Original scientific paper
https://doi.org/10.33180/InfMIDEM2020.201

Informacije MIDEM
Journal of Microelectronics,
Electronic Components and Materials
Vol. 50, No. 2(2020), 71 – 86

# Hardware Implementation of Residue Multipliers based Signed RNS Processor for Cryptosystems

*Elango Sekar, Sampath Palaniswami*

*Department of Electronics and Communication Engineering, Bannari Amman Institute of Technology, Erode, Tamil Nadu, India*

**Abstract:** The Residue Number System (RNS) characterize large integer numbers into smaller residues using moduli sets to enhance the performance of digital cryptosystems. A parallel Signed Residue Multiplication (SRM) algorithm, VLSI parallel array architecture for balanced ($2^n$-1, $2^n$, $2^n$+1) and unbalanced ($2^k$-1, $2^k$, $2^k$+1) word-length moduli are proposed which in turn are capable of handling signed input numbers. Balanced $2^n$-1 SRM is used as a reference to design an unbalanced $2^k$-1 and $2^k$+1. The synthesized results show that the proposed $2^n$-1 SRM architecture achieves 17% of the area, 26% of speed, and 24% of Power Delay Product (PDP) improvement compared to the Modified Booth Encoded (MBE) architectures discussed in the review of the literature. The proposed $2^n$+1 SRM architecture achieves 23% of the area, 20% of speed, and 22% of PDP improvement compared to recent counterparts. There is a significant improvement in the results due to the fully parallel coarsely grained approach adopted for the design, which is hardly attempted for signed numbers using array architectures. Finally, the proposed SRM modules are used to design {$2^n$-1, $2^n$, $2^n$+1} special moduli set based RNS processor, and the real-time verification is performed on Zynq (XC7Z020CLG484-1) Field Programmable Gate Array (FPGA).

**Keywords:** signed modulo multiplication; Very Large Scale Integration (VLSI); FPGA; computer arithmetic; RNS

# Strojna implementacija množilnikov ostankov na osnovi predznačenega RNS procesorja za sisteme kriptiranja

**Izvleček:** Številski sistem ostankov velike celoštevilske cifre v manjše ostanke na osnovi setov modulov za povečanje učinkovitosti sistemov kriptiranja. Predlagan je algoritem vzporednega množenja predznačenih ostankov (SRM) v VLSI paralelni arhitekturi za uravnotežen ($2^n$-1, $2^n$, $2^n$+1) in neuravnotežen ($2^k$-1, $2^k$, $2^k$+1) modul dolžine besede. Uravnotežen SRM je uporabljen kot referenca za načrtovanje neuravnoteženega algoritma. Rezultati kažejo, da predlagana arhitektura zajema 17% prostora, 26% hitrosti in 22% izboljšanost PDP glede na trenutne arhitekture. Izboljšava je dosežena na osnovi paralelnega načrtovanja. Verifikacija v realnem času je izvedena na Zynq FPGA.

**Ključne besede:** predznačeno množenjem odulov; VLSI; FPGA; računalniška arithmetika; RNS

** Corresponding Author's e-mail: elangos@bitsathy.ac.in*

## 1 Introduction

In cloud computing and the Internet of Things (IoT), data security is one of the major concerns for service providers. Therefore a dedicated hardware cryptography support is needed for modern electronic devices [1],[2],[3][4][5]. In recent years, Elliptic Curve Cryptography (ECC) [6] has received scientific interest as it ensures more security through hard underlying mathematical problems. It leads to an increase in the length of the key, and as a result, performing faster arithmetic operations on larger integers have become the bottleneck problem. RNS based arithmetic operation [7,8] is a solution through which residue multiplication has become the heart of computation architecture. The natu-

ral defense offered by RNS against attacks is another reason for the selection of residue arithmetic as the prime candidate in cryptosystems [9,10].

Similarly to the above operation, modular exponentiation [11] is a time-critical operation that is widely used in cryptographic algorithms like RSA. The modular exponentiation operation is performed in the form of residue multiplication. Therefore, the employment of efficient high-speed residue multiplication is vital in public-key encryption and decryption.

Typical hardware implementation of the RNS based application is dependent on the chosen moduli set. The selection of RNS Moduli [12] and the width of the residue decide the efficiency and performance of the cryptosystems. A $\{2^n-1,2^n,2^n+1\}$ special moduli set representation is a pairwise relatively co-prime standard RNS. These moduli set has a unique advantage in which two or more numbers do not have the same representation. Special moduli set shows better representational efficiency [12] compared to that of other moduli set and also maintains a good balance between the different moduli in a given moduli-set. Based on the number of bits used to represent the input, moduli and residue output are classified into balanced and unbalanced word-length moduli multiplication [13] [14].

Modified Booth Encoded (MBE) modulo multiplication scheme is relatively faster and can handle both signed, and unsigned numbers, the researcher's attention turned towards it, and many modifications of the same are reported in recent years [15,16,17,18,19,20]. The residue multipliers based on diminished-1 input representation in array and bit pair recoding booth algorithm are seen in [16,17,21]. Based on the conducted survey, it is evident that there is no work based on a signed array modulo multiplication scheme reported in the literature. The reasons for the above could be based on the complexity in handling the Partial Product (PP) and poor speed performance. This is one of the reasons that have highly motivated us to attempt a proposal on an array-based high-speed area-efficient parallel SRM module for RNS. In this work, the high-speed performance is achieved by a new multiplication methodology incorporating parallelism in PP generation and addition process.

Six significant contributions for this work include (i) an SRM algorithm for $2^n-1$, $2^n+1$ and $2^n$ balanced word-length moduli (ii) an SRM for $2^k-1$, $2^k+1$ and $2^k$ unbalanced word – length moduli (iii) Mathematical modeling of SRM algorithm for each moduli (iv) VLSI characterization of proposed SRM algorithm in terms of high-speed area-efficient Carry Save Adder (CSA) architecture and very high-speed Han Carlson parallel prefix-based SRM array architecture (v) Functional verification of the proposed modules in FPGA and synthesis in ASIC (vi) Design of RNS Processor to demonstrate the effectiveness of the proposed algorithm.

The paper is structured as follows: In Section 2, the related works connected to residue multipliers with various moduli sets performance are analyzed. In Section 3, characteristic equation, algorithm, and VLSI architecture are presented for both balanced ($2^n-1$, $2^n$, $2^n+1$) and unbalanced ($2^k-1$, $2^k$, $2^k+1$) word-length moduli. The design of the RNS processor is given in section 4. In section 5, Synthesis results, performance analysis, and RNS processor implementation are presented. The conclusion for the proposed work is drawn in section 6.

## 2 Review of Existing Work

An MBE based $2^n-1$ multiplication module to reduce the number of PPs is presented in [22]. The results show a significant improvement in area and delay. However, they fail to address power consumption. A radix-8 booth encoded RNS $2^n-1$ multiplier [14] using unbalanced word length of moduli supporting sizeable dynamic range with adaptable delay to achieve less area and power consumption is presented. The same authors have designed a radix-8 $2^n-1$ & $2^n+1$ multiplier with a balanced word length of moduli in [18] using various modulo properties. The author claims that less area and power are achieved by using CSA in [14] and parallel prefix adders in [18] for efficient addition operations with a slight increase in delay for lower bit width. Improved booth selector and encoder architecture consist of MUX, and the EXOR gate for the $2^n-1$ MBE multiplication algorithm is presented in [23]. The architecture improves the speed performance and efficiency, but the introduction of MUX in selector architecture leads to a slight increase in area requirement, and also power consumption is not discussed.

A compact ordinary array structure [15] based $2^n+1$ multiplication scheme by grouping the PPs and modify the correction bit are presented. The PP is reduced by the CSA tree, and the final carry propagation addition is carried out by prefix structure in order to achieve better area and delay performance in which the power consumption is not discussed. By introducing a new PP formation scheme, a binary-weighted representation based modulo $2^n+1$ multiplier is presented in [19] and is extended to implement a multiply-add unit. The authors have achieved less area and power consumption with similar delay performance compared to [15]. A radix-4 MBE architecture with a diminished-1 input representation and dadda tree reduction scheme, which

can handle zero operands with better speed and area, is discussed in [16]. A compressor structure is introduced in [24] for PP reduction. This work achieves less power, delay, and consumes less area compared to [15].

A hybrid input representation approach with a radix-4 booth encoding scheme utilizing one binary-weighted operand and diminished-1 input representation for the other operand is explained in [17]. The architecture supports both odd and even value of n. The authors have achieved a compact area with an enhanced speed compared to the existing multipliers. The radix-8 booth encoded $2^n$+1 multiplier for balanced word length moduli is designed in [18] using hard multiple generators, bias, and adders. The authors claim that the area and power reduction is accomplished compared to radix-4 and array type multiplier. However, there is an increase in operation time. In [20], the authors have improved the hard multiple generator method with a minimum number of bias terms compared to [18]. Two novel methods to increase the performance and to improve the efficiency of the radix-8 modulo $2^n$+1 multiplier are explained in [20]. The first method significantly reduces the amount of bias, and the second one is new hard multiple generators based on a parallel-prefix structure computes carry only for odd positions. These schemes result in a lightweight parallel-prefix adder for the computation of triple the number with significant area-saving and improved fan-out. It achieves less area and power compared to the radix-8 booth multiplier [18]. There is an increase in HMG delay compared to [18] and almost maintains the same delay performance for multiplier operation compared to [18].

The problem in MBE based architecture is that it requires an efficient booth selector and encoder compared to the array-based architectures. The former scheme reduces the number of PPs and improves speed performance. However, it invites additional hardware costs during implementation. Our proposed work is an entirely different approach compared to [18], [20] designed to address the above issues. In the proposed approach, split array type architecture is considered for implementing the $2^n$+1 operation, which occupies less area compared to the MBE scheme. Array architecture is a non-encoded architecture compared to the booth, so it does not require hard multiples for processing the PPs. The problem of an increased number of PPs in an array is addressed in the proposed scheme by splitting array structure into four segments, and full parallelism is maintained in PP additions also. The parallelism in the architecture ensures improved speed by maintaining the area advantage of the general array structure. The handling of signed numbers in array architecture is another reason for which the array scheme is less explored for data processing applications. The representation of signed numbers is addressed in the proposed architecture using appropriate constants.

# 3 Proposed Work

## 3.1 Proposed balanced word-length SRM

In balanced word-length modulo multiplication, the number of bits required representing the input, moduli, and output bits are summarized in Table 1. The type mentioned above of multiplication called balanced residue multiplication as it maintains a balanced bit-width between input, output, and moduli representation, as given in Table 1. In literature, the design problem of $2^n$-1 and $2^n$+1 residue multiplication is achieved through MBE schemes, whereas the possibilities of addressing this problem using array architecture are hardly considered, especially for signed numbers. The hierarchical approach for signed array multiplication presented in [25]. The motivation behind this work is the regularity in VLSI implementation and the reduced area budget offered by the array architectures compared to MBE architecture. The delay problem usually found in array architecture compared to the MBE scheme is addressed here using hierarchy based processing of the input bits and parallel addition structure. For comparative analysis, the adder structure is realized using CSA and Han Carlson parallel prefix [26] based schemes. The mathematical background, algorithm, and architecture of proposed residue $2^n$-1, $2^n$+1, and $2^n$ multiplications are presented in the following subsections.

**Table 1:** Balanced word-length moduli representation

| Moduli | $2^n$-1 | $2^n$ | $2^n$+1 |
|---|---|---|---|
| Number of input bits A & B | n | | |
| Moduli representation bits | n | n | n+1 |
| Number of output bits - P | n | n | n+1 |

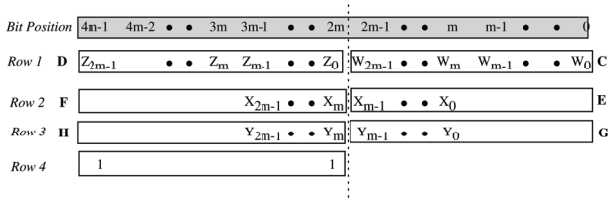


**Figure 1:** Intermediate PPs arrangement (nxn) [25]

### 3.1.1 Proposed $2^n$-1 SRM

The $2^n$-1 modulo multiplication module is one of the essential operations in the RNS independent arithmetic channel. The mathematical background, algorithm,

and the proposed architectures for the signed $2^n$-1 residue multiplier are given below.

*Mathematical modeling*

Consider the 2's complement signed number representation of two binary numbers A and B as given in Eq. (1) & (2)

$$A = -a_{n-1}2^{n-1} + \sum_{i=0}^{n=-2} a_i 2^i \quad (1)$$

$$B = -b_{n-1}2^{n-1} + \sum_{i=0}^{n=-2} b_i 2^i \quad (2)$$

The $2^n$-1 residue product representation is given in Eq. (3)

$$P = |A \times B|_{2^n-1} = \left| \left( -a_{n-1}2^{n-1} + \sum_{i=0}^{n-2} a_i 2^i \right) \times \left( -b_{n-1}2^{n-1} + \sum_{i=0}^{n-2} b_i 2^i \right) \right|_{2^n-1} \quad (3)$$

**Step 1. Partitioning of Input bits and Generation of intermediate PPs W, X, Y, Z using hierarchical partitioning multiplier [25]**

**Step 2. PP arrangement:**

The generated PPs are arranged [25], and a constant is added, as shown in Fig. 1 where m=n/2.

**Step 3. Rearrangement of Intermediate PPs:**

Fig. 2 shows the rearrangement of PPs, and the addition process flow carried out for the $2^n$-1 residue multiplication, and the corresponding mathematical operations are given in Eq. (4) – (6). The notations and operators used in this mathematical modeling are summarized in Table 2 and Table 3 respectively.

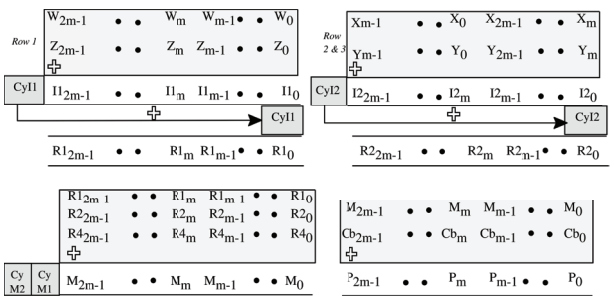


**Figure 2:** PP Rearrangement and addition process $2^n$-1 (nxn)

**Table 2:** Notations used in mathematical modeling

| Notations | Description |
|---|---|
| $A_H$, $A_L$, $B_H$, and $B_L$ | Higher and Lower bits of A & B inputs. |
| $C_b$ | Compensation Bits |
| $M_{i+1} \| M_i$ | Overflow bits of $M_{i-1}$ addition process |
| $Cy_{I1}$ | One bit Carry of $I_1$ addition process that has to be IEAC (Inverted End around Carry) |
| $Cy_{I2}$ | One bit Carry of $I_2$ addition process that has to be IEAC |
| $Cy_{I3}$ | One bit Carry of $I_3$ addition process that has to be IEAC |
| $R_{1c}$ | Carry Bit of $R_1$ (or) Overflow bit of $R_1$ |
| $R_{2c}$ | Carry Bit of $R_2$ (or) Overflow bit of $R_2$ |
| $R_{3c}$ | Carry Bit of $R_3$ (or) Overflow bit of $R_3$ |
| $Cy_{Mi}$ | n/2 Overflow carry bits of Mi addition process. If No overflow occurred n/2 bit zeros is considered |
| $Cy_{Qi}$ | One bit Carry of Qi addition process that has to be IEAC |

**Table 3:** Operators used in mathematical modeling

| Decimal Format: a=12; b=8; n=4<br>Binary Format: a=1100; b=1000; n=0100 | | | |
|---|---|---|---|
| **Operator** | **Description/ Functionality** | **Example** | **Result** |
| $\cdot$ | AND | $(a_i \cdot b_i)$ | $(1000)_2$ |
| $\|$ | OR | $(a_i \| b_i)$ | $(1100)_2$ |
| $\overline{a}$ | NOT | $\overline{a}$ | $(0011)_2$ |
| $\overline{\cdot}$ | NAND | $\overline{(a_i \cdot b_i)}$ | $(0111)_2$ |
| $\oplus$ | EXOR | $(a_i \oplus b_i)$ | $(0111)_2$ |
| $\overline{\oplus}$ | EXNOR | $\overline{(a_i \oplus b_i)}$ | $(1000)_2$ |
| + | Addition | $(a_i + b_i)$ | $(20)_{10}$ |
| $\|\|$ | Modulus | $\|a \times b\|_{2^n-1}$ | $(6)_{10}$ |
| $\Sigma$ | Summation | $\sum_{i=0}^{3} a_i$ | $(12)_{10}$ |
| $\Sigma\,\Sigma$ | Double Summation | $\sum_{j=0}^{3}\sum_{i=0}^{3} (a_i \cdot b_i) 2^{i+j}$ | $(96)_{10}$ |
| - | Subtraction | $(a - b)$ | $(4)_{10}$ |

| a b | Multiplication | a b | $(96)_{10}$ |
|---|---|---|---|
| X | Multiplication | $(a \times b)$ | $(96)_{10}$ |
| / | Division | $\left(n/2+1\right)$ | $(3)_{10}$ |
| $\left\|(\times)^{-1}\right\|_{2^n+1}$ | Multiplicative Inverse | $\left\|(a\times b)^{-1}\right\|_{2^n+1}$ | $(14)_{10}$ |
| $\|\|$ | Concatenation | $(a \,\|\|\, b)$ | $(11001000)_2$ |

$$M_{i-1}=\sum_{i=1}^{n}\left(W_{i-1}+Z_{1i-1}\right)2^{i-1}+\sum_{i=\frac{n}{2}+1}^{n}\left(X_{1i-1}+Y_{1i-1}\right)2^{i-1}$$
$$+\sum_{i=1}^{\frac{n}{2}}\left(X_{1i-1}+Y_{1i-1}\right)2^{i-1}+2^{0}+2^{n-1} \quad (4)$$

The final product is

$$P_{i-1}=\left|A\times B\right|_{2^n-1}=\sum_{i=1}^{n}\left(M_{i-1}+C_{bi-1}\right)2^{i-1} \quad (5)$$

The compensation bits are expressed as

$$C_{bi-1}=\left(\sum_{i=1}^{n}(Sub)2^{i-1}\right)+\left(\sum_{i=1}^{2}\left(Add_{i-1}\right)2^{i-1}\right) \quad (6)$$

Where

$$Ad_0=M_{i+1}\bullet\overline{M_i}\;;\; Ad_1=M_{i+1}\bullet M_i;\; Sub=\overline{M_{i+1}}\bullet\overline{M_i}$$

*Algorithm*

The proposed $2^n$-1 SRM algorithm is given below

1. Input: A & B (A, B → n-bit signed numbers), where *n = 4,8,16,32,etc..*
2. Output $P \leftarrow \left|A\times B\right|_{2^n-1}$, where *P ← n bit*
3. Intermediate PPs Generation →*W,X,Y,Z*
4. Rearrange the Intermediate PPs into 4 rows as in Fig. 1.
5. Split the arrangement in Fig. 1 into two equal halves
   LSP (Least Significant Plane) ← *Bit_Pos(0 to (n-1))*
   MSP (Most Significant Plane) ← *Bit_Pos(n to (2n-1))*
6. Fold the MSP towards LSP side as given in Fig. 2.
7. M ← *Sum (LSP, Folded MSP, EAC)*
8. P ← *Sum (M,$C_b$)*

*Architecture*

The architecture of proposed $2^n$-1 residue multiplication is shown in Fig. 3. The architecture consists of three stages, namely the partitioning stage, intermediate PPs generation stage, and adder stage. The four parallel modules in the intermediate PP generation stage M-I, M-II, M-III, M-IV indicates the hardware required for computing W, X, Y, Z given in [25]. The four independent parallel addition process observed in the architecture is the main reason for achieving high performance in the proposed array architecture. The compensation bits are gets added in the final stage to obtain modulo results. CSA and Han-Carlson parallel prefix adder structure is incorporated in Fig. 3 in order to analyze the performance. The results of the proposed work are further discussed in Section 5.

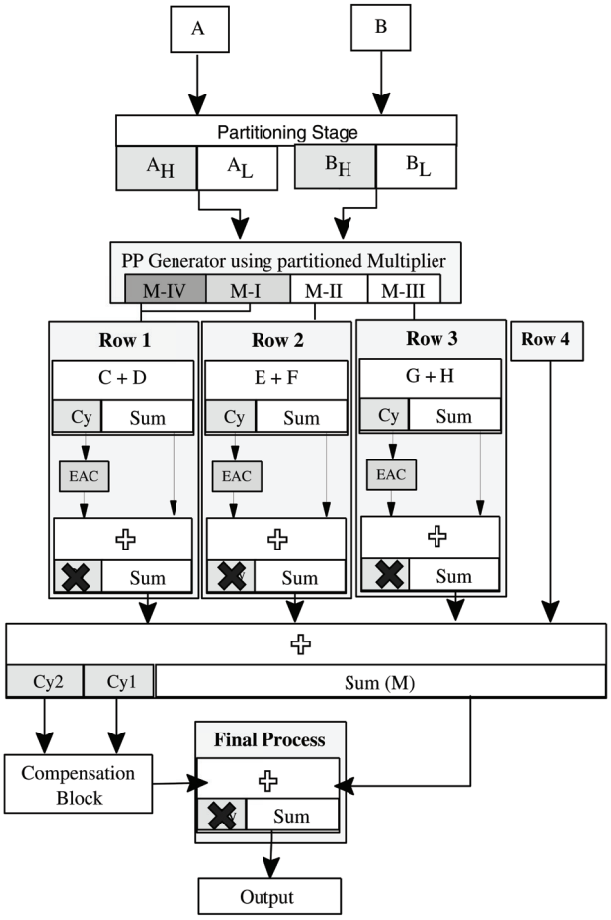


**Figure 3:** Architecture of $2^n$-1 SRM

*3.1.2. Proposed $2^n$+1 SRM*

The $2^n$+1 residue multiplication problem is considered as a demanding operation in RNS Processor due to the increase in moduli output range compared to $2^n$ and $2^n$-1 multiplications, as represented in Table 1. In the proposed scheme, the increased moduli output range is regulated using the diminished-1 approach for both multiplier and multiplicand. The primary advantage of using the proposed scheme is that this architecture can handle exceptional cases like 'all-zeros' input and

'all-ones' input, which consecutively produce the correct results. This architecture handles the bit positions higher than n-1 by complementing and mapping them to the LSBs. The mathematical background, algorithm, and the proposed architectures for signed $2^n+1$ residue multipliers are given in the below subsections.

*Mathematical modeling*

The diminished-1 representation of binary inputs A and B are modified as A' & B', which is given in Eq. (7) – (8)

$$A' = \left(-a_{n-1}2^{n-1} + \sum_{i=0}^{n-2} a_i 2^i\right) - 1 \tag{7}$$

$$B' = \left(-b_{n-1}2^{n-1} + \sum_{i=0}^{n-2} b_i 2^i\right) - 1 \tag{8}$$

The residue product P is given by the following Eq.(9)

$$P = \left|A \times B\right|_{2^n+1} = \left|\left(A' \times B'\right) + A' + B' + C_b\right|_{2^n+1} \tag{9}$$

The methodology and arrangements of PP are the same as step 1 and step 2 of signed $2^n$-1, but the inputs are A' and B'. The final product is obtained by rearranging the PPs of Fig. 1 in such a way to obtain the result of $2^n$+1 residue multiplication. Fig. 4 shows the rearrangement of PPs, the position of PPs, and the addition process flow carried out for the $2^n$+1 multiplication, and the same is represented in Eq. (10) – (20). The mathematical operations performed between Row 1 to Row 4 are given below

**Row 1:**

$$I_{1(i-1)} = \left(\sum_{i=1}^{n}\left(W_{i-1} + \overline{Z_{1i-1}}\right)2^{i-1} + 1\right) \tag{10}$$

$$R_{1(i-1)} = \sum_{i=1}^{n}\left(I_{1(i-1)}\right)2^{i-1} + \left(\overline{Cy_{I1}}\right)2^0 \tag{11}$$

**Row 2:**

$$I_{2(i-1)} = \sum_{i=\frac{n}{2}+1}^{n}\left(X_{1\left(i-(m+1)\right)}\right)2^{i-1} + \left(\sum_{i=1}^{\frac{n}{2}}\left(\overline{X_{1(m+i-1)}}\right)2^{i-1}\right) + \sum_{i=\frac{n}{2}+1}^{n}(1)2^{i-1} + \left(\sum_{i=1}^{\frac{n}{2}}(0)2^{i-1}\right) + 1 \tag{12}$$

$$R_{2(i-1)} = \sum_{i=1}^{n}\left(I_{2(i-1)}\right)2^{i-1} + \left(\overline{Cy_{I2}}\right)2^0 \tag{13}$$

**Row 3:**

$$I_{3(i-1)} = \sum_{i=\frac{n}{2}+1}^{n}\left(Y_{1\left(i-(m+1)\right)}\right)2^{i-1} + \left(\sum_{i=1}^{\frac{n}{2}}\left(\overline{Y_{1(m+i-1)}}\right)2^{i-1}\right) + \sum_{i=\frac{n}{2}+1}^{n}(1)2^{i-1} + \left(\sum_{i=1}^{\frac{n}{2}}(0)2^{i-1}\right) + 1 \tag{14}$$

$$R_{3(i-1)} = \sum_{i=1}^{n}\left(I_{3(i-1)}\right)2^{i-1} + \left(\overline{Cy_{I3}}\right)2^0 \tag{15}$$

**Row 4:**

$$R_{4(i-1)} = (0)2^{n-1} + \sum_{i=1}^{n-1}(1)2^{i-1} \tag{16}$$

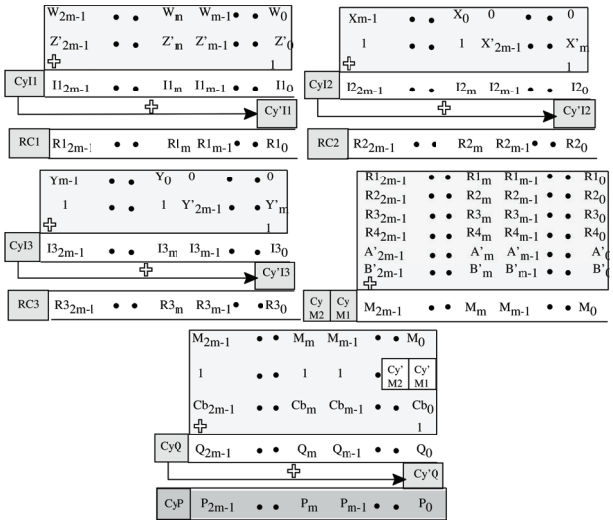


**Figure 4:** PP Rearrangement and addition process $2^n$+1 (nxn)

Finally, all four rows get added as per the following equations.

$$M_{(i-1)} = \sum_{i=1}^{n}\left(R_{1(i-1)} + R_{2(i-1)} + R_{3(i-1)} + R_{4(i-1)} + A_{i-1} + B_{i-1}\right)2^{i-1} \tag{17}$$

$$Q_{i-1} = \left(\sum_{i=1}^{n}\left(M_{i-1} + C_{bi-1}\right)2^{i-1}\right) + \sum_{i=1}^{2}\left(\overline{Cy_{Mi-1}}\right) + \sum_{i=1}^{2}\left(\overline{Cy_{Mi-1}}\right)2^{i-1} + \left(\sum_{i=3}^{n}(1)2^{i-1}\right) + 1 \tag{18}$$

Where $C_b$ is given in Eq. (19)

$$C_{bi-1} = \left(\sum_{i=3}^{n}(1)2^{i-1}\right) + \left(\sum_{i=1}^{2}\left(Ad_{i-1} + \overline{Sub_{i-1}}\right)2^{i-1}\right) + 1$$

$$Ad_0 = A'[n-1] \oplus B'[n-1]$$

$$Ad_1 = A'[n-1] \mid B'[n-1]$$

$$Sub_0 = \left(R_{1c} \oplus R_{2c} \oplus R_{3c}\right)$$

$$Sub_1 = \left(R_{1c} \cdot R_{2c}\right) \mid \left(R_{2c} \cdot R_{3c}\right) \mid \left(R_{3C} \cdot R_{1c}\right) \tag{19}$$

The $2^n+1$ multiplication is given in Eq. (20)

$$P_{[n:0]}=\left|A\times B\right|_{2^n+1}=\sum_{i=1}^{n}\left(Q_{(i-1)}\right)2^{i-1}+\overline{Cy_{Qi}} \quad (20)$$

*Algorithm*
The proposed $2^n+1$ SRM algorithm is given below

1. Input: A & B (A, B→ n-bit signed numbers), where *n=4,8,16,32,etc..*
2. Output $P \leftarrow \left|A\times B\right|_{2^n+1}$, where *P ← n+1 bit*
3. A′ ← *Diminished-1 (A);* B′ ← *Diminished-1 (B);*
3. Intermediate PPs Generation →*W,X,Y,Z*
4. Rearrange the Intermediate PPs into 4 rows as shown in Fig. 1.
5. Split the arrangement in Fig. 1 into
   LSP ← *Bit_Pos(0 to (n-1))*
   MSP ← *Bit_Pos (n to (2n-1))*
6. Fold the MSP towards LSP as given in Fig. 4.
7. $R_1$←*Sum (LSP, 2's Comp. (MSP), IEAC);*
8. $R_2$←*Sum (LSP, 2's Comp. (MSP), IEAC);*
9. $R_3$←*Sum (LSP, 2's Comp. (MSP), IEAC);*
10. $R_4$←*Sum (LSP, 2's Comp. (MSP), IEAC);*
11. M ← *Sum ((Rx,) A′, B′)*, where *x =1,2,3,4*
12. P ← *Sum (M, 2's Complement ($Cy_M$),$C_b$, IEAC)*

*Architecture*
The overall architecture arrangement of $2^n+1$ is similar to that of $2^n-1$ except for the fact that it has some additional modules to perform 2's complement operation and Inverted End Around Carry (IEAC), as shown in Fig. 5. However, the compensation generation scheme is complicated compared to $2^n-1$ architecture.

### *3.1.3 Signed $2^n$ residue multiplier*

*Mathematical modeling*
The operation required to obtain Module I (W) follows the same pattern as in $2^n-1$. The X & Y are given in Eq. (21) and (22). Z is not required for computing $2^n$ because it has a higher weight position compared to $2^n$ value.

$$X=\left(\overline{a_{n-1}\cdot b_0}\right)2^{m-1}+\sum_{i=m}^{n-2}\left(a_i\cdot b_0\right)2^{i-m}+\sum_{j=1}^{m-1}\sum_{i=m}^{n-1-j}\left(a_i\cdot b_j\right)2^{i+j-m} \quad (21)$$

$$Y=\left(\overline{b_{n-1}\cdot a_0}\right)2^{m-1}+\sum_{i=m}^{n-2}\left(b_i\cdot a_0\right)2^{i-m}+\sum_{j=1}^{m-1}\sum_{i=m}^{n-1-j}\left(b_i\cdot a_j\right)2^{i+j-m} \quad (22)$$

The final $2^n$ product is given in Eq. (23)

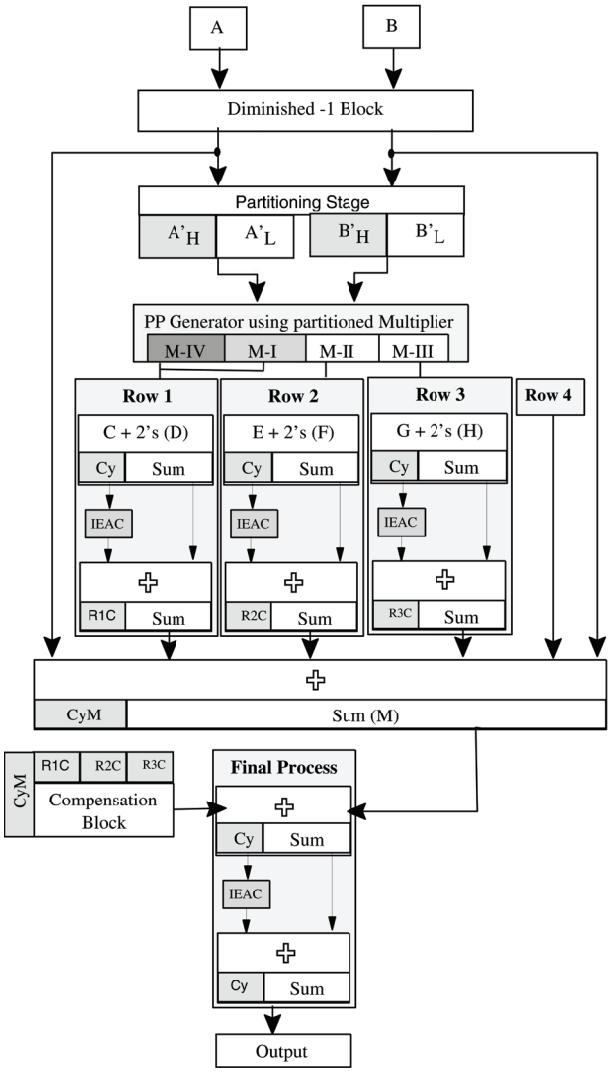


**Figure 5:** Architecture of $2^n+1$ SRM

$$P=\left|A\times B\right|_{2^n}=\sum_{i=0}^{2m-1}W_i 2^i+\sum_{k=m}^{2m-1}\left(X_{k-m}+Y_{k-m}\right)2^k \quad (23)$$

## *3.2 Proposed unbalanced word-length SRM*

The unbalanced word-length moduli multiplier typically used in applications different bit-width proportion between input, moduli, and output is required. In unbalanced word-length residue multiplication, the number of bits required to represent the input, moduli, and output bit-width, which are summarized in Table 4. The strategy followed to design $2^k-1$module is derived from the $2^n-1$ balanced module. However, the $2^k+1$ is not derived from the $2^n+1$ balanced module because it may lead to comparatively complex architecture with more delay penalty. Instead, $2^n-1$ balanced design is converted to an unbalanced $2^k+1$ by modifying the final result of $2^n-1$ multiplication.

**Table 4:** Unbalanced word-length moduli representation

| Moduli | $2^k$-1 | $2^k$ | $2^k$+1 |
|---|---|---|---|
| Number of input bits A & B | n | | |
| Moduli representation bits | k | k | k+1 |
| Number of output bits -P | k | k | k+1 |

*3.2.1. Proposed $2^k$-1 and $2^k$+1 SRM*

*Mathematical modeling*

Let us consider the n bit output of balanced $2^n$-1 multiplication given in Eq. (5). It is split into two halves $P_L$ and $P_H$, as shown in Fig. 6 to obtain the result k=n/2 & k=n/4 bits, and the corresponding equations are given in (24) – (25).

For k=n/2

$$\left.\begin{aligned} P_L &= P\left[\frac{n}{2}\text{-}1:0\right] \\ P_H &= P\left[n\text{-}1:\frac{n}{2}\right] \end{aligned}\right\} 2^n\text{-}1(\text{Output})$$

$$\left.\begin{aligned} UP_{i-1} &= \sum_{i=1}^{\frac{n}{2}}\left(P_{i-1}+P_{\frac{n}{2}+i-1}\right)2^{i-1} \\ P2_{i-1} &= \sum_{i=1}^{\frac{n}{2}}\left(UP_{i-1}\right)2^{i-1}+C_{out}2^0 \end{aligned}\right\} 2^k\text{-}1 \qquad (24)$$

$$\left.\begin{aligned} UP_{i-1} &= \sum_{i=1}^{\frac{n}{2}}\left(P_{i-1}+\overline{P_{\frac{n}{2}+i-1}}\right)2^{i-1}+2^0 \\ P2_{[k:0]} &= \sum_{i=1}^{\frac{n}{2}}\left(UP_{i-1}\right)2^{i-1}+\overline{C_{out}2^0} \end{aligned}\right\} 2^k+1$$

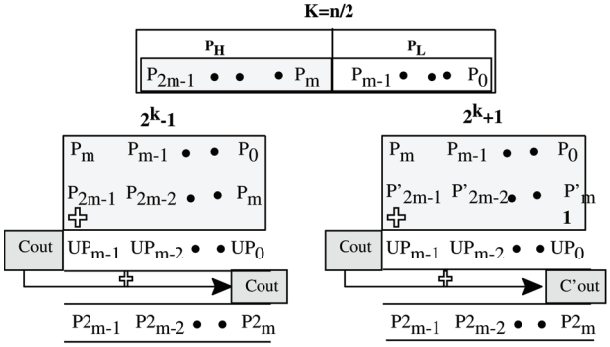


**Figure 6:** Unbalanced PP Rearrangements and addition process

**For k=n/4**

k=n/4 design is derived from k=n/2. The output of k=n/2 acts as an input for the k=n/4 design.

$$\left.\begin{aligned} P2_L &= P2\left[\frac{n}{2}\text{-}1:0\right] \\ P2_H &= P2\left[n\text{-}1:\frac{n}{2}\right] \end{aligned}\right\} 2^n\text{-}1(\text{output})$$

$$\left.\begin{aligned} NP_{i-1} &= \sum_{i=1}^{\frac{n}{2}}\left(P2_{i-1}+P2_{\frac{n}{2}+i-1}\right)2^{i-1} \\ P4_{i-1} &= \sum_{i=1}^{\frac{n}{2}}\left(NP_{i-1}\right)2^{i-1}+C_{out}2^0 \end{aligned}\right\} 2^k\text{-}1 \qquad (25)$$

$$\left.\begin{aligned} UP_{i-1} &= \sum_{i=1}^{\frac{n}{2}}\left(P2_{i-1}+\overline{P2_{\frac{n}{2}+i-1}}\right)2^{i-1}+2^0 \\ P4_{[k:0]} &= \sum_{i=1}^{\frac{n}{2}}\left(UP_{i-1}\right)2^{i-1}+\overline{C_{out}2^0} \end{aligned}\right\} 2^k+1$$

*Algorithm*

The proposed SRM algorithm for the unbalanced $2^k$-1 and $2^k$+1 is given below

1. Input: A & B (A, B → n bit signed numbers), where *n=4,8,16,32,etc..*
2. Output $P \leftarrow |A\times B|$, where *P ← k bit for- $2^k$-1 and k+1 bit for $2^k$+1*
3. Consider Eq.(5) - $P \leftarrow |A\times B|_{2^n-1}$,
4. Split the Eq. (5) into two equal halves
   $P_H$ ← *Bit_Pos(0 to (n/2)-1)*
   $P_L$ ← *Bit_Pos((n/2) to n-1)*
5. Fold the $P_H$ towards $P_L$ side as mentioned in Fig. 6.

**If ($2^k$-1) Operation**

6. $P_2$ = Sum($P_L$, $P_H$, EAC ) → k=n/2
7. $P_4$ = Sum($P_{2L}$, $P_{2H}$, EAC ) → k=n/4
8. $P_8$ = Sum($P_{4L}$, $P_{4H}$, EAC ) → k=n/8

**If ($2^k$+1) Operation**

6. $P_2$ = Sum($P_L$, 2's ($P_H$), EAC ) → k=n/2
7. $P_4$ = Sum($P_{2L}$, 2's ($P_{2H}$), EAC ) → k=n/4
8. $P_8$ = Sum($P_{4L}$, 2's ( $P_{4H}$), EAC ) → k=n/8

*Architecture*

The unbalanced SRM architecture for $2^k$-1 and $2^k$+1 is depicted in Fig. 7. The architecture is derived from proposed $2^n$-1 SRM.

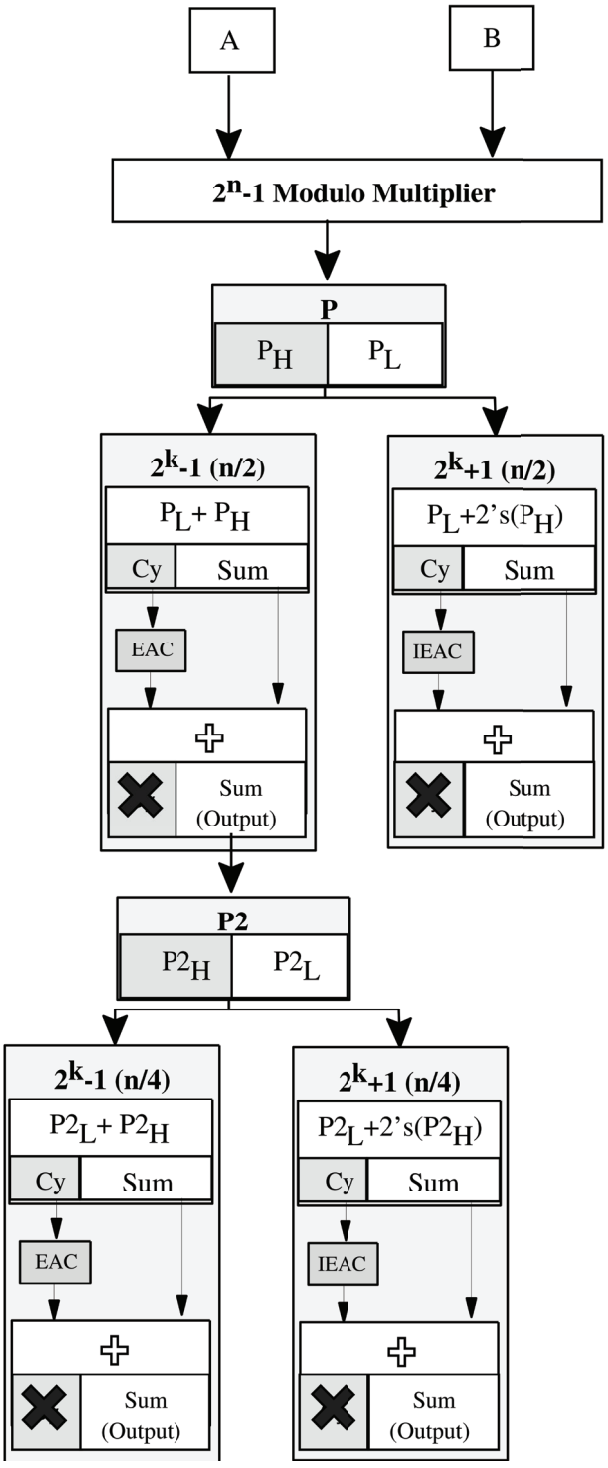


**Figure 7:** Architecture of $2^k$-1 & $2^k$+1 SRM

*3.2.2. $2^k$ SRM*

The residue multiplication $P = |A \times B|_{2^k}$ is derived from a $2^n$ balanced residue multiplier equation. The characteristic equations of $2^k$ unbalanced residue multiplication are given in Eq. (26)

$$P=|A\times B|_{2^k}=\begin{cases} RP_{i-1}=\sum_{i=1}^{\frac{n}{2}}(P_{i-1})2^{i-1} \rightarrow k=\dfrac{n}{2} \\ P4_{i-1}=\sum_{i=1}^{\frac{n}{4}}(P_{i-1})2^{i-1} \rightarrow k=\dfrac{n}{4} \\ P8_{i-1}=\sum_{i=1}^{\frac{n}{8}}(P_{i-1})2^{i-1} \rightarrow k=\dfrac{n}{8} \end{cases} \quad (26)$$

## 4 RNS processor

### 4.1 Architecture

In general, the cryptographic algorithm requires many rounds of arithmetic operations in order to create the ciphertext. Instead of doing such lengthy arithmetic operations in binary representation, residue values can be used to save the area and time budget. The proposed balanced and unbalanced word-length residue multipliers are used for implementing special moduli set based RNS computing platforms, as given in Fig. 8. The RNS processing system consists of three blocks, namely Forward Converter (FC), Independent Modulo Arithmetic Processing Unit (IMAPU), and Reverse Converter (RC) [13], [27]. The proposed SRM architectures are used to design arithmetic channels and RC. The FC and RC blocks convert the binary number to residue number and vice versa. The IMAPU block consists of application-based arithmetic operations or any other desired operations in modulo representation. The RC operation can be performed using the Chinese Remainder Theorem (CRT) [28] or Mixed Radix Conversion (MRC) [29]. In this paper, the MRC technique [13,27] is considered for the conversion in the RC block. The characteristic equations of MRC given in Eq. (27) – (29) shows that the operation can be done by modulo subtractions, multiplicative inverses, and residue multiplication. Here the multiplicative inverse is computed using the Extended Euclidean algorithm (EECD) [30]. From [13,27] the decoded number is expressed in the following form for MRC technique

$$X=Z_N m_{N-1} m_{N-2} \cdots\cdots m_1 + \cdots\cdots + Z_3 m_2 m_1 + Z_2 m_1 + Z_1 \quad (27)$$

where $0<Z_i<m_i$

The mixed-radix digits are derived as,

$$\begin{aligned} Z_1 &= x_1 \\ Z_2 &= \left| |m_1^{-1}|_{m_2} \times (x_2 - Z_1) \right|_{m_2} \\ Z_3 &= \left| \left| (m_2 m_1)^{-1} \right|_{m_3} \times (x_3 - Z_2 m_1 + Z_1) \right|_{m_3} \end{aligned} \quad (28)$$

The finalized equation is derived for the value of N bit as,

$$Z_N = \left|\left(m_1 m_2 m_3 \cdots\cdots m_{N-1}\right)^{-1}\right|_{m_N} \times \left|\left(x_N - \left(x_{N-1} m_{N-2} \cdots\cdots Z_2 m_1 + z_1\right)\right)\right|_{m_N} \quad (29)$$

Where $m_1$,$m_2$,$m_3$ are moduli sets, and $x_1$, $x_2$, $x_3$ are residue output of IMAPU.

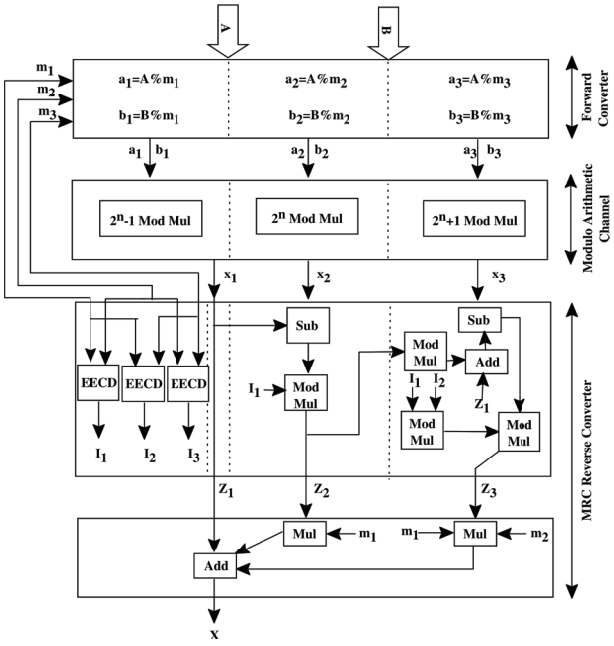


**Figure 8:** Hardware realization of signed RNS processor

The effectiveness of the proposed multiplier is tested by designing decoupled modulo arithmetic channels and memoryless MRC reverse converter, as shown in Fig. 8. An example calculation depicting the dataflow in the architecture is given in Table 5.

**Table 5:** RNS processor computation

| A=600 | B=600 | $m_1$=255 | $m_2$=256 | $m_3$=257 |
|---|---|---|---|---|
| Forward Conversion | | | | |
| $a_1=\|600\|_{255}=90$ | $a_2=\|600\|_{256}=88$ | | | $a_3=\|600\|_{257}=86$ |
| $b_1=\|600\|_{255}=90$ | $b_2=\|600\|_{256}=88$ | | | $b_3=\|600\|_{257}=86$ |
| IMAPU | | | | |
| $x_1=\|90x90\|_{255}= 195$ | $x_2=\|88x88\|_{256}=64$ | | | $x_3=\|86x86\|_{257}= 200$ |
| MRC based Reverse Conversion | | | | |
| $Z_1 = 195$ | $Z_2 = \|\|255\|^{-1}_{256}x(64-195)\|_{256}$ $= \|255x-131\|_{256}$ $= 131$ | | | |
| $Z_3 = \|(\|\|255\|^{-1}x\|256\|^{-1}\|_{257})x(200-(131x255)+195)\|_{257}$ $= \|32768x(200-33600)\|_{257}$ $= 5$ | X=(5x255x256)+ ((131x255)+195) =326400+33405+195 X=360000 | | | |
| X=3,60,000 | | | | |

### *4.2 Range analysis*

The permissible number ranges for balanced and unbalanced word-length residue multipliers are shown in Table 6. The bit-width required to represent triple moduli set {$2^n$-1, $2^n$, $2^n$+1} balanced system is 3n+1bits whereas the maximum number of bits required for unbalanced moduli {$2^k$+1,$2^k$, $2^k$-1} system is 3k+1.

## *5 Results and Discussions*

### *5.1 FPGA synthesis*

The architecture level functional verification of the proposed design is coded using Verilog HDL and simulated in the Xilinx ISIM tool. The results corresponding to hardware architectures are synthesized in Xilinx Synthesis Technology (XST) for balanced and unbalanced type residue multipliers. The results of the proposed architecture with CSA (Proposed-I) and prefix-based adders (Proposed-II) are presented in Table 7 and Table 8, respectively.

**Table 7:** FPGA synthesis results of balanced word-length SRM

| Multiplier | n | Xilinx Zynq FPGA Board (XC7Z020CLG484-1) | | | |
|---|---|---|---|---|---|
| | | Proposed - I | | Proposed - II | |
| | | LUT (No's) | Delay (ns) | LUT (No's) | Delay (ns) |
| $2^n$ | 8 | 37 | 11.5 | 37 | 11.4 |
| | 16 | 186 | 26.2 | 203 | 21.7 |
| | 32 | 928 | 70.3 | 1026 | 45.1 |
| $2^n$-1 | 8 | 112 | 18.6 | 144 | 17.7 |
| | 16 | 530 | 51.3 | 697 | 28.7 |
| | 32 | 2134 | 155.7 | 2848 | 53.0 |
| $2^n$+1 | 8 | 235 | 29.4 | 270 | 27.4 |
| | 16 | 688 | 85.8 | 861 | 44.1 |
| | 32 | 2215 | 196.7 | 3705 | 79.8 |

**Table 6:** Range analysis of triple moduli set RNS processor

| | Balanced Word-Length Moduli | | | Unbalanced Word-Length Moduli | | |
|---|---|---|---|---|---|---|
| Moduli | $2^n$-1 | $2^n$ | $2^n$+1 | $2^k$-1 | $2^k$ | $2^k$+1 |
| Number of Input Bits – A & B | n | | | | | |
| Number of Output Bits -P | n | n | n+1 | k | k | k+1 |
| Permissible Number Range | | | | | | |
| Input Range (Signed Integers) | $\left[-\frac{2^n}{2} \leftrightarrow \frac{2^n}{2}-1\right]$ | | | | | |
| Input Range (Unsigned Integers) | [0 ↔ 2n-1] | | | | | |
| Output Range -P | [0 ↔ $2^n$-2] | [0 ↔ $2^n$-1] | [0 ↔ $2^n$] | [0 ↔ $2^k$-2] | [0 ↔ $2^k$-1] | 0 ↔ $2^k$] |
| Dynamic Range of the Moduli | R= {$2^{3n}$-$2^n$ } | | | R= {$2^{3k}$-$2^k$ } | | |
| Permissible Range (Signed Integers) | $R=\left\{-\frac{\left[\left(2^{3n}-2^n\right)\right]}{2} \leftrightarrow \frac{\left[\left(2^{3n}-2^n\right)\right]}{2}-1\right\} \rightarrow \mathrm{Even}(n)$<br>$R=\left\{-\frac{\left[\left(2^{3n}-2^n\right)-1\right]}{2} \leftrightarrow \frac{\left[\left(2^{3n}-2^n\right)-1\right]}{2}\right\} \rightarrow \mathrm{Odd}(n)$ | | | $R=\left\{-\frac{\left[\left(2^{3k}-2^k\right)\right]}{2} \leftrightarrow \frac{\left[\left(2^{3k}-2^k\right)\right]}{2}-1\right\} \rightarrow \mathrm{Even}(k)$<br>$R=\left\{-\frac{\left[\left(2^{3k}-2^k\right)-1\right]}{2} \leftrightarrow \frac{\left[\left(2^{3k}-2^k\right)-1\right]}{2}\right\} \rightarrow \mathrm{Odd}(k)$ | | |
| Permissible Range (Unsigned Integers) | $R=\left\{0 \leftrightarrow \left(2^{3n}-2^n\right)-1\right\}$ | | | $R=\left\{0 \leftrightarrow \left(2^{3k}-2^k\right)-1\right\}$ | | |

**Table 8:** FPGA synthesis results of unbalanced word-length SRM

| Mul. | n | Zynq FPGA Board (XC7Z020CLG484-1) | | | | | |
|---|---|---|---|---|---|---|---|
| | | k=n/2 | | k=n/4 | | k=n/8 | |
| | | LUT (No's) | Delay (ns) | LUTs (No's) | Delay (ns) | LUT (No's) | Delay (ns) |
| Proposed -I | | | | | | | |
| $2^k$ | 8 | 6 | 8 | - | - | - | - |
| | 16 | 41 | 15 | 5 | 7.9 | - | - |
| | 32 | 250 | 43 | 41 | 13.2 | 5 | 7.9 |
| $2^k$-1 | 8 | 120 | 20 | - | - | - | - |
| | 16 | 645 | 67 | 654 | 68.8 | - | - |
| | 32 | 2577 | 162 | 2597 | 166.5 | 2605 | 168 |
| $2^k$+1 | 8 | 121 | 19 | - | - | - | - |
| | 16 | 649 | 70 | 654 | 78.1 | - | - |
| | 32 | 2587 | 173 | 1580 | 176.0 | 2599 | 179 |
| Proposed -II | | | | | | | |
| $2^k$ | 8 | 5 | 8 | - | - | - | - |
| | 16 | 44 | 13 | 5 | 7.8 | - | - |
| | 32 | 255 | 25 | 44 | 15.0 | 5 | 7.8 |
| $2^k$-1 | 8 | 156 | 18 | - | - | - | - |
| | 16 | 735 | 52 | 753 | 55.0 | - | - |
| | 32 | 2967 | 157 | 3018 | 61.9 | 3049 | 164 |
| $2^k$+1 | 8 | 163 | 21 | - | - | - | - |
| | 16 | 960 | 54 | 980 | 36.0 | - | - |
| | 32 | 2996 | 160 | 3056 | 162.5 | 3081 | 165 |

**LUT – Look Up Table & LE- Logic Element*

**Table 9:** ASIC synthesis results of balanced word-length SRM

| Mul. | n | Technology | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 180 nm | | | | 90 nm | | | | 45 nm | | | |
| | | Area ($\mu m^2$) | Power ($\mu W$) | Delay (ns) | PDP (pJ) | Area ($\mu m^2$) | Power ($\mu W$) | Delay (ns) | PDP (pJ) | Area ($\mu m^2$) | Power ($\mu W$) | Delay (ns) | PDP (pJ) |
| $2^n$ | | | | | | | | | | | | | |
| Proposed - I | 8 | 2164 | 217 | 1.3 | 0.3 | 682 | 30 | 0.8 | 0.02 | 369 | 20 | 0.5 | 0.0 |
| | 16 | 10438 | 1677 | 7.4 | 12 | 2967 | 308 | 4.2 | 1.3 | 1604 | 200 | 3.6 | 0.7 |
| | 32 | 39171 | 6095 | 26.8 | 163 | 12273 | 1218 | 15.2 | 18.5 | 6557 | 739 | 13 | 9.4 |
| Proposed - II | 8 | 2044 | 228 | 1.2 | 0.3 | 652 | 51 | 0.8 | 0.0 | 357 | 32 | 0.8 | 0.0 |
| | 16 | 11302 | 1809 | 6.7 | 12 | 3167 | 407 | 4.7 | 1.9 | 1712 | 232 | 4.1 | 1.0 |
| | 32 | 45784 | 6316 | 23.5 | 148 | 15787 | 1264 | 16.7 | 21 | 8435 | 769 | 14 | 10.7 |
| $2^n$-1 | | | | | | | | | | | | | |
| [22] | 8 | 8668 | 849 | 6.8 | 6 | 2733 | 169 | 4.2 | 1 | 1477 | 108 | 4.0 | 0.43 |
| | 16 | 34382 | 4328 | 29.1 | 126 | 9770 | 797 | 16.4 | 13 | 5281 | 513 | 14.3 | 7 |
| | 32 | 125346 | 19176 | 86.5 | 1658 | 39270 | 3835 | 48.9 | 188 | 20981 | 2323 | 41.2 | 96 |
| [14] | 8 | 8148 | 781 | 6.2 | 5 | 2569 | 156 | 3.8 | 1 | 1389 | 101 | 3.7 | 0.37 |
| | 16 | 32663 | 3983 | 26.5 | 105 | 9282 | 732 | 15.0 | 11 | 5017 | 473 | 13.0 | 6 |
| | 32 | 119079 | 17259 | 78.7 | 1357 | 37306 | 3452 | 44.5 | 154 | 19932 | 2089 | 37.5 | 78 |
| [23] | 8 | 8235 | 798 | 6.4 | 5 | 2597 | 158 | 3.9 | 1 | 1404 | 103 | 3.8 | 0.38 |
| | 16 | 32663 | 4069 | 27.0 | 110 | 9282 | 748 | 15.3 | 11 | 5017 | 483 | 13.3 | 6 |
| | 32 | 117825 | 18025 | 80.4 | 1449 | 36913 | 3607 | 45.5 | 164 | 19722 | 2185 | 38.3 | 84 |
| Proposed - I | 8 | 6518 | 789 | 5.1 | 4 | 2055 | 163 | 3.2 | 1 | 1111 | 124 | 3.1 | 0.38 |
| | 16 | 26447 | 4058 | 22.2 | 90 | 7516 | 824 | 12.5 | 10 | 4063 | 537 | 10.9 | 6 |
| | 32 | 97926 | 18282 | 67.0 | 1225 | 30679 | 3689 | 37.9 | 140 | 16392 | 2211 | 31.9 | 71 |
| Proposed - II | 8 | 6708 | 809 | 4.7 | 4 | 2081 | 169 | 2.8 | 0 | 1141 | 127 | 2.8 | 0.35 |
| | 16 | 28937 | 4147 | 20.1 | 83 | 8120 | 845 | 11.3 | 10 | 4389 | 548 | 9.7 | 5 |
| | 32 | 114459 | 18447 | 58.8 | 1084 | 35859 | 3722 | 33.7 | 125 | 19159 | 2234 | 27.5 | 61 |
| $2^n$+1 | | | | | | | | | | | | | |
| [21] | 8 | 17659 | 1058 | 13.8 | 15 | 5588 | 209 | 8.6 | 2 | 3020 | 135 | 8 | 1 |
| | 16 | 63121 | 5999 | 56.5 | 339 | 18179 | 1104 | 31.6 | 35 | 9827 | 713 | 27.8 | 20 |
| | 32 | 254585 | 30713 | 170 | 5221 | 79827 | 6145 | 96 | 590 | 41349 | 3719 | 81 | 301 |
| [15] | 8 | 15365 | 1049 | 13.6 | 14 | 4874 | 207 | 8.3 | 2 | 2634 | 135 | 8 | 1 |
| | 16 | 59211 | 5894 | 54.9 | 324 | 16899 | 1085 | 31 | 34 | 9135 | 699 | 27 | 19 |
| | 32 | 237894 | 28951 | 168 | 4864 | 73352 | 5792 | 95 | 550 | 39659 | 3508 | 80 | 281 |
| [16] | 8 | 13961 | 944 | 11.1 | 10 | 4402 | 188 | 6.8 | 1 | 2380 | 122 | 6.6 | 1 |
| | 16 | 53425 | 5216 | 51.6 | 269 | 15182 | 961 | 29.1 | 28 | 8207 | 618 | 25.4 | 16 |
| | 32 | 210141 | 25176 | 159.6 | 4018 | 65835 | 5036 | 90.3 | 455 | 35175 | 3049 | 76 | 232 |
| [17] | 8 | 13251 | 870 | 12.2 | 11 | 4142 | 173 | 7.5 | 1 | 2239 | 112 | 7.2 | 1 |
| | 16 | 52376 | 4658 | 46.7 | 218 | 14871 | 859 | 26.4 | 23 | 8039 | 554 | 23 | 13 |
| | 32 | 224280 | 22582 | 141.1 | 3186 | 68951 | 4518 | 79.8 | 361 | 37280 | 2736 | 67.2 | 184 |
| [18] | 8 | 12565 | 954 | 13.3 | 13 | 3962 | 190 | 8.2 | 2 | 2142 | 122 | 7.9 | 1 |
| | 16 | 48617 | 5306 | 57 | 302 | 13816 | 977 | 34 | 33 | 7468 | 629 | 29 | 18 |
| | 32 | 187025 | 26057 | 155 | 4039 | 58593 | 5211 | 81 | 422 | 31305 | 3155 | 78 | 246 |
| [19] | 8 | 15058 | 975 | 14 | 14 | 4776 | 194 | 9 | 2 | 2582 | 125 | 8.5 | 1 |
| | 16 | 52698 | 5130 | 55.1 | 283 | 15040 | 945 | 32 | 30 | 8130 | 610 | 28 | 17 |
| | 32 | 166526 | 23163 | 154 | 3567 | 51347 | 4633 | 80 | 371 | 27762 | 2805 | 74 | 208 |
| [20] | 8 | 8796 | 916 | 12.4 | 11 | 2773 | 182 | 8.3 | 2 | 1499 | 118 | 7.4 | 1 |
| | 16 | 38893 | 4882 | 44.6 | 218 | 11053 | 899 | 26.6 | 24 | 5974 | 581 | 22.5 | 13 |
| | 32 | 158971 | 21889 | 134.7 | 2948 | 49804 | 4379 | 64.7 | 283 | 26610 | 2650 | 60.5 | 160 |
| Proposed - I | 8 | 8356 | 908 | 11.7 | 11 | 2635 | 186 | 7.9 | 1 | 1424 | 143 | 7.1 | 1 |
| | 16 | 33059 | 4787 | 41.4 | 198 | 9395 | 972 | 24.5 | 24 | 5078 | 637 | 20.2 | 13 |
| | 32 | 122408 | 21797 | 120.2 | 2620 | 38349 | 4439 | 57.6 | 256 | 20489 | 2670 | 55.6 | 148 |
| Proposed - II | 8 | 8831 | 929 | 11 | 10 | 2825 | 192 | 7.2 | 1 | 1495 | 148 | 6.5 | 1 |
| | 16 | 36171 | 4841 | 37 | 179 | 10279 | 1004 | 22 | 22 | 5556 | 653 | 18.1 | 12 |
| | 32 | 143074 | 22127 | 102 | 2257 | 44857 | 4472 | 50.7 | 227 | 23982 | 2683 | 49.8 | 134 |

### *5.2 ASIC synthesis*

#### *5.2.1 Performance analysis*

From Table 9, the area comparison of $2^n$-1 SRM shows that the proposed architecture I & II requires less area compared to other multipliers [14][22][23]. The synthesis results show that the proposed design I occupy 17% - 22%, and design II occupies a 10% lesser area than existing modulo MBE. Delay analysis indicates that the proposed-I has a 17% - 24% speed improvement, and Proposed-II excels in speed by 26% - 30%. Power analysis shows that the total power required for the design is almost the same compared to recent works.

In $2^n$+1 SRM architectures, the proposed designs outperforms the other multipliers in area efficiency and speed improvement [15,16,17,18,19,20,21]. Proposed architecture I save area in the range of 23% - 44%, whereas the proposed architecture II reduces the area in the range of 10% - 32% compared to existing MBE architectures. The speed improvement of proposed-I and II lies between the ranges of 10% - 35% and 20% – 39%, respectively. The power profiles of the proposed

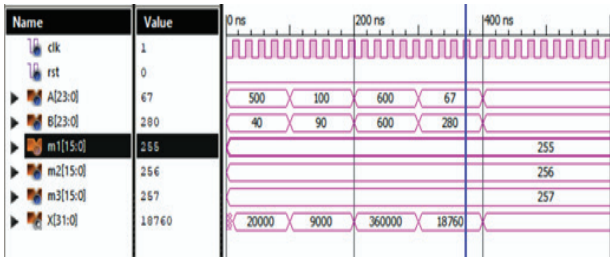


**Figure 9:** Simulation result of RNS processor

multipliers are almost the same as that of recent works. Since the proposed unbalanced residue multipliers are derived from proposed balanced residue multipliers, they follow the same trend in the area, delay, and power metrics, which are presented in Table 10.

The core problem addressed in this work is the improvement of speed performance of residue signed array multiplier, which generally consumes less area than its booth type counterparts. To achieve this objective, an enormous parallel operation from start to end is envisioned, designed, and implemented. It is inferred

**Table 10:** ASIC results (90 nm) of unbalanced word-length SRM

| Mul. | n | k=n/2 | | | | k=n/4 | | | | k=n/8 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Area ($\mu m^2$) | Power ($\mu W$) | Delay (ns) | PDP (pJ) | Area ($\mu m^2$) | Power ($\mu W$) | Delay (ns) | PDP (pJ) | Area ($\mu m^2$) | Power ($\mu W$) | Delay (ns) | PDP (pJ) |
| $2^k$ | | | | | | | | | | | | | |
| Proposed - I | 8 | 173 | 11.4 | 0.7 | 0.01 | - | - | - | - | - | - | - | - |
| | 16 | 640 | 32 | 2.5 | 0.08 | 124.2 | 5.4 | 1.6 | 0.01 | - | - | - | - |
| | 32 | 3224 | 241 | 8.0 | 1.93 | 739.2 | 26.4 | 5.8 | 0.15 | 142 | 13 | 1.5 | 0.02 |
| Proposed - II | 8 | 110 | 3.8 | 0.7 | 0.00 | - | - | - | - | - | - | - | - |
| | 16 | 637 | 45 | 2.7 | 0.12 | 110.7 | 10.7 | 1.5 | 0.02 | - | - | - | - |
| | 32 | 3564 | 254 | 8.5 | 2.16 | 732.6 | 36 | 4.5 | 0.16 | 129 | 20 | 2.6 | 0.05 |
| $2^k$-1 | | | | | | | | | | | | | |
| Proposed - I | 8 | 2172 | 181 | 3.6 | 1 | - | - | - | - | - | - | - | - |
| | 16 | 7757 | 905 | 14.1 | 13 | 8146 | 934 | 15.2 | 14.2 | - | - | - | - |
| | 32 | 31413 | 3953 | 42.15 | 167 | 32984 | 4085 | 45.2 | 184.7 | 33314 | 4142 | 47 | 195 |
| Proposed - II | 8 | 2196 | 186 | 3.3 | 1 | - | - | - | - | - | - | - | - |
| | 16 | 8405 | 950 | 12.8 | 12 | 8659 | 986 | 13.9 | 13.7 | - | - | - | - |
| | 32 | 36643 | 3993 | 36.8 | 147 | 37742 | 4145 | 39.5 | 164 | 38122 | 4188 | 41 | 173 |
| $2^k$+1 | | | | | | | | | | | | | |
| Proposed - I | 8 | 2212 | 203 | 3.7 | 0.7 | - | - | - | - | - | - | - | - |
| | 16 | 7846 | 964 | 14.3 | 14 | 8084 | 996 | 15.3 | 15 | - | - | - | - |
| | 32 | 31634 | 4043 | 42.6 | 172 | 32584 | 4184 | 45.5 | 190 | 32911 | 4241 | 48 | 204 |
| Proposed - II | 8 | 2240 | 209 | 3.6 | 0.7 | - | - | - | - | - | - | - | - |
| | 16 | 8575 | 1021 | 13.4 | 14 | 8821 | 1053 | 14.4 | 15 | - | - | - | - |
| | 32 | 36924 | 4142 | 38.2 | 158 | 38402 | 4264 | 40.1 | 171 | 39937 | 4323 | 45 | 192 |

**Table 11:** FPGA and ASIC synthesis Results of RNS Processor

| Parameter | FPGA Synthesis | | Parameter | ASIC Synthesis | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | 180 nm | | 90 nm | | 45 nm | |
| | Proposed - I | Proposed - II | | Proposed - I | Proposed - II | Proposed - I | Proposed - II | Proposed - I | Proposed - II |
| Number of LUTs | 23490 | 26508 | Area ($\mu m^2$) | 230237 | 241028 | 73676 | 77129 | 41259 | 43192 |
| | | | Power (mW) | 36 | 41 | 15 | 18 | 10 | 12 |
| Delay (ns) | 936 | 875 | Delay (ns) | 870 | 700 | 440 | 325 | 170 | 145 |

from the analysis that proposed designs have significant improvement in speed and area performance.

*5.2.2 Hardware Implementation of RNS Processor*

RNS processing examples discussed in Section 4 and the architecture is shown in Fig. 8 is simulated, and ISIM simulated results are shown in Fig. 9. The synthesis of the RNS Processor is done for both FPGA and ASIC platforms. The results for the same are presented in Table 11. The synthesized netlist of the RNS processor is implemented by targeting to the Xilinx Zynq board (XC7Z020CLG484-1).

## 6 Conclusion

A new array signed residue multiplication scheme for balanced ($2^n$-1, $2^n$+1, $2^n$) and unbalanced ($2^k$-1, $2^k$+1, $2^k$) word-length moduli are proposed in this paper. The proposed architecture with enormous parallelism is realized by incorporating CSA and Han-Carlson prefix adder structures into it. The existing and proposed multipliers are synthesized in both ASIC and FPGA technologies. From the synthesis results, the proposed-I $2^n$-1 residue multiplication scheme saves 17% area. However, the scheme with prefix structure achieves 26% speed and 24% PDP improvement compared to state of the art MBE $2^n$-1 residue multipliers. Similarly, a balanced $2^n$+1 proposed-I saves 23% area requirement. Speed and PDP improvement of proposed-II is 20% and 22 %, respectively, compared to the state of the art $2^n$+1 residue multipliers. The unbalanced multipliers derived from the balanced multiplier follows the same trend. Finally, the proposed residue arithmetic modules are used in arithmetic channel creation, reverse converter design of {$2^n$-1, $2^n$, $2^n$+1} triple moduli set RNS Processor and the same is implemented as hardware using Zynq (XC7Z020CLG484-1) device for real-time verification. The results indicate that the proposed designs can be efficiently utilized to improve the speed and area performances of RNS based cryptographic applications like RSA and ECC. The results also show that the proposed-I SRM architecture implemented using CSA may be used for area constrained RNS applications, and the Proposed-II SRM architecture using prefix can be used for high-speed applications.

## 7 Conflict of Interest

We have no conflict of interest to declare.

## 8 References


1. Sousa, L., Antao, S., Martins, P., " Combining Residue Arithmetic to Design Efficient Cryptographic Circuits and Systems," IEEE Circuits Syst. Mag., vol.16, no.4, pp.6-32, 2016. https://doi.org/10.1109/MCAS.2016.2614714
2. Celesti, A., Fazio, M., Villari, M., Puliafito, A., "Adding long-term availability, obfuscation, and encryption to multi-cloud storage systems," J. Netw. Comput. Appl., vol.59, pp. 208–218, 2016. https://doi.org/10.1016/j.jnca.2014.09.021
3. Chang, C.H., Molahosseini, A.S., Zarandi, A.A.E., Tay, T.F., "Residue number systems: A new paradigm to datapath optimization for low-power and high-performance digital signal processing applications," IEEE Circuits Syst. Mag., vol.15, no.5,pp.26–44,2015. https://doi.org/10.1109/MCAS.2015.2484118
4. Molahosseini, A.S., Zarandi, A.A.E., Martins, P., Sousa, L., "A multifunctional unit for designing efficient RNS-based datapaths," IEEE Access, vol.5, pp. 25972–25986,2017. https://doi.org/10.1109/ACCESS.2017.2766841

5. Patronik, P., Piestrak, S.J., "Hardware/Software Approach to Designing Low-Power RNS-Enhanced Arithmetic Units," IEEE Trans. Circuits Syst. I Regul. Pap., vol.64, no.5 , pp.1031–1039 , 2017. https://doi.org/10.1109/TCSI.2017.2669108
6. He, D., Chen, Y. & Chen, J., "An Id-Based Three-Party Authenticated Key Exchange Protocol Using Elliptic Curve Cryptography for Mobile-Commerce Environments," Arabian Journal for Science and Engineering (2013) 38: 2055. https://doi.org/10.1007/s13369-013-0575-4
7. Esmaeildoust, M., Schinianakis, D., Javashi, H., Stouraitis, T., Navi, K., "Efficient RNS implementation of elliptic curve point multiplication over GF(p)," IEEE Trans. Very Large Scale Integr. Syst. vol.21, no.8, pp.1545–1549, 2013. https://doi.org/10.1109/TVLSI.2012.2210916
8. Ding, J., Li, S., "A Modular Multiplier Implemented with Truncated Multiplication," IEEE Trans. Circuits Syst. II Express Briefs. vol.65, no.11, pp.1713–1717,2018. https://doi.org/10.1109/TCSII.2017.2771239
9. Antao, S., Sousa, L., "The CRNS framework and its application to programmable and reconfigurable cryptography," ACM Trans. Archit. Code Optim. , vol.9,no.4,pp.1–25,2013. https://doi.org/10.1145/2400682.2400692
10. Sung-Ming Y., Kim S., Lim S., Moon S., "RSA Speedup with Residue Number System Immune against Hardware Fault Cryptanalysis," In: Kim K. Information Security and Cryptology — ICISC 2001 Lecture Notes in Computer Science, vol 2288. Springer, Berlin, Heidelberg, 2002.
11. Fathy, K.A., Bahig, H.M. & Ragab A.A., "A Fast Parallel Modular Exponentiation Algorithm," Arabian Journal for Science and Engineering vol. 43, 903, 2018. https://doi.org/10.1007/s13369-017- 2797-3
12. Wei Wang, Swamy, M.N.S., Ahmad, M.O., "Moduli selection in RNS for efficient VLSI implementation," In: Proceedings of the 2003 International Symposium on Circuits and Systems, 2003. ISCAS '03. pp. 512-515, IEEE 2003.
13. Omondi, A., Premkumar, B., Residue number systems, Published by Imperial College Press and Distributed By World Scientific Publishing Co. , 2014.
14. Muralidharan, R., Chang, C.H., "Radix-8 booth encoded modulo $2^n$-1 multipliers with adaptive delay for high dynamic range residue number system," IEEE Trans. Circuits Syst. I Regul. Pap. vol.58,no.5,pp.982–993,2011. https://doi.org/10.1109/TCSI.2010.2092133
15. Vergos, H.T., Efstathiou, C., "Design of efficient modulo $2^n$+1 multipliers," IET Comput. Digit. Tech. vol.1,pp.49-57,2007. https://doi.org/10.1049/iet-cdt:20060026
16. Chen, J.W., Yao, R.H. "Efficient modulo $2^n$+1 multipliers for diminished-1 representation," IET Circuits, Devices Syst., vol.4, pp.291-300,2010. https://doi.org/10.1049/iet-cds.2009.0284
17. Chen, J.W., Yao, R.H., Wu, W.J., "Efficient modulo $2^n$+1 multipliers," IEEE Trans. Very Large Scale Integr. Syst., vol.19,no.12, pp.2149–2157, 2011. https://doi.org/10.1109/TVLSI.2010.2080330
18. Muralidharan, R., Chang, C.H., "Area-power efficient modulo $2^n$-1 and modulo $2^n$+1 Multipliers for {$2^n$-1, $2^n$, $2^n$+1} Based RNS," IEEE Trans. Circuits Syst. I Regul. Pap., vol.59, no.10,pp.2263–2274 , 2012. https://doi.org/10.1109/TCSI.2012.2185334
19. Efstathiou, C., Moshopoulos, N., Axelos, N., Pekmestzi, K., "Efficient modulo $2^n$+1 multiply and multiply-add units based on modified Booth encoding," Integr. VLSI J., vol.47, pp.140–147 ,2014. https://doi.org/10.1016/j.vlsi.2013.04.001
20. Mirhosseini, S.M., Molahosseini, A.S., Hosseinzadeh, M., Sousa, L., Martins, P., "A Reduced-Bias Approach with a Lightweight Hard-Multiple Generator to Design a Radix-8 Modulo $2^n$ + 1 Multiplier," IEEE Trans. Circuits Syst. II Express Briefs, vol.64, no.7, pp.817–821, 2017. https://doi.org/10.1109/TCSII.2016.2601285
21. Efstathiou, C., Vergos, H.T., Dimitrakopoulos, G., Nikolos, D., "Efficient diminished-1 modulo $2^n$+1 multipliers," IEEE Trans. Comput., vol.54, pp.491–496,2005. https://doi.org/10.1109/TC.2005.63
22. Efstathiou, C., Vergos, H.T., Nikolos, D., "Modified booth modulo $2^n$ - 1 multipliers," IEEE Trans. Comput., vol.53, 370–374,2004. https://doi.org/10.1109/TC.2004.1261842
23. Li, L., Hu, J., Chen, Y., "Modified booth encoding modulo ($2^n$-1) multipliers," IEICE Electron. Express, vol.9, no.5, pp.352–358, 2012. https://doi.org/10.1587/elex.9.352
24. Alaie, M.A., Timarchi, S., "Efficient modulo $2^n$+1 multiplier," Int. J. Comput. Aided Eng. Technol. , vol.8, no.3, pp. 260-276, 2016. https://doi.org/10.1504/ijcaet.2016.077604
25. S Elango , P Sampath, "Implementation of High Performance Hierarchy Based Parallel Signed Multiplier for Cryptosystems, J Circuit Syst. Comp., vol. 29, no. 13, pp. 2050214-1-2050214-25, 2020. https://doi.org/10.1142/S021812662050214X
26. Gupta, T., Sharma, J.B., "Han–Carlson adder based high-speed Vedic multiplier for complex multipli-

cation," Microsyst. Technol., vol.24, pp. 3901–3906, 2018.
https://doi.org/10.1007/s00542-018-3872-8
27. Ananda Mohan, P. V., Residue number systems: Theory and applications, Springer International Publishing, Cham, 2016.
28. Noorimehr, M.R., Hosseinzadeh, M. & Farshidi, R. 'High Speed Residue to Binary Converter for the New Four-Moduli Set $\{2^{2n}, 2^n+1, 2^{n/2}+1, 2^{n/2}-1\}$ Arabian Journal for Science and Engineering, vol.39, 2887,2014.
https://doi.org/10.1007/s13369-014-0963-4
29. Sousa, L., Antão, S.: MRC-based RNS reverse converters for the four-moduli sets $\{2^n-1, 2^n, 2^n+1, 2^{2n+1}-1\}$,"IEEE Trans. Circuits Syst. II Express Briefs,vol.59,no.4,pp.244–248,2012.
https://doi.org/10.1109/TCSII.2012.2188456
30. Sedjelmaci, S.M., "A parallel extended GCD algorithm," J. Discret. Algorithms, vol.6, pp.526 – 538,2008.
https://doi.org/10.1016/j.jda.2006.12.009



Arrived: 27. 10. 2019
Accepted: 31. 03. 2020