ICIC Express Letters
Volume **10**, Number **12**, December **2016**
 ISSN 1881-803X
pp. **2963–2969**

AREA-EFFICIENT MULTIFUNCTION MODULO $(2^n \pm 1)$ SQUARER DESIGN USING MODIFIED BOOTH ENCODING SCHEME

Chao-Tsung Kuo[1,*], Tso-Bing Juang[2] and Yu-Ming Cheng[1]

[1]Department of Electronic Engineering
National Quemoy University
No. 1, University Rd., Jinning Township, Kinmen County 892, Taiwan
*Corresponding author: ctkuo@nqu.edu.tw; ming19940608@gmail.com

[2]Department of Computer Science and Information Engineering
National Pingtung University
No. 4-18, Minsheng Rd., Pingtung City 90003, Taiwan
tsobing@mail.nptu.edu.tw

Received May 2016; accepted August 2016

Abstract. *The residue number system (RNS) has been used for many applications such as cryptography, communication components, digital signal processing, digital filter, Fermat number transform (FNT), encryption operation and partial encryption of international data encryption algorithm (IDEA), which can provide significant speed savings compared with binary system. In this paper, an area-efficient multifunction RNS modulo* $(2^n \pm 1)$ *squarer is proposed. By using common partial product arrays, our proposed modulo* $(2^n \pm 1)$ *squarer based on modified booth encoding schemes could perform both modulo* $(2^n + 1)$ *squaring and modulo* $(2^n - 1)$ *squaring operations on the same hardware as demanded. Our proposed squarer can achieve significant 24.19% and 46.68% area savings compared with the hardware required for individual modulo* $(2^n + 1)$ *squarer and modulo* $(2^n - 1)$ *squarer for* $n = 8$ *and 16, respectively. Our hardware is implemented using Xilinx Spartan 3E FPGA.*
Keywords: Field programmable gate array (FPGA), Residue number system (RNS), Modulo $(2^n \pm 1)$ squarer, Computer arithmetic

1. **Introduction.** The residue number system (RNS) can be applied to cryptography, communication components, digital signal processing, digital filter, Fermat number transform (FNT), encryption operation and partial encryption of international data encryption algorithm (IDEA). The benefit of using RNS can provide significant speedup over binary system due to the fact that limited carry propagation is required in RNS.

So far, $\{2^n+1, 2^n-1\}$ is one of the most commonly used moduli sets for RNS operations applied in digital signal processing, pseudorandom number generation, cryptography and digital filter [1-15]. There were many works such as modulo $(2^n + 1)$ squarer [7], modulo $(2^n + 1)$ multipliers [8,9,11,12,14,15] and modulo $(2^n - 1)$ multipliers [10-12] presented in previous literature. Among these modulo squaring and multiplication computations, partial products (PP) and carry save adder (CSA) are commonly adopted in modulo $(2^n + 1)$ [11,13,15] or modulo $(2^n - 1)$ adder [11] which is used to get the final modulo computation result. In the classification of modulo input/output format, diminished-1 [7,8,13] and weighted [9,11,12,14,15] representatives are the commonly used. In diminished-1 usage, every number is subtracted by one so that n-bit adders can be used instead of using $(n + 1)$-bit adders for modulo operations. Considering the tradeoff of delay time and area cost, diminished-1 input and weighted output representation are used in our proposed modulo $(2^n \pm 1)$ squarer design.

In our work, our motivation is to develop a multifunction RNS modulo $(2^n \pm 1)$ squarer using modified booth encoder scheme which can be operated for two squaring functions

on the same hardware for slightly increased delay. Since the squaring operations are different from the operations for multiplication or multiplication and accumulation (MAC) proposed in [15], our work is to propose more area-efficient implementation based on improving previous work proposed in [7].

The remainder of this paper is organized as follows. In Section 2, we will review previous related methods for modulo RNS multiplication and squaring operations. In Section 3, an area-efficient modulo $(2^n \pm 1)$ squarer using modified booth encoding scheme is proposed. Field programmable gate array (FPGA) hardware implementation will be presented in Section 4. Finally, Section 5 draws the conclusions.

2. **Previous Related Methods for Modulo RNS Multiplication and Squaring Operations.** The architecture for modulo $(2^n \pm 1)$ squaring and multiplications are very similar in all aspects for hardware implementation, with only one difference being that the number of partial products in modulo $(2^n \pm 1)$ squarer is fewer than that in modulo $(2^n \pm 1)$ multiplication. Therefore, in this section, we will firstly describe the basic operations of modulo $(2^n + 1)$ and modulo $(2^n - 1)$ multipliers, and then discuss the operations required for modulo $(2^n + 1)$ squarer in previous literature. In previous literature, there were many works about modulo $(2^n + 1)$ multipliers [8,9,11,12,14,15] and modulo $(2^n - 1)$ multipliers [10-12]. The mathematical equations about modulo $(2^n + 1)$ and modulo $(2^n - 1)$ multipliers are shown in Equation (1) and Equation (2), respectively.

Let $A = \sum_{i=0}^{n} 2^i a_i$, $B = \sum_{i=0}^{n} 2^i b_i$,

$$|A \times B|_{2^n+1} = \left| \sum_{i=0}^{n} a_i 2^i \sum_{j=0}^{n} b_j 2^j \right|_{2^n+1} = \left| \sum_{i=0}^{n} \left(\sum_{j=0}^{n} PP_{i,j} 2^{i+j} \right) \right|_{2^n+1} \tag{1}$$

Let $A = \sum_{i=0}^{n-1} 2^i a_i$, $B = \sum_{i=0}^{n-1} 2^i b_i$,

$$\begin{aligned} |A \times B|_{2^n-1} &= \left| \sum_{i=0}^{n-1} a_i 2^i B \right|_{2^n-1} = \left| \sum_{i=0}^{n-1} a_i \times (b_{n-i-1} b_{n-i-2} \dots b_0 \dots b_{n-i}) \right|_{2^n-1} \\ &= \left| \sum_{i=0}^{n-1} PP_i \right|_{2^n-1} \end{aligned} \tag{2}$$

In 2005, Vergos and Efstathiou [7] presented diminished-1 modulo $(2^n + 1)$ squarer, and Vergos's mathematical equation about diminished-1 based modulo $(2^n + 1)$ squarer is shown in Equation (3) and Equation (4). In Equation (3) and Equation (4), A is $(n + 1)$ bits input number, A_{-1} is denoted as diminished-1 representation, Q denotes the squaring result of A_{-1} modulo $(2^n + 1)$, Q_{-1} represents diminished-1 of output Q, pp_i is the ith row partial product and C_i denotes the ith row correction factor. We can observe that the Vergos's method proposed in [7] can still be improved in hardware area and delay consideration, which will be described in Section 3.

Let $A_{-1} = a_{n-1} a_{n-2} \dots a_1 a_0$,

$$\left| A_{-1}^2 \right|_{2^n+1} = \left| \sum_{i=0}^{n-1} \sum_{j=0}^{n-1} x_{i,j} 2^{|i+j|_n} \right|_{2^n+1} = \left| \sum_{i=0}^{n-1} (pp_i + C_i) \right|_{2^n+1} \tag{3}$$

$$Q_{-1} + 1 = \left| (A_{-1} + 1)^2 \right|_{2^n+1}, \; Q_{-1} = \left| \left| A_{-1}^2 \right|_{2^n+1} + |2 \times A_{-1}|_{2^n+1} \right|_{2^n+1} \tag{4}$$

To the best of our knowledge, there is not any multifunction modulo $(2^n \pm 1)$ squarer using the same hardware being proposed. In this work, we will propose area-efficient multifunction modulo $(2^n \pm 1)$ squarer using modified booth encoding.

3. **Proposed Area-Efficient Modulo ($2^n \pm 1$) Squarer Using Modified Booth Encoding.** The derived mathematical equation of our proposed multifunction RNS modulo $(2^n \pm 1)$ squarer using modified booth encoder is shown in Equation (5) and Equation (6). Let A_{-1} be an n-bit unsigned binary number denoted as $A_{-1} = a_{n-1}a_{n-2}\ldots a_1a_0$. $|\cdot|_{2^n\pm1}$ is denoted as modulo (2^n+1) or modulo (2^n-1) operation. It should be noted that we use diminished-1 input A_{-1} representative in our work. A_{-1} represents the input value A subtracted by one. The output Q is the final result which is represented by weighted representations to fit the correct numbers. In Equation (6), W is the compensation factor for modulo (2^n+1) squaring operation, which includes the inversion of A_{-1} and Z vector which will be shown in Figure 5.

Let $A_{-1} = \sum_{i=0}^{n-1} 2^i a_i$,

$$\begin{aligned}
\left|A_{-1}^2\right|_{2^n\pm1} &= |A_{-1} \times A_{-1}|_{2^n\pm1} \\
&= |A_{-1} \times [-a_n(2^n-1) + a_{n-1}2^{n-1} + \cdots + a_2 2^2 + a_1 2 + a_0]|_{2^n\pm1} \\
&= |A_{-1} \times [-a_n(2^{n+1}-1) + a_n 2^n + a_{n-1}2^{n-1} + \cdots + a_2 2^2 + a_1 2 + a_0]|_{2^n\pm1} \\
&= |A_{-1} \times [2^n(a_n + a_{n-1} - 2a_{n+1}) + 2^{n-2}(a_{n-2} + a_{n-3} - 2a_{n-1}) + \cdots \\
&\quad +(a_0 + a_n - 2a_1)]|_{2^n\pm1} \\
&= \left|\sum_i A_{-1} \times 2^{2i}(a_{2i-1} + a_{2i} - 2a_{2i+1})\right|_{2^n\pm1}
\end{aligned} \tag{5}$$

$$|Q|_{2^n\pm1} = \left|A_{-1}^2\right|_{2^n\pm1} = \begin{cases} \left|\sum_i^{PP_i}\right|_{2^n-1}, & \text{for modulo } 2^n-1 \\ \left|\sum_i^{PP_i} + W\right|_{2^n+1}, & \text{for modulo } 2^n+1,\ W \text{ is compensation factor} \end{cases} \tag{6}$$

The block diagram of proposed multifunction RNS modulo $(2^n \pm 1)$ squarer using modified booth encoder is shown in Figure 1. These blocks contain booth encoder, booth selector, partial product addition array and carry save adder array. Figure 2 is our proposed partial product hardware architecture of multifunction RNS modulo $(2^8 \pm 1)$ squarer. In Figure 2, S is the control signal which is used to control module (2^n+1) or module (2^n-1). BE and BS represent booth encoder and booth select circuit block. PP is the partial product value of each block. The corresponding circuits of booth encoder (BE)

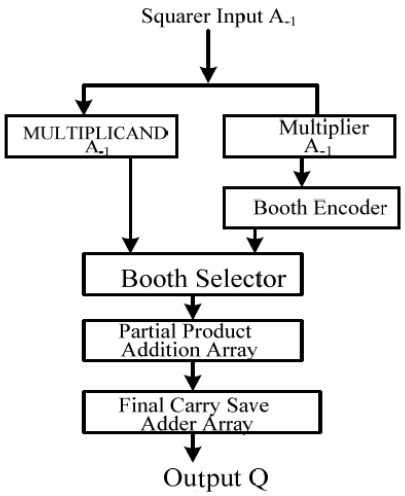

FIGURE 1. Block diagram of proposed modulo $2^n \pm 1$ squarer

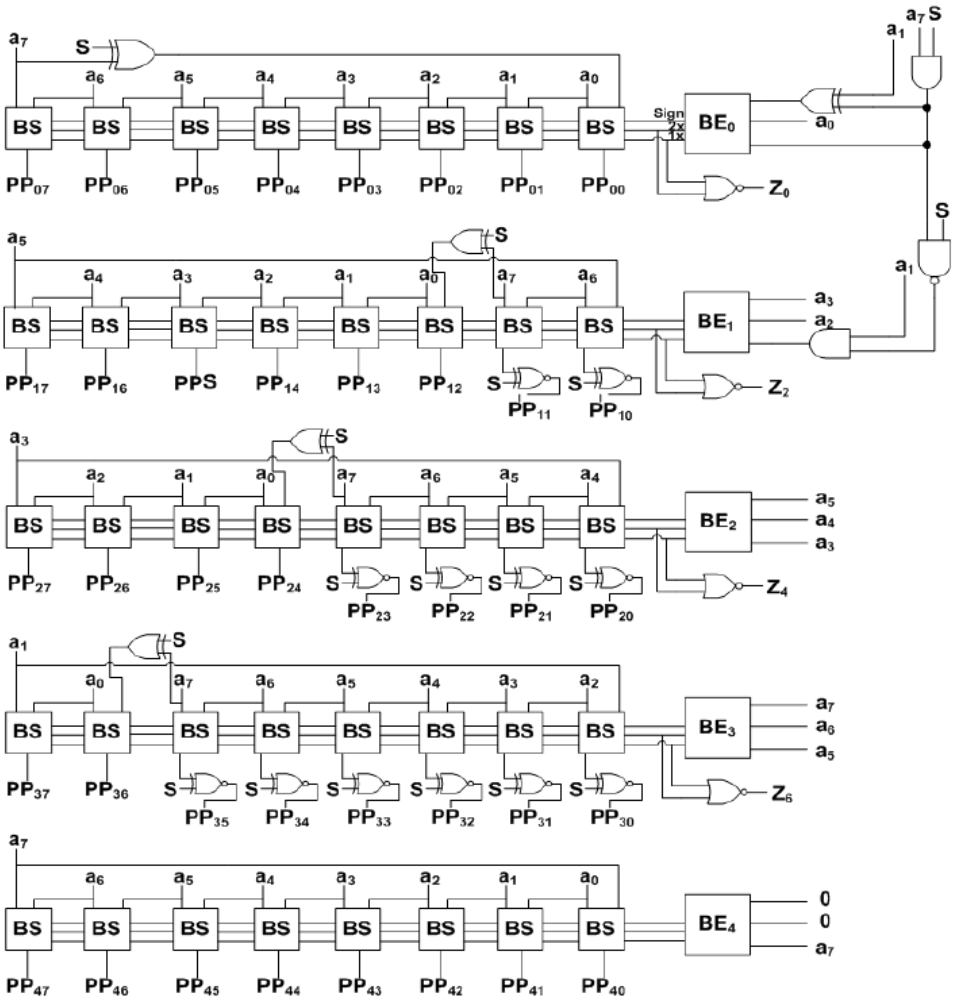

FIGURE 2. Proposed architecture for generating partial products in modulo $(2^8 \pm 1)$ squarer

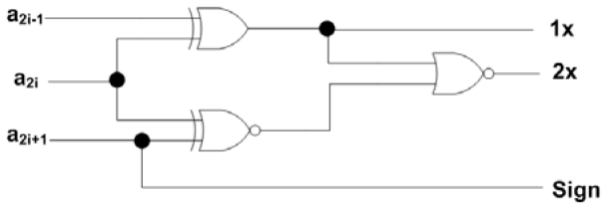

FIGURE 3. The architecture of booth encoder (BE) circuit [12]

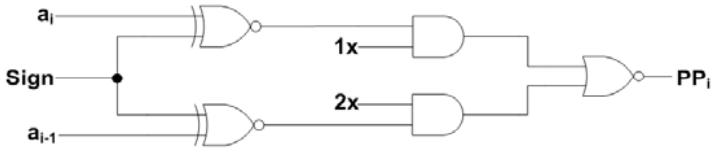

FIGURE 4. The architecture of booth selector (BS) circuit [15]

and booth select (BS) are shown in Figure 3 and Figure 4, respectively. In Figure 3, BE is used to decide multiply one (1x) or multiply two (2x) and produce the value of output Z. The architecture of proposed carry save adder (CSA) array in modulo $(2^n \pm 1)$ squarer is shown in Figure 5. In Figure 5, taking n to be 8, S signal is used to control modulo

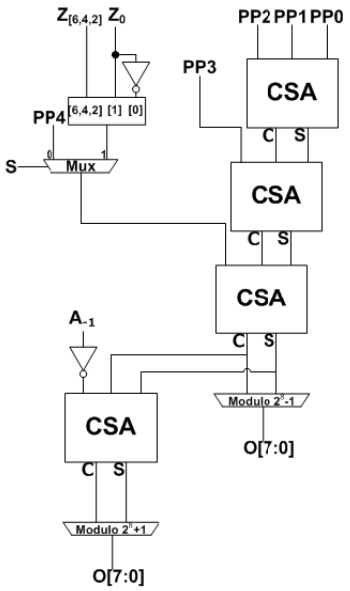

FIGURE 5. Proposed multifunction RNS modulo $(2^8 \pm 1)$ squarer carry save adder (CSA) arrays

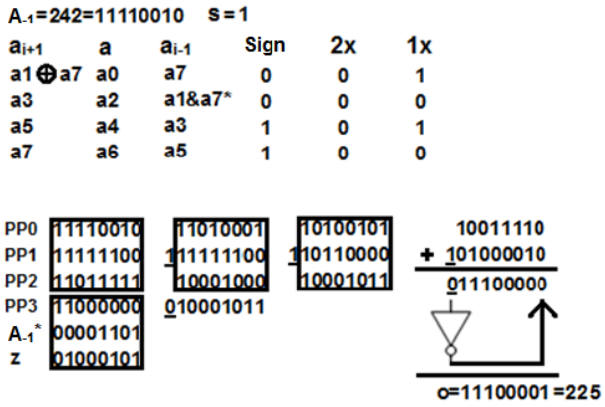

FIGURE 6. Numerical example of our proposed 242^2 modulo $(2^8 + 1)$ squarer

(2^n+1) or modulo $(2^n - 1)$ ($S = 0$ is for modulo $(2^n - 1)$ and $S = 1$ is for modulo $(2^n + 1)$ squaring operation, respectively). Compensation factor W is needed in modulo $(2^n + 1)$, and it includes the inversion of input A_{-1} and Z vector. Partial product array can be summed to produce output in modulo (2^n-1). The final calculation result of modulo $(2^8 \pm 1)$ squarer can be shown in $O[7:0]$.

Figure 6 is the numerical example of our proposed modulo $(2^8 + 1)$ squarer, taking A_{-1} equal to 242 as example, using modified booth encoding schemes, the corresponding first row 001 to obtain $pp_0 = 11110010$, second row 000 to obtain $pp_1 = 11111100$, third row 101 to obtain $pp_2 = 11011111$ and fourth row 100 to obtain $pp_3 = 11000000$. A^*_{-1} is the inversion of A_{-1}, and Z is obtained from booth encoder which is shown in Figure 2 and Figure 5. It should be noted that the value of the 1st, 3rd, 5th and 7th bits in Z will be set to zero. Therefore, we can easily obtain the result of 242^2 modulo $(2^8 + 1)$ squarer to be 225 using carry save adder. Figure 7 is the numerical example of our proposed modulo $(2^8 - 1)$ squarer. The mathematical procedure is using carry save adder for each partial product (PP).

4. **FPGA Hardware Implementation.** We have designed our proposed multifunction modulo $(2^n \pm 1)$ squarer with Xilinx Spartan 3E FPGA. Since the hardware of our proposed

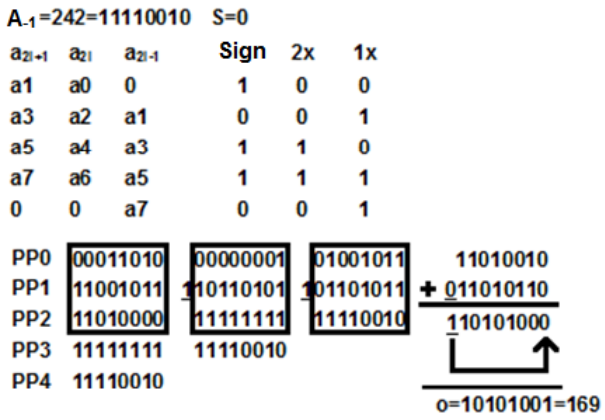

FIGURE 7. Numerical example of our proposed 242^2 modulo $(2^8 - 1)$ squarer

TABLE 1. Area/delay comparison of the proposed work compared with individual squarers (The hardware is implemented using Xilinx Spartan 3E FPGA)

Items	Proposed modulo $(2^n - 1)$ squarer		Proposed modulo $(2^n + 1)$ squarer		**Proposed modulo $(2^n \pm 1)$ squarer**	
n	8	16	8	16	**8**	**16**
Delay time (ns)	18.557	20.594	17.957	21.972	**24.774**	**24.477**
Area (LUT)	103	428	112	446	**163**	**466**

TABLE 2. Area savings of the proposed work compared with combined squarer (The hardware is implemented using Xilinx Spartan 3E FPGA)

Items	Combined proposed modulo $(2^n - 1)$ and modulo $(2^n + 1)$ squarer		**Proposed modulo $(2^n \pm 1)$ squarer**	
n	8	16	**8**	**16**
Area (LUT)	215	874	**163**	**466**
Area savings	–	–	**24.19%**	**46.68%**

squarer can be set to perform two different squaring operations, which is different from Vergos's method [7] or other work for modulo squarer, in this paper, our proposed modulo $(2^n \pm 1)$ squarer design is only compared with the hardware for combined proposed modulo $(2^n + 1)$ and proposed modulo $(2^n - 1)$ squarer, in which 'combined' is represented as the combination of modulo (2^n+1) and modulo (2^n-1) squarer for hardware implementations. The hardware implementation of the proposed work compared with combined proposed modulo $(2^n + 1)$ and proposed modulo $(2^n - 1)$ squarer method is shown in Table 1 and Table 2. The power number of n in this modulo $(2^n \pm 1)$ squarer is taken 8 and 16 for hardware implementation. In Table 1 and Table 2, we can observe that our proposed modulo $(2^n \pm 1)$ squarer could achieve area saving of 24.19% $(n = 8)$, 46.68% $(n = 16)$ compared with combined modulo $(2^n + 1)$ and $(2^n - 1)$ squarer using the original circuits.

5. **Conclusions.** In this paper, we have proposed an area-efficient multifunction RNS modulo $(2^n \pm 1)$ squarer. Using common partial product arrays and the same original circuits, our proposed modulo $(2^n \pm 1)$ squarer based on modified booth encoding schemes could perform the function of both modulo (2^n+1) squaring and modulo (2^n-1) squaring on the same hardware with tolerable delay. Also, our proposed work can achieve significant 24.19% and 46.68% area saving compared with the hardware summation of individual modulo $(2^n + 1)$ squarer and modulo $(2^n - 1)$ squarer for modulo 2 based of power value

n, n being equal to 8 and 16 respectively. Our hardware is implemented using Xilinx Spartan 3E FPGA. Our proposed multifunction modulo $(2^n \pm 1)$ squarer using modified booth encoding scheme can be applied to many applications such as cryptography, Fermat number transform (FNT), encryption operation and partial encryption of international data encryption algorithm (IDEA).

REFERENCES

[1] K. Kaluri, W. F. Leong, K.-H. Tan, L. Johnson and M. Soderstrand, FPGA hardware implementation of an RNS FIR digital filter, *Conference Record of the 35th Asilomar Conference on Signals, System and Computers*, pp.1340-1344, 2001.

[2] J. Ramirez et al., RNS-enabled digital signal processor design, *Electronics Letters*, vol.38, no.6, pp.266-268, 2002.

[3] G. L. Brenocchi, G. C. Cardarilli, A. Del Re, A, Nannarelli and M. Re, Low-power adaptive filter based on RNS components, *Proc. of the IEEE International Symposium on Circuits and Systems*, pp.3211-3214, 2007.

[4] M. A. Soderstrand et al., Residue number system arithmetic, *Modern Application in Digital Signal Processing*, 1986.

[5] A. Nannarelli, M. Re and G. C. Cardarilli, Tradeoffs between residue number system and traditional FIR filter, *Proc. of the IEEE International Symposium on Circuits and Systems*, pp.305-308, 2001.

[6] G. C. Cardarilli, A. Nannarelli and M. Re, Reducing power dissipation in FIR filters using the residue number system, *Proc. of the 43rd IEEE Midwest Symposium on Circuits and System*, pp.320-323, 2000.

[7] H. T. Vergos and C. Efstathiou, Diminished-1 modulo $2^n + 1$ squarer design, *IEE Proceedings – Computer and Digital Techniques*, vol.152, no.5, pp.561-566, 2005.

[8] H. T. Vergos and D. Nikolos, Efficient diminished-1 modulo $2^n + 1$ multipliers, *IEEE Trans. Computers*, vol.51, no.4, pp.491-496, 2005.

[9] H. T. Vergos and C. Efstathiou, Design of efficient modulo $2^n + 1$ multipliers, *IET Comput. and Digital Tech.*, vol.1, no.1, pp.49-57, 2007.

[10] H. T. Vergos and C. Efstathiou, Modified booth 1's complement and modulo $2^n - 1$ multipliers, *Proc. of the 7th IEEE International Conference on Electronics, Circuits and Systems*, vol.2, pp.637-640, 2000.

[11] R. Zimmerman, Efficient VLSI implementation of modulo $(2^n \pm 1)$ addition and multiplication, *Proc. of the 15th IEEE Symposium on Computer Arithmetic*, pp.158-167, 1999.

[12] T.-B. Juang and J.-H. Huang, Multifunction RNS modulo $2^n \pm 1$ multipliers based on modified booth encoding, *IEEE Asia Pacific Conference on Circuits and Systems*, pp.515-518, 2012.

[13] H. T. Vergos, C. Efstathiou and D. Nikolos, High speed parallel-prefix modulo $2^n + 1$ adders for diminished-one operands, *Proc. of the 15th IEEE Symposium on Computer Arithmetic*, pp.211-217, 2001.

[14] J. W. Chen, R. H. Yao and W. J. Wu, Efficient modulo $2^n + 1$ multipliers, *IEEE Trans. Very Large Scale Integration System*, vol.19, no.12, 2011.

[15] C. Efstathiou, N. Moshopoulos, N. Axelos and K. Pekmestzi, Efficient modulo $2^n + 1$ multiply and multiply-add units based on modified booth encoding, *Integration, the VLSI Journal*, vol.47, pp.140-147, 2014.