Efficient Modular Adder Designs Based on Thermometer and One-Hot Coding

Fereshteh Jafarzadehpour, Amir Sabbagh Molahosseini, *Senior Member, IEEE*, Azadeh Alsadat Emrani Zarandi, *Member, IEEE*, and Leonel Sousa, *Senior Member, IEEE*

***Abstract*— Residue number systems (RNSs) are efficient alternatives to positional number systems, providing fast and power-efficient computational systems. The key feature of the RNS benefitting modern embedded systems and the Internet-of-Thing (IoT) edge devices is its energy efficiency. Modular addition is the most important and frequent operation applied on the components of RNS, including arithmetic units in the channels as well as forward and reverse converters. The small and medium dynamic range requirements of low-power embedded and edge devices make the usage of the thermometer coding (TC) and one-hot coding (OHC) viable, reducing the power consumption and improving the energy efficiency of modulo addition in comparison to regular binary representations. Based on these techniques, this paper presents two new energy-efficient modular adders, which, due to the carry-free internal computations, are also highly performing. The proposed modular adders based on the TC and OHC result in average improvements of 38% and 34.5% for the delay, 27% and 14.5% for the circuit area, 29.5% and 6.3% for energy consumption, and about 54.9% and 44.2% for the area-delay product (ADP), respectively, in comparison with the related state of the art.**

***Index Terms*— Computer arithmetic, modular addition, one-hot coding (OHC), residue number system (RNS), thermometer coding (TC).**

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], in particular, have been used for DSP [5] and cryptography [6], supporting high-speed, 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 (with the exception of moduli like 2^n). Moreover, the usage of the TC format to represent residues removes the need for forward converters, whenever inputs values are coded using analog-to-residue 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

Manuscript received December 22, 2018; revised April 22, 2019; accepted May 22, 2019. Date of publication June 27, 2019; date of current version August 23, 2019. This work was supported in part by the Portuguese funds through Fundação para a Ciência e a Tecnologia (FCT) under Grant UID/CEC/50021/2019. (*Corresponding author: Amir Sabbagh Molahosseini.*)

F. Jafarzadehpour and A. Sabbagh Molahosseini are with the Department of Computer Engineering, Kerman Branch, Islamic Azad University, Kerman 7635131167, Iran (e-mail: f.jafarzadeh@iauk.ac.ir; amir@iauk.ac.ir).

A. A. Emrani Zarandi is with the Department of Computer Engineering, Shahid Bahonar University of Kerman, Kerman 7616913439, Iran (e-mail: a.emrani@uk.ac.ir).

L. Sousa is with the R&D Instituto de Engenharia de Sistemas eComputadores (INESC-ID), Instituto Superior Técnico (IST), University of Lisbon, Lisbon 1000-029, Portugal (e-mail: Leonel.sousa@ist.ul.pt).

Digital Object Identifier 10.1109/TVLSI.2019.2919609

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 2^n channel exploiting binary representations.

This paper presents 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 a large number of multiplexers (MUXs), the herein proposed adders have significant practical experimental improvements regarding the latency, area, and energy consumption.

The remaining of this paper is organized as follows. First, the RNS and its associated modular arithmetic are introduced, and then the TC and OHC together with the corresponding state-of-the-art modular adder architectures are briefly reviewed in Section II. Then, the new structures for modular addition based on OHC and TC are proposed in Section III and implemented and compared with previous works in Section IV. Finally, conclusions are drawn in Section V.

II. Background

In this section, the background on RNS, OHC, and TC is introduced.

A. Residue Number Systems

An RNS supported on a moduli-set of n pairwise relatively prime numbers $\{m_1, m_2, \ldots, m_n\}$ allows for the integers in $[0, M-1]$, with $M = m_1 \times m_2 \times \ldots \times m_n$, to be uniquely represented by their remainders modulo $m_1, m_2, \ldots, m_n$.

A *forward conversion* maps the binary weighted representation of a number X onto the residues $(x_1, x_2, \ldots, x_n)$ associated with the moduli in the set

$$x_i = |X|_{m_i} = |X_{MB-1} \ldots X_1 X_0|_{m_i}, \quad i = 1 \ldots n \tag{1}$$

where $MB = \lceil \log_2 m_i \rceil$ [4].

Considering two RNS numbers A and B

$$A = (a_1, a_2, \ldots, a_n) \tag{2}$$

$$B = (b_1, b_2, \ldots, b_n). \tag{3}$$

The *modular addition* of these two RNS numbers can be performed as follows [20]:

$$S = A + B = (s_1, s_2, \ldots, s_n) \tag{4}$$

with

$$s_i = |a_i + b_i|_{m_i} = \begin{cases} a_i + b_i, & \text{if } a_i + b_i < m_i \\ a_i + b_i - m_i, & \text{if } a_i + b_i \geq m_i. \end{cases} \tag{5}$$

The *reverse conversion* maps the residues in the RNS domain $(x_1, x_2, \ldots, x_n)$ into their equivalent weighted binary representations, using a method like the mixed-radix conversion (MRC) [4]

$$X = v_n \prod_{i=1}^{n-1} m_i + \ldots + v_3 m_2 m_1 + v_2 m_1 + v_1 \tag{6}$$

TABLE I
TC Example

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

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

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

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).

In [18], it is stated that the number bits required to represent all the remainders modulo m_i are m_i. However, it should be mentioned that, since the leftmost bit is always zero, these numbers can, in fact, be represented with only $m_i - 1$ bits.

In order 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 m_i, the sum should be decreased by m_i. 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:

$$\bar{B} = \bar{b}_2 \bar{b}_3 \bar{b}_4 \ldots \bar{b}_{m-1}\ 1, \quad \text{if } (B \neq 00\ldots0). \tag{8}$$

Herein, as in [18], the starting index of the bits is considered to be 1, i.e., b_1. In other words, the rightmost bit, b_1, 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, $\bar{B}$, 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 ($b_1 = 0$). In general, the complement of B can be calculated as

$$\bar{B} = (\bar{b}_2 \wedge b_1)(\bar{b}_3 \wedge b_1) \ldots (\bar{b}_{m-1} \wedge b_1) b_1. \tag{9}$$

In [18], the number of bits required for the TC representations is considered to be the value of the modulo. In this situation, $\bar{B}$ can be computed as follows:

$$\bar{B} = \bar{b}_1 \bar{b}_2 \bar{b}_3 \ldots \bar{b}_{m-1} \bar{b}_m. \tag{10}$$

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 b_m 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).

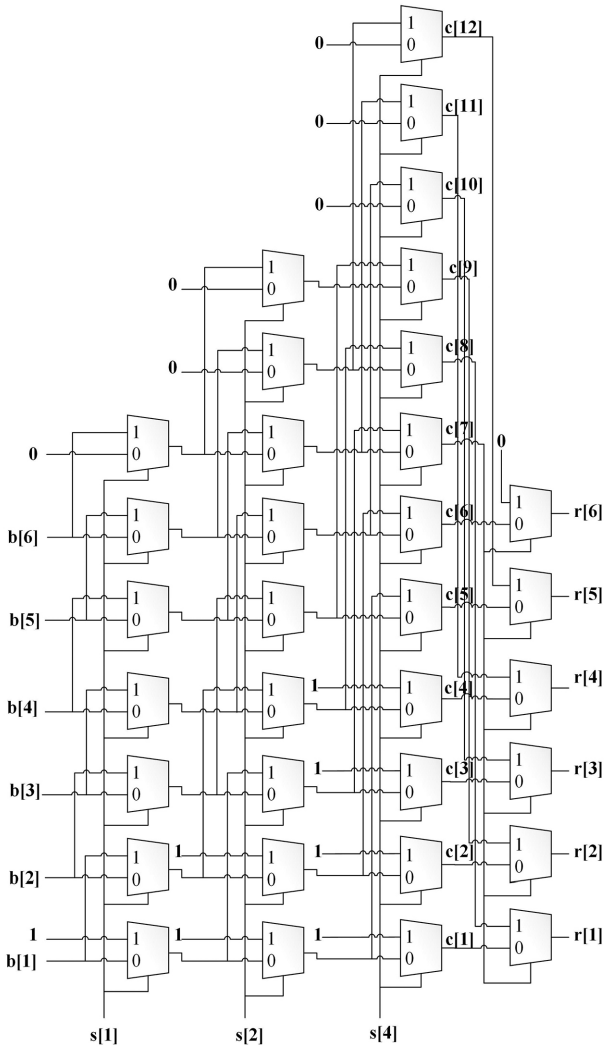

Fig. 1. Modulo-7 TCR-based adder [18].

TABLE II
OHC EXAMPLE

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

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

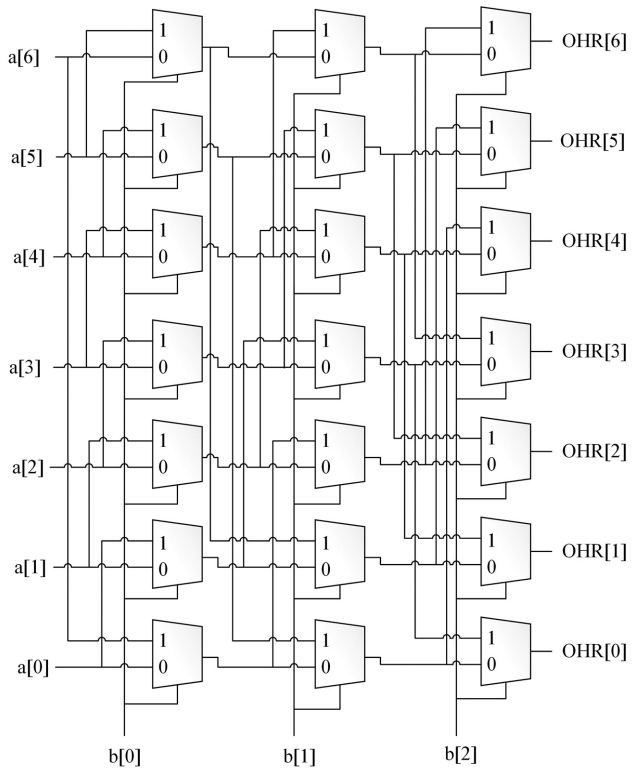

Fig. 2. OHR-based modulo-7 adder [19].

of B modulo m corresponds to

$$\bar{B} = b_1 b_2 b_3 \ldots b_{m-1} b_0 \tag{11}$$

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 $\bar{B}$ is calculated without inverters, by reordering the bits.

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 $b_0 = 1$, then A will be one bit circularly shifted to the left. Similarly, if b_1 and b_2 are equal to 1, then a will be shifted left two and four positions, corresponding to the weights of the bits, respectively.

III. Proposed Modular Adders

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

A. Thermometer-Based Modular Adder

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, and also for computing the sum, the order of the bits of B is reversed, which means the rightmost bit becomes the most left bit, and so on. After that, bitwise AND and NOR logic operations are applied to the inputs A and B. If any bit of the output of AND gates is 1, then $A + B$ is equal to or greater than the modulo. More concretely, if exactly one bit of these outputs becomes 1, this means that $A + B = m$ and the result should be 0. When two bits of the outputs are 1, the result becomes 1, and this process is continued whenever more bits of the output take the value 1. The outputs of the NOR gates are used to compute the result of the addition when $A + B < m$, as it will be observed in Lemma 1.

Lemma 1: Consider two TCR numbers, A and B of $m-1$ bits. The condition $A + B \ge m$ is verified with the bit cl, and the result of $A + B$ *mod* m represented with $m-1$ bits is computed with the following relations:

$$\text{Sum} = \begin{cases} \text{SUM}_1, & \text{if } cl = 1 (A + B \ge m) \\ \text{SUM}_0, & \text{if } cl = 0 (A + B < m) \end{cases} \tag{12}$$

where

$$cl = \vee_{i=1}^{m-1} (a_i \wedge b_{m-i}) \tag{13}$$

$$\text{SUM}_1 = \left(\sum_{i=1}^{m-1} a_i \wedge b_{m-i} \right) - 1 \tag{14}$$

$$\text{SUM}_0 = (m-1) - \left(\sum_{i=1}^{m-1} \overline{a_i \vee b_{m-i}} \right). \tag{15}$$

Proof: Both A and B are the TCR numbers in arithmetic modulo m, which means that $m-1$ bits are required to represent each

$$A = a_{m-1} \ldots a_2 a_1 = \underbrace{0 \ldots 0}_{a_{\text{zeros}}} \underbrace{1 \ldots 1}_{a_{\text{ones}}} \tag{16}$$

$$B = b_{m-1} \ldots b_2 b_1 = \underbrace{0 \ldots 0}_{b_{\text{zeros}}} \underbrace{1 \ldots 1}_{b_{\text{ones}}}. \tag{17}$$

In (16) and (17), the total number of bits, i.e., $a_{\text{zeros}} + a_{\text{ones}}$ or $b_{\text{zeros}} + b_{\text{ones}}$, is equal to $m-1$. Hence, if the total number of 1's in both A and B is greater than or equal to the modulo, m, there will not be enough space in the result to store them all. In other words, the second operand can only accept as many bits with the value of 1 from the first operand as its number of zero bits. Otherwise, the sum will be equal to or greater than the modulo. In order to detect this situation, the order of the bits of the second operand can be reversed, and then a bitwise AND operation is performed, i.e., $a_i \wedge b_{m-i}$

$$A = \underbrace{0 \ldots 0}_{a\_\text{zeros}} \underbrace{1 \ldots 1}_{k} \underbrace{1 \ldots 1}_{a_{\text{ones}} - k} \tag{18}$$

$$B_{\text{reverse}} = \underbrace{1 \ldots 1}_{b_{\text{ones}} - k} \underbrace{1 \ldots 1}_{k} \underbrace{0 \ldots 0}_{b_{\text{zeros}}} \tag{19}$$

$$A \wedge B_{\text{reverse}} = \underbrace{0 \ldots 0}_{a\_\text{zeros}} \underbrace{1 \ldots 1}_{k} \underbrace{0 \ldots 0}_{b\_\text{zeros}}. \tag{20}$$

The overlap between the 1 bits of A and the reversed version of B means that the addition of A and B exceeds or is equal

to m in the following situations:

$$\begin{cases} A+B<m, & \text{if } k=0 \\ A+B=m, & \text{if } k=1 \\ A+B>m, & \text{if } k>1. \end{cases} \tag{21}$$

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

$$A \wedge B_{\text{reverse}} = (a_{m-1} \wedge b_1) \ldots (a_2 \wedge b_{m-2})(a_1 \wedge b_{m-1}). \tag{22}$$

To detect the existence of at least a bit 1 among the resulting bits in (22), the following formulation can be adopted:

$$cl = (a_{m-1} \wedge b_1) \vee \ldots \vee (a_2 \wedge b_{m-2}) \vee (a_1 \wedge b_{m-1}). \tag{23}$$

Hence, (13) is achieved, i.e., $cl = 1$ and $cl = 0$ mean that $A + B \geq m$ and $A + B < m$, respectively.

Equation (20) can also be used to obtain the result of the modular addition whenever $A + B \geq m$. 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

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

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 SUM_1 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

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

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

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

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

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

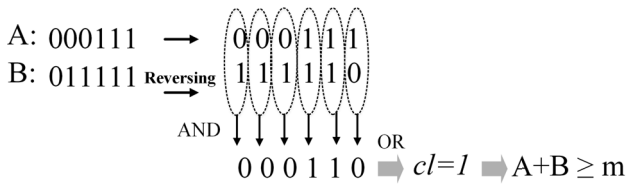

Fig. 3. Thermometer addition example when $A + B \geq m$.

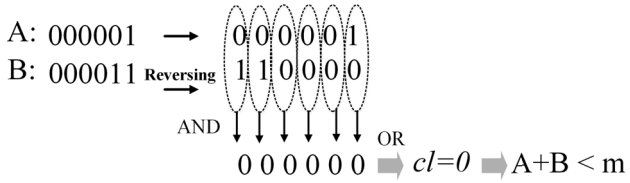

Fig. 4. Thermometer addition example when $A + B < m$.

Alternatively, the number of zero bits in SUM_0 can be achieved by counting the number of overlapping zero bits between A and the reversed version of B using the bitwise NOR operation $(\sum_{i=1}^{m-1} \overline{a_i \vee b_{m-i}})$, which can be represented as

$$\overline{A \vee B_{\text{reverse}}} = \underbrace{0\ldots0}_{b_{\text{ones}}}\underbrace{1\ldots1}_{k}\underbrace{0\ldots0}_{a_{\text{ones}}}. \tag{29}$$

Then, the subtraction $(m-1) - (\sum_{i=1}^{m-1} \overline{a \vee b_{m-i}})$ is computed to achieve the number of 1s in the final sum, leading to (15).

Example 1: Let us assume that $A = 3$ and $B = 5$ and compute $|3 + 5|_7$. Both numbers are represented in TCR in Fig. 3. It can be seen that $(a_2 \wedge b_5)$ and $(a_3 \wedge b_4)$ are equal to one, which means that $cl = 1$ and $A + B \geq m$. To achieve the sum, according to (14)

$$\text{SUM}_1 = \left(\sum_{i=1}^{6} a_i \wedge b_{m-i}\right) - 1 = 2 - 1 = 1.$$

According to (24), the TCR representation of the result is

$$\text{SUM}_1 = 000001$$

which is the correct result ($|3 + 5|_7 = 1$).

Example 2: Let us assume that $A = 1$ and $B = 2$. Both numbers are represented in TCR in Fig. 4. It can be seen that none of the outputs of the AND gates takes the value "1." This means $cl = 0$ and leads to the conclusion that $A + B < m$. According to (15), the sum is

$$\text{SUM}_0 = (7-1) - \left(\sum_{i=1}^{6} a_i \vee \bar{b}_{m-i}\right) = 6 - 3 = 3.$$

According to (28), the TCR representation of the result is

$$\text{SUM}_0 = 000111$$

which is the correct result ($|1 + 2|_7 = 3$).

The proposed design for the TCR-based modulo adder is shown in Fig. 5, for the case when $m = 7$. When $A+B >= m$, the result is stored in SUM_1, while for the other case,

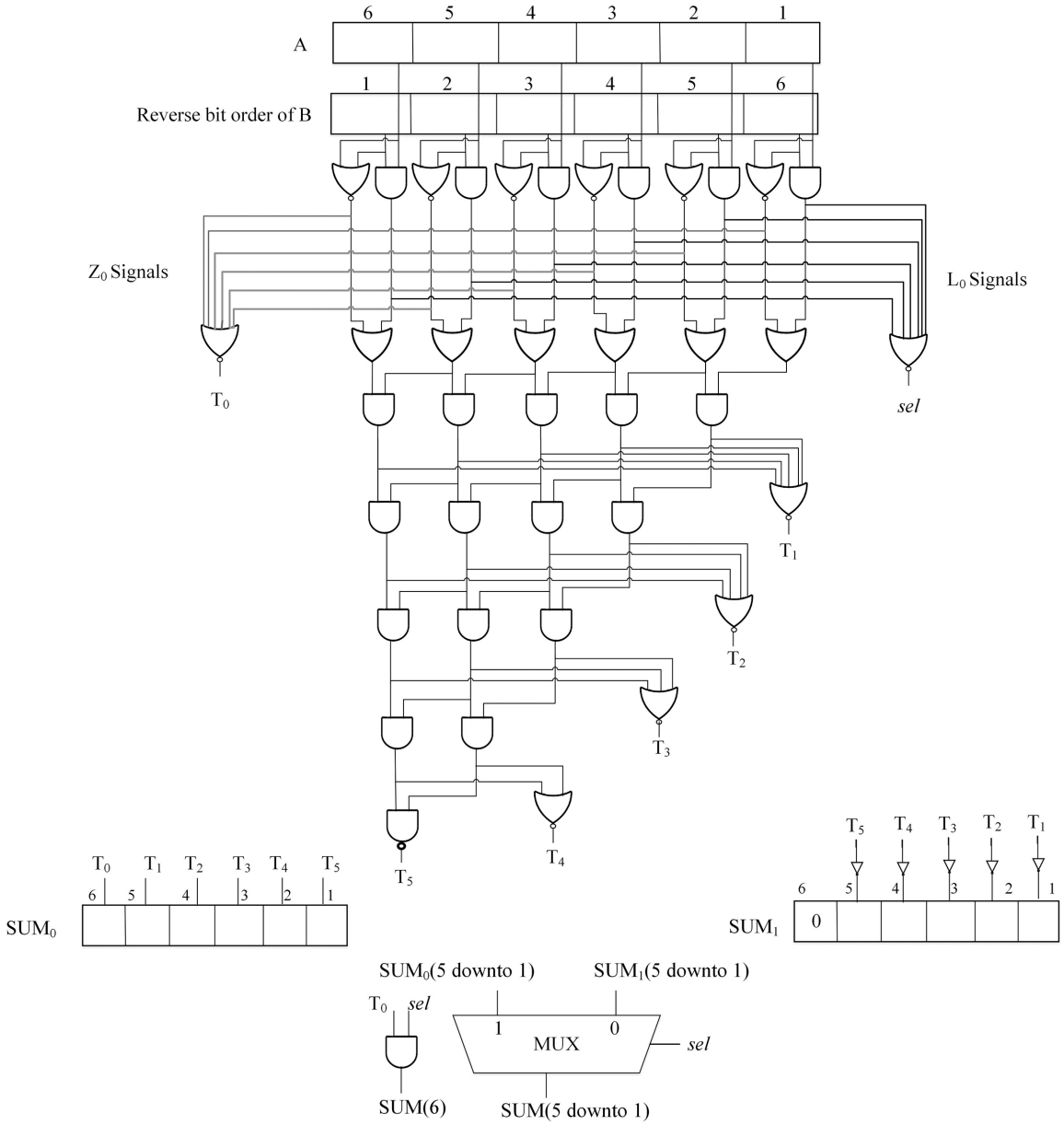

Fig. 5. Proposed TCR-based modulo-7 adder.

$A + B < m$, the result is placed in SUM_0. As mentioned before, if at least one of the ANDs' output bits in the first level gets the value 1, the result of the modular addition of A and B is equal to or greater than m. Otherwise, the result is less than m. L_0 signals are connected to the NOR gate with six inputs. Based on the output of this gate (*sel*), SUM_0 or SUM_1 is selected (*sel* is the complement of*cl*). It should be noted that some multi-input gates in Fig. 5 can be implemented using the tree structures of 2-input gates without impacting the delay. Let us analyze the operation of the circuit to compute SUM_0 and SUM_1.

1) SUM_0 *Circuit:* As observed in Fig. 5, B, with the bits in the reverse order, and A are the inputs of the NOR and AND gates in the first level. When $A + B < m$, the output of all the AND gates in the first level becomes 0, and the number of output bits of the NOR gates with value "1" is used to identify the number of zeros in SUM_0. Therefore, if at least one of the Z_0 signals becomes one, the number of zeros in the result is also at least one. Since, with TC, 0s are placed in the bits located on the left-hand side, the value of the left bit of SUM_0 is equal to the T_0 signal.

If at least the output of two NOR gates of the first level becomes one, the number of bits with 0 in the result will be at least two. As it was mentioned before, if the output of more than one NOR gate becomes one, these 1 bits are located sequentially. Therefore, the AND gates can detect whenever there are two sequential bits with the value 1. The output of the NOR gate with five inputs (T_1) specifies the 5th bit of the SUM_0. In the same way, if at least three signals of Z_0 become one, this means that there are at least 3 bits of the result with

the value 0, and so on. Finally, if T_5, the output of the NAND gate, becomes "0," which means that the outputs of all the NOR gates in the first level are "1," all the 6 bits of the sum are "0."

2) SUM_1 Circuit: Whenever $A+B >= m$, the result of the modular addition of A and B is SUM_1. The output of all the NOR gates in the first level becomes 0, and the outputs of the AND gates in the first level are used to calculate SUM_1. If exactly one of the outputs of the AND gates becomes one, the result of $A+B$ is equal to the modulo m and the sum takes the value of zero. As it was mentioned before, if $A+B >= m$, the difference between the number of bits of the result with the value 1 and the number of AND gates with output 1 is one. Hence, if at least two L_0 signals have the value 1, there is at least one bit of the result with the value "1." In this situation, the output of the NOR gate with five inputs (T_1) takes the value zero. Since, in TC, ones are placed sequentially at the right-hand side, the value of the rightmost bit of SUM_1 is the complemented value of T_1. Similarly, having at least three signals of L_0 with the value "1" results in the least two bits taking the value "1," and so on.

Whenever the result of $A + B$ becomes equal to or greater than m, the result has at least one bit with the value zero, placed on the left-hand side. In order to add two modulo m integers A and B, the addition result is in the range 0 to $2m-2$. When $A+B \geq m$, the result of the addition takes the maximum value of $m-2$ and there is at least one zero bit. It should be noted that the six L_0 signals are used to compute the MUX' selector. If at least one of these signals takes the value of one, the output of the six-input NOR gate, which generates the *sel* signal, becomes 0, selecting SUM_1 as the final result. Otherwise, when all six signals are zero, SUM_0 will be outputted. The structure of the proposed TCR adder in Fig. 5 can be easily generalized for a general value of m.

B. One-Hot-Based Modular Adder

Herein, we use the bit indexing of [19], wherein the starting index of bits is considered to be 0. Because only one bit of the OHR operands has the value "1" (the bit position defines the value of the integer), there are m^2 possible combinations of A and B.

Lemma 2: In the one-hot modular addition, exactly m combinations of all the possible m^2 combinations of A and B result in the same output. Therefore, each bit of the result is set to 1 for m combinations of A and B.

Proof: If A has the value of k ($a_k = 1$), for each i, there is exactly one value of B for which the result of $A + B$ mod m is i. We start by considering the two following states.

1) $k \leq i$: In this case, B can be calculated from the formula: $A + B = i$. Therefore, B has the value of $i-k$ ($b_{i-k} = 1$).
2) $k > i$: In this case, B can be calculated as: $A + B = i+m$, and thus, the only possible choice of B is $m+i-k$ ($b_{i+m-k} = 1$).

where k is an integer ranging from 0 to $m-1$.

For each value of k, there is exactly one choice of B resulting in $SUM_i = 1$. Therefore, m combinations of A and B produce the value of i in the modular addition output.

The following relation shows the m combinations for which the modular addition of A and B is equal to i (SUM_i has the value of one):

$$SUM_i = \left(\vee_{k=0}^{i} (a_k \wedge b_{i-k}) \right) \vee \left(\vee_{k'=i+1}^{m-1} (a_{k'} \wedge b_{i+m-k'}) \right). \tag{30}$$

The first term of (30) includes $i+1$ combinations of A and B, while the second takes into account $m-i-1$ combinations, leading to a total of $(i+1)+(m-i-1) = m$ combinations.

When $A = B$, the bitwise AND of A and B easily identifies the position of the one-value bits both in A and B. In this case, the output of only one AND gate becomes 1, while all the other bits are 0. Therefore, we can use m AND gates for the m combinations of $A = B$.

When $A \neq B$, the bitwise OR of A and B indicates two positions of bits with the value of one (a bit position in A and another in B). It is not important which of these two positions belongs to B or A since addition is commutative. If these two bit positions are i and j (we can detect this case by ANDing the outputs of the OR gates in bit positions i and j), two combinations of A and B are possible: 1) $A_i = 1$ and $B_j = 0$ and 2) $A_i = 0$ and $B_j = 1$. Therefore, for detecting the m^2-m combinations of $A \neq B$, m 2-input OR gates are used in the first level and $(m/2) = (m!/(m-2)!2!) = (m(m-1)/2)$ 2-input AND gates are required in the second level (for ANDing all the combinations of two outputs of the OR gates in the first level). As already mentioned, each AND gate in the second level covers two combinations of A and B, and so all the $(m/2)*2$ possible combinations are covered.

Fig. 6 shows the proposed OHR adder for a general value of m. The numbers inside the OR gates indicate the number of inputs (these OR gates can also be implemented using two-input OR gates). Note that AND(i) and OR(i) represent the bits resulting from the ANDing and ing of the inputs A_i and B_i, respectively. It can be seen that m AND gates at the first level are used to detect the m combinations of A and B for $A = B$. In other words, if both A and B have the same value, they have the same bit position with " 1," and the other bit positions are zero according to the definition of OHC. For identifying the other cases, $A \neq B$ (the remaining m^2-m combinations of A and B), first, the OR of A and B in the first level is computed, and then all the combinations of two OR gate outputs are ANDed. The output of each AND gate in the second level allows to identify the two combinations of $A \neq B$.

With this approach, all the m^2 combinations of modular additions of A and B in OHC can be detected with $m(m+1)/2$ AND gates (m AND gates in the first level for $A = B$ and $m(m-1)/2$ AND gates in the second level for $A \neq B$).

The following two cases are individually considered in Fig. 6.

1) Odd m: Each bit of the SUM in this case, when m is odd and $A = B$, can be calculated from (31), where i is the

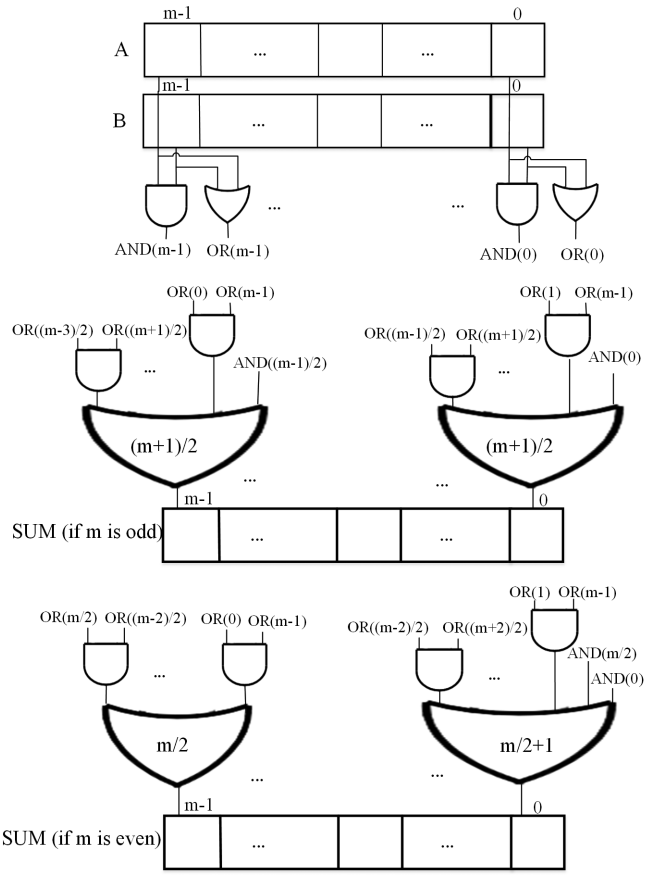

Fig. 6. Proposed OHR modulo adder for general m.

bit number

$$\mathrm{SUM}_i = \begin{cases} \mathrm{AND}\left(\dfrac{i}{2}\right), & \text{if } m \text{ is odd, } A=B \text{ and } i \text{ is even} \\ \mathrm{AND}\left(\dfrac{m+i}{2}\right), & \text{if } m \text{ is odd, } A=B \text{ and } i \text{ is odd.} \end{cases} \tag{31}$$

In general, when m is odd, the SUM can be calculated with

$$\mathrm{SUM}_i = \mathrm{AND}\left(\frac{i}{2}\right) \vee \left(\underbrace{\vee_{k=0}^{\frac{i}{2}-1}}_{\frac{i}{2}-1\geq 0} (\mathrm{OR}(k) \wedge \mathrm{OR}(i-k)) \right) \vee \left(\underbrace{\vee_{k'=i+1}^{\lfloor \frac{m+i}{2} \rfloor}}_{i\leq m-2} (\mathrm{OR}(k') \wedge \mathrm{OR}(m+i-k')) \right) \quad (m \text{ is odd and } i \text{ is even}) \tag{32}$$

$$\mathrm{SUM}_i = \mathrm{AND}\left(\frac{m+i}{2}\right) \vee \left(\vee_{k=0}^{\lfloor \frac{i}{2} \rfloor} (\mathrm{OR}(k) \wedge \mathrm{OR}(i-k)) \right) \vee \left(\underbrace{\vee_{k'=i+1}^{(\frac{m+i}{2})-1}}_{i\leq m-2} (\mathrm{OR}(k') \wedge \mathrm{OR}(m+i-k')) \right) \quad (m \text{ is odd and } i \text{ is odd}). \tag{33}$$

TABLE III
NUMBER OF COMBINATIONS COVERED BY (32)

Part of (32)	Number of combinations
$\mathrm{AND}(\frac{i}{2})$	1
$\underbrace{(\vee_{k=0}^{\frac{i}{2}-1}}_{\frac{i}{2}-1\geq 0} OR(k) \wedge OR(i-k))$	$\frac{i}{2}$ *2 =i
$\underbrace{(\vee_{k'=i+1}^{\lfloor \frac{m+i}{2} \rfloor}}_{i\leq m-2} OR(k') \wedge OR(m+i-k'))$	$(\lfloor \frac{m+i}{2} \rfloor$ -i)*2

(m is odd and i is even)

TABLE IV
NUMBER OF COMBINATIONS COVERED BY (33)

Part of (33)	Number of combinations
$\mathrm{AND}(\frac{m+i}{2})$	1
$((\vee_{k=0}^{\lfloor \frac{i}{2} \rfloor}(OR(k) \wedge OR(i-k))$	$(\lfloor \frac{i}{2} \rfloor + 1)$ *2
$\underbrace{(\vee_{k'=i+1}^{(\frac{m+i}{2})-1}}_{i\leq m-2} (OR(k') \wedge OR(m+i-k'))$	$(\left(\frac{m+i}{2}\right) - 1$ -i)*2

(m is odd and i is odd)

Equation (32) is used for the bits with even index and (33) is used for the bits with odd index.

According to Lemma 2, only m combinations from all the possible m^2 combinations of A and B result in the same output. Equations (32) and (33) confirm this point. Table III shows all combinations that cause $\mathrm{SUM}_i = 1$ in (32). The total number of combinations that are covered by (32) $[n_{(32)}]$ is shown in the following:

$$\begin{aligned} n_{(32)} &= 1 + i + \left(\left\lfloor \frac{m+i}{2} \right\rfloor - i \right) * 2 \\ &= 1 + i + \left(\left\lfloor \frac{m}{2} \right\rfloor + \frac{i}{2} - i \right) * 2 = 1 + \left(\frac{m-1}{2} \right) * 2 \\ &= m \qquad (m \text{ is odd and } i \text{ is even}). \end{aligned} \tag{34}$$

Table IV shows all combinations that lead to $\mathrm{SUM}_i = 1$ in (33). The total number of combinations covered by (33) $[n_{(33)}]$ is shown in the following:

$$\begin{aligned} n_{(33)} &= 1 + \left(\left\lfloor \frac{i}{2} \right\rfloor + 1 \right) * 2 + \left(\left(\frac{m+i}{2} \right) - 1 - i \right) * 2 \\ &= 1 + \left(\left(\frac{i-1}{2} \right) + 1 \right) * 2 + (m - i - 2) \\ &= m \qquad (m \text{ is odd and } i \text{ is odd}). \end{aligned} \tag{35}$$

2) Even m: In this case, each of the m combinations of $A = B$ produces even values of SUM. Equation (36) shows the value of each bit in the result when m is even and $A = B$

$$\mathrm{SUM}_i = \begin{cases} \left(\mathrm{AND}\left(\dfrac{i}{2}\right) \vee \mathrm{AND}\left(\dfrac{m+i}{2}\right) \right), \\ \qquad \text{if } m \text{ is even, } A = B \text{ and } i \text{ is even} \\ 0, \quad \text{if } m \text{ is even, } A = B \text{ and } i \text{ is odd.} \end{cases} \tag{36}$$

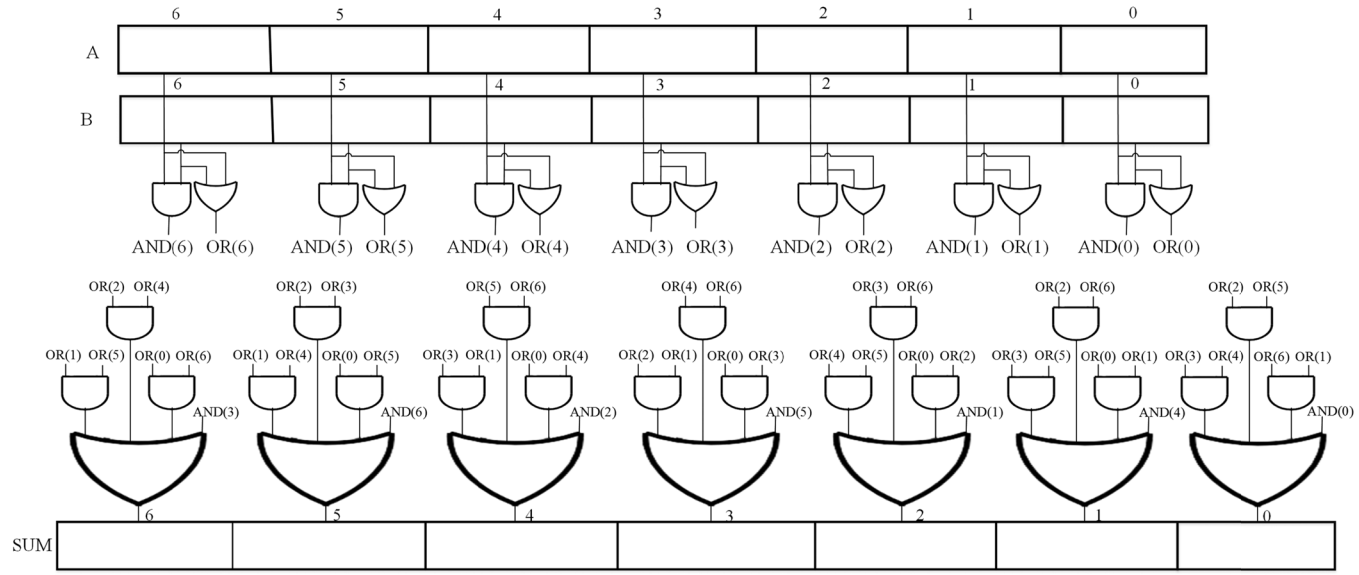

Fig. 7. Proposed OHR modulo adder for $m = 7$.

In general, when m is even, the SUM can be calculated from

$$\mathrm{SUM}_i = \mathrm{AND}\left(\frac{i}{2}\right) \vee \mathrm{AND}\left(\frac{i+m}{2}\right) \vee \left(\underbrace{\vee_{k=0}^{\frac{i}{2}-1} (\mathrm{OR}(k) \wedge \mathrm{OR}(i-k))}_{\frac{i}{2}1 \geq 0} \right) \vee \left(\underbrace{\vee_{k'=i+1}^{\left(\frac{m+i}{2}\right)-1} (\mathrm{OR}(k') \wedge \mathrm{OR}(m+i-k'))}_{i \leq m-2} \right) \quad (m \text{ is even and } i \text{ is even}) \tag{37}$$

$$\mathrm{SUM}_i = \left(\vee_{k=0}^{\left\lfloor \frac{i}{2} \right\rfloor} (\mathrm{OR}(k) \wedge \mathrm{OR}(i-k)) \right) \vee \left(\vee_{k'=i+1}^{\left\lfloor \frac{m+i}{2} \right\rfloor} (\mathrm{OR}(k') \wedge \mathrm{OR}(m+i-k')) \right) \quad (m \text{ is even and } i \text{ is odd}). \tag{38}$$

Equation (37) is used for the bits with even index and (38) for the bits with odd index. Both (37) and (38) also cover m combinations of A and B.

Example: Equations (32) and (33) are used to implement an OHR-based adder modulo $m = 7$. The bits of even index of SUM are produced by (32)

$$\begin{aligned}
\mathrm{SUM}_0 &= \mathrm{AND}(0) \vee \left(\vee_{k'=0+1}^{\left\lfloor \frac{7+0}{2} \right\rfloor} (\mathrm{OR}(k') \wedge \mathrm{OR}(7+0-k')) \right) \\
&= \mathrm{AND}(0) \vee (\mathrm{OR}(1) \wedge \mathrm{OR}(6)) \vee (\mathrm{OR}(2) \wedge \mathrm{OR}(5)) \\
&\quad \vee (\mathrm{OR}(3) \wedge \mathrm{OR}(4)) \\
\mathrm{SUM}_2 &= \mathrm{AND}\left(\frac{2}{2}\right) \vee \left(\vee_{k=0}^{\frac{2}{2}-1} (\mathrm{OR}(k) \wedge \mathrm{OR}(2-k)) \right) \\
&\quad \vee \left(\vee_{k'=2+1}^{\left\lfloor \frac{7+2}{2} \right\rfloor} (\mathrm{OR}(k') \wedge \mathrm{OR}(7+2-k')) \right) \\
&= \mathrm{AND}(1) \vee (\mathrm{OR}(0) \wedge \mathrm{OR}(2)) \vee (\mathrm{OR}(3) \wedge \mathrm{OR}(6)) \\
&\quad \vee (\mathrm{OR}(4) \wedge \mathrm{OR}(5)) \\
\mathrm{SUM}_4 &= \mathrm{AND}\left(\frac{4}{2}\right) \vee \left(\vee_{k=0}^{\frac{4}{2}-1} (\mathrm{OR}(k) \wedge \mathrm{OR}(4-k)) \right) \\
&\quad \vee \left(\vee_{k'=4+1}^{\left\lfloor \frac{7+4}{2} \right\rfloor} (\mathrm{OR}(k') \wedge \mathrm{OR}(7+4-k')) \right) \\
&= \mathrm{AND}(2) \vee (\mathrm{OR}(0) \wedge \mathrm{OR}(4)) \vee (\mathrm{OR}(1) \wedge \mathrm{OR}(3)) \\
&\quad \vee (\mathrm{OR}(5) \wedge \mathrm{OR}(6)) \\
\mathrm{SUM}_6 &= \mathrm{AND}\left(\frac{6}{2}\right) \vee \left(\vee_{k=0}^{\frac{6}{2}-1} (\mathrm{OR}(k) \wedge \mathrm{OR}(6-k)) \right) \\
&= \mathrm{AND}(3) \vee (\mathrm{OR}(0) \wedge \mathrm{OR}(6)) \vee (\mathrm{OR}(1) \wedge \mathrm{OR}(5)) \\
&\quad \vee (\mathrm{OR}(2) \wedge \mathrm{OR}(4)).
\end{aligned}$$

The odd bits of SUM are obtained from (33)

$$\begin{aligned}
\mathrm{SUM}_1 &= \mathrm{AND}\left(\frac{7+1}{2}\right) \vee \left(\vee_{k=0}^{\left\lfloor \frac{1}{2} \right\rfloor} (\mathrm{OR}(k) \wedge \mathrm{OR}(1-k)) \right) \\
&\quad \vee \left(\vee_{k'=1+1}^{\left(\frac{7+1}{2}\right)-1} (\mathrm{OR}(k') \wedge \mathrm{OR}(7+1-k')) \right) \\
&= \mathrm{AND}(4) \vee (\mathrm{OR}(0) \wedge \mathrm{OR}(1)) \vee (\mathrm{OR}(2) \wedge \mathrm{OR}(6)) \\
&\quad \vee (\mathrm{OR}(3) \wedge \mathrm{OR}(5)) \\
\mathrm{SUM}_3 &= \mathrm{AND}\left(\frac{7+3}{2}\right) \vee \left(\vee_{k=0}^{\left\lfloor \frac{3}{2} \right\rfloor} (\mathrm{OR}(k) \wedge \mathrm{OR}(3-k)) \right) \\
&\quad \vee \left(\vee_{k'=3+1}^{\left(\frac{7+3}{2}\right)-1} (\mathrm{OR}(k') \wedge \mathrm{OR}(7+3-k')) \right) \\
&= \mathrm{AND}(5) \vee (\mathrm{OR}(0) \wedge \mathrm{OR}(3)) \vee (\mathrm{OR}(1) \wedge \mathrm{OR}(2)) \\
&\quad \vee (\mathrm{OR}(4) \wedge \mathrm{OR}(6)) \\
\mathrm{SUM}_5 &= \mathrm{AND}\left(\frac{7+5}{2}\right) \vee \left(\vee_{k=0}^{\left\lfloor \frac{5}{2} \right\rfloor} (\mathrm{OR}(k) \wedge \mathrm{OR}(5-k)) \right) \\
&= \mathrm{AND}(5) \vee (\mathrm{OR}(0) \wedge \mathrm{OR}(3)) \\
&\quad \vee (\mathrm{OR}(1) \wedge \mathrm{OR}(2)) \vee (\mathrm{OR}(4) \wedge \mathrm{OR}(6)).
\end{aligned}$$

The proposed modulo adder for $m = 7$ is shown in Fig. 7. This adder has a simpler structure and less delay than [19]. The other feature of this design is that both operands are represented in OHC, while in [19], one of the operands has to be encoded in binary.

IV. Performance Evaluation

In this section, the proposed modular adders are evaluated and compared with the previous state-of-the-art TCR and OHR-based adders [18], [19]. With that purpose, both technology agnostic analyses, based on the unit-gate model [22], as well as the transistor count, and experimental evaluation, based on application-specific integrated circuits (ASICs), are performed. Note that the only work that has reported TCR-based modular adders is [18], and the latest design of OHR-based modular adders is reported in [19].

A. Technology Agnostic Assessment

The proposed TCR-based adder in Fig. 5 consists of the SUM_0 and SUM_1 producing gates, an AND gate, and an $(m-2)$-bit 2×1 MUX. Therefore, its area can be formulated as follows:

$$A_{\text{Proposed TCR Adder}} = A_{\text{SUM}_0\,\&\,\text{SUM}_1\text{Producer}} + A_{\text{AND}} + (m-2)A_{\text{MUX2}\times1} \quad (39)$$

where A_k denotes the area of the component k, and

$$\begin{aligned} A_{\text{SUM}_0\,\&\,\text{SUM}_1\ \text{Producer}} &= 2(m-1)A_{\text{AND/NOR}} + (1+2+\cdots+(m-1)) \\ &\quad \times A_{\text{AND/OR/NAND}} \\ &\quad +(A_{\text{NOR}} + A_{\text{NOR}_3} + A_{\text{NOR}_4} + \cdots + A_{\text{NOR}_{m-2}}) \\ &\quad +2A_{\text{NOR}_{m-1}} + (m-2)A_{\text{NOT}}. \end{aligned} \quad (40)$$

It should be mentioned that A_{NOR_i} in (40) indicates the area of a NOR gate with i inputs for $i \geq 3$, and the "/" symbol means "or." Moreover, the delay of the critical path of the TCR adder herein proposed can be estimated as

$$D_{\text{Proposed TCR Adder}} = D_{\text{SUM}_1\text{Producer}} + D_{\text{MUX2}\times1} \quad (41)$$

where D_k denotes the delay of the component k, and

$$D_{\text{SUM}_1\ \text{Producer}} = (m-2)D_{\text{AND}} + D_{\text{OR}} + D_{\text{NAND}} + D_{\text{not}}. \quad (42)$$

The area of the TCR adder of [18] is estimated as follows:

$$\begin{aligned} A_{\text{TCR Adder[18]}} &= A_{\text{Shifter control circuit}} + (m + (m+2) + (m+2+2^2) \\ &\quad + \ldots + (2m-2))A_{\text{MUX2}\times1} + (m-1)A_{\text{MUX2}\times1} \end{aligned} \quad (43)$$

where

$$A_{\text{Shifter control circuit}} = (m-2)A_{\text{XOR}}. \quad (44)$$

Equation (45) is used for computing the delay of the TCR-based adder of [18]

$$D_{\text{TCR Adder[18]}} = D_{\text{Shifter control circuit}} + (\lceil \text{Log}^m \rceil + 1)D_{\text{MUX2}\times1} \quad (45)$$

TABLE V
Area and Delay, Estimated With Unit-Gate Model, for the Proposed Thermometer-Based Modular Adder and [18]

modulo (m)	Proposed TRC-based adder modulo m		TRC-based adder modulo m in [18]	
	Area	Delay	Area	Delay
5	37	7	78	12
7	69	9	112	14
8	88	10	129	14
9	109	11	191	16
11	157	13	231	18

where

$$D_{\text{Shifter control circuit}} = \lceil \text{Log}^{m-1} \rceil D_{\text{XOR}}. \quad (46)$$

According to the unit-gate model, the basic 2-input gates (AND, OR, NAND, and NOR) count as one unit for the delay and the area, with the exception of the XOR/XNOR gates which count as two units [22]. In that model, more complex cells are built from the basic two-input gates. Moreover, one bit 2×1 MUX counts as three and two gates for the area and the delay, respectively [23]. By applying the unit-gate model, (39)–(46) can be rewritten as the following (note that each i-input basic gate can be realized using $i-1$ 2-input gates):

$$A_{\text{Proposed TCR Adder\_unit gate}} = 7m - 11 + \frac{m(m-1) + (m-3)(m-2)}{2} \quad (47)$$

$$D_{\text{Proposed TCR Adder\_unit gate}} = m + 2 \quad (48)$$

$$\begin{aligned} A_{\text{TCR Adder[18] \_unit gate}} &= 2(m-2) + 3(m + (m+2) + (m+2+2^2) \\ &\quad + \ldots + (2m-2)) + 3(m-1) \end{aligned} \quad (49)$$

$$D_{\text{TCR Adder[18]\_unit gate}} = 2(\lceil \text{Log}^{m-1} \rceil + \lceil \text{Log}^m \rceil + 1). \quad (50)$$

Table V compares the TCR-based adder herein proposed with [18] based on the unit-gate model for moduli 5, 7–9, and 11. With our proposal, the delay and area values have been improved on average by 33% and 39.5%, respectively.

The area of the OHR-based adder proposed herein can be estimated as follows:

$$A_{\text{Proposed OHR Adder}} = A_{\text{First Level}} + A_{\text{second Level}} + A_{\text{last level}} \quad (51)$$

where

$$A_{\text{First Level}} = mA_{\text{AND}} + mA_{\text{OR}} \quad (52)$$

$$A_{\text{second Level}} = \frac{m(m-1)}{2}A_{\text{AND}} \quad (53)$$

$$A_{\text{last Level (multi input OR}'s)} = \frac{m(m-1)}{2}A_{\text{OR}}. \quad (54)$$

It should be mentioned that there are $m(m-1)/2$ two-input AND gates in the second level of the proposed

OHR-based adder. Furthermore, the last level includes m OR gates with $((m+1)/2)$-inputs for odd values of m. Each i-input OR gate can be implemented with $(i-1)$ two-input OR gates. Therefore, a total of $m(m-1)/2$ two-input OR gates are required for the last level of the OHR-based adder for odd values of m. In the case of an even m, according to Fig. 6, half of the last level of OR gates has $m/2$ inputs, and the other has $(m/2)+1$ inputs. Therefore, again a total of $m(m-1)/2$ two-input OR gates are required for the case of even values of m. Moreover, the delay of the critical path of the proposed OHR-based adders includes

$$D_{\text{Proposed OHR Adder}} = D_{\text{First Level}} + D_{\text{second Level}} + D_{\text{last level}} \tag{55}$$

where

$$D_{\text{First Level}} = \text{MAX}D_{\text{OR}}, D_{\text{AND}} \tag{56}$$

$$D_{\text{second Level}} = D_{\text{AND}} \tag{57}$$

$$D_{\text{last Level}} = \left\lceil \text{Log}^{\frac{m+1}{2}} \right\rceil D_{\text{OR}}. \tag{58}$$

By applying the unit-gate model to (51)–(58), the following formulas are obtained:

$$\begin{aligned} A_{\text{Proposed OHR Adder\_unit gate}} &= \underbrace{2m}_{A_{\text{First Level}(\text{AND}/\text{OR}'s)}} + \underbrace{\frac{m(m-1)}{2}}_{\text{A second Level (AND}'s)} \\ &+ \underbrace{\frac{m(m-1)}{2}}_{\text{A last Level (multi input OR}'s)} = m^2 + m \end{aligned} \tag{59}$$

$$D_{\text{Proposed OHR Adder\_unit gate}} = \underbrace{2}_{D_{\text{First \& Second Level}}} + \underbrace{\left\lceil \log^{\frac{m+1}{2}} \right\rceil}_{D_{\text{Last Level (multi-input OR)}}} \tag{60}$$

while for [19]

$$A_{\text{OHR Adder[19]\_unit gate}} = 3 * m\lceil \text{Log}^m \rceil \tag{61}$$

$$D_{\text{OHR Adder[19]\_unit gate}} = 2 * \lceil \text{Log}^m \rceil. \tag{62}$$

Table VI shows the estimate of the area and the delay of the proposed OHR-based adder as well as of [19]. According to the unit-gate model, the proposed OHR-based adders result in an average improvement of 31.7% and 12.2% in delay and area, respectively.

Similar to [19], a comparison is performed based on the number of transistors required for implementing the logic gates, as shown in Table VII. We consider complementary metal–oxide–semiconductor (CMOS) designs for all the required gates. Therefore, basic two-input digital gates can be implemented with four transistors. In addition, a two-input XOR gate requires 12 transistors [24]. A MUX also requires eight transistors by using inverting MUXs [24] (an extra NOR gate is required when the number of consecutive MUXs is odd). For [18] (see Fig. 1), MUXs for which one of the inputs is 0 are implemented by a single AND gate, and MUXs for which one of the inputs is 1 are implemented by one AND and one OR gate. Note that a MUX or an XOR gate can also be implemented with transmission gates (TGs) with a reduced number of transistors. However, TGs have the problem of voltage drop and high internal capacitances due to the direct exposure of the junction capacitors to the signals which are passing TGs [25], and therefore, they are not considered here. In the first level of the proposed OHR-based adder (see Fig. 7), NAND and NOR gates can be used instead of AND and OR gates. With this technique, each bit of the SUM can be calculated by a *compound gate* [24] with $2m$ transistors for a modulo m. Fig. 8 shows the structure of this *compound gate* for $m = 7$. It can be seen from Table VII that the proposed TCR and OHR modular adders require less transistors than [18] and [19], respectively.

TABLE VI

AREA AND DELAY, ESTIMATED WITH UNIT-GATE MODEL, FOR THE PROPOSED ONE-HOT MODULAR ADDER AND [19]

Modulo (m)	Proposed One-Hot modulo adder		One-Hot modulo adder of [19]	
	Area	Delay	Area	Delay
5	30	4	45	6
7	56	4	63	6
8	72	5	72	6
9	90	5	108	8
11	132	5	132	8

TABLE VII

TRANSISTOR COUNT COMPARISON

modulo (m)	TCR-based adders modulo m		OHR-based adders modulo m	
	Proposed	[18]	Proposed	[19]
5	140	216	90	136
7	250	312	154	188
8	314	360	192	214
9	384	546	234	296
11	542	662	330	360

B. Experimental Evaluation

All the proposed modular adders together with [18] and [19] were described in VHDL and verified using ModelSim. Circuits for all these adders were implemented using the 65-nm TSMC CMOS logic salicide process (1-poly, 9-metal). Cadence RTL Compiler tools version v11.20-s012_1 were used for synthesizing the designs and the Cadence Encounter and NanoRoute tools (versions v09.12-s159 and v09.12-s013, respectively) for placing and routing. Power consumption was obtained from the placed-and-routed circuit specifications, for 20% of switching activity. The Cadence Encounter power reporting tool was used to measure the total power, including the dynamic and leakage components of the power. Experimental results are presented in Tables VIII and IX for the same moduli considered in [19]. Also, the percentage of energy savings of the proposed adders, when compared to the previous designs, is shown in Fig. 9.

TABLE VIII
EXPERIMENTAL RESULTS FOR THE PROPOSED THERMOMETER-BASED MODULAR ADDER AND [18]

modulo (m)	Proposed thermometer based modulo adder				Thermometer based modulo adder [18]			
	Delay (ns)	Power (mW)	Allocated area (um^2)	Energy (pJ)	Delay (ns)	Power (mW)	Allocated area (um^2)	Energy (pJ)
5	0.292	0.755	488	0.2205	0.583	0.525	743	0.3061
7	0.35	0.974	820	0.3409	0.573	0.7671	1111	0.4395
8	0.432	0.8588	1069	0.371	0.672	0.7645	1354	0.5137
9	0.422	1.127	1289	0.4756	0.645	1.064	1884	0.686
11	0.498	1.152	1823	0.5737	0.724	1.292	2411	0.9354

TABLE IX
EXPERIMENTAL RESULTS FOR THE PROPOSED OHC MODULO ADDER AND [19]

modulo (m)	Proposed OHR Adder				OHR Adder [19]			
	Delay (ns)	Power (mW)	Allocated area (um^2)	Energy (pJ)	Delay (ns)	Power (mW)	Allocated area (um^2)	Energy (pJ)
5	0.203	1.475	501	0.2994	0.301	1.034	640	0.3112
7	0.215	1.734	761	0.3728	0.311	1.225	841	0.3810
8	0.216	1.886	880	0.4074	0.319	1.399	979	0.4463
9	0.227	2.088	1037	0.474	0.377	1.404	1318	0.5293
11	0.244	2.346	1414	0.5724	0.389	1.572	1600	0.6115

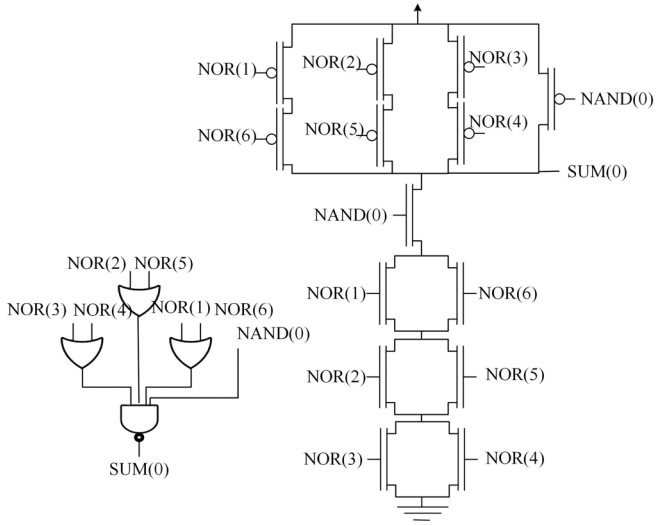

Fig. 8. Structure of a compound CMOS gate for calculating SUM(0) when $m = 7$ (by using nand and nor gates instead of and and or gates in the first level of Fig. 7).

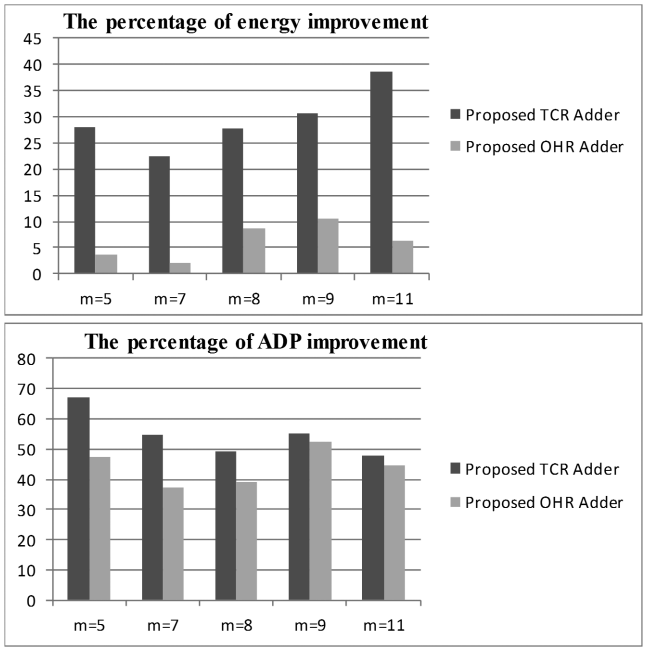

Fig. 9. Comparison of the performance of the proposed TC- and OHC-based adders.

Table VIII presents the experimental results for the TCR-based modular adders. Experimental results show that the speed, area, and energy for moduli 5, 7–9, and 11 are improved on average by 38%, 27.5%, and 29.5% respectively. It should be noted that due to the way of representing numbers in OHC and TC, the number of transistors switching in the computational circuits is significantly less than for the regularly binary coded operands. For example, for the proposed OHR-based modular adder, whenever the inputs are changed at most the output of four gates may change from zero to one (including two OR gates in the top, one AND gate in the middle, and one of the 4-input OR gates in the bottom of Fig. 6), and at most the output of four gates may change from one to zero (including two OR gates in the top, one AND gate in the middle, and one of the 4-input OR gates in the bottom). The output of the other gates remains zero without any switching activity. For the binary computations, it is possible to have all the output bits changing by changing the values at the input.

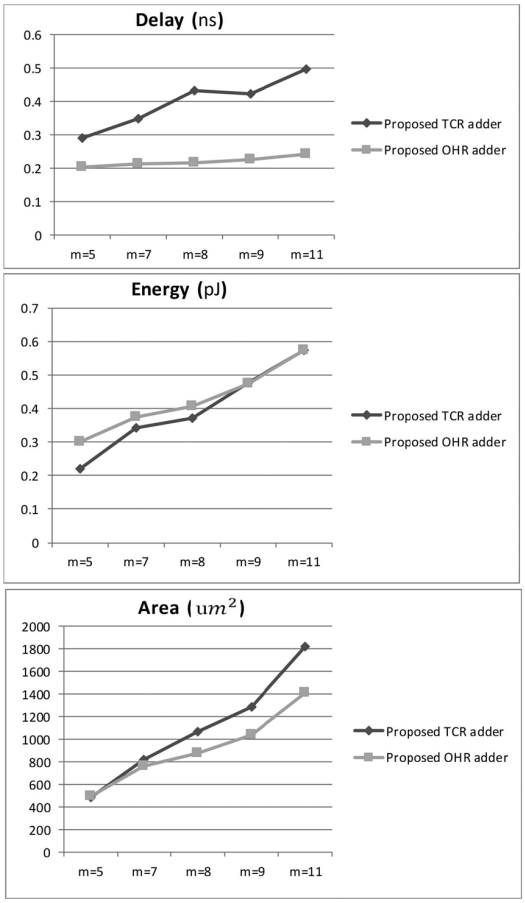

Fig. 10. Comparing the performance of the proposed TC- and OHC-based adders.

Therefore, the power consumption of the suggested design in Fig. 6 is much less than the hardware structures supported on binary inputs. Since, in both [18] and [19], the effectiveness of TCR and OHR-based designs is proved, in comparison with regular binary designs, and since the proposed designs have better performance than [18] and [19], it can be concluded that they also have better performance than regular modular binary adders; with the exception of moduli with the shape 2^n, since this type of modulo does not require any additional modular circuitry, it can be implemented with regular binary adders [20].

Table IX shows the experimental results of the proposed OHR-based modular adders. It can be seen that the delay and area of the suggested OHR-based adders are improved in comparison with [19]. The delay of the proposed adder for the moduli 5, 7–9, and 11 is improved in average 34.5%, while the improvement of energy (PDP) is 6.3%.

It should be noted that one of the operands of the design in [19] is represented in binary. The same assumption is also made herein to simulate [19]. However, if the circuits required to convert one-hot encodings to their equivalent binary representations are considered, the required power and energy for those adders grow.

Fig. 9 represents the percentage of the improvement of the PDP (i.e., energy) and the area-delay-product (ADP) of the proposed TCR and OHR-based modular adders, in comparison with the designs in [18] and [19]. It can be observed that the proposed TC-based modular adder results in 28%, 22.4%, 27.8%, 30.7%, and 38.7% energy improvement for moduli 5, 7–9, and 11, respectively. Moreover, the ADP has been improved 67.1%, 54.9%, 49.2%, 55.2%, and 48% for the same moduli. Similarly, according to Fig. 9, the suggested OHR-based modular adders result in 3.79%, 2.14%, 8.72%, 10.45%, and 6.39% energy reduction for moduli 5, 7–9, and 11, respectively. Finally, the ADP improvement percentages are 47.2%, 37.44%, 39.13%, 52.62%, and 44.57%.

Although the power consumption of the proposed adders (except for TCR adders and $m = 11$ in Tables VIII and IX) is higher than [18] and [19], due to more interconnections and the usage of multi-input gates, the delay is significantly reduced by avoiding the MUX-approach and making simplifications at the logic level. In the end, the significant reduction of the delay allows us to improve the energy efficiency for all cases in comparison to [18] and [19]. Fig. 10 compares the performance of the proposed adders. Although the representation of numbers with the OHC requires one more bit than with the TC, the delay and the occupied area of the proposed OHR adder design are better than those of the proposed TCR adder design, while the energy consumption is very close for the two approaches. As indicated in [19], the one-hot-based modular adders require less area and impose lower delays than the corresponding binary modular adders. Therefore, since our one-hot modulo adder has a better performance than [19], it can be concluded that the suggested one-hot modulo adder has a better performance than the corresponding binary adders for the moduli 5, 7–9, and 11.

Furthermore, OHC reduces the circuit activity factor due to the use of just one active signal for each input value. Therefore, OHC circuits have better energy-efficiency than the corresponding binary adders, being extremely useful for moduli with small values [19].

V. Conclusion

In this paper, 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 2^n). 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 energy-efficient 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.

Fereshteh Jafarzadehpour received the B.Sc. degree (Hons.) from the Shahid Bahonar University of Kerman, Kerman, Iran, in 2008, and the M.Sc. degree (Hons.) in computer architecture from the Science and Research Branch, Islamic Azad University, Tehran, Iran, in 2015. She is currently working toward the Ph.D. degree at the Department of Computer Engineering, Kerman Branch, Islamic Azad University, Kerman.

She was ranked 13th in the 12th National Scientific Olympiad in the field of computer engineering in 2007. Her current research interests include computer arithmetic, residue number systems, and VLSI design.

Amir Sabbagh Molahosseini (SM'18) received the B.Sc. degree from the Shahid Bahonar University of Kerman, Kerman, Iran, in 2005, and the M.Sc. and Ph.D. degrees (Hons.) in computer engineering from the Science and Research Branch, Islamic Azad University, Tehran, Iran, in 2007 and 2010, respectively.

He was a Visiting Researcher with the Signal Processing Systems Group, R&D Instituto de Engenharia de Sistemas eComputadores (INESC-ID), Instituto Superior Tecnico (IST), University of Lisbon, Lisbon, Portugal. He is currently an Assistant Professor with the Department of Computer Engineering, Kerman Branch, Islamic Azad University, Kerman. His current research interests include computer arithmetic and alternative computing systems.

Dr. Sabbagh Molahosseini is currently an Associate Editor of the *Journal of Electrical and Computer Engineering*.

Azadeh Alsadat Emrani Zarandi (M'16) received the B.Sc. degree from the Shahid Bahonar University of Kerman, Kerman, Iran, in 2006, the M.Sc. degree (Hons.) from the Isfahan University of Technology, Isfahan, Iran, in 2008, and the Ph.D. degree in computer engineering from the Science and Research Branch, Islamic Azad University, Tehran, Iran, in 2015.

She was a Visiting Researcher with the Signal Processing Systems Group, R&D Instituto de Engenharia de Sistemas eComputadores (INESC-ID), Instituto Superior Tecnico (IST), University of Lisbon, Lisbon, Portugal. She is currently an Assistant Professor with the Computer Engineering Department, Shahid Bahonar University of Kerman. Her current research interests include computer arithmetic, inexact computing, and wireless sensor networks.

Leonel Sousa (SM'03) received the Ph.D. degree in electrical and computer engineering from the Instituto Superior Tecnico (IST), University of Lisbon (UL), Lisbon, Portugal, in 1996.

He is currently Full Professor with IST, UL, where he is also a Senior Researcher with the R&D Instituto de Engenharia de Sistemas eComputadores (INESC-ID). He has contributed to more than 200 papers in journals and international conferences, for which he received several awards, such as the DASIP'13 Best Paper Award, the SAMOS'11 Stamatis Vassiliadis Best Paper Award, the DASIP'10 Best Poster Award, and the Honorable Mention Award UTL/Santander Totta for the quality of the publications in 2009. His current research interests include VLSI architectures, computer architectures, parallel computing, computer arithmetic, and signal processing systems.

Dr. Sousa is a Fellow of the IET and a Distinguished Scientist of ACM. He has contributed to the organization of several international conferences, namely the Program Chair and the General and Topic Chair, and has given keynotes in some of them. He has edited four special issues in international journals, and he is currently an Associate Editor of the IEEE TRANSACTIONS ON COMPUTERS, IEEE ACCESS, the *IET Electronics Letters*, and the *Journal of Real-Time Image Processing* (Springer).