CHAPTER 1

INTRODUCTION

Modern digital signal processing systems require improved energy-efficiency, namely for emerging applications such as deep learning and Internet-of-Things (IoT). Unconventional number systems and arithmetic have been investigated recently to achieve specialized efficient embedded systems for those applications. Residue Number Systems (RNSs), in particular, have been used for digital signal processing (DSP) and cryptography, 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 inter-modulo operations. However, RNS arithmetic operations, such as additions and multiplications, are performed much more frequently than forward and reverse conversions. 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). The one-hot residue (OHR) has been considered for designing RNS modular arithmetic circuits based on circular shifting. 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. RNSs based on OHC have also been used on DSP applications due to their high speed. Alternatively, there are other types of coding, such as the thermometer coding (TC) that can be applied to enhance the performance of RNS modular arithmetic.

The thermometer is a 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. This type of coding is a sub-class of Golomb coding's, used in a variety of applications, including neural networks and data compression. Moreover, the TC together with distributed arithmetic can lead to fast implementations of modular arithmetic circuits.

With existent analog to digital converters (ADC) that directly convert analog inputs to residues encoded in the TC format, engineers and researchers are increasingly interested in modular adders for TC. Since there is no carry propagation in the modular addition of two TC numbers, the addition for small moduli can be done faster when compared to designs for

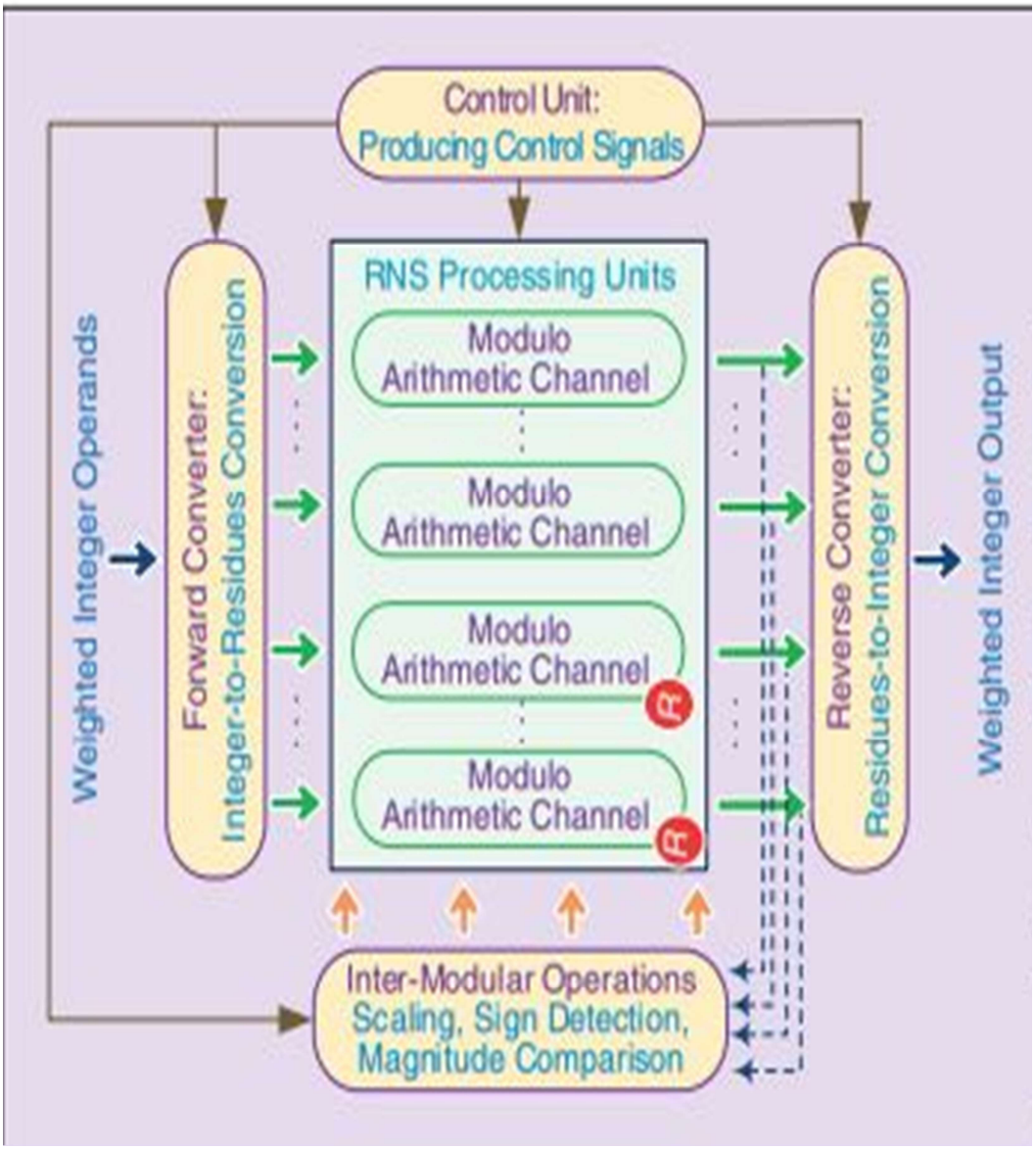
positional representations (with the exception of moduli like 2^n). Moreover, the usage of the TC format to represent residues removes the need for forward converters, whenever inputs values are coded using analog-to-residue converters. Similarly, modular adders based on OHC operate in a carry free manner, having a simple structure for different moduli. To set up efficient RNS systems with both thermometer or one hot coding, small moduli should be selected.

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, 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 applications.

CHAPTER 2

RESIDUE NUMBER SYSTEM

Residue Number System (RNS) is a non-weighted number system which was proposed by Garner back in 1959 to achieve fast implementation of addition, subtraction and multiplication operations in special-purpose computations. Unfortunately, RNS did not turn out as a popular alternative to two's complement number system in those days. The rigidity of instruction set architectures of the market-dominant computers and microprocessors then has been the main barrier to sustain the development of RNS-based applications.



Overview of Residue Number Systems

Another advantageous feature of the RNS system is that the operation between each pair of residue digits can be performed in a parallel and in independent manner from the other pairs of residues, as long as the resultant value does not exceed the legitimate range provided by the moduli set used.

Shorter bit-lengths of residues and inherent parallelism suggest that RNS combined with DA would be ideal for inner products computation. The performance gain is dependent on the data bit-length.

RNS has received varying degrees of attention from researchers in the past for implementing computer arithmetic. This number system provides a fundamental methodology for partitioning a large dynamic range system into a number of smaller but independent channels over which computations may be performed in parallel in a carry free manner between the channels.

For example, using the [7,8,9] moduli set, the decimal number X=179 will be represented using 4,3,8 residue set and the decimal number Y=254 will be represented using 2,6,2 residue set. Arithmetic operation of these two integers can be equally performed in modular arithmetic using their residues as follows:

$$179 \; \Theta \; 254 \cong 4,3,8 \; \Theta \; 2,6,2$$

$$= 402, 306, 802$$

where the arithmetic operator Θ can be +, −, and ×.

For example: $179+254 \cong 4,3,8 + 2,6,2$

$$= 4+2,3+6, 8+2$$

$$= 6,9,10$$

$$= 6,1,1$$

after executing the modulo operations |6| 7, |9| 8 & |10| 9. A reverse RNS-decimal conversion operation will then give the answer as $6,1,1 \cong 433$. Subtraction can be similarly executed with addition through using the two's complement of the subtrahend.

2.1 APPLICATIONS:

RNS has been widely known as an alternative number system that can accelerate the speed of applications dominated by a large number of

additions, subtractions and multiplications. The examples highlighted in this section include the well-known applications of RNS in DSP, cryptography, and memory module as well as its lesser-known applications in network and cloud computing, which use the features of RNS to satisfy certain critical constraints in security, power consumption and performance.

2.1.1 DIGITAL SIGNAL PROCESSING:

Finite impulse response (FIR) filter is one of the most frequently used functions in DSP systems. It is an application that is most well suited to demonstrate the benefits of RNS. An innovative use of RNS in DSP systems was reported on the implementation of one-dimensional discrete wavelet transform (DWT)

2.1.2 COMMUNICATIONS AND NETWORKING:

Software-defined networking (SDN) is an approach to simplify computer networking through abstraction of lower-level functionality. The network services are man- aged by decoupling the control plane that decides the traffic flow from the data plane that forwards traffic to the selected destination. OpenFlow is a well-known protocol to realize SDN. However, OpenFlow-enabled switching network has difficulty to meet the core network demands in its reactive mode because the total number of end-to-end active flows in a given route is severely constrained by the hardware flow table implementation in a multi-vendor core network. Firstly, the complex rule insertion creates a bottleneck for the maxi- mum flow rate. At the most only a few hundred flows per second can be set up.

Besides, the delay for flow insertion and flow modification can vary significantly. Secondly, the time taken for the flow to be fully functional is severely affected by the control channel latency. Emulated experiments show that a control channel latency of 300 *m*s will result in 20 s to reach full flow rate across 32 switches. The use of expensive and power-hungry fast content addressable memories to address the stateful requirement for the active flows imposes scalability, responsiveness, cost and power consumption problems for the application of SDN in core networks.

2.1.3 CRYPTOGRAPHY:

Public key cryptography is fundamentally used to assure confidentiality, authenticity and no reputability in electronic communications without requiring a covert channel for key exchange between the communication parties. Encryption and decryption in public key algorithm involve computationally costly large modular multiplications and modular exponentiations due to the large keys required for security reason. This has important implications for their practical use in mobile devices and light weight applications.

CHAPTER 3

MODULAR ADDERS

Modular adder is the primary element in the implementation of RNS based applications such as digital signal processing, reverse converters, multipliers, etc. Basically, modular adders are of two types: generic modular adders and special modular adders.

The moduli set used for modulo adders are of two types: those are specific moduli set and arbitrary moduli set. A lot of research has been going on in the field of implementation of general modular adders, which are nothing but modular adders with the specific moduli set. The moduli set consist of 3, 4 or 5 modulus values.

The modular adders are designed by using three different design methods, those are 1. Purely combinational logic circuit method, 2 look-up table method and 3. hybrid implementation.

Modulo is the operation of finding the Remainder when you divide two numbers, where moduli is of the form (2^n ± 1). Here 7 is the base.

EXAMPLE:

(5+6) mod 7= 11%7 = 4

Key advantages in the modular arithmetic are twofold. Firstly, is that the dynamic range (DR) of the residue set is confined to the value of the moduli and hence is normally very much smaller compared to the decimal number that they represent. Secondly, parallel arithmetic operations can be carried out among the pairs of residues independent of each other. Both these mean that the arithmetic circuits required can be of much simpler circuitry and can be much faster when compared to conventional number addition.

CHAPTER 4

CODING METHODOLOGIES

4.1 BINARY CODE:

For BC based modular arithmetic, the general approach to implement modulo adder for generic modulus is to make use of the architecture as shown in Fig. 4.1.1 although there are other possible variations that are based on specific hardware approach and those using multiple adders designs when specific moduli sets are considered. In Fig. 4.1.1, two binary adders are used to implement the following modular addition:

$$|A+B|_m = \begin{cases} A+B & \text{if } A+B < m \\ A+B-m & \text{otherwise} \end{cases}$$

Circuit Fig. 4.1.1(a) uses one binary adder to perform the addition of the two operands, and to first provide an intermediate sum. The modulus is then subtracted from intermediate sum, which can be performed through addition with the two's complement of, as shown. The carry-out bit from the subtraction is then used to control the output multiplexer, which determines whether the output should be or, in effect performing the modulo operation.

A variation of the circuit is shown in Fig. 4.1.1(b) where a three operands adder is used to calculate the in parallel, to achieve faster operation at the expense of more complicated binary adder design. Hence, although there is no carry propagation between the residue channels of different moduli, there is still localized carry propagation occurring within the residue channel since the residues themselves are encoded in BC and hence operate based on the binary adders.

Furthermore, the modulo adders of Fig. 4.1.1 needs the carry bit of the second binary adder in order to generate the final result. As such, the performance of these modulo adder depends very much on the carry propagation performance of the underlying binary adders used in the implementation. For example, the binary adders of Fig. 4.1.1 can be based on the ripple carry full

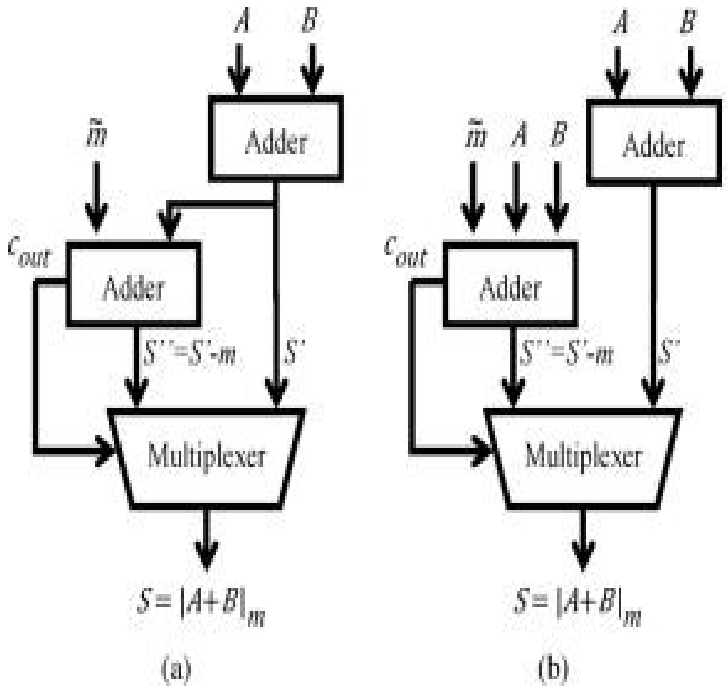


Figure 4.1.1 Modulo-m Adders for BC Residues

adder which is fast but at much higher logic gates cost. As the carry bit propagation originates from the BC format used in encoding the residues, we propose an approach similar to earlier TCR to tackle this inefficiency by using an alternative encoding format, the one-hot code, to be used for the modulo adder.

The OHC based modulo adder can be implemented using shifter-based circuit to perform the addition operation without the carry propagation property.

Furthermore, the modulo operation can be performed in a very simple manner compared to the BC version. As such, it is used in conjunction with the proposed TCR base DA-RNS described earlier.

Example: (5+6) mod 7=?

1 0 1

\+ 1 1 0

1 1 0 1 = 1 1

(1 1 + 0) mod 7 = 4

4.2 THERMOMETER CODE:

Number that is encoded using the TC format number system is equivalent to base-1 binary number system where the magnitude of an integer datum is represented by the number of '1' bits in the bit pattern of the number.

With TC, 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. As an example, the numbers

Table (1): TC REPRESENTATION

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

As such, TC number system is not a place-value number system and the position of the '1' bit within the bit pattern of the number is not relevant. In practice, a TC format integer always has all its '1' bits grouped in a sequence that resides toward one end of the datum, followed by a sequence of '0' bits such that the total number of bits used corresponds to the DR of the particular system.

For example, for a TC encoded number that has a DR of 0 to 10, the TC representation of a decimal 7 would be encoded with ten '1' bits and five '0' bits as 00011111111 where the subscript 1 is added to indicate explicitly that the binary bit pattern is of base-1 nature.

In comparison, with a base-2 BC format number system that possesses the same DR of 0 to10, the decimal 7 would be encoded by using only 4 bits as 01112. Hence TC format is usually not efficient for wide DR system but is suitable for RNS based system since the DR of the residues are usually very much smaller compared to the

binary number they represent. Residues that are represented using TC format will be known as thermometer code residue (TCR) hereafter

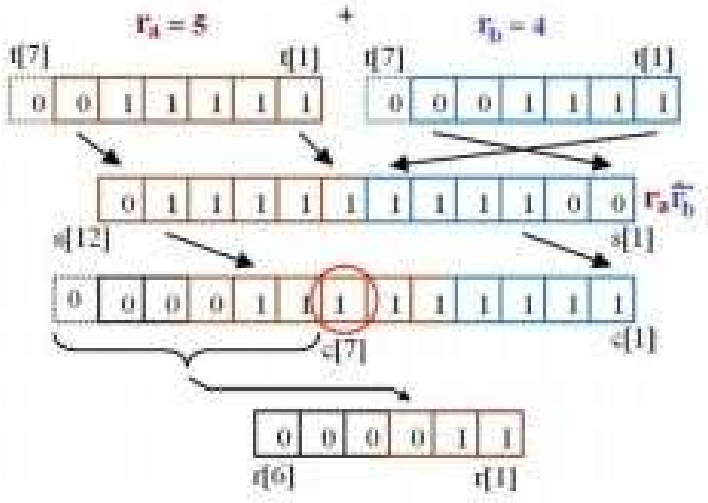


FIGURE 4.2.1: Modular Addition for TC Based Residues

By using TCRs, modular addition of two operand consists of bit concatenating and bit shifting of the concatenated augend and the addend, followed by taking the modulo of the intermediate sum to retain the result within the original length of the modular thermometer residue. The modular thermometer residue addition can be described in a sequence of steps that in actuality is performed by hardwired circuitry in one automatic sequence of operation as will be described later.

Fig. 4.2.1 is an example used to illustrate the principle of the modular addition using thermometer residues. The two input operands are r_a= 5 and r_b= 4, both of which are 7-bit TCRs of modulus 7 (indicated as modulus-7 hereinafter). Each residue is hence indicated as consisting of 7 bits, t [7:1]. However, in RNS, the maximum value of a residue is always 1 less than the value of its modulus, which means that t [7] will always be '0' for r_a and r_b.

Hence t [7] is would normally not be needed to be included in actual implementation to reduce transistors count, as bit of '0' does not contribute to the value in a base-1 TC number system. The two residues are then concatenated to form a 12-bit s[12:1] intermediate sum denoted as r_a r_b where r_b are presented with its bits in the reverse order (e.g., by simply hardwiring the positions of its bits in the reverse order) such that all '1' bits of the two thermometer residues reside beside each other as shown. The next step is to shift the intermediate sum s[12:1] to form a normalized thermometer residue, where all its '1'

bits reside toward one end of the TC datum. To do this, the 12-bit intermediate sum is logically shifted toward the c [1] position in the next stage by removing the '0's situated starting at s [1], with extra '0's padded at the other end to retain the length of the TC datum.

This forms a 12-bit normalized thermometer residue c [12:1], with c [13] (as well as c [14]) having implicit bit values of '0'. Checking the value of bit c [7] of this normalized concatenated thermometer residue indicates that the intermediate sum consists of at least 7 '1', thus exceeding the value of the modulus of 7. The upper half of the intermediate sum is hence selected as the final answer in the form of the 6-bit thermometer residue r [6:1], which corresponds to the expected result for this modular addition of ra and rb: |5+4|7= 2.

Two key points to note in the addition operation just described is first it does not make use of any binary adder as in the case of the conventional base-2 binary modular arithmetic. As such there is no carry bit propagation involved during the addition process. The second is that the operation to take the modulo of the intermediate sum consists of just a single bit test, which is extremely simple to implement as will be shown.

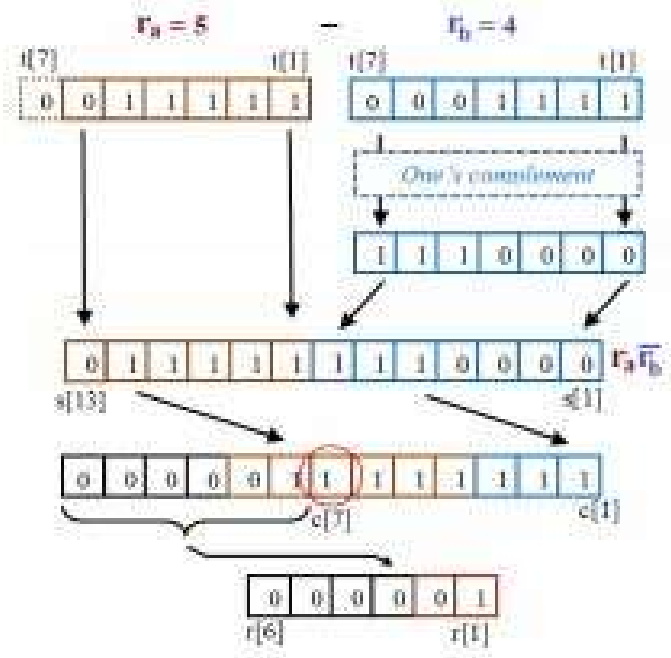


Figure 4.2.2: Modular subtraction for TC based residues

Modular subtraction for TCR consists of concatenating the minuend with the additive inverse of the subtrahend, which is simply obtained as the one's complement of the TCR

subtrahend. Fig. 4.2.2 shows an example of the subtraction operation of two integers $r_a = 5$ being the minuend and $r\dot{b} = 4$ being the subtrahend, both of which are 7-bit modular thermometer residues of modulus-7.

For modular subtraction where the subtrahend is larger than the minuend, the TCR representation also has the advantage of always producing the correct positive result directly.

Consider the case of two integers $r_a = 4 = 00011111$ being the minuend and $r\dot{b} = 5 = 00111111$ being the subtrahend, both of which are 7-bit TCR of modulus-7. As before, subtraction involves getting the additive inverse of the subtrahend to obtain $r_b = 11000001$ followed by the concatenation with r_a to obtain $r_a\, r_b$ and the modulo operation as shown below:

$$|0001111_1 - 1100000_1|_7 = |\,00011111100000_1|_7$$

$$= |000111111_1|_7$$

$$= 0111111_1$$

$$= 6_{10}$$

The final value of decimal 6 corresponds to the expected answer for the modular arithmetic operation which can be derived based on the BC format equivalent as follows:

$$r = |4-5|_7$$

$$= |-1|_7$$

$$= |-1 + 7|_7$$

$$= 6$$

Yet another advantage of the base-1 TCR over the base-2 BC equivalent is that there is no ambiguity for cases when subtrahend has a value of zero, since the one's complement of value zero in thermometer code will give a value equal to that of the modulus, which becomes zero after executing the modulo operation.

Hence the TCR additive inverse of X=0 in modulus-m system is m, which revert to 0 after executing the modulo-m operation. Multiplication in RNS is a well-known complicated issue. A common approach is to use LUT with index transform techniques if needed. Multiplication is hence the main weakness of TCR based system.

However, if the scope of the TCR usage is constraint to normal digital signal processing, where the most common computation is the MAC operation, then there is an elegant and efficient solution to address the TCR's multiplication limitation in the MAC operation. The approach is to make use of the distributed arithmetic (DA) principle [4] that cleverly removes the multiplication involved in the MAC calculation, or the inner-product calculation. Instead, only modular addition is needed to be performed.

In summary, the three fundamental arithmetic operations, addition, subtraction and multiplication can all be performed by just using the modular addition process. As such, the efficiency of the modular adder is of utmost importance in TC based modular arithmetic operation found in many signal processing algorithms and can therefore considerably influence their overall performance. Next section will hence present the design of a modular adder for data based on the TCR format.

Convention RNS representation of the residues themselves continue to use the base-2 BC format. In this paper, a base-1 thermometer code (TC) format based RNS method is presented to overcomes the various limitations of the base-2 BC based system as shall be described. The motivation of using the TC based residues introduced here is also due to the availability of a novel RNS based zero-crossing folding ADC [1] which directly converts its input analog signal to digital outputs in the RNS's residues representation that are encoded in TC format as illustrated Fig. 4.2.3. Hence there is no binary to RNS forward conversion involved in generating the RNS based data, which is one of the main practical hindrance for RNS based signal processing in real world applications.

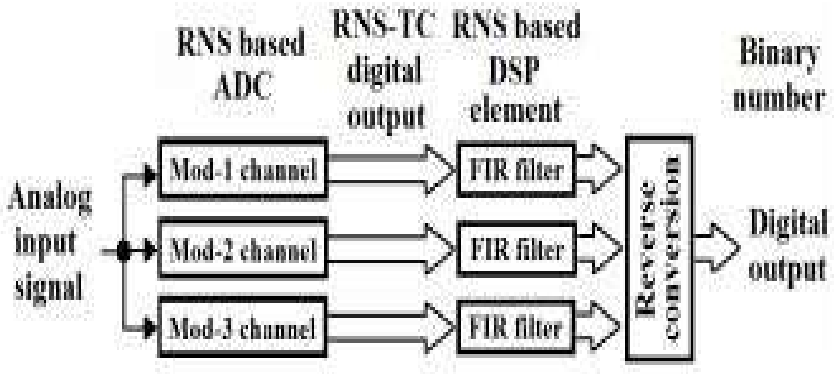


Figure 4.2.3 RNS ADC with RNS signal processing

The RNS based ADC hence produces multiple channels of residues each encoded in TC format. Instead of converting these residues to conventional binary representation, it is

proposed that the TC modular arithmetic operations presented here are used to process the TC data directly, such as performing the digital filtering using FIR filter as shown in Fig. 1, further reducing the overhead of performing the TC to BC encoding process.

It is to emphasize that the TC based method exploits the small dynamic range property of the residues by representing them using the TC format where the number of bits required would not be excessive and hence remains practical. Hence practical implementation would be most feasible by using multiple smaller size moduli, rather than a few medium size moduli.

4.2.1 CIRCUIT REALISATION OF TCR:

Modular addition of two TCRs involves the concatenating of the two residues, follow by logical shifting of the intermediate sum and then a bit test for the modulo operation. Fig. 5 shows the block diagram of a practical circuit that can be used as a TCR modular adder based on the same principle, but implement in a slightly different manner for better hardware efficiency. Two residues ra and rb are applied to the circuit input.

k

Instead of concatenating the two TCRs and then normalizing the intermediate sum by removing all the '0' bits in one of the operands, the modulo adder circuit pads appropriate number of '1' bits to the TCR operand ra, based on the value of the other TCR ra. This hence changes the number of '1' bit(s) in rb based on the value of rb, effectively performing the TC concatenating-based addition operation.

Hence the addition process operates based on bit incrementing, which can be simply implemented using the log shifter circuit as shown in below figure, which is for a modulus-7 system.

The shifter circuit consists of multiple groups of multiplexers connected in series. The multiplexers in a group are connected in such a way that upon the assertion of their select control signal S(x), the group will perform a 'vertical' logical bit shift of x=2^k , where k=0,1,2,…, to produce the concatenated output c[1:2m-2]. With appropriate encoded shift control signals S(x), the total number of groups need to perform up to m bits of bit shifting in a log shifter is equal to log2m

(E.g. for m=7 in Fig. 4.2.6, number of shifter groups needed =3).

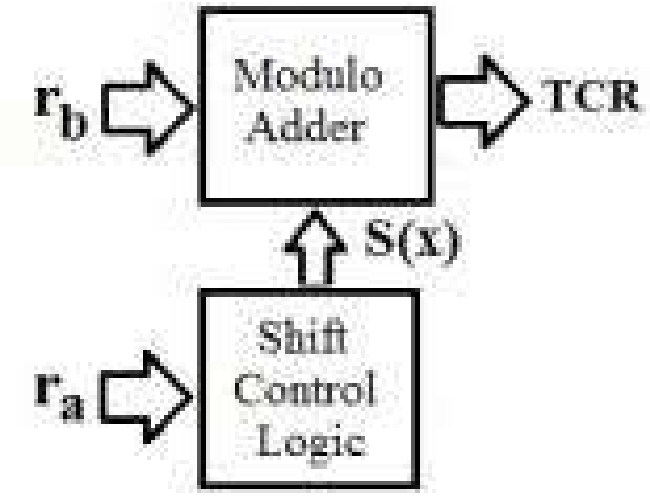


Figure 4.2.4: Modular adders for TCR'S

The bit shifting is controlled by a shifter control logic circuit that decodes the number of '1' bits in ra. As ra is in the TC format while the S(x) is in the BC format, this circuit hence performs a TC to BC encoding operation, which can be expressed in Boolean expressions as follows.

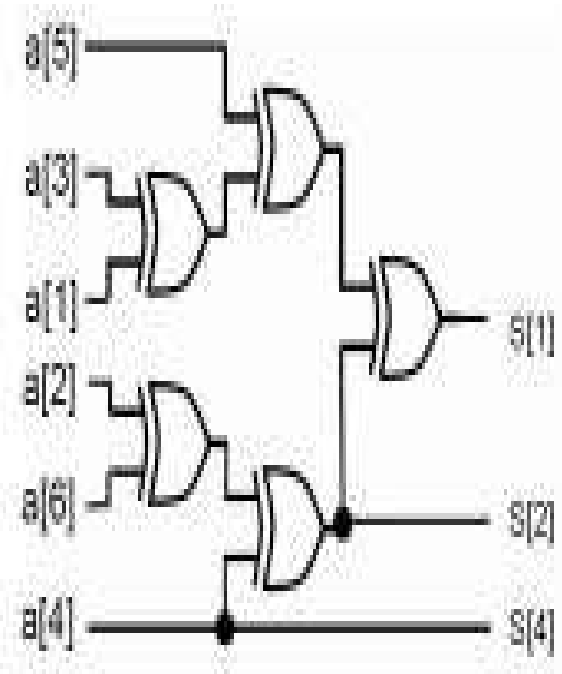


Figure 4.2.5: Shifter Control Circuit for Modulis-7 Adder

$$S[1] = a[1] \oplus a[2] \oplus a[3] \oplus a[4] \oplus a[5] \oplus \ldots$$
$$S[2] = a[2] \oplus a[4] \oplus a[6] \oplus a[8] \oplus \ldots$$
$$S[4] = a[4] \oplus a[8] \oplus a[12] \oplus \ldots$$
$$S[8] = a[8] \oplus a[16] \oplus \ldots$$

Hence the shift control logic circuit can be implemented by combinatorial logic circuit such as the one shown in Fig. 4.2.5, which is to be used for the modulo-7 adder as shown in Fig. 4.2.6.

For corresponding TC modular adder, the number of multiplexers needed per group to concatenate the input operands varies between m and (2m-2), with 3 or 4 groups of multiplexers to perform the bit shifting. In addition, one group of multiplexers consists of (m-1) units are used to perform the modulo operation. The shifting control circuits are assumed to be implemented based on the Boolean expression derived logic gates such as the one shown in Fig 4.2.5

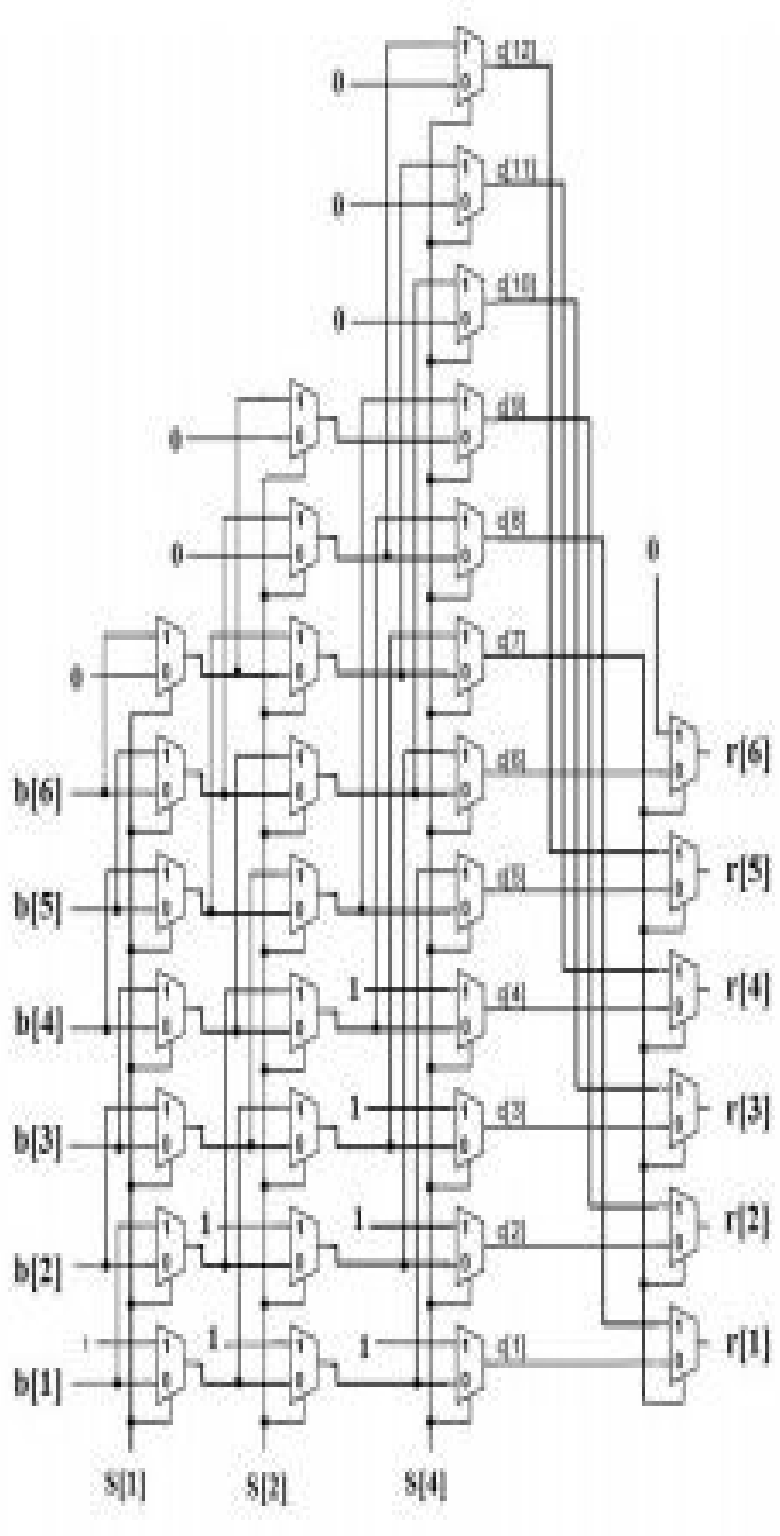


Figure 4.2.6: Modulo Adder for modulus-7 TCR's

4.3 ONE HOT CODE:

A one-hot code consists of n bits, but can only have 1 bit asserted at any one time. Hence it is also known as 1-out-of-n encoding scheme. One-hot code is normally used for decoding the address bits for LUT purpose. When it is used to represent the residue in RNS, it is referred to as the one-hot residue (OHR) [11].

In the one-hot code format, the value of the residue corresponds directly to the asserted bit position. Compared to the TCR, the OHR uses one extra bit in order to encode the value corresponding to 0. For example, a residue digit with value of 5 in a modulus-7 system would be encoded as {0100000}, while a value of 0 would be encoded as {0000001}.

TABLE (2): OHR REPRESENTATION

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

The OHC is usually used to address look up tables (LUTs), and at the output of some linear circuits like FIR filters. K+1 bit is 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 2 shows the numbers between 0 and 7 encoded in an OHC.

While the value of an OHR is intuitively clear from its bit pattern, it lacks

formal mathematical properties (e.g. base-1, base-2) and poses difficulty in using it for general mathematical purposes. Nevertheless, its unique usefulness for representing residues lies in the automatic execution of the modulo operation when performing addition and subtraction based on circular shifting technique.

Consider two modulus-7 residues and which have numerical value of 4 and 5 respectively expressed in OHR format.

ra = 0010000

rb = 0100000

The modular sum of these two operands can be found by executing a circular shift operation to one of the operands, based on the value of the other operand. In this example, will be circular shifted five positions (based on the value of) to its left (assuming its highest value bit position is on the left) which will make its '1' bit wrap around to the lowest value bit position such that eventually becomes {0000100}. This implies a numerical value of 2, which is consistent with This unique property exhibits a very efficient way to perform automatic modulo operation during the addition.

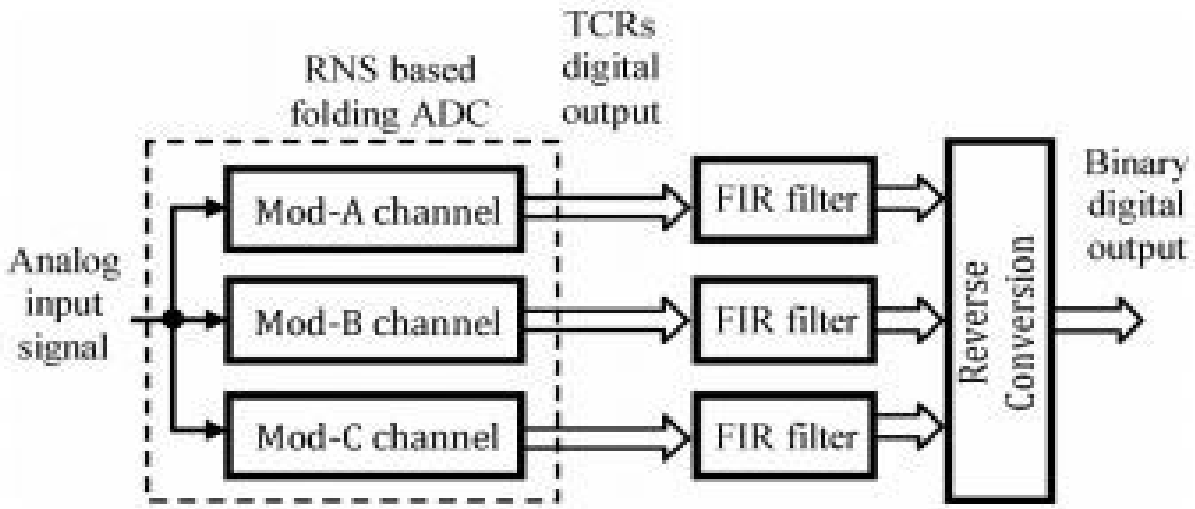


Figure 4.2.7: RNS-ADC with pre-processing FIR filter

The proposed system also uses modulo adder based on the one-hot code (OHC) format that provides a simple and fast method to perform the modulo-accumulation encountered in DA. This combination provides an elegant solution to practical problems faced in the implementation of conventional BC based DA-RNS systems. As

such, it enables benefits in using RNS with the DA to be truly realized in very efficient manner using simple circuit design.

One important practical consideration in using RNS based modular arithmetic is the forward conversion required to first convert a conventional number to its residues. This is likely to be a costly operation and will hinder wide adoption of RNS in real world applications. Our motivation for using the TC encoded residue (TCR) is the availability of a novel RNS based folding ADC where the data generated during the conversion are inherently in the TCR format.

As such, there is no extra overhead needed to first convert a datum to its RNS representation. Further arithmetic operations required in signal processing can hence be performed in TCR directly, before it is eventually converted to the conventional binary representation.

Fig.4.2.7 illustrates this concept where individual RNS based FIR sub-filter is integrated within each modulus channel of the RNS ADC to perform signal pre-processing before eventually outputting the data in conventional binary number representation. Nevertheless, the proposed scheme can also be used with conventional systems where the binary representation is first converted to the TC based RNS representation during the forward conversion, instead of the usual BC based RNS representation.

CIRCUIT REALISATION OF ONE HOT METHOD:

The below figure shows the circuit schematic of a modulo-7 adder used in the modular accumulator based on this design. The adder consists of multiplexers arranged to formed a log based circular shift. The input a[n] and the output OHR[n] are both OHR encoded while the b[n] input is BC encoded.

The BC encoded b[n] controls the amount of circular shift that is applied to a[n]. This effectively executes $|A+B|_7$, with the modulo 7 operation automatically performed as the OHR bits encoder to present the output in the conventional binary representation wrap around. The OHC format at the output is also convenient if it is to be used to

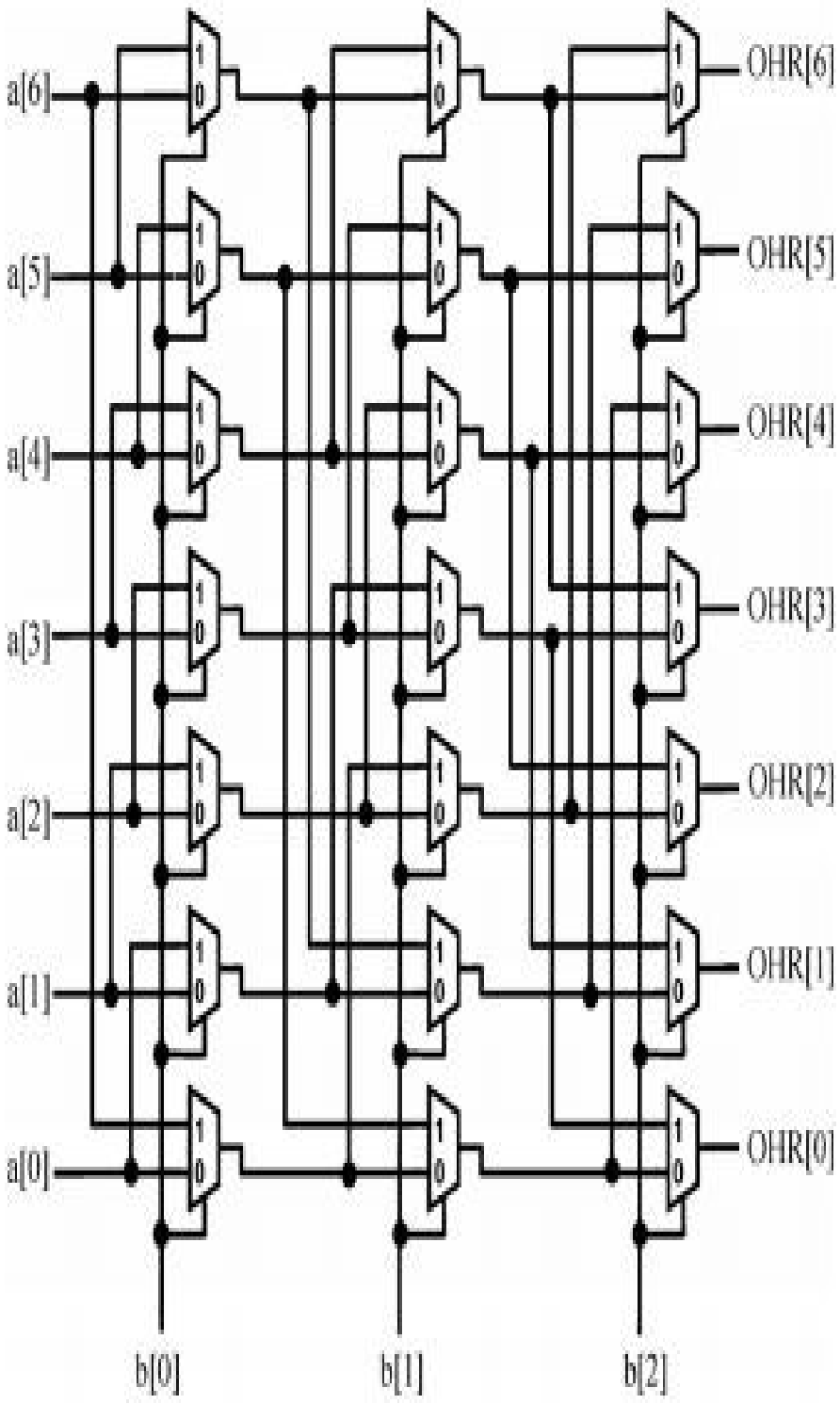


Figure 4.2.8: OHR based modulo-7 adder

address a LUT such as a residue-to-binary encoder to present the output in the conventional binary representation.

Compared to the BCR modulo adder, this shifter based OHR modulo adder implementation is much faster as there is no logic gate delay involved in the operation, nor does it have carry propagation issues. The operating speed is solely determined by the delay of the signal passing through the multiplexers.

In addition, the number of transistors used to implement the shifter based modulo adder is lower when compared to that in BCR modulo adder based on the ripple carry adders, which is the most area efficient (but slowest) implementation for a binary adder.

CHAPTER-5

PROPOSED METHODS

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.

5.1 PROPOSED THERMOMETER METHOD:

An important aspect to apply the proposed method to add two modulo m residues, ($0 \leq A,B < m$) 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, etc. After that, a 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 $A+B \geq m$ 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:

$$Sum= \begin{cases} SUM1 \ if \ cl=1 \ (A+B \geq m) \\ SUM0 \ if \ cl=0 \ (A+B<m) \end{cases} \ldots\ldots\ldots\ldots\ldots\ldots (1)$$

where

$$cl=\vee i=1 m-1 (ai \wedge bm-i) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (2)$$

$$SUM1=(\sum_{i=1}^{m-1} ai \wedge bm-i) -1 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (3)$$

$$SUM0=(m-1) -(\sum_{i=1}^{m-1} \overline{ai \vee bm-i}) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (4)$$

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

$$A=am-1 \ldots a2a1=0 \ldots 0_{\smile} \ azeros \quad 1 \ldots 1_{\smile} \ aones \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (5)$$

$$B=bm-1 \ldots b2b1=0 \ldots 0_{\smile} \ bzeros \quad 1 \ldots 1_{\smile} \ bones \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (6)$$

The total number of bits, i.e. $azeros$+$aones$ or $bzeros$+$bones$, 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. $ai \wedge bm-i$:

$$A=\underbrace{0\ldots0}_{a\_zeros}\quad 1\ldots1_{\sqcup}\, k\quad 1\ldots1_{\sqcup}\, aones-k \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (7)$$

$$Breverse=1\ldots1_{\sqcup}\, bones-k\quad 1\ldots1_{\sqcup}\, k0\ldots0_{\sqcup}\, b\_zeros\ldots\ldots\ldots\ldots \quad (8)$$

$$A \wedge Breverse=0\ldots0_{\sqcup}\, a\_zeros\quad 1\ldots1_{\sqcup}\, k0\ldots0_{\sqcup}\, b\_zeros \ldots\ldots\ldots\ldots\ldots \quad (9)$$

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:

$$\begin{cases} A+B<m \text{ if } k=0 \\ A+B=m \text{ if } k=1 \\ A+B>m \text{ if } k>1 \end{cases} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (10)$$

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, it can be rewritten as:

$$A \wedge Breverse=(am-1 \wedge b1)\ldots(a2 \wedge bm-2)\,(a1 \wedge bm-1) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (11)$$

To detect the existence of at least a bit 1 among the resulting bits in (11), the following formulation can be adopted:

$$cl=(am-1 \wedge b1) \vee \ldots \vee (a2 \wedge bm-2) \vee (a1 \wedge bm-1) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (12)$$

Hence, (2) is achieved, i.e. cl=1 and cl=0 mean that $A+B \geq m$ and $A+B<m$, respectively.

It 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:

$$SUM1=0\ldots0_{\sqcup}\, b\_zeros0\ldots0_{\sqcup}\, a\_zeros1\ldots1_{\sqcup}\, k-1\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (13)$$

When k=1, $A+B$ is equal to the modulo m. The number of *bits* with the value 1 in (11) is equal to the number of 1s of the modular addition plus one. There is no situation where two non-sequential bits 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 bit in should be counted using the formula $\Sigma ai \wedge bm-im-1i=1$, and then decreased by one to achieve the final sum, corresponding to $SUM1$ in (3).

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:

$$A=0\ldots0_{\smile}\, azeros-k \quad 0\ldots0_{\smile}\, k \quad 1\ldots1_{\smile}\, aones \ldots\ldots\ldots\ldots\ldots\ldots (14)$$

$$Breverse=1\ldots1_{\smile}\, bones \quad 0\ldots0_{\smile}\, k \quad 0\ldots0_{\smile}\, bzeros-k \ldots\ldots\ldots\ldots (15)$$

$$A \vee Breverse=1\ldots1_{\smile}\, bones \; 0\ldots0_{\smile}\, k \quad 1\ldots1_{\smile}\, aones \ldots\ldots\ldots\ldots\ldots\ldots (16)$$

The sum in the case $A+B<m$ will have $(aones+bones)$ 1 bits together with k bits of 0, i.e.:

$$SUM0=0\ldots0_{\smile}\, k1\ldots1_{\smile}\, bones \quad 1\ldots1_{\smile}\, aones\ldots\ldots\ldots\ldots\ldots\ldots\ldots (17)$$

Alternatively, the number of zero bits in $SUM0$ can be achieved by counting the number of overlapping zero bits between A and the reversed version of B using the bitwise NOR operation $(\sum_{i=1}^{m-1} \overline{ai \vee bm-i})$, which can be represented as:

$$\overline{A \vee Breverse}= 0\ldots0_{\smile}\, bones1\ldots1_{\smile}\, k0\ldots0_{\smile}\, aones \ldots\ldots\ldots\ldots\ldots\ldots\ldots (18)$$

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

Example 1: Let us assume that $A=3$, $B=5$ and compute |3+5|7 . Both numbers are represented in TCR in Fig. 3. It can be seen that $(a2 \wedge b5)$ and $(a3 \wedge b4)$ are equal to one, which means that cl=1 and $A+B \geq m$. To achieve the sum, according to (3):

$SUM1=(\sum_{i=1}^{6} ai \wedge bm-i)-1 = 2-1=1$

According to (13), the TCR representation of the result is: $SUM1$=000001

which is the correct result (|3+5|7=1).

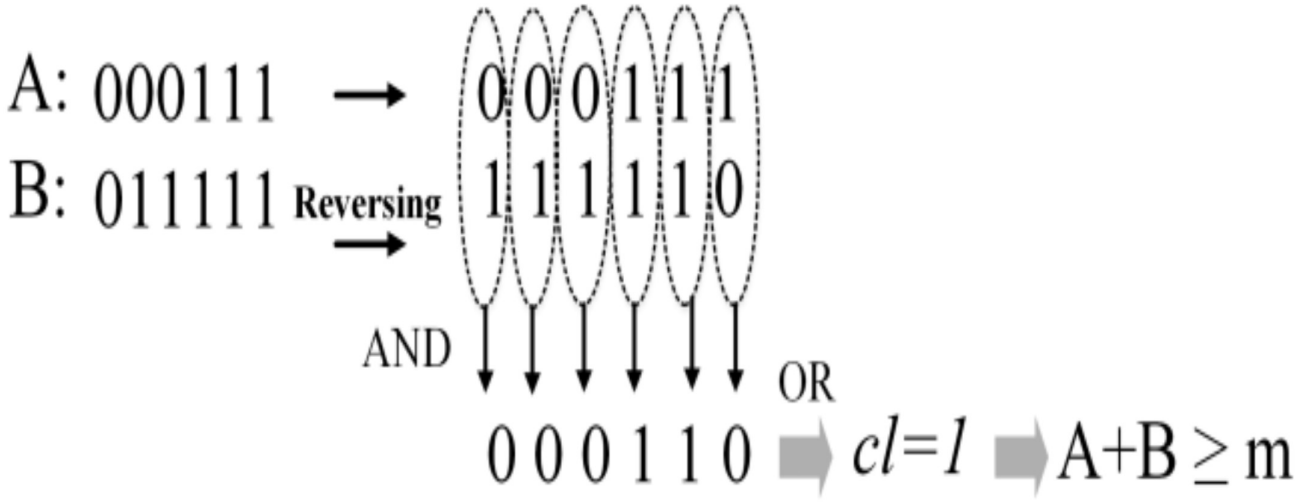


Figure 5.1.1 *Thermometer addition example when A + B >=m*

Example 2: Let us assume that *A=1* and *B=2*. Both numbers are represented in TCR in Fig. 5.1.2 It can be seen that none of the outputs of the AND gates take the value '1'. This means *cl*=0, and leads to the conclusion that *A+B<m*.

According to the above equations, the sum is:

$SUM0 = (7-1) - (\sum_{i=1}^{6} \overline{ai \vee bm-i}) = 6-3=3$

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

$SUM0$=000111

which is the correct result (|1+2|7=3).

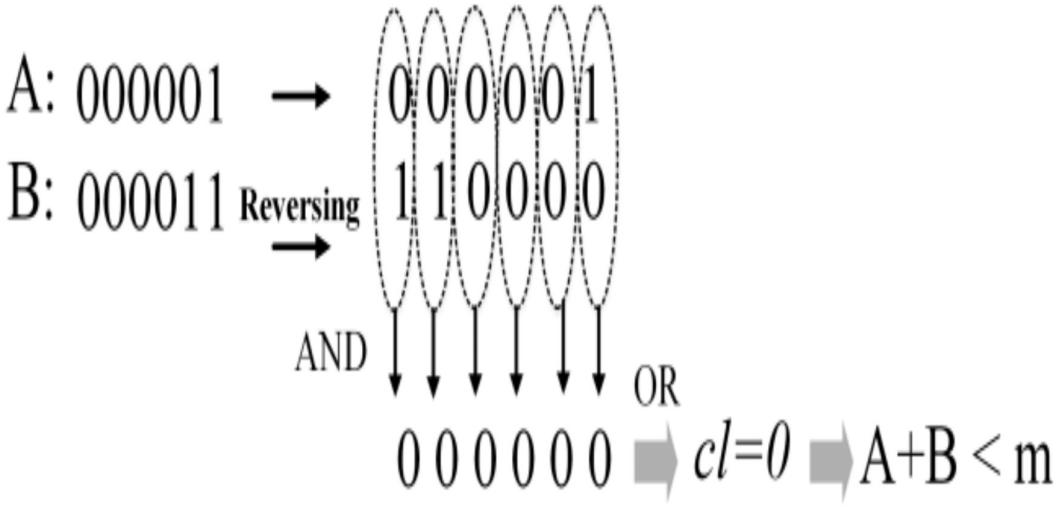


Figure 5.1.2: Thermometer addition example when A + B <m

The proposed design for the TCR based modulo adder is illustrated in Fig. 5, for the case when m=7. When $A+B>=m$, the result is stored in $SUM1$, while for the other case, $A+B<m$, the result is placed in $SUM0$. As mentioned before, if at least one of the *AND*s' 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. *L0* signals are connected to the NOR gate with 6 inputs. Based on the output of this gate (*sel*), $SUM0$ or $SUM1$ is selected (*sel* is the complement of *cl*). It should be noted that some multi-input gates in Fig. 5.1.3 can be implemented using tree structures of 2-input gates without impacting the delay. Let us analyze the operation of the circuit to compute $SUM0$ and $SUM1$.

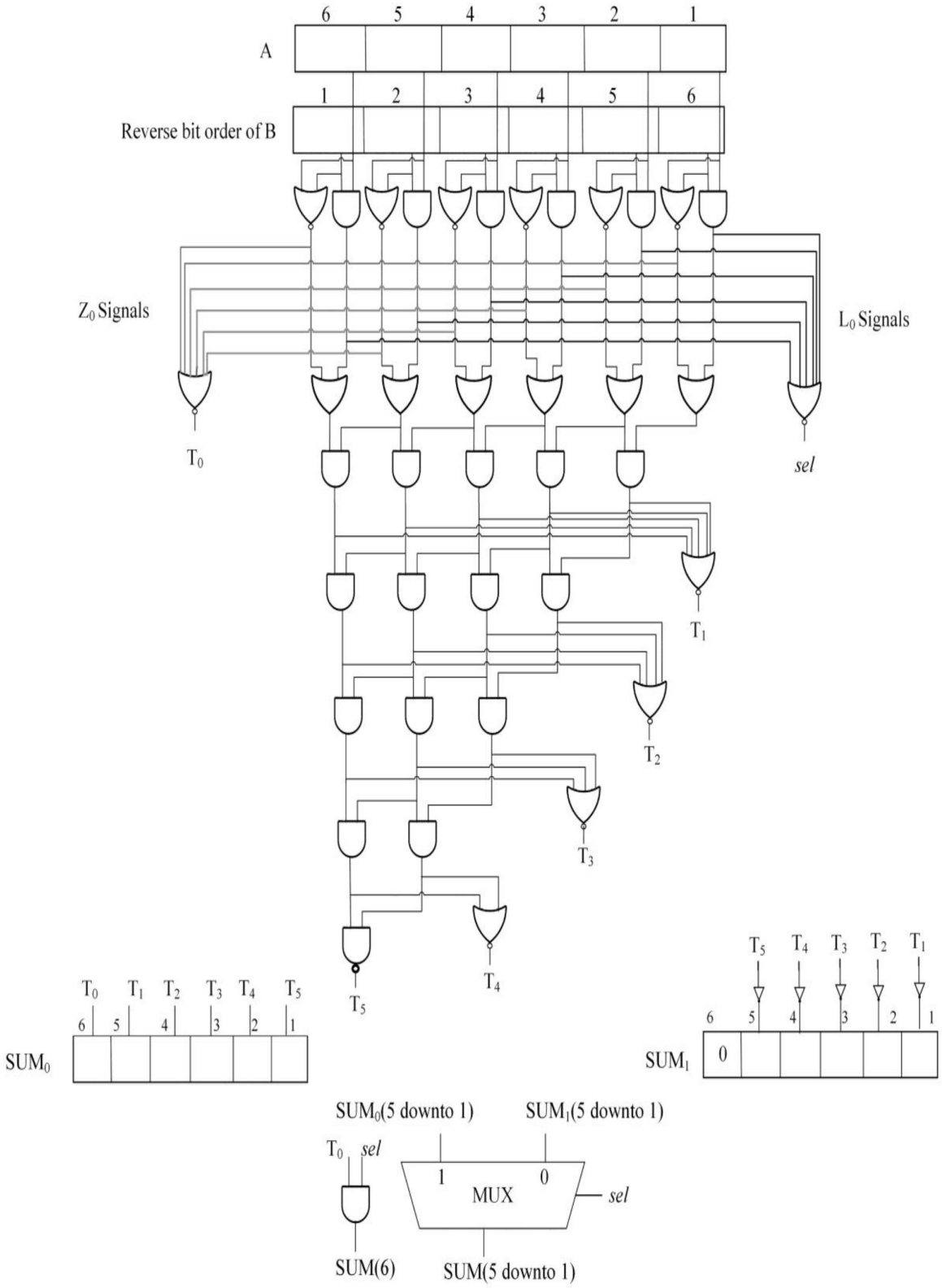


Figure 5.1.3: Proposed TCR based modulo-7 adder

$SUM0$ circuit :

As observed in Fig. 5.1.3, *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 $SUM0$.

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 side, the value of the left bit of the $SUM0$ 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 one-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 5 inputs (*T*1) specifies the 5th bit of the $SUM0$. In the same way, if at least 3 signals of *Z*0 become one, this means that there are at least 3 bits of the result with the value 0, etc. 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'.

$SUM1$ circuit :

Whenever *A+B>=m*, the result of the modular addition of *A* and *B* is $SUM1$. 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 $SUM1$. 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 5 inputs (*T*1) takes the value zero.

Since in TC, ones are placed sequentially at the right-hand side, the value of the right most bit of $SUM1$ 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 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 $L0$ signals are used to compute the MUX' selector.

If at least one of these signals takes the value of one, the output of the 6-input NOR gate, which generates the *sel* signal, becomes 0, selecting $SUM1$ as the final result. Otherwise, when all six signals are zero, $SUM0$ will be outputted. The structure of the proposed TCR adder in Fig. 5.1.3 can be easily generalized for a general value of m.

5.2 PROPOSED ONE HOT METHOD:

Herein, we use the bit indexing of [8], 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 $m2$ possible combinations of A and B.

Lemma 2: In the One-Hot modular addition, exactly m combinations of all the possible $m2$ 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 (ak=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 $i-k$ ($bi-k$=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 $m+i-k$ ($bi+m-k$=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 SUMi=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 (SUMi has the value of one).

$$SUMi=(\bigvee_{K=0}^{I} (ak \wedge bi-k)) \vee (\bigvee_{K'=I+1}^{M-1} (ak' \wedge bi+m-k')) \quad \text{..... (19)}$$

The first term of (19) 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 position i and j), two combinations of A and B are possible: 1) $Ai=1$ and $Bj=0$; and 2) $Ai=0$ and $Bj=1$.

Therefore, for detecting the $m2–m$ combinations of $A \neq B$, m 2-input OR gates are used in the first level and $(m2)=m!(m-2)!2!=m(m-1)2$ 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 $(m2)*2$ possible combinations are covered.

Fig. 5.2.1 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 ORing of the inputs Ai and Bi, 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 m2–m combinations of A and B), firstly the OR of A and B in the first level are 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 two combinations of $A \neq B$.

With this approach, all the $m2$ combinations of modular additions of A and B in one-hot coding 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 (20), where i is the bit number.

$$SUMi=\begin{cases} AND(i\backslash 2)\ if\ m\ is\ odd, A=B\ and\ i\ is\ even \\ AND(m+i\backslash 2)\ if\ m\ is\ odd, A=B\ and\ i\ is\ odd \end{cases} \quad \text{(20)}$$

In general, when m is odd, The SUM can be calculated with (21) and (22) ((21) is used for the bits with even index and (22) is used for the bits with odd index).

$$SUMi=AND(i\backslash 2)\vee(\bigvee_{k=0}^{i\backslash 2-1}(OR(k)\wedge OR(i-k)))\vee(\bigvee_{k'=i+1}^{|m+i\backslash 2|}(OR(k')\wedge OR(m+i-k')))$$

(m is odd and i is even)…… (21)

$$SUMi=AND(m+i\backslash 2)\vee(\bigvee_{k=0}^{[i\backslash 2]}(OR(k)\wedge OR(i-k)))\vee(\bigvee_{k'=i+1}^{(m+i\backslash 2)}(OR(k')\wedge OR(m+i-k')))$$

(m is odd and i is odd)…… (22)

According to Lemma 2, only m combinations from all the possible $m2$ combinations of A and B result in the same output. (21) and (22) confirm this point.

TABLE 3

NUMBER OF COMBINATIONS COVERED BY (21)

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

(m is odd and i is even)

Above table shows all combinations that cause $SUMi$=1 in (21). The total number of combinations that are covered by (21) (n(21)) is shown in (23).

$$n(21)=1+i+(\lfloor m+i2\rfloor - i)*2$$
$$= 1+i+(\lfloor m2\rfloor+i2-i)*2=1+(m-12)*2$$
$$=m \text{ (m is odd and i is even)} \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (23)$$

Table IV shows all combinations that lead to $SUMi$=1 in (22). The total number of combinations covered by (22) (n(22)) is shown in (24).

$$n(22)= 1+(\lfloor i2\rfloor+1) *2 +((m+i2)-1 -i)*2$$
$$= 1+((i-12)+1)*2 +(m-i-2)$$
$$=m \text{ (m is odd and i is odd)} \ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (24)$$

TABLE 4

NUMBER OF COMBINATIONS COVERED BY (22)

Part of (33)	Number of combinations
$AND(\frac{m+i}{2})$	1
$((V_{k=0}^{\lfloor \frac{i}{2} \rfloor}(OR(k) \wedge OR(i-k))$	$(\lfloor \frac{i}{2} \rfloor + 1)*2$
$(V_{\underbrace{k'=i+1}_{i<m-2}}^{(\frac{m+i}{2})-1} (OR(k') \wedge OR(m+i-k'))$	$((\frac{m+i}{2}) - 1 - i)*2$
(m is odd and i is odd)	

2) even *m*

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

$$SUMi= \begin{cases} (AND(i\backslash 2) \vee AND(m+i\backslash 2)) \\ \quad if\ m\ is\ even, A=B\ and\ i\ is\ even \\ 0 \quad if\ m\ is\ even, A=B\ and\ i\ is\ odd \end{cases} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \quad (25)$$

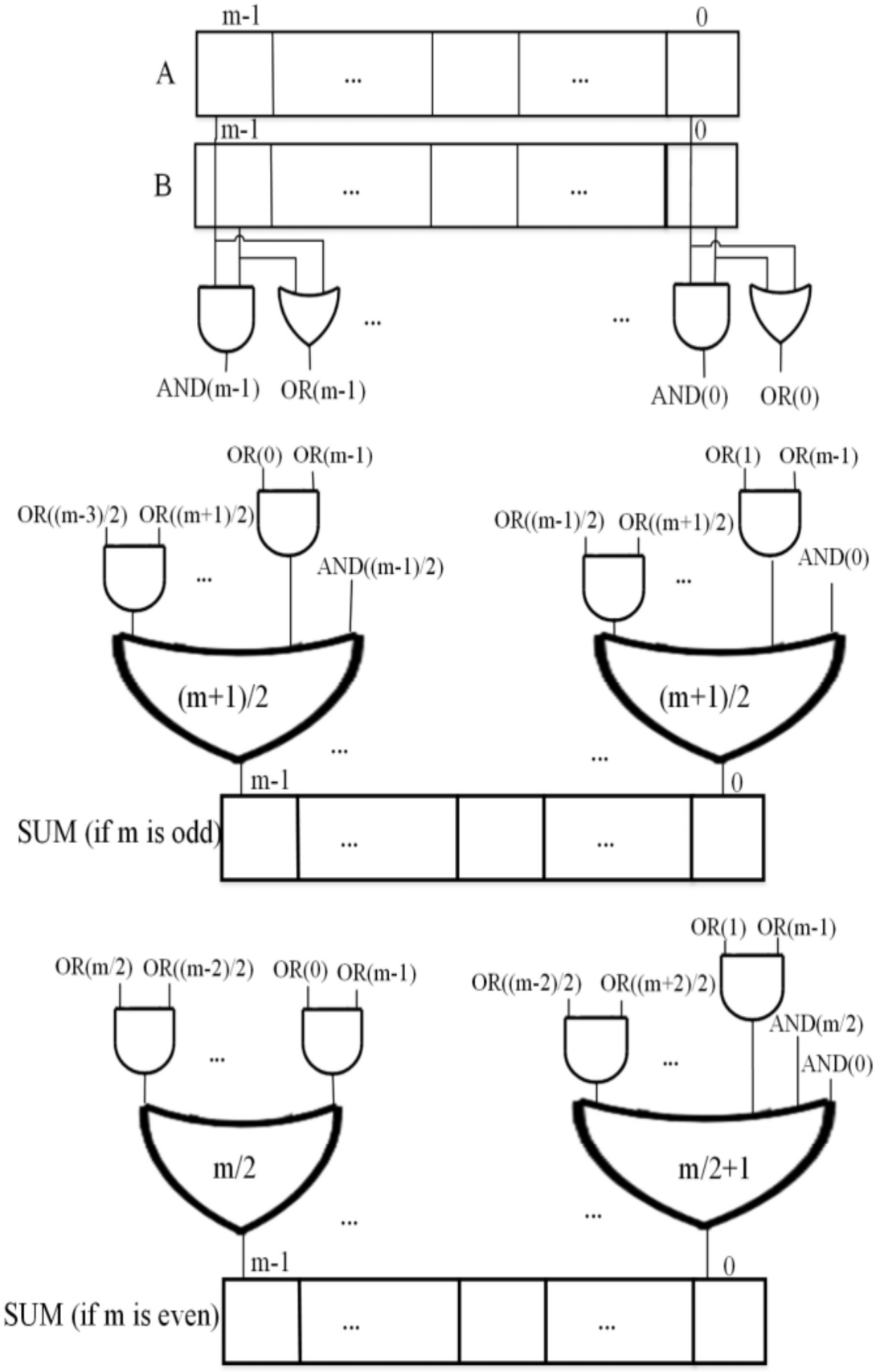


Figure 5.2.1 :Proposed OHR modulo adder for general m.

In general, when *m* is even, the SUM can be calculated from (26) and (27) ((26) is used for the bits with even index and (27) for the bits with odd index).

$$SUMi=AND(i\backslash 2)\vee AND(i+m\backslash 2)\vee(\bigvee_{k=0}^{i\backslash 2-1} \; \sqcup \; i\backslash 2-1\geq 0(OR(k)\wedge OR(i-k)))\vee(\bigvee_{k'=i+1}^{[m+i\backslash 2]} \; \sqcup \; i\leq m-2(OR(k')\wedge OR(m+i-k')))$$

(m is even and i is even)…… (26)

$$SUMi=(\bigvee_{k=0}^{[i\backslash 2]} \; (OR(k)\wedge OR(i-k)))\vee(\bigvee_{k'=i+1}^{[m+i\backslash 2]} \; (OR(k')\wedge OR(m+i-k')))$$

(m is even and i is odd)…… (27)

Equations (26) and (27) also cover *m* combinations of *A* and *B*.

Example: (21) and (22) are used to implement an OHR-based adder modulo *m*=7.The bits of even index of SUM are produced by (21):

$SUM0$=*AND (0)*$\vee$ ($\bigvee_{K'=0+1}^{[7+0\backslash 2]} \; (OR(k')\wedge OR(7+0-k'))$)

=*AND (0)* $\vee$ *(* $OR(1)\wedge OR(6)$*)* $\vee$ *(* $OR(2)\wedge OR(5)$*)* $\vee$ *(* $OR(3)\wedge OR(4)$*)*

$SUM2=AND(2\backslash 2)\vee(\bigvee_{K=0}^{2\backslash 2-1} \; (OR(k)\wedge OR(2-k)))\vee(\bigvee_{K'=2+1}^{[7+2\backslash 2]} \; (OR(k')\wedge OR(7+2-k'))$)

=*AND (1)* $\vee$ *(* $OR(0)\wedge OR(2)$*)* $\vee$ *(* $OR(3)\wedge OR(6)$*)* $\vee$ *(* $OR(4)\wedge OR(5)$*)*

$SUM4=AND(4\backslash 2)\vee(\bigvee_{K=0}^{4\backslash 2-1} \; (OR(k)\wedge OR(4-k)))\vee(\bigvee_{K'=4+1}^{[7+4\backslash 2]} \; (OR(k')\wedge OR(7+4-k'))$)

=*AND (2)* $\vee$ *(* $OR(0)\wedge OR(4)$*)* $\vee$ *(* $OR(1)\wedge OR(3)$*)* $\vee$ *(* $OR(5)\wedge OR(6)$*)*

$SUM6=AND(6\backslash 2)\vee(\bigvee_{K=0}^{6\backslash 2-1} \; (OR(k)\wedge OR(6-k))$)

=*AND (3)* $\vee$ *(* $OR(0)\wedge OR(6)$*)* $\vee$ *(* $OR(1)\wedge OR(5)$*)* $\vee$ *(* $OR(2)\wedge OR(4)$*)*

The odd bits of SUM are obtained from (33):

$SUM1=AND(7+1\backslash 2)\vee(\bigvee_{k=0}^{1\backslash 2} \; (OR(k)\wedge OR(1-k)))\vee(\bigvee_{k'=i+1}^{7+1\backslash 2-1} \; (OR(k')\wedge OR(7+1-k'))$)

=AND(4) $\vee$ ($OR(0)\wedge OR(1)$) $\vee$ ($OR(2)\wedge OR(6)$) $\vee$ ($OR(3)\wedge OR(5)$)

$SUM3=AND(7+3\backslash 2)\vee(\bigvee_{k=0}^{3\backslash 2} \; (OR(k)\wedge OR(3-k)))\vee(\bigvee_{k'=3+1}^{(7+3\backslash 2)-1} \; (OR(k')\wedge OR(7+3-k'))$)

=AND(5) $\vee$ ($OR(0)\wedge OR(3)$) $\vee$ ($OR(1)\wedge OR(2)$) $\vee$ ($OR(4)\wedge OR(6)$)

$SUM5=AND(7+5\backslash 2)\vee(\bigvee_{k=0}^{5\backslash 2} \; (OR(k)\wedge OR(5-k)))$

= AND(5) $\vee$ ($OR(0)\wedge OR(3)$) $\vee$ ($OR(1)\wedge OR(2)$) $\vee$ ($OR(4)\wedge OR(6)$).

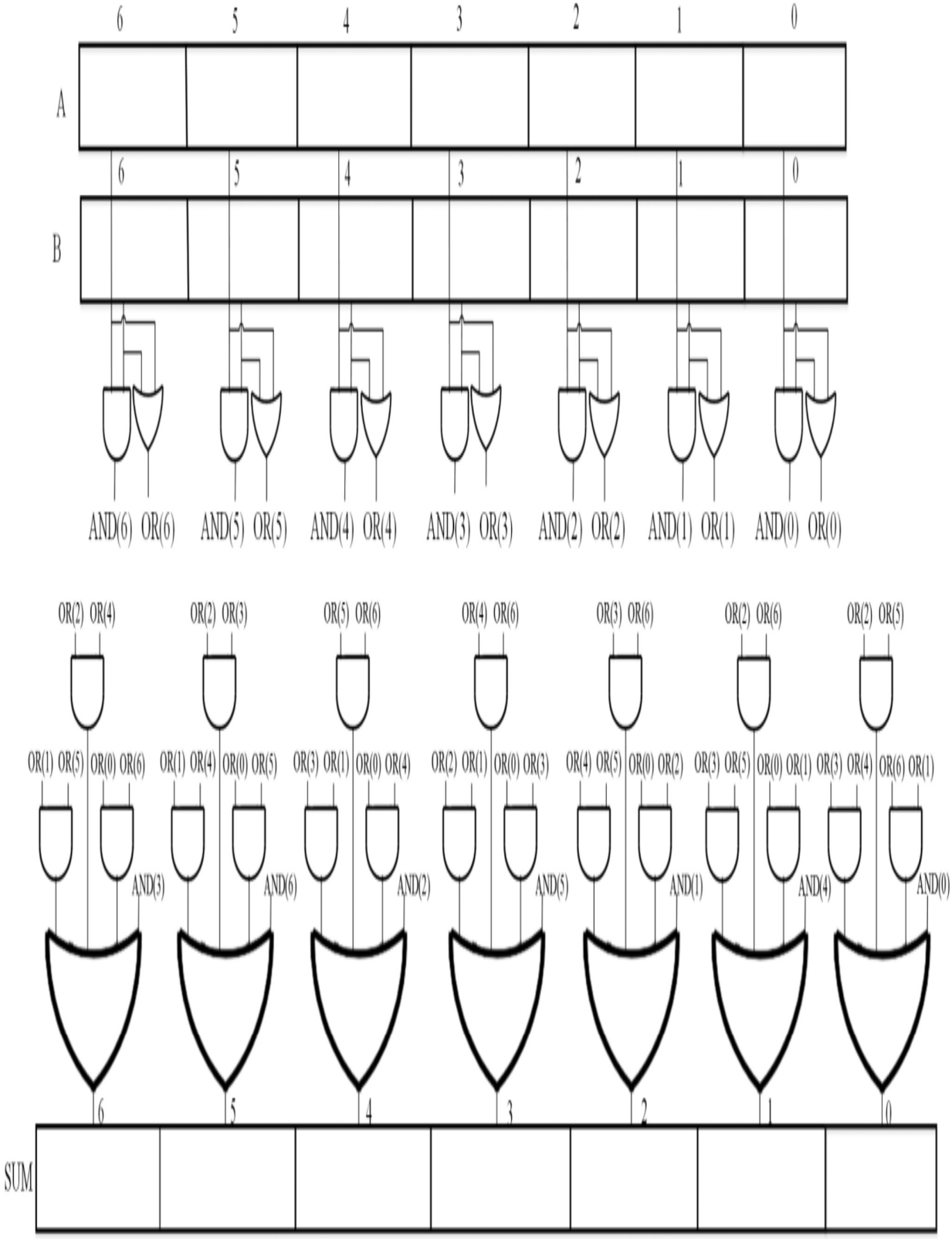


Figure 5.2.2 : Proposed OHR modulo adder for m=7

The proposed modulo adder for m=7 is shown in Fig. 7. This adder has a simpler structure and less delay than [8]. The other feature of this design is that both operands are represented in OHC, while in [8] one of the operands has to be encoded in binary.

CHAPTER 6

PERFORMANCE EVALUATION

In this section, the proposed modular adders are evaluated and compared with previous state-of-the-art TCR and OHR-based adders. With that purpose, both technology agnostic analyses, based on the unit-gate model, 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, and the latest design of OHR-based modular adders is reported.

6.2.1 TECHNOLOGY AGNOSTIC ASSESSMENT:

The proposed TCR-based adder in Fig. 5 consists of the $SUM0$ and $SUM1$ producing gates, an AND gate, and a (m-2)-bit 2×1 Multiplexer. Therefore, its area can be formulated as follows:

$$A_{Proposed\ TCR\ Adder}$$

$$=SUM0\ \&\ SUM1\ Producer+AAND+(m-2)\ AMUX\ 2\times1 \quad \text{.....(28)}$$

where Ak denotes the area of the component k, and:

$$ASUM0\&SUM1\ Producer$$

$$=2(m-1)AAND/NOR+(1+2+\cdots+(m-1))AAND/OR/NAND+(ANOR+ANOR3+ANOR4+\cdots+ ANORm-2)+2ANORm-1+(m-2)ANOT \quad \text{.....(29)}$$

It should be mentioned that $ANORi$ in (40) indicates the area of a NOR gate with i inputs for $i\geq3$, and the '/' symbol means 'or'. Moreover, the delay of the critical path of the TCR adder herein proposed can be estimated as:

$$DProposed\ TCR\ Adder=DSUM1\ Producer+DMUX\ 2\times1 \quad \text{..... (30)}$$

where Dk denotes delay of the component k, and:

$$DSUM1\ Producer=(m-2)DAND+DOR+DNAND+Dnot \quad \text{..... (31)}$$

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

$$A_{TCR\ Adder}[18]$$

$$=AShifter\ \ control\ \ circuit+(m+(m+2)+(m+2+22)+\cdots+(2m-2))AMUX\ 2\times1+(m-1)\ AMUX\ 2\times1 \quad \text{.....(32)}$$

where:

$$AShifter\ control\ circuit=(m-2)A_{XOR} \quad \text{.....(33)}$$

Equation (34) is used for computing the delay of the TCR-based adder of [7].

$$D\ TCR\ Adder[18]$$

$$=DShifter\ control\ circuit+(\lceil Logm\rceil+1)DMUX\ 2\times1 \quad \text{(34)}$$

where:

$$DShifter\ control\ circuit=\lceil Log\hat{}m-1\rceil DXOR \quad \text{(35)}$$

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. In that model, more complex cells are built from the basic two-input gates. Moreover, one-bit 2×1 multiplexer counts as three and two gates for the area and the delay, respectively. By applying the Unit-Gate model, (28) to (35) can be rewritten as (36) to (39) (note that each *i*-input basic gate can be realized using *i*-1 2-inputs gates).

$$A_{Proposed\ TCR\ Adder\_unit\ gate}$$

$$=7m-11+\frac{m(m-1)+(m-3)(m-2)}{2} \quad \text{(36)}$$

$$D_{Proposed\ TCR\ Adder\_unit\ gate}$$

$$=m+2 \quad \text{(37)}$$

$$A_{\ TCR\ Adder[7]\_unit\ gate}$$

$$=2(m-2)+3(m+(m+2)+(m+2+22)+\cdots+(2m-2))+3(m-1) \quad \text{(38)}$$

$$D_{\ TCR\ Adder[18]\_unit\ gate}$$

$$=2(\lceil Logm-1\rceil+\lceil Logm\rceil+1) \quad \text{(39)}$$

Table V compares the TCR-based adder herein proposed with [7] based on the Unit-Gate model, for moduli 5, 7, 8, 9 and 11. With our proposal, the delay and area values have been improved on average by 33% and 39.5%, respectively.

TABLE 5

AREA AND DELAY ESTIMATED WITH UNIT-GATE MODULE 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

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

$$A_{Proposed\ OHR\ Adder} = A_{First\ Level} + A_{second\ Level} + A_{last\ level} \quad \text{.....(40)}$$

where:

$$A_{First\ Level} = mA_{AND} + mA_{OR} \quad \text{..... (41)}$$

$$A_{second\ Level} = \frac{m(m-1)}{2} A_{AND} \quad \text{.....(42)}$$

$$A_{last\ Level\ (multi\ input\ OR's)} = \frac{m(m-1)}{2} A_{OR} \quad \text{..... (43)}$$

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*-inputs 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:

$$D_{Proposed\ OHR\ Adder} = D_{First\ Level} + D_{second\ Level} + D_{last\ level} \quad \ldots\ldots (44)$$

where:

$$D_{First\ Level} = MAX\{D_{OR}, D_{AND}\} \quad \ldots\ldots (45)$$

$$D_{second\ Level} = D_{AND} \quad \ldots\ldots (46)$$

$$D_{last\ Level} = \lceil Log\frac{m+1}{2}\rceil D_{OR} \quad \ldots\ldots (47)$$

By applying the Unit-Gate model to (51) to (58), the following formulas are obtained:

$$A_{Proposed\ OHR\ Adder\_unit\ gate} = \underbrace{2m}_{A First\ Level\ (AND/OR's)} + \underbrace{m(m-1)\backslash 2}_{A Second\ Level\ (AND's)} + \underbrace{m(m-1)\backslash 2}_{A Last\ Level\ (multi\ input\ OR's)} = m^2+m \quad \ldots\ldots (48)$$

$$D_{Proposed\ OHR\ Adder\_unit\ gate} = \underbrace{2}_{D First\ \&\ Second\ Level} + \underbrace{\lceil Log^{\wedge}m+1\backslash 2\rceil}_{D Last\ Level\ (multi-input\ OR)} \quad ..(49)$$

while for [8]:

$$A_{\ OHR\ Adder[8]\_unit\ gate} = 3*m\lceil Log^{\wedge}m\rceil \quad \ldots\ldots (50)$$

$$D_{\ OHR\ Adder[8]\_unit\ gate} = 2*\lceil Log^{\wedge}m\rceil \quad \ldots\ldots (51)$$

Table VI shows the estimate of the area and the delay of the proposed OHR-based adder as well as of [7]. According to the Unit-Gate model, the proposed OHR-based adders results in an average improvement of 31.7% and 12.2% in delay and area, respectively.

TABLE 6

AREA AND DELAY ESTIMATED UNIT-GATE MODUL 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

Similarly, 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 multiplexer also requires 8 transistors by using inverting multiplexers [24] (an extra not gate is required when the number of consecutive multiplexers is odd). For [18] (Fig. 1), multiplexers for which one of the inputs is 0, are implemented by a single AND gate, and multiplexers for which one of the inputs is 1 are implemented by one AND and one OR gate.

Note that a multiplexer 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 (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 2*m* transistors for a modulo *m*

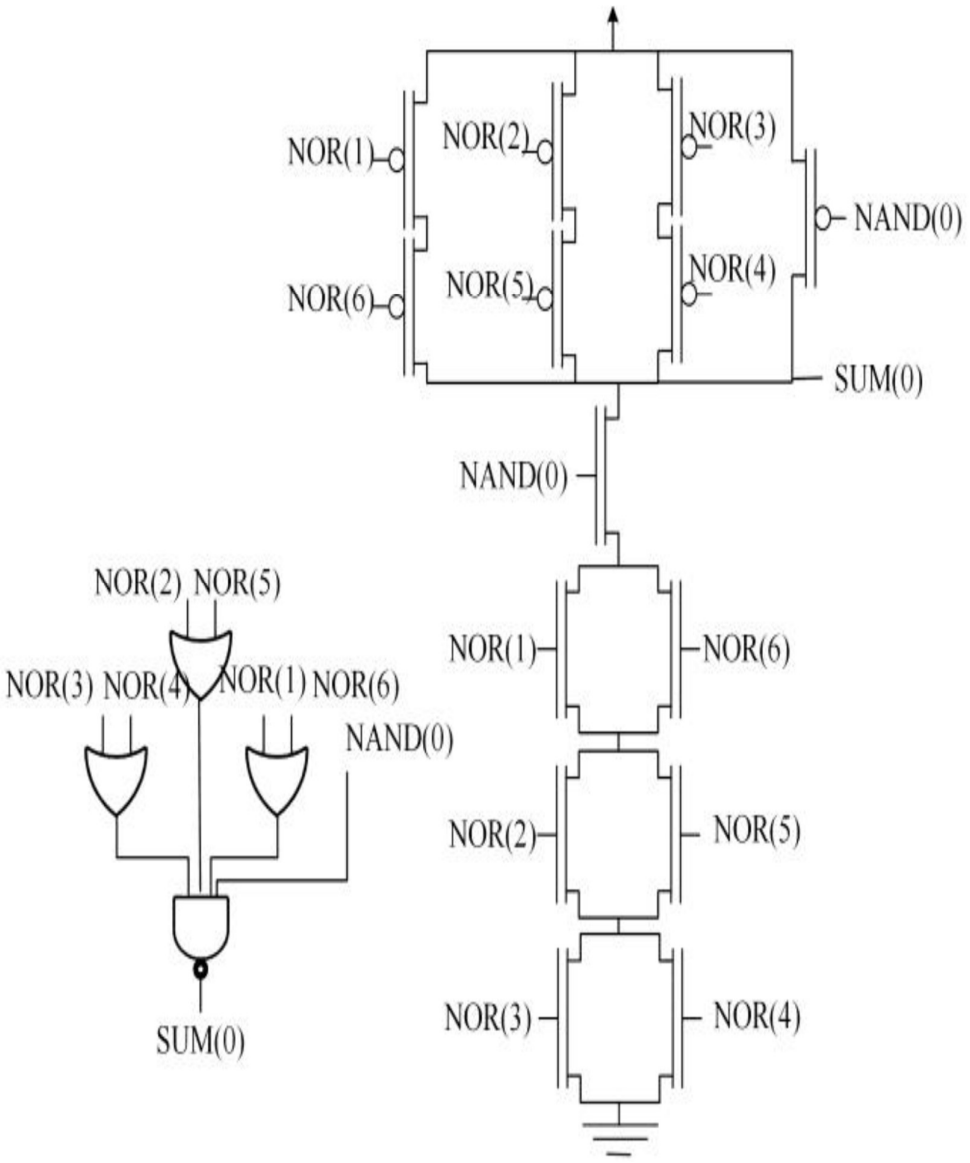


Figure.6.1.1 The structure of a *compound* CMOS *gate* for calculating SUM(0) when *m*=7 (by using NAND and NOR gates instead of AND and OR gates in the first level of Fig. 6.1.1).

. Fig. 6.1.1 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 7

TRANSISTOR COUNT COMPARISION

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

6.2 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 was used for synthesizing the designs and the Cadence Encounter and Nano Route 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 8 and 9 for the same moduli considered in [19]. Also, the percentage of energy savings of the proposed adders, when compared to the previous designs, are shown in Fig. 6.2.1.

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, 8, 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 One-Hot and thermometer coding, 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. 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 are 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 have also better performance than regular modular binary adders; with the exception of moduli with the shape $2n$, since this type of modulo does not require any additional modular circuitry, and can be implemented with regular binary adders [20].

Table 9 shows the experimental results for 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, 8, 9 and 11 is improved in average 34.5%, while the improvement of energy (power-delay-product) 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 herein also made 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 power-delay product (i.e. energy), and area-delay-product (ADP) of the proposed TCR and OHR-based modular adders.

TABLE 8

EXPERIMENTAL RESULTS FOR THR PROPOSED THERMOMETER BASED MODULO 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 9

EXPERIMENTAL RESULTS FOR THR PROPOSED OHC MODULO ADDER AND [18]

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

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, 8, 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, 8, 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 multiplexer-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

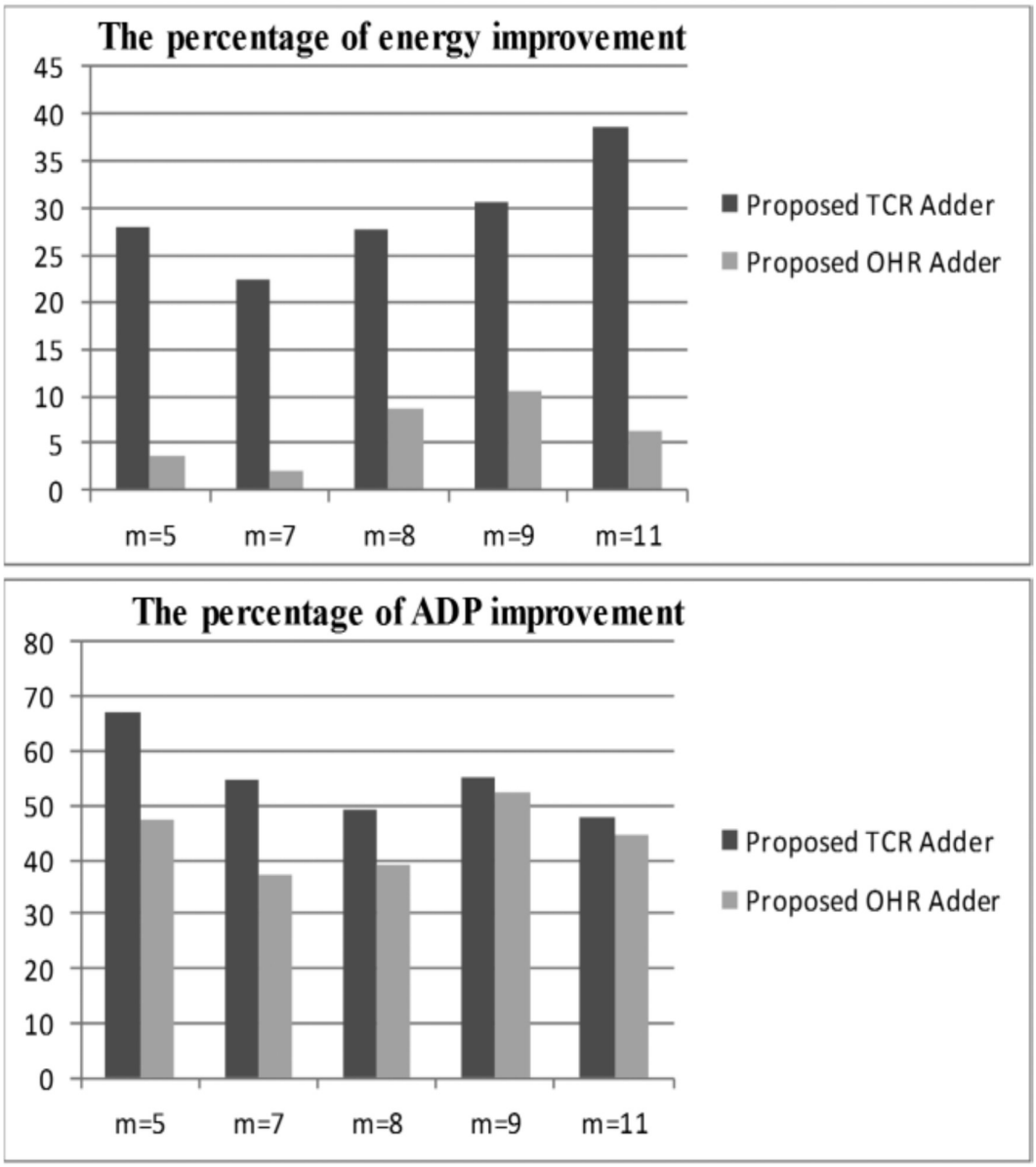


Figure 6.2.1. Comparison of the performance of the proposed thermometer and one-hot coding-based adders.

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 better performance than [19], it can be concluded that the suggested One-Hot modulo adder has better performance than the corresponding binary adders for the moduli 5, 7, 8, 9 and 11.

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

CHAPTER 7
CONCLUSION

In this paper, two new classes of efficient modular adders are proposed, for Thermometer Coding (TC) and One-Hot Coding (OHC). The main advantages of the proposed adders are their high performance and low-cost, making them useful for example for Residue Number Systems (RNSs) based on small moduli sets, used for digital signal processing embedded systems and IoT.

For the first time, the conventional multiplexer-based design of OHC and TC adders are replaced by a novel approach based on a small number of logical gates.

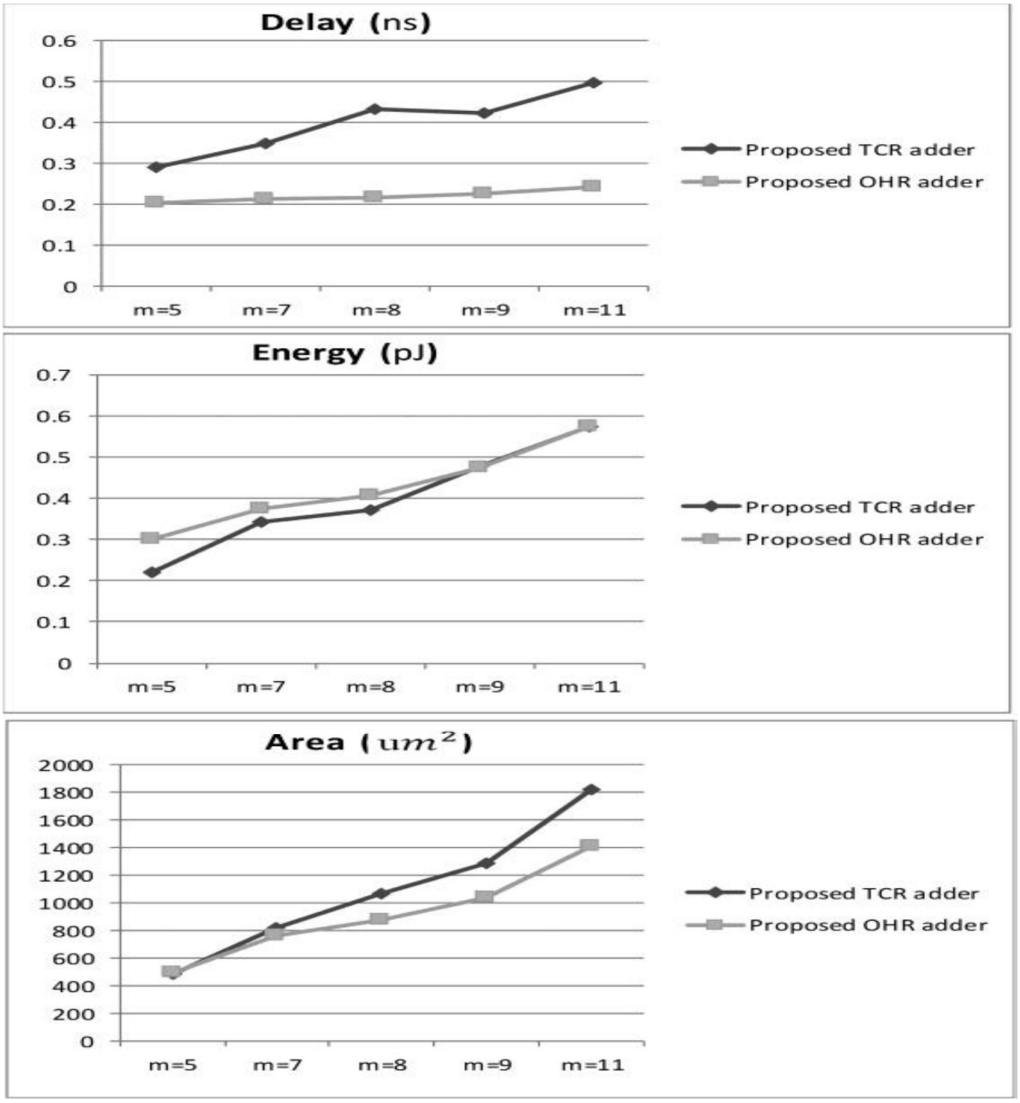


Figure. 7.1. Comparing the performance of the proposed thermometer and one-hot coding-based adders.

Since TC and OHC modular adders do not require carry propagation, their structures for small moduli become simpler, 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 work are easily extended to design other units for modular arithmetic, such as subtractors.

CHAPTER 8

REFERENCES

[1] A.S. Molahosseini, L. Sousa and C.H. Chang (Eds.), *Embedded Systems Design with Special Arithmetic and Number Systems*, 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 Journal of Solid-State Circuits*, vol. 52, no. 1, pp. 127-138, 2017.

[3] M. Alioto (Ed.), *Enabling the Internet of Things: from Integrated Circuits to Integrated Systems*, Springer, 2017.

[4] P.V.A. Mohan, *Residue Number Systems: Theory and Applications*, Springer, 2016.

[5] C.H. Chang, A.S. Molahosseini, A.A. Emrani 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 and Systems Magazine*, vol. 15, no. 4, pp. 26-44, 2015.

[6] L. Sousa, S. Antão, and P. Martins, "Combining Residue Arithmetic to Design Efficient Cryptographic Circuits and Systems," *IEEE Circuits and Systems Magazine*, vol. 16, no. 4, pp. 6-32, 2016.

[7] M. Labafniya and M. Eshghi, "An efficient adder/subtracter circuit for one-hot residue number system," Intl. Conf. on Electronic Devices, Systems and Applications (ICEDSA), pp.121-124, 2010.

[8] M. Hosseinzadeh, S. JafaraliJassbi, K. Navi, "A novel multiple valued logic OHRNS adder circuit for modulo (*rn*-1)," The 4th international conference on advanced engineering computing and applications in sciences, pp.166-170, 2010.

[9] D. Kheirandish, A. Safari, Y. Kong, "Using one hot residue number system (OHRNS) for digital image processing," The 16th CSI International Symposium on Artificial Intelligence and Signal Processing, pp. 064-067, 2012

[10] W. A. Chren, "One-hot residue coding for low delay-power product CMOS design," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 45, no. 3, pp. 303-313, 1998.

[11] H. V. Jayashree, J. Vuggirala and G. N Patil, " Design of High Speed Area Efficient OHRNS Data Path Subsystems for FFT Implementation," 4th International Conference on Electronics and Communication Systems (ICECS), 2017.

[12] S. Kak, "Generalized unary coding", *Circuits Systems and Signal Processing*, vol. 35, no. 4, pp. 1419-1426, 2016.

[13] S.W. Golomb, Run-length encodings. IEEE Trans. Inf. Theory IT–12, 399–401,1966.

[14] R. F. Rice, R. Plaunt, "Adaptive variable-length coding for efficient compression of spacecraft television data,"*IEEE Trans. Commun.,* vol.16, pp. 889–897, 1971.

[15] S. Jayashri and P. Saranya," Reconfigurable FIR Filter Using Distributed Arithmetic Residue Number System Algorithm Based on Thermometer Coding", International Conference on Communication and Signal Processing, 2014.

[16] C. H. Vun, A. B. Premkumar, and W. Zhang, "Sum of Products: Computation using Modular Thermometer Codes", International

Symposium on Intelligent Signal Processing and Communication Systems (ISPACS), Dec. 2014.

[17] C. H. Vun and A. Premkumar, "RNS encoding based folding ADC," in Proc. *IEEE International Symposium on Circuits and Systems*, pp. 814–817, 2012.

[18] C. H. Vun and A. Premkumar, "Thermometer code based modular arithmetic," in Proc. *Spring Congress on Engineering and Technology (S-CET)*, pp. 534–538, 2012.

[19] C. H. Vun, A. Premkumar, and W. Zhang, "A new RNS based da approach for inner product computation," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 60, no. 8, pp. 2139–2152, 2013.

[20] K. Navi, A.S. Molahosseini and M. Esmaeildoust, "How to Teach Residue Number System to Computer Scientists and Engineers," *IEEE Transactions on Education*, vol. 54, no. 1, pp. 156-163, 2011.

[21] S. A. White, "Applications of distributed arithmetic to digital signal processing: a tutorial review," IEEE ASSP Magazine, vol. 6, no. 3, pp. 4-19, 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*, fourth edition, Addison-Wesley, 2011.

[25] X. Chen and N.A. Touba, *Fundamentals of CMOS design*, Chapter 2 in *Electronic Design Automation*, L.T. Wang, Y.W. Chang and K.T. Cheng (Eds.), Morgan Kaufmann, 2009