Implementation of Effective Modular Adder Designs Using One-Hot Coding and Thermometer on FPGA

Mr.D.Satyanarayana [(1)] Dr.A.Ranganayakulu [(2)] Syed Yasmeen [(3)] Alasandalapalli Swaroopa [(4)] Thirumala Usharani Katari [(5)] Thumbeti Swetha [(6)]

[1,2] *Krishna Chaithanya Institute Of Technology & Sciences, Ece Department, Markapur, Andhra Pradesh.*

[3,4,5,6,] *Krishna Chaithanya Institute Of Technology & Sciences, UG Student-ECE, Markapur, Andhra Pradesh.*

Abstract In this project, Positional number systems can be effectively replaced by residue number systems (RNSs), which offer quick and power-efficient computational systems. The RNS's energy efficiency is a vital aspect that helps current embedded systems and Internet-of-things (IoT) edge devices. The most significant and commonly used operation on RNS components, such as the forward and reverse converters and the arithmetic units in the channels, is modular addition. The use of thermometer coding (TC) and onehot coding (OHC) is feasible due to the small and medium dynamic range needs of low power embedded and edge devices. This lowers power consumption and improves the energy efficiency of modulo addition when compared to normal binary representations. This project proposes two novel energy-efficient modular adders that are also highly performing because of the carry-free internal computations, based on these methodologies. Comparing the suggested modular adders based on the TC and OHC to the related state of the art, the improvements are on average 38% and 34.5% for the delay, 27% and 14.5% for the circuit area, 29.5% and 6.3% for energy consumption, and roughly 54.9% and 44.2% for the area-delay product (ADP), respectively and this design can be implemented on Xilinx Vivado on FPGA Simulator.

Keywords: One Hot Codings,5G Communications, FPGA, Verilog HDL, Thermometer coding for Encoding Deocders.

I.INTRODUCTION

Modern digital signal processing (DSP) systems [1] require improved energy efficiency, namely, for emerging applications such as deep learning [2] and Internet-of-Things (IoT) [3]. Unconventional number systems and arithmetic have been investigated recently to achieve specialized efficient embedded systems for those applications [1]. Residue number systems (RNSs) [4] have been used for DSP [5] and cryptography [6], supporting highspeed, low-power, and fault-tolerant computations. Mapping weighted number representations into residues and vice versa, i.e., forward and reverse conversion, are essential but complex intermodulo operations. However, RNS arithmetic operations, such as additions and multiplications, are performed much more frequently than forward and reverse conversions [1]. Therefore, efficient modular adders are essential to achieve RNS-based high-performance and highly efficient embedded computing systems.

One way to increase the efficiency of modular arithmetic units, i.e., modulo adders, subtractors, and multipliers, is by using the one-hot coding (OHC) [7]–[9]. The one-hot residue (OHR) has been considered for designing RNS modular arithmetic circuits based on circular shifting [10]. The OHC circuits based on barrel shifters show a power-delay product (PDP) reduction of up to 85% in comparison to the conventional positional encoding, because they significantly reduce

the circuit's activity factor [10]. RNSs based on OHC have also been used on DSP applications due to their high speed [9], [11]. Alternatively, there are other types of coding, such as the thermometer coding (TC) [12] that can be applied to enhance the performance of RNS modular arithmetic. The thermometer is the unary coding, in which the number of 1's corresponds to the magnitude of the displayed number. This means that the Hamming distance between numbers represented in TC has a linear relationship to its difference [12]. This type of coding is a subclass of Golomb coding's [13] used in a variety of applications, including neural networks and data compression [13], [14]. Moreover, the TC together with distributed arithmetic can lead to fast implementations of modular arithmetic circuits [15], [16].

With existent analog-to-digital converters (ADCs) that directly convert analog inputs to residues encoded in the TC format [17], engineers and researchers are increasingly interested in modular adders for TC [18]. Since there is no carry propagation in the modular addition of two TC numbers, the addition of small moduli can be done faster when compared to designs for positional representations (apart from moduli like 2n). Moreover, the usage of the TC format to represent residues removes the need for forward converters, whenever inputs values are coded using analog-toresidue converters [17]. Similarly, modular adders based on OHC operate in a carry free manner [19], having a simple structure for different moduli. To set up efficient RNS systems with TC or OHC, small moduli should be selected. This makes this approach more suitable to a class of applications that includes low power embedded and IoT edge devices, where a small dynamic range is required [1]. For applications requiring larger dynamic ranges, sets with a higher number of moduli should be selected, while also possibly including a modulo 2n channel exploiting binary representations.

This project creates new efficient structures for designing modular adders based on OHC and TC. The proposed adders are designed using novel digital circuits supported on specific features of OHC and TC. In comparison to previous works [18], [19], which proposed adders based on shifting and were implemented using many multiplexers (MUXs), the herein proposed adders have significant practical experimental improvements regarding the latency, area, and energy consumption.

2.LITERATURE SURVEY

A. Residue Number Systems :

An RNS supported on a moduli-set of n pairwise relatively prime numbers {m1, m2,..., mn} allows for the integers in [0, M − 1], with M = m1 × m2 × ... × mn, to be uniquely represented by their remainders modulo m1, m2,..., mn. A forward conversion maps the binary weighted representation of a number X onto the residues (x1, x2,..., xn) associated with the moduli in the set xi = |X|mi = |X M B−1 ... X1X0|mi, i = 1 ... n (1) where M B = log2 mi [4]. Considering two RNS numbers A and B A = (a1, a2,..., an) (2) B = (b1, b2,..., bn). (3) The modular addition of these two RNS numbers can be performed as follows [20]: S = A + B = (s1,s2,...,sn) (4) with si = |ai + bi|mi = ai + bi, if ai + bi < mi ai + bi − mi, if ai + bi ≥ mi . (5) The reverse conversion maps the residues in the RNS domain (x1, x2,..., xn) into their equivalent weighted binary representations, using a method like the mixed-radix conversion (MRC) [4] X = vn n −1 i=1 mi + ... + v3m2m1 + v2m1 + v1 (6).

the first coefficient v_1 is equal to x_1, and for the other coefficients, we have that

$$v_n = |(((x_n - v_1)|m_1^{-1}|_{m_n} - v_2)|m_2^{-1}|_{m_n} - \cdots - v_{n-1})|m_{n-1}^{-1}|_{m_n}|_{m_n} \quad (7)$$

where $|m_i^{-1}|_{m_j}$ denotes the multiplicative inverse of m_i modulo m_j.

B. Thermometer Coding:

With TC [18], the value of each number is expressed as the number of ones in a string of bits. Since, in this coding, no weight is assigned to bit positions, it is not important where the ones are placed. However, for simplicity, these 1s are usually placed at one end of the string [18]. As an example, the numbers between 0 and 7 are represented as shown in Table I. Table I shows that by increasing the range of numbers, the amount of required bits in a TC representation grows at a fast pace. Therefore, the TC is not suitable to represent large numbers. Nevertheless, the decomposition of numbers in a set of small residues makes TC-based fast circuits suitable for RNS. Therefore, combining TC with RNS improves the performance and efficiency of arithmetic systems. The application of the thermometer code to RNS residues is herein designated TC for RNS (TCR).

TABLE I
TC EXAMPLE

Regular Representation	TC
0	0000000
1	0000001
2	0000011
3	0000111
4	0001111
5	0011111
6	0111111
7	1111111

In [18], it is stated that the number bits required to represent all the remainders modulo mi are mi . However, it should be mentioned that, since the leftmost bit is always zero, these numbers can, in fact, be represented with only mi − 1 bits. To perform an addition in TC, one needs to count the number of ones in the two operands. If this number is greater than or equal to mi , the sum should be decreased by mi . As an example, to add the two numbers A = 001111 and B = 011111 modulo 7 in TC, since a total of 9 bits take the value of one when considering all the bits of the two representations, one removes 7 of them to obtain the result of 000011.
In order to perform a subtraction in TC, the complement of the subtrahend (B) has to be computed and added to the minuend (A). When B is not zero, the complement of B can be easily calculated as follows: $B^{\neg} = b^{\neg}2b^{\neg}3b^{\neg}4 \ldots b^{\neg}m{-}1\ 1$, if (B! = 00 ... 0). (8) Herein, as in [18], the starting index of the bits is 1, i.e., b1. In other words, the rightmost bit, b1,lets one know whether a number is greater than or equal to 0. For instance, if B is equal to 000011 modulo 7, its complement, $B^{\neg}$, takes the value of 011111. Moreover, the complement of B = 000000 is also B = 000000, which can be easily identified by checking the value of the first bit (b1 = 0). In general, the complement of B can be calculated as $B^{\neg} = (b^{\neg}2 \wedge b1)(b^{\neg}3 \wedge b1)\ldots(b^{\neg} m{-}1 \wedge b1)b1$. (9) In [18], the number of bits required for the TC representations is considered to be the value of

the modulo. In this situation, B^{-} can be computed as follows: $B^{-} = b^{-}1b^{-}2b^{-}3 \ldots b^{-}m-1b^{-}m$. (10)

By considering the representation in (10), the complement of zero takes the value of the modulo, which does not cause any problem for the modular addition. However, the modular adder suggested in this paper only uses m − 1 bits, and thus bm is not used.

By using the distributed arithmetic approach [21], modular multiplication based on the TC can be done using modular adders. Therefore, an efficient structure for designing TC-based modular adders plays an important role to enhance the performance of RNS based on TC. Fig. 1 presents the TC-based modulo 7 adder architecture proposed in [18]. In this adder, one of the operands is converted from the TC to the binary format using a customized decoder, shown as s[1], s[2], and s[4] in Fig. 1 (where the indexes indicate the weight of the bit in the binary number system, e.g., s[1]+2s[2]+4s[4]). A number of 1s equal to the value of the first operand is added to the second operand (b[6] ... b[1]). If the number of 1s in the result exceed m −1, the m initial 1s are removed from the final result (r in Fig. 1).

For example, let us assume that B is equal to 6, represented in the TCR format 111111, and that S is equal to 5, represented in the binary format s[1] = 1, s[2] = 0, and s[4] = 1. According to Fig. 1, the s[1] forces the MUXs in the first level to shift one bit left b, and insert a 1 in the rightmost bit. Next, the MUXs in the second level transfer their inputs to the outputs without inserting 1s since s[2] = 0 (if s[2] was 1, then two 1s would be added to the right bits). Similarly, the MUXs in the third level perform 4-bit left shifting while inserting four 1s into the right bit positions. The result of this step is 011111111111 (c signals in Fig. 1). Since, in this example, the modulo is 7, c[7] plays a key role in modulo reduction and is used to select the inputs of the MUXs in the last level. If c[7] is equal to 1, the result exceeded m − 1, and thus, the value of the modulo should be subtracted from the result, i.e., c[8]–c[12] are forwarded to the output. Otherwise, c[7] = 0 and c[1]–c[6] will constitute the output (the result is less than the modulo value 7).

C. One-Hot Coding:

The OHC is usually used to address lookup tables (LUTs) and at the output of some linear circuits such as FIR filters [17]. K +1 bits are required to represent numbers between 0 and K in this coding. With OHC, only one bit takes the value of one and the others are zero. The value of the number in this coding is defined by the relative position of the bit with value “1.” Table II shows the numbers between 0 and 7 encoded in an OHC. The OHC requires one more bit than the TC. When the OHC is used to represent residues in RNS, it is named OHR [21]. The modular addition of two numbers A and B in this type of coding can be computed with shifts. To perform an addition, one operand should be circularly shifted a number of positions defined by the other operand. To perform (A − B) mod m, the complement of B should be added to A. The complement of B modulo m corresponds to $B^{-} = b1b2b3 \ldots bm-1b0$ (11) where bits are indexed from 0 to m − 1. As an example, the complement of 2 (0000100) modulo 7 is 5 (0100000), and the complement of 0 is zero. It is easily observed that B^{-} is calculated without inverters, by reordering the bits.

TABLE II
OHC EXAMPLE

Regular Representation	HOC
0	00000001
1	00000010
2	00000100
3	00001000
4	00010000
5	00100000
6	01000000
7	10000000

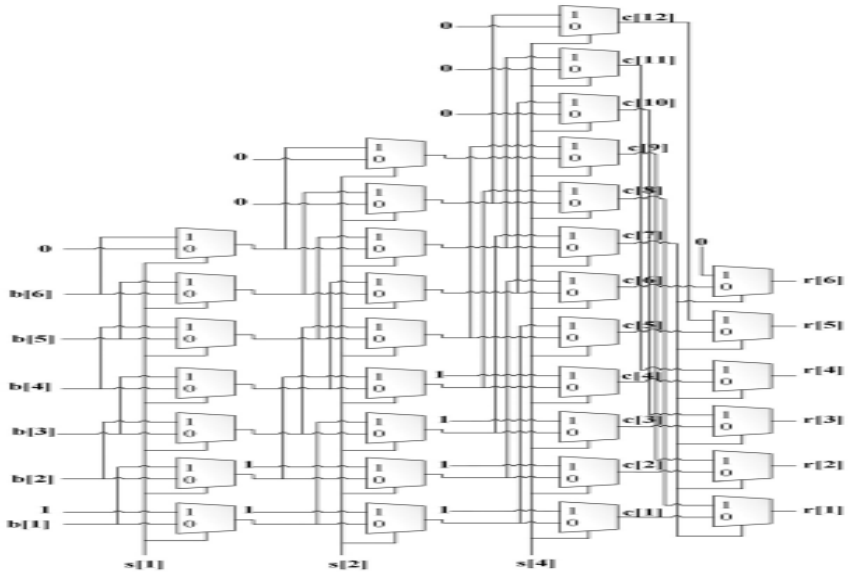


Fig. 1. Modulo-7 TCR-based adder

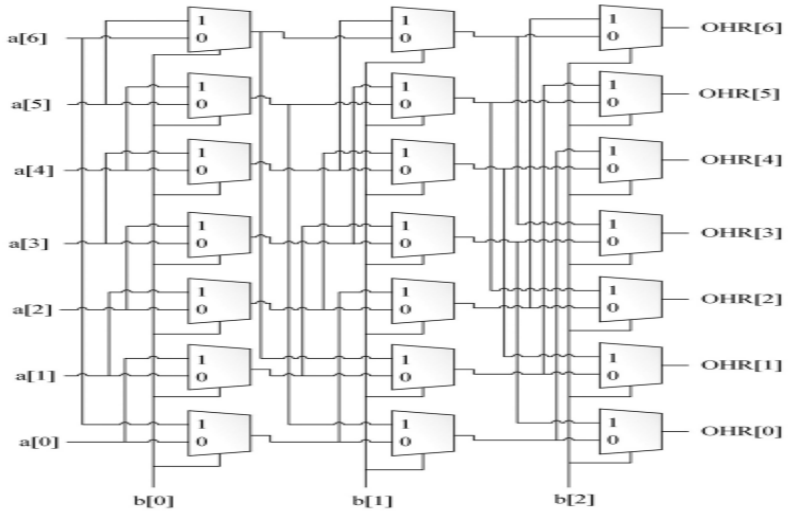


Fig. 2. OHR-based modulo-7 adder.

Fig. 2 presents an OHR modulo 7 adder [19]. The second operand, i.e., B, is in binary format, represented by b[2]b[1]b[0] in Fig. 2. These bits operate as selectors for the MUXs at the different levels. Modular addition for OHC is performed by just circularly left shifting one of the

operands a number of positions given by the value of the other. In Fig. 2, the operand A is circularly left shifted according to the value of B. Since B is in the binary format, if b0 = 1, then A will be one bit circularly shifted to the left. Similarly, if b1 and b2 are equal to 1, then a will be shifted left two and four positions, corresponding to the weights of the bits, respectively.

II.EXISTING SYSTEM AND PROPOSED SYSTEM

EXISTING METHOD

The OHC is usually used to address lookup tables (LUTs) and at the output of some linear circuits such as FIR filters [17]. K +1 bits are required to represent numbers between 0 and K in this coding. With OHC, only one bit takes the value of one and the others are zero. The value of the number in this coding is defined by the relative position of the bit with value "1." Table II shows the numbers between 0 and 7 encoded in an OHC. The OHC requires one more bit than the TC. When the OHC is used to represent residues in RNS, it is named OHR [21]. The modular addition of two numbers A and B in this type of coding can be computed with shifts. To perform an addition, one operand should be circularly shifted a number of positions defined by the other operand. To perform (A − B) mod m, the complement of B should be added to A. The complement of B modulo m corresponds to $B^{-} = b1b2b3 \ldots bm{-}1b0$ (11) where bits are indexed from 0 to m − 1. As an example, the complement of 2 (0000100) modulo 7 is 5 (0100000), and the complement of 0 is zero. It is easily observed that B^{-} is calculated without inverters, by reordering the bits.

TABLE II

OHC EXAMPLE

Regular Representation	HOC
0	00000001
1	00000010
2	00000100
3	00001000
4	00010000
5	00100000
6	01000000
7	10000000

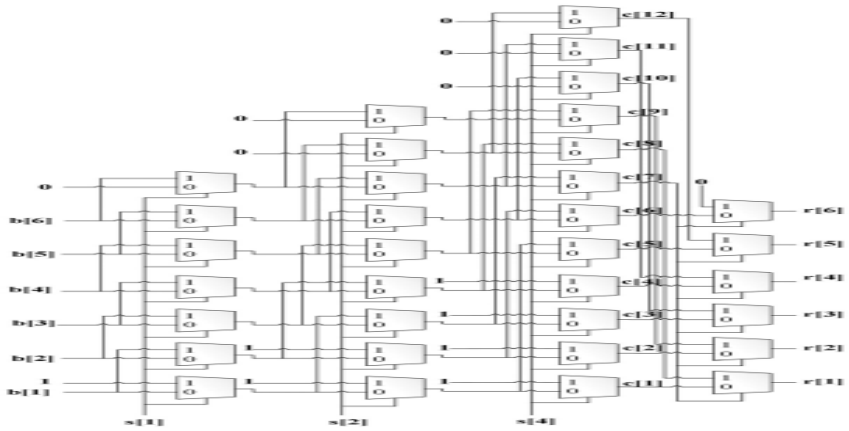


Fig. 1. Modulo-7 TCR-based adder

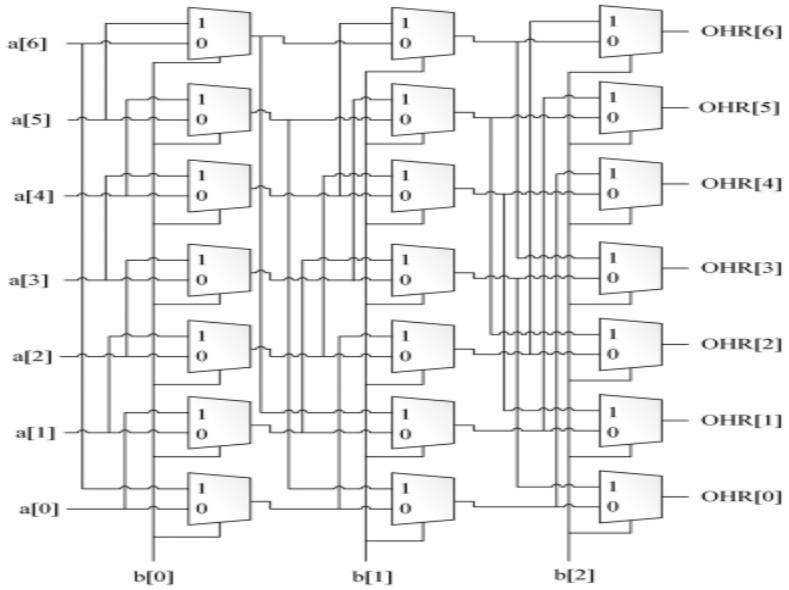


Fig. 2. OHR-based modulo-7 adder.

Fig. 2 presents an OHR modulo 7 adder [19]. The second operand, i.e., B, is in binary format, represented by b[2]b[1]b[0] in Fig. 2. These bits operate as selectors for the MUXs at the different levels. Modular addition for OHC is performed by just circularly left shifting one of the operands several positions given by the value of the other. In Fig. 2, the operand A is circularly left shifted according to the value of B. Since B is in the binary format, if b0 = 1, then A will be one bit circularly shifted to the left. Similarly, if b1 and b2 are equal to 1, then a will be shifted left two and four positions, corresponding to the weights of the bits, respectively.

2.PROPOSED SYSTEM

The proposed designs for OHR- and TCR-based modular adders are presented here. The proposed hardware structures for modular addition require less circuit area and less delay in comparison to the state of the art.

An important aspect to apply the proposed method to add two modulo m residues($0 \le A, B < m$) represented in TCR is to identify whether A + B >= m or not. With this aim, to m in the following situations: A + B < m, if k = 0 A + B = m, if k = 1 A + B > m, if k > 1. (21) Therefore, one just needs to apply the OR operation to all the outputs of the bitwise AND operation to check whether at least one of these outputs is 1 (meaning that the sum is equal to or greater than m) or all of them are 0 (meaning that the sum will be less than the modulo). Hence, (20) can be rewritten as A ∧ B reverse = (am−1∧b1)...(a2∧bm−2)(a1∧bm−1). (22) To detect the existence of at least a bit 1 among the resulting bits in (22), the following formulation can be adopted: cl = (am−1 ∧ b1) ∨ ... ∨ (a2 ∧ bm−2) ∨ (a1 ∧ bm−1). (23)

Hence, (13) is achieved, i.e., cl = 1 and cl = 0 mean that A + B ≥ m and A + B < m, respectively. Equation (20) can also be used to obtain the result of the modular addition whenever A + B ≥ m. k determines the number of 1s that are in excess of the m − 1 bits. Since the sum of the regular addition of A and B has to be reduced by m, this can be achieved by considering the overlapping 1 bits minus one as the result is

$$\mathrm{SUM}_1 = \underbrace{0\ldots0}_{b\_\text{zeros}}\underbrace{0\ldots0}_{a\ \text{zeros}}\underbrace{1\ldots1}_{k-1}. \qquad (24)$$

When k = 1, A + B is equal to the modulo m. The number of bits with the value 1 in (22) is equal to the number of 1s of the modular addition plus one. There is no situation where two nonsequential bits in (22) become one, whereas between them there are bits with the zero value. Therefore, the correct modular addition for k = 1 is 0. For the other cases, wherein k > 1, just k−1 1s should be placed in the final TCR sum. Hence, the number of overlapping 1 bits in (22) should be counted using the formula $\sum_{i=1}^{m-1} a_i \wedge b_{m-i}$, and then decreased by one to achieve the final sum, corresponding to SUM1 in (14).

For the case A + B < m, the number of bits with the zero value is the key for performing the modular addition. As in the previous condition, the bit-reversed representation of B is considered and, on the top of that, the bitwise OR operation with A is applied is

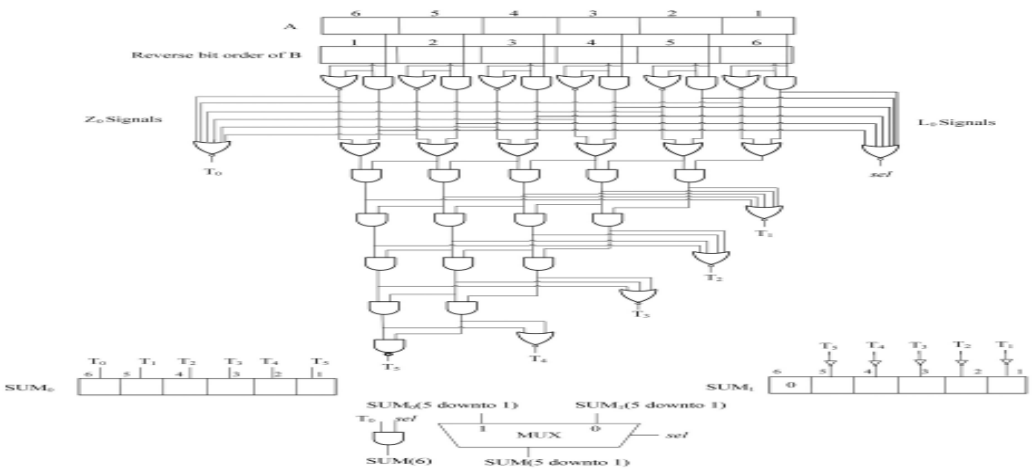
$$A = \underbrace{0\ldots0}_{a_{\text{zeros}}-k}\underbrace{0\ldots0}_{k}\underbrace{1\ldots1}_{a_{\text{ones}}} \qquad (25)$$

$$B_{\text{reverse}} = \underbrace{1\ldots1}_{b_{\text{ones}}}\underbrace{0\ldots0}_{k}\underbrace{0\ldots0}_{b_{\text{zeros}}-k} \qquad (26)$$

$$A \vee B_{\text{reverse}} = \underbrace{1\ldots1}_{b_{\text{ones}}}\underbrace{0\ldots0}_{k}\underbrace{0\ldots0}_{a_{\text{ones}}}. \qquad (27)$$

The sum in the case $A + B < m$ will have $(a_{\text{ones}} + b_{\text{ones}})$ 1 bits together with k bits of 0, that is

$$\mathrm{SUM}_0 = \underbrace{0\ldots0}_{k}\underbrace{1\ldots1}_{b_{\text{ones}}}\underbrace{1\ldots1}_{a_{\text{ones}}}. \qquad (28)$$



A Ternary Carry Ripple (TCR) modulo-7 adder is a digital circuit that performs addition in ternary (base-3) arithmetic and outputs the result modulo-7. In ternary addition, each digit can take one of three values: 0, 1, or 2. The modulo-7 operation ensures that the result stays within the range of 0 to 6.

Here's a simple TCR-based modulo-7 adder using a ripple carry mechanism:

Ternary Carry Ripple Modulo-7 Adder:

Input Digits (A, B):

The input digits are ternary digits, and in this example, let's use A and B as the two ternary numbers to be added.

Ternary Full Adder:

Design a ternary full adder that takes two ternary inputs (A and B) along with a ternary carry-in (Cin). The full adder produces a ternary sum (S) and a ternary carry-out (Cout).

Ripple Carry Mechanism:

Connect multiple instances of the ternary full adder in a ripple carry configuration. The carry-out from each stage is fed as the carry-in to the next stage.

Modulo-7 Operation:

After obtaining the sum from the last stage of the adder, apply the modulo-7 operation to ensure that the result is within the range of 0 to 6.

Output:

The output of the TCR-based modulo-7 adder is the modulo-7 sum of the input ternary digits.

Ternary Full Adder Logic:

The logic for a ternary full adder is more complex than a binary full adder due to the additional ternary digit values. Here's a simplified representation of the logic for a ternary full adder:

Inputs: A, B, Cin (Ternary inputs)

Outputs: S, Cout (Ternary sum and carry-out)

S = (A + B + Cin) % 3 // Ternary addition with modulo-3 operation

Cout = (A + B + Cin) / 3 // Ternary division (carry-out)

Example:

Let's consider an example where A = 201 (in ternary) and B = 110 (in ternary):

Ternary Addition:

Use the ternary full adder to perform the addition, considering the ternary carry-in from the previous stage (initial carry-in is 0):

```
   201
 + 110
 ------
  0210   (Ternary sum)
```

Ripple Carry:

Propagate the carry from one stage to the next in a ripple carry fashion.

Modulo-7 Operation:

Apply the modulo-7 operation to the final sum:

0210 (Ternary sum) mod 7 = 6

So, the result of adding 201 and 110 in ternary and taking the modulo-7 is 6.

III. RESULTS AND ANALYSIS DISCUSSION

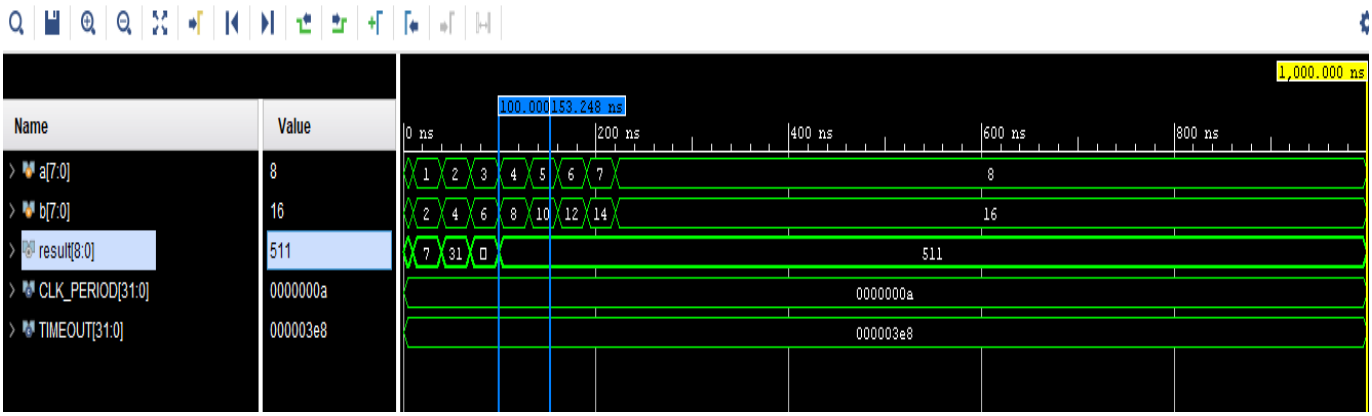


FIG.1.SIMULATION OUTPUT

```
Starting static elaboration
Completed static elaboration
INFO: [XSIM 43-4323] No Change in HDL. Linking previously generated obj files to create kernel
INFO: [USF-XSim-69] 'elaborate' step finished in '3' seconds
Vivado Simulator 2018.1
Time resolution is 1 ps
Input: a =   1, b =   2
Result:   7
Input: a =   2, b =   4
Result:  31
Input: a =   3, b =   6
Result: 127
Input: a =   4, b =   8
Result: 511
Input: a =   5, b =  10
Result: 511
Input: a =   6, b =  12
Result: 511
Input: a =   7, b =  14
Result: 511
Input: a =   8, b =  16
Result: 511
relaunch_sim: Time (s): cpu = 00:00:01 ; elapsed = 00:00:12 . Memory (MB): peak = 2283.066 ; gain = 0.000
```

FIG.2.OUTPUT

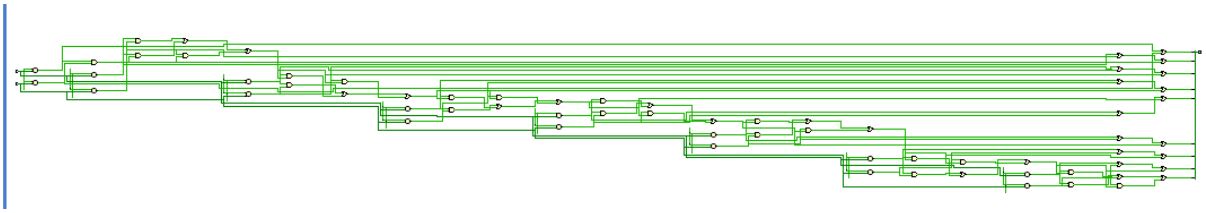


FIG3.RTL SCHEMATIC DIAGRAM

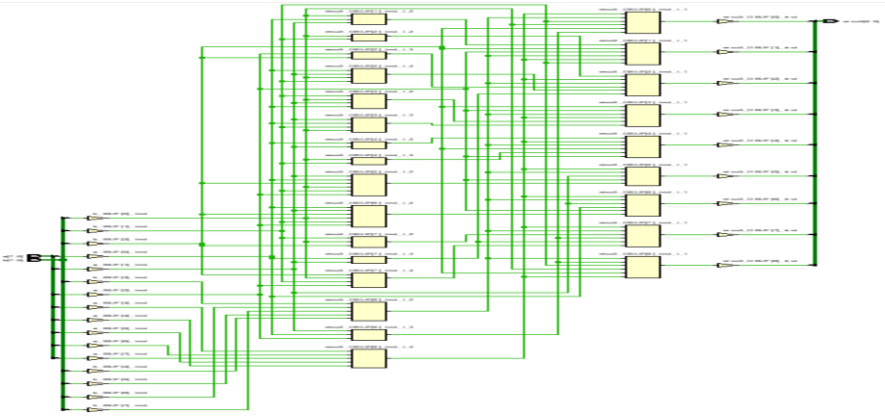


FIG.4.SYNTHESIS DESIGN

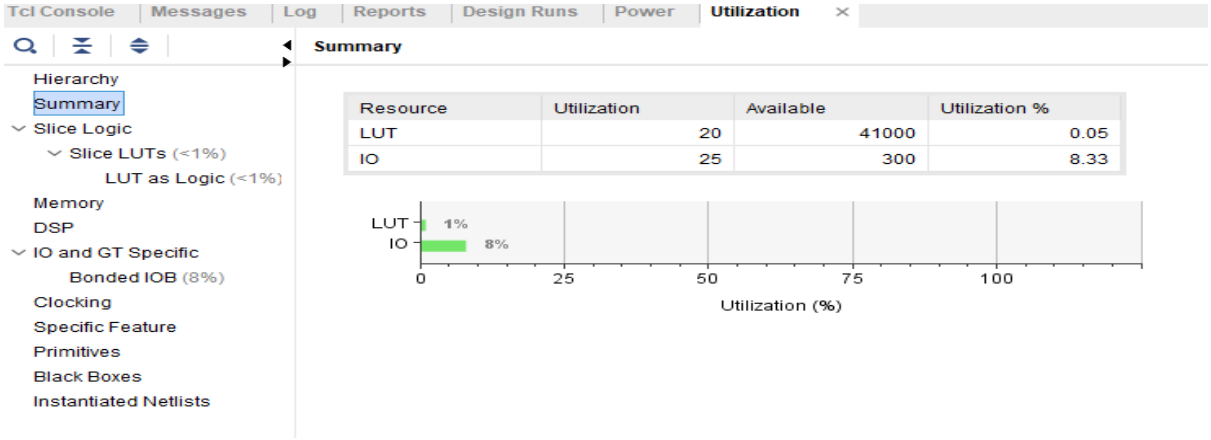


FIG.5.UTILIZATION OF LUTs AND FFs

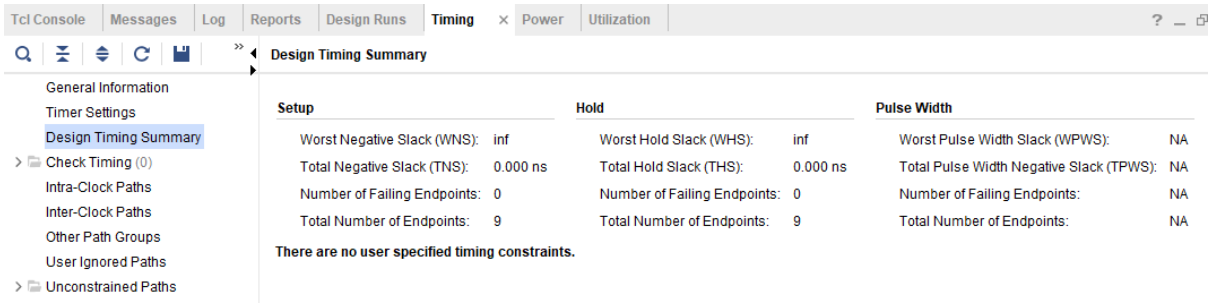


FIG.6.TIMING REPORT

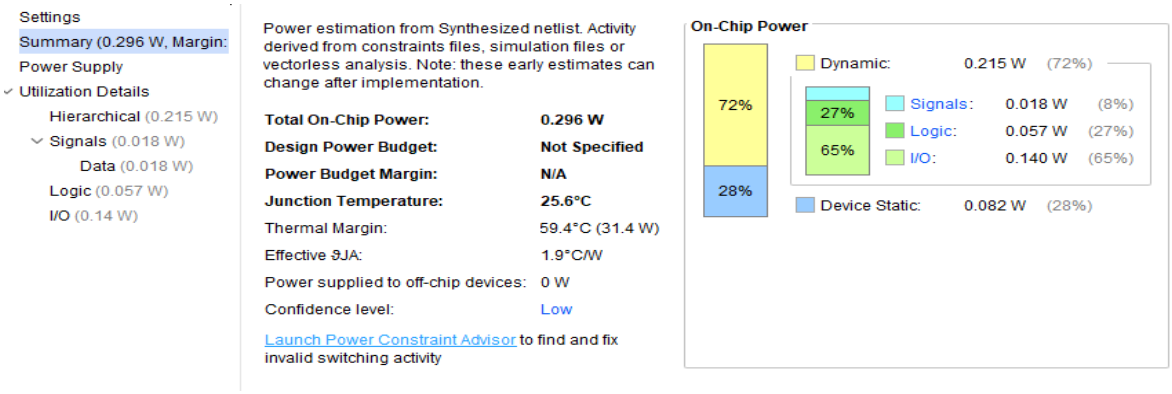


FIG.7.POWER REPORT

CONCLUSION

In this project, two new classes of efficient modular adders are proposed for TC and OHC. The main advantages of the proposed adders are their high performance and low cost, making them useful, for example, for RNSs based on small moduli sets, used for DSP embedded systems and IoT. For the first time, the conventional MUX-based design of OHC and TC adders is replaced by a novel approach based on a small number of logical gates. Since TC and OHC modular adders do not require carry propagation, their structures for small moduli become simpler and more efficient and have lower delay than binary modular adders (except for moduli with the shape of 2n). Performance analyses and experimental results have shown the significant impact

of these improvements. Moreover, the formulation and architectures introduced in this paper are easily extended to design other units for modular arithmetic, such as subtractors.

REFERENCES

[1] A. S. Molahosseini, L. Sousa and C. H. Chang, (Eds.), Embedded Systems Design with Special Arithmetic and Number Systems. New York, NY, USA: Springer, 2017.

[2] Y. H. Chen, T. Krishna, J. S. Emer, and V. Sze, "Eyeriss: An energyefficient reconfigurable accelerator for deep convolutional neural networks," IEEE J. Solid-State Circuits, vol. 52, no. 1, pp. 127–138, Jan. 2017.

[3] M. Alioto (Ed.), Enabling the Internet of Things: From Integrated Circuits to Integrated Systems. New York, NY, USA: Springer, 2017.

[4] P. V. A. Mohan, Residue Number Systems: Theory and Applications. New York, NY, USA: Springer, 2016.

[5] C.-H. Chang, A. S. Molahosseini, A. A. E. Zarandi, and T. F. Tay, "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. 4, pp. 26–44, 4th Quart., 2015.

[6] L. Sousa, S. Antão, and P. Martins, "Combining residue arithmetic to design efficient cryptographic circuits and systems," IEEE Circuits Syst. Mag., vol. 16, no. 4, pp. 6–32, 4th Quart., 2016.

[7] M. Labafniya and M. Eshghi, "An efficient adder/subtracter circuit for one-hot residue number system," in Proc. Intl. Conf. Electron. Devices, Syst. Appl. (ICEDSA), Apr. 2010, pp. 121–124.

[8] M. Hosseinzadeh, S. JafaraliJassbi, and K. Navi, "A novel multiple valued logic OHRNS adder circuit for modulo (RN-1)," in Proc. 4th Int. Conf. Adv. Eng. Comput. Appl. Sci., Aug. 2010, pp. 166–170.

[9] D. K. Taleshmekaeil, A. Safari, and Y. Kong, "Using one hot residue number system(OHRNS) for digital image processing," in Proc. 16th CSI Int. Symp. Artif. Intell. Signal Process., May 2012, pp. 64–67.

[10] W. A. Chren, Jr., "One-hot residue coding for low delay-power product CMOS design," IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process., vol. 45, no. 3, pp. 303–313, Mar. 1998.

[11] H. V. Jayashree, J. Vuggirala, and G. N. Patil, "Design of high speed area efficient OHRNS data path subsystems for FFT implementation," in Proc. 4th Int. Conf. Electron. Commun. Syst. (ICECS), Feb. 2017, pp. 148–152.

[12] S. Kak, "Generalized Unary Coding," Circuits, Syst. Signal Process., vol. 35, no. 4, pp. 1419–1426, Apr. 2016.

[13] S. W. Golomb, "Run-length encodings (Corresp.)," IEEE Trans. Inf. Theory, vol. 12, no. 3, pp. 399–401, Jul. 1966.

[14] R. F. Rice and R. Plaunt, "Adaptive variable-length coding for efficient compression of spacecraft television data," IEEE Trans. Commun. Technol., vol. COMT-19, no. 6, pp. 889–897, Dec. 1971.

[15] S. Jayashri and P. Saranya, "Reconfigurable FIR filter using distributed arithmetic residue number system algorithm based on thermometer coding," in Proc. Int. Conf. Commun. Signal Process., Apr. 2014, pp. 1991–1995.

[16] C. H. Vun, A. B. Premkumar, and W. Zhang, "Sum of products: Computation using modular thermometer codes," in Proc. Int. Symp. Intell. Signal Process. Commun. Syst. (ISPACS), Dec. 2014, pp. 141–146.

[17] C. H. Vun and A. B. Premkumar, "RNS encoding based folding ADC," in Proc. IEEE Int. Symp. Circuits Syst., May 2012, pp. 814–817.

[18] C. H. Vun and A. Premkumar, "Thermometer code based modular arithmetic," in Proc. Spring Congr. Eng. Technol., May 2012, pp. 534–538.

[19] C. H. Vun, A. B. Premkumar, and W. Zhang, "A new RNS based DA approach for inner product computation," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 60, no. 8, pp. 2139–2152, Aug. 2013.

[20] K. Navi, A. S. Molahosseini, and M. Esmaeildoust, "How to teach residue number system to computer scientists and engineers," IEEE Trans. Edu., vol. 54, no. 1, pp. 156–163, Feb. 2011.

[21] S. A. White, "Applications of distributed arithmetic to digital signal processing: A tutorial review," IEEE ASSP Mag., vol. 6, no. 3, pp. 4–19, Jul. 1989.

[22] R. Zimmermann, "Binary adder architectures for cell-based VLSI and their synthesis," Ph.D. dissertation, Dept. Inf. Technol. Elect. Eng., ETH Zürich, Zürich, Switzerland, 1997.

[23] A. S. Molahosseini, K. Navi, O. Hashemipour, and A. Jalali, "An efficient architecture for designing reverse converters based on a general three-moduli set," J. Syst. Archit., vol. 54, no. 10, pp. 929–934, 2008.

[24] N. H. E. Weste and D. M. Harris, CMOS VLSI Design: A Circuits and Systems Perspective, 4th Ed. Reading, MA, USA: Addison-Wesley, 2011.

[25] X. Chen and N. A. Touba, "Fundamentals of CMOS Design," in Electronic Design Automation, L. T. Wang, Y. W. Chang and K. T. Cheng (Eds.), Burlington,, MA, USA: Morgan Kaufmann, 2009. Chap. 2.