{A} + \underbrace{(2^{4n+2}-2^{2n+1}-1)x_2}_{B}\Big|_{2^{2n+1}+1} \quad (21)$$

$$y_2 = \left|\left|\frac{X}{k}\right|\right|_{2^{2n+1}} = \Big|\Big|\underbrace{(2^{2n+1}-1)(2^{2n}+1)(x_1)}_{E} + \underbrace{((x_2)2^{4n+2}-2^{2n+1}-1\ )}_{F} + \underbrace{(2^{2n+1}+1)(2^{2n})(x_3)}_{G}\Big|_{m_1 m_3}\Big|_{2^{2n+1}} \quad (22)$$

$$y_3 = \left|\left|\frac{X}{k}\right|\right|_{2^{2n+1}-1} = \Big|\underbrace{(2^{4n+2}-2^{2n+1}-1)x_2}_{C} + \underbrace{(2^{2n+1}+1)(2^{2n})x_3}_{D}\Big|_{2^{2n+1}-1} \quad (23)$$

Where $m_1 m_3 = 2^{4n+2}-1$, applying axioms 1, 2, 3 and lemma 1 to (21), (22), and (23), we have the highly simplified equations presented in theorem 2.

Illustrative Example 1: Given the moduli set $\{2^{2n+1}+1, 2^{2n+1}, 2^{2n+1}-1\}$, perform the scaling of an integer X= 89 for n=1.

Solution

Given the moduli set $\{2^{2n+1}+1, 2^{2n+1}, 2^{2n+1}-1\}$; When n=1, the moduli set is {9, 8, 7} and M= $9 \times 8 \times 7 = 504$, X=$89 < 504$.

x_i= X mod m_i for i = 1, 2 and 3 as [8, 1, 5] and

$y_1 = |-x_1 + x_2|_{m_1} = |-8+1|_9 = 2$

$y_2 = ||(2^{4n+1}+2^{2n}-1)(x_1) + (x_2)(2^{4n+2}-2^{2n+1}-1) + (2^{4n+1}+2^{2n})(x_3)|_{2^{4n+2}-1}|_{2^{2n+1}} = ||(35\times 8)+(55\times 1)+(36\times 5)|_{63}|_8$= 3 and

$y_3 = |-x_2 + x_3\ |_{m_3} = |-1+5|_7$= 4.

2, 3, and 4 is the RNS representation of 11, the scaled results. This is summarized in table 1.

Table 1. Numerical Example of Proposed Scaler for Moduli Set {9, 8, 7}, Scaling Factor K = 8, n =1, X = 89, and x_i= (8, 1, 5)

$y_1 = \left\|\left\|\frac{X}{K}\right\|\right\|_{m_1} = \left\|\left\lfloor\frac{89}{8}\right\rfloor\right\|_9 = 2$	$\|1-8\|_9 = 2$
$y_2 = \left\|\left\|\frac{X}{K}\right\|\right\|_{m_2} = \left\|\left\lfloor\frac{89}{8}\right\rfloor\right\|_8 = 3$	$\|\|(35\times 8)+(55\times 1)+(36\times 5)\|_{63}\|_8 = 3$
$y_3 = \left\|\left\|\frac{X}{K}\right\|\right\|_{m_3} = \left\|\left\lfloor\frac{89}{8}\right\rfloor\right\|_7 = 4$	$\|5-1\|_7 = 4$

4. HARDWARE IMPLEMENTATION

The hardware realization of the proposed scaling algorithm is discussed below. We implement the algorithm for channels one, two and three. A general architecture is proposed for the proposed scaling algorithm. Given the moduli set $\{2^{2n+1}+1, 2^{2n+1}, 2^{2n+1}-1\}$, the residue digits $x_1, x_2\ and\ x_3$ relative to the moduli set are $2n+2\ bits,\ 2n+1\ bits\ and\ 2n+1 bits$ respectively.

For the implementation of the $m_1$channel given by; $y_1 = \left\|\left\|\frac{X}{K}\right\|\right\|_{m_1} = |x_2 - x_1|_{m_1}$we shall have the following;

$$y_1 = |x_2 + \bar{x}_1|_{m_1} \quad (24)$$

Since x_1 is a $2n+2$ bit and x_2 a 2n+1 bit integers, then this can be written as

$$x_1 = \underbrace{(x_{1,2n+1}x_{1,2n}\dots x_{1,1}x_{1,0})}_{2n+2} \quad (25)$$

$$x_2 = \underbrace{(x_{2,2n}x_{2,2n-1}\dots x_{2,1}x_{2,0})}_{2n+1} \quad (26)$$

$$\bar{x}_1 = \underbrace{(\bar{x}_{1,2n+1}\bar{x}_{1,2n}\dots \bar{x}_{1,1}\bar{x}_{1,0})}_{2n+2} \quad (27)$$

$|-x_1|_{2^{2n+1}+1} = |2^{2n+1}+1-x_1|_{2^{2n+1}+1} = \bar{x}_1$, where $\bar{x}_1$is the complement of x_1.

The two numbers to be added are of different bit lengths. To do so we ensure that they are of the same bit lengths. This can be done by adding a zero bit at the MSB position of x_2, and take the one's complement of x_1.

$$|x_2 + \bar{x}_1|_{2^{2n+1}+1} = \Big|\overbrace{(0x_{2,2n}x_{2,2n-1}\dots x_{2,1}x_{2,0})}^{2n+2} + \underbrace{(\bar{x}_{1,2n+1}\bar{x}_{1,2n}\dots \bar{x}_{1,1}\bar{x}_{1,0})}_{2n+2}\Big|_{2^{2n+1}+1} \quad (28)$$

The two numbers $x_2\ and\ x_1$ are now of the same bit length that is $2n+2$ bit and can be added together. The generation of y_1 requires two carry propagate adders (CPA). The two operands $x_2\ and\ x_1$ are added by the first CPA with an End Around Carry (EAC) to produce a sum and a carry bit of 1. The second CPA adds the sum and carry bit in $2^{2n+1}+1$ modulo adder to produce a sum.

$$y_3 = |x_3 - x_2|_{m_3} \tag{29}$$

$$y_3 = |x_3 - x_2|_{2^{2n+1}-1} \tag{30}$$

$$|x_3 - x_2|_{2^{2n+1}-1} = \left| \overbrace{(x_{3,2n}x_{3,2n-1}\dots x_{3,1}x_{3,0})}^{2n+1} - \underbrace{(x_{2,2n}x_{2,2n-1}\dots x_{2,1}x_{2,0})}_{2n+1} \right|_{2^{2n+1}-1} \tag{31}$$

The two numbers $x_2\ and\ x_3$are of the same bit length $(2n+1)$and so can be added together. To do so, the complement of x_2 is taken, resulting in the following equation below.

$$|x_3 + \bar{x}_2|_{2^{2n+1}-1} = \left| \overbrace{(x_{3,2n}x_{3,2n-1}\dots x_{3,1}x_{3,0})}^{2n+1} + \underbrace{(\bar{x}_{2,2n}\bar{x}_{2,2n-1}\dots \bar{x}_{2,1}\bar{x}_{2,0})}_{2n+1} \right|_{2^{2n+1}-1} \tag{32}$$

The result y_3 can thus be generated using two operands $x_2\ and\ x_3$, a CPA, and an inverter. The inverter will generate the complement of x_2 and the CPA will add the two operands $x_3\ and\ \bar{x}_2$ to generate the required results.

The m_2 channel can be implemented to generate y_2 as follows.

$$y_2 = ||(2^{4n+1} + 2^{2n} - 1)(x_1) + (2^{4n+2} - 2^{2n+1} - 1)(x_2) + (2^{4n+1} + 2^{2n})(x_3)|_{2^{4n+2}-1}|_{2^{2n+1}} \tag{33}$$

$$x_1 = \underbrace{(x_{1,2n+1}x_{1,2n}\dots x_{1,1}x_{1,0})}_{2n+2} \tag{34}$$

$$x_2 = \underbrace{(x_{2,2n}x_{2,2n-1}\dots x_{2,1}x_{2,0})}_{2n+1} \tag{35}$$

$$x_3 = \underbrace{(x_{3,2n}x_{3,2n-1}\dots x_{3,1}x_{3,0})}_{2n+1} \tag{36}$$

The negation of x_2 will require the complement of (35). This is done below.

$$-x_2 = \underbrace{(\bar{x}_{2,2n}\bar{x}_{2,2n-1}\dots \bar{x}_{2,1}\bar{x}_{2,0})}_{2n+1} \tag{37}$$

Now we expand equation (33) as follows:

$$y_2 = \left\| \underbrace{(2^{4n+1}(x_1) + 2^{2n}(x_1) - (x_1))}_{G} + \overbrace{(2^{4n+2}(x_2) - 2^{2n+1}(x_2) - (x_2))}^{H} + \underbrace{(2^{4n+1}(x_3) + 2^{2n}(x_3))}_{I} \right|_{2^{4n+2}-1} \Big|_{2^{2n+1}} \tag{38}$$

Let $g_1 = (2^{4n+1}(x_1),\ g_2 = 2^{2n}(x_1),\quad g_3 = (-x_1)\ ,\ h_1 = 2^{4n+2}(x_2), h_2 = -2^{2n+1}(x_2)\ ,$

$$h_3 = (-x_2),\ i_1 = 2^{4n+1}(x_3), \text{and } i_2 = 2^{2n}(x_3)$$

Now we define the following properties of modulo 2^n-1 arithmetic proposed by [9].

Property 1: Multiplying an n-bit binary number x by r power of two in modulo 2^n-1 is equivalent to a circular left shift operation.

$\rightarrow |2^r x|_{2^n-1} = CLS_n\ (x,r),$ where $CLS_n\ (x,r),$ denotes a circular left shift of n-bit binary number x by r bits to the left.

Property 2: $|-2^r x|_{2^n-1} = |2^r \bar{x}|_{2^n-1} = CLS_n\ (\bar{x},r)$ where $\bar{x}$ is the one's complement of x.

Applying properties 1 and 2 to equation (38) will yield the following:

$$g_1 = |(2^{4n+1}(x_1)|_{2^{4n+2}-1} = CLS_{4n+2}\ (x_1, 4n+1)$$

$$g_1 = \overbrace{(x_{1,2n+1}x_{1,2n}\dots x_{1,1}x_{1,0}\underbrace{000\dots0)}_{4n+1}}^{6n+3} \tag{39}$$

$$g_2 = |(2^{2n}(x_1)|_{2^{4n+2}-1} = CLS_{4n+2}\ (x_1, 2n)$$

$$g_2 = \overbrace{(x_{1,2n+1}x_{1,2n}\dots x_{1,1}x_{1,0}\underbrace{000\dots0)}_{2n}}^{4n+2} \tag{40}$$

$$g_3 = (-x_1) = \underbrace{(\bar{x}_{1,2n+1}\bar{x}_{1,2n}\dots \bar{x}_{1,1}\bar{x}_{1,0})}_{2n+2} \tag{41}$$

$$h_1 = |(2^{4n+2}(x_2)|_{2^{4n+2}-1} = CLS_{4n+2}\ (x_2, 4n+2)$$

$$h_1 = \overbrace{(x_{2,2n}x_{2,2n-1}\dots x_{2,1}x_{2,0}\underbrace{00\dots\dots00)}_{4n+2}}^{6n+3} \tag{42}$$

$$h_2 = |(-2^{2n+1}(x_2)|_{2^{4n+2}-1} = CLS_{4n+2}\ (x_2, 2n+1)$$

$$h_2 = \overbrace{(\bar{x}_{2,2n}\bar{x}_{2,2n-1}\dots \bar{x}_{2,1}\bar{x}_{2,0}\underbrace{11\dots\dots11)}_{2n+1}}^{4n+2} \tag{42}$$

$$h_3 = |(-x_2)|_{2^{4n+2}-1} = \underbrace{(\bar{x}_{2,2n}\bar{x}_{2,2n-1}\dots \bar{x}_{2,1}\bar{x}_{2,0})}_{2n+1} \tag{43}$$

$$i_1 = |(2^{4n+1}(x_3)|_{2^{4n+2}-1} = CLS_{4n+2}\ (x_3, 4n+1)$$

$$i_1 = \overbrace{(x_{3,2n}x_{3,2n-1}\dots x_{3,1}x_{3,0}\underbrace{00\dots\dots00}_{4n+1})}^{6n+2} \quad (44)$$

$$i_2 = |(2^{2n}(x_3)|_{2^{4n+2}-1} = CLS_{4n+2}\,(x_3, 2n)$$

$$i_2 = \overbrace{(x_{3,2n}x_{3,2n-1}\dots x_{3,1}x_{3,0}\underbrace{00\dots\dots00}_{2n})}^{4n+1} \quad (45)$$

$$G = g_1 + g_2 + g_3 \quad (46)$$

$$H = h_1 + h_2 + h_3 \quad (47)$$

$$I = i_1 + i_2 \quad (48)$$

Equation (38) can now be written as:

$$y_2 = ||G + H + I|_{2^{4n+2}-1}|_{2^{2n+1}} \quad (49)$$

The implementation of channel two will require a Carry Save Adder with End Around Carry (CSA EAC) to add G, H and I. This will generate a sum (s_i) and a carry bit (c_i). A Carry Propagate Adder (CPA) will be required to add the sum and the carry bit in modulo 2^{2n+1} to obtain the desired result y_2. The proposed architecture of the algorithm is shown in figure 1.

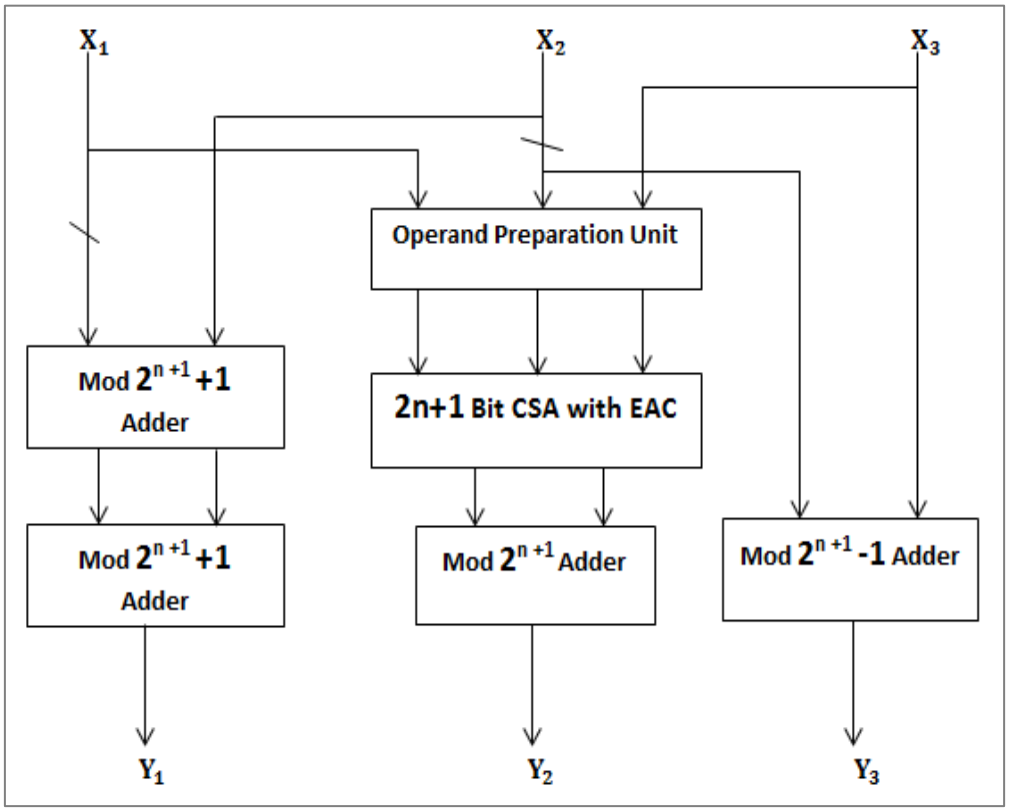

Figure 1: Hardware architecture of proposed RNS scaling algorithm

4.1 Performance Evaluation

In order that we evaluate the performance of the proposed scaling algorithm, it is compared with similar scaling scheme proposed by Chip- Hong Chang and Jeremy Yung Shern Low in 2011.

The Unit- Gate model which is used to analyze the delay and area consumption of the Chang Scaler is adapted to do the analysis. The model is adapted in order that we have the same terms for unbiased comparison. The proposed algorithm is **FA** based architecture. The Unit – Gate model asserts that, a two input monotonic gate such as AND or NAND gate is said to have one unit of area and one unit of delay. And a XOR gate consumes two units of area and two units of delay. An inverter is deemed to have a negligible fraction of a unit and therefore, has **zero** units of area and delay. Based on the model adapted, FA has **seven** units of area and **four** units of delay.

The area and delay for each residue channel can independently be evaluated by analyzing the area and time complexity of the logic gate implementation adapted for the hardware architecture. According to [30], the area and delay for the diminished – one mod 2^n+1 adder are $4.5n\lceil\log_2 n\rceil + 0.5n + 6$ and $2\lceil\log_2 n\rceil + 3$ units respectively. From Fig.1, we require an inverter to complement x_1 . The operands $\bar{x}_1$ and x_2 are first added using a Carry Propagate Adder with End Around Carry. A sum and a carry bit is generated and then added by another Carry Propagate Adder with a constant carry - in bit of 1 to yield a sum. Since an inverter has zero units of area and delay, the focus is on the adders. The area and delay of the first channel are: $2(4.5n\lceil\log_2 n\rceil+0.5n+6)$ and $2(2\lceil\log_2 n\rceil+3)$ respectively. The channel two shown in Fig 1 requires 2n+1 bit CSA with EAC and modulo 2^{2n+1} adder to implement. The CSA requires 2n+1 FAs to be implemented and a FA has seven units of area and four units of delay. The CSA will therefore require 7n units of area and 4 units of delay. The CPA will require $6n\lceil\log_2 2n+1\rceil + 24n$ units of

area and $2\lceil \log_2 2n \rceil + 3$ of delay. The total area for the second channel would be $6n\lceil \log_2 n \rceil$ + 31n units. The total delay for the same channel would be $2\lceil \log_2 n \rceil$+4 units of clock cycles. Channel three will be implemented using an inverter, for the one's complement of one operand and a Ling modulo 2^{2n+1} -1 adder. Based on the Ling modulo adder [31], estimated area and delay for channel three are $3n\lceil \log_2 2n - 1 \rceil$ + 12n and $2\lceil \log_2 n - 1 \rceil$ +3 respectively. We present the decomposition of area and delay of each channel in the tables 2 and 3 respectively.

Table 2: Estimation of Unit-Gate Model Area of Proposed (P) and State of the Art (C) Schemes(S)

S	Channel	A_{CPA}	A_{other}	A_{Total}
P	m_1	$2(4.5n\lceil \log_2 n \rceil + 0.5n + 6)$	0	$2(4.5n\lceil \log_2 n \rceil + 0.5n + 6)$
C	m_1	$3n\lceil \log_2 n - 1 \rceil + 12n$	0	$3n\lceil \log_2 n - 1 \rceil + 12n$
P	m_2	$6n\lceil \log_2 n \rceil + 24n$	7n	$6n\lceil \log_2 n \rceil + 31n$
C	m_2	$6n\lceil \log_2 2n \rceil + 24n$	15n	$6n\lceil \log_2 2n \rceil + 39n$
P	m_3	$3n\lceil \log_2 n - 1 \rceil + 12n$	0	$3n\lceil \log_2 n - 1 \rceil + 12n$
C	m_3	$4.5n\lceil \log_2 2n \rceil + 0.5n + 6$	7n+6	$4.5n\lceil \log_2 2n \rceil + 7.5n + 12$

Table 3: Estimation of Unit-Gate Model Delay of Proposed (P) and State of the Art(C) Schemes(S)

S	Channel	A_{CPA}	A_{other}	A_{Total}
P	m_1	$2(2\lceil \log_2 n \rceil + 3)$	0	$2(2\lceil \log_2 n \rceil + 3)$
C	m_1	$2\lceil \log_2 n - 1 \rceil + 3$	0	$2\lceil \log_2 n - 1 \rceil + 3$
P	m_2	$2\lceil \log_2 n \rceil + 3$	4	$2\lceil \log_2 n \rceil + 7$
C	m_2	$2\lceil \log_2 n \rceil + 3$	5	$2\lceil \log_2 n \rceil + 8$
P	m_3	$2\lceil \log_2 n - 1 \rceil + 3$	0	$2\lceil \log_2 n - 1 \rceil + 3$
C	m_3	$2\lceil \log_2 n \rceil + 3$	6	$2\lceil \log_2 n \rceil + 9$

5. CONCLUSION

In this paper, an efficient RNS scaling algorithm based on the new moduli set $\{2^{2n+1}+1, 2^{2n+1}, 2^{2n+1}-1\}$ is proposed. The full adder based implementation is used. The proposed algorithm has been evaluated based on dynamic range (DR), area and delay and compared with the state of the art scheme [4]. The proposed Algorithm outperforms the scheme in [4] in terms of DR, area and delay with the percentages as 98%, 18.4% and 21.7% respectively in favour of the proposed algorithm.

6. REFERENCES

[1] Neha, S. 2008. An Overview of Residue Number System. National Seminar on Devices, Circuits and Communication.

[2] Stouraitis, T. and Paliouras, V. 2001. Considering the Alternatives in low-power design," Circuits and Devices Magazine, IEEE, Vol. 17, No. 4, Pp. 22–29.

[3] Soudris, D., Dasygenis, M., Mitroglou, K., Tatas, K., and Thanailakis 2002. A Full adder Based Methodology for Scaling Operations in Residue Number System, Electronics, Circuits and Systems. 9th International Conference on Vol. 3, 891-894.

[4] Molahosseini, A. S., Navi, K., Dadkhah, C., Kavehei, O., and Timachi 2010. Efficient Reverse converter Designs for the New 4- Moduli Sets and based on new CRTs Circuits and Systems I: Regular Papers, IEEE Transactionson Vol. 57 No. 4. 823-835.

[5] Chang, C. H. and Low, J. 2011. Simple, Fast, and Exact RNS Scaler for the Three-Moduli Set, Circuits and Systems I: Regular Papers, IEEE Transactions on, Vol. 58, No. 11, 2686–2697.

[6] Safari, A. and Kong, Y. 2012. Simple, fast and synchronous hybrid scaling scheme for the 8-bit Moduli Set, Journal of Emerging Trends in Computing and Information Sciences, Vol. 3, No. 6, 949–956.

[7] Kong, Y. and Philip, B. 2009. Fast Scaling in the Residue Number System, IEEE Trans. Very Large Scale Integer (VLSI) Syst. Vol. 17 No. 3, (Mar. 2009) 443-447.

[8] Gbolagade, K. A. 2010. Efficient Reverse Conversion in Residue Number System Processors. PhD. Thesis Delft University of Technology the Netherlands.

[9] Szabo N. S. and Tanaka R. I. (1967), Residue Arithmetic and its Applications to Computer Technology. McGraw-Hill New York, 1967, Vol. 24.

[10] O'Keefe, K. H. and Wright, J. L. 1973. Remarks on Base Extension for Modular Arithmetic. Computers, IEEE Transactions on, Vol. 100, No. 9, 833–835.

[11] Jullien, G. A. 1978. Residue Number Scaling and Other Operations Using ROM Arrays, Computers, IEEE Transactions on, Vol. 100, No. 4, 325–336.

[12] Taylor F. J. and Huang C. H (1981), a Floating-Point Residue Arithmetic Unit. Journal of the Franklin Institute, Vol. 311, No. 1, Pp. 33–53, 1981.

[13] Taylor, F. J. and Huang, C. H. 1982. An Auto Scale Residue Multiplier. Computers, IEEE Transactions on, Vol. 100, No. 4, 321–325.

[14] Miller, D. D. and Polky, J. N. 1984. An Implementation of the LMS Algorithm in the residue number system. Circuits and Systems, IEEE Transactions, Vol. 31, No. 5, Pp. 452–461.

[15] Griffin, M. S. M. and Taylor, F. 1989. Efficient Scaling in the Residue Number System, in Int. Conf. Acoust. Speech, Signal Process, Glasgow, U.K. 1075–1078.

[16] Shenoy, M. A. P and Kumaresan, K. 1989. A Fast and Accurate RNS Scaling Technique for high Speed Signal

Processing. IEEE Trans. Acoust. Speech, Signal Process, Vol. 37, No. 6. (June 1989), 929 – 937.

[17] Ulman, Z. D. Czyzak, M. and Zurida, J. M. 1993. Effective RNS Scaling Algorithm with the Chinese Remainder Theorem Decomposition. in Proc. IEEE Pacific Rim Conf. Commun. Computers, Signal Process., Victoria, BC, (May 1993), 528-531.

[18] Barsi, F. and Pinotti, M. C. 1995. Fast base extension and precise Scaling in RNS for Look- up Table implimentations. IEEE Trans. Systems and Process, Vol. 43, No. 10, (October 1995), 2427- 2430.

[19] Garcia, A. and Lloris, A. 1999. A look- up Scheme for Scaling in RNS. IEEE Transactions Comput. Vol. 48, No.7, (July1999), 748-751.

[20] Meyer-Base, U. and Stouraitis, T. 2003. New Power -of - 2 RNS Scaling Scheme for Cell-based IC Design, IEEE Trans. Very Large Scale Integer, (VLSI) Syst., Vol. 11, No. 2, (April 2003), 280-283.

[21] Benardson, P. 1985. Fast Memoryless, Over 64 bits, Residue-to-Binary converter. Circuits and Systems IEEE Transactions on Vol. 32. No. 3, (Mar. 1985), 298-300.

[22] Mohan, P. V. A. 2007. RNS -To Binary Converter for a New Three-Moduli set. IEEE Transaction on Circuits and Systems-II, Vol. 54 No. 9, 775-779.

[23] Dasygenis, M., Mitroglou, K., Soudris, D., and Thanailakis, 2008. A Full Adder Based Methodology for the Design of Scaling Operations in Residue Number System. Circuits and Systems I: Regular Papers, IEEE Transactions on Vol. 55, No. 2, (Mar. 2008), 546-558.

[24] Bernocchi, G. L., Cardarili, G. C., Nannarelli, A., Re M. 2007. Low Power Adaptive Filter Based on RNS Components. Proc. IEEE International Symposium Circuits Systems, New Orleans, LA, 3211-3214.

[25] Bankas, E. K., and Gbolagade, K. A., 2013. An Effective New CRT Based Reverse Converter for a Novel Moduli Set $\{2^{2n+1}- 1, 2^{2n+1}, 2^{2n}- 1\}$. International Journal of VLSI Design and Communication Systems (December, 2013), 4(6):1-11.

[26] Omondi, A. and Premkumar, B. 2007. Residue Number System Theory and Implementation. Imperial College Press.

[27] Gbolagade, K. A. and Cotofana, S.D. 2009b. A Reverse Converter for the new 4 – Moduli set $\{2^n+3, 2^n+2, 2^n+1, 2^n\}$. Submitted to IEEE Newcastaisa Toulouse, France. (July, 2009).

[28] Daabo, M. I. and Gbolagade, K. A. 2012. Overflow Detection Scheme in RNS Multiplication before Forward Convertion. Journal of computing. 4(12):13-16.

[29] Bankas, E. K., and Gbolagade, K. A., 2015. New MRC Adder-Based Reverse Converter for the Moduli Set $\{2^n, 2^{2n+1} - 1, 2^{2n+2} - 1\}$. Oxford Journals, Science & Mathematics, Computer Journal Volume 58, Issue 7, 1566-1572.

[30] Dimitrakopoulos, G., Nikolos, D. G., Vergos, H. T., Nikolos, D., Efstathiou, C. 2005. New Architectures for modulo 2^n - 1 Adders. in Proc. IEEE Int. Conf. Electrn., Circuits, Syst., Gammarth, Tunisia. (Dec. 2005). 1- 4.

[31] Vergos, H. T., Efstathiou, C., and Nikolos, D. 2002. Diminished – One Modulo 2n + 1 Adder Design. IEEE Transaction Computers, Vol. 51, No. 12, (Dec. 2002), 1389-1399.