Thermometer code based modular arithmetic

Vun, Chan Hua; Premkumar, Annamalai Benjamin

2012

Vun, C. H., & Premkumar, A. B. (2012). Thermometer code based modular arithmetic. 2012 Spring Congress on Engineering and Technology, pp1-5.

<https://hdl.handle.net/10356/99796>

<https://doi.org/10.1109/SCET.2012.6342081>

© 2012 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works. The published version is available at: [<http://dx.doi.org/10.1109/SCET.2012.6342081>].

Downloaded on 26 Apr 2023 20:50:06 SGT

Thermometer Code based Modular Arithmetic

Chan Hua Vun and Benjamin Premkumar

School of Computer Engineering

Nanyang Technological University

Singapore 639798

Email: {aschvun, asannamalai}@ntu.edu.sg

Abstract—This paper presents a novel approach to perform modular arithmetic addition and subtraction using base-1 thermometer code data format for operands corresponding to the residues of the same modulus. Two n -bit thermometer code operands are first concatenated and logically shifted to produce a normalized $2n$ -bit thermometer code intermediate sum. Modulo operation is then applied to this $2n$ -bit intermediate sum to produce an n -bit datum corresponding to the modular sum of the two input operands. This approach greatly simplifies the modular addition operation by eliminating the carry bit propagation during the arithmetic operation encountered when using the conventional base-2 binary code data format. It also enables practical applications of modular arithmetic for signal processing algorithms in a very efficient way. Circuit for implementing the modular arithmetic units using multiplexers and basic logic gates are also described in this paper.

Keywords—Modular arithmetic; Thermometer code residue; RNS

I. INTRODUCTION

In digital signal processing, data used in the processing routine are usually represented in conventional binary number system using the base-2 binary code (BC) encoded format. But it is well known that the speed of arithmetic operations such as addition can be significantly enhanced by limiting the carry bit propagation. Apart from using more efficient carry bit propagation binary adder circuits, one alternative way to minimize carry propagation and hence to improve speed is to employ number theory transforms to reduce the bit-length of the data. One of these representations that is simple and efficient is the residue number system (RNS).

RNS is a number system in which an integer X within the range $0 \leq X < P$ can be uniquely represented by a set of remainders (known as residues) when it is divided by a set of pairwise prime positive integers (called the moduli), with P equal to the product of the moduli. Hence using a $[m_1, m_2, \dots, m_M]$ moduli set, an integer $X < P = \prod_{i=1}^M m_i$ can be uniquely represented by its residues as $X \equiv \langle x_1, x_2, \dots, x_M \rangle$. The main feature of these residues is their small dynamic range (DR) such that they can be encoded in much shorter bit-length. In addition, RNS also allows concurrent arithmetic operation to be implemented in a parallel manner among the various channels of the residues. As such, arithmetic operations for data using residues can be executed much faster when compared to equivalent operations based on its corresponding full binary representation.

Figure 1: RNS ADC with RNS signal processing

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 overcome 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. 1. 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.

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.

The remainder of the paper is organized as follows. Section II provides the background of RNS and the complication of the BC based modular arithmetic. Section III explains the principle of the TC based modular arithmetic. Section IV describes the circuit needed to implement the proposed technique, while Section V presents a performance comparison of the proposed circuit against the conventional BC based circuit for modulo addition. Section VI provides the concluding remarks on the new technique proposed in the paper.

II. BACKGROUND

A. RNS and modular arithmetic

RNS [2] 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 $\langle 4,3,8 \rangle$ residue set and the decimal number $Y=254$ will be represented using $\langle 2,6,2 \rangle$ residue set. Arithmetic operation of these two integers can be equally performed in modular arithmetic using their residues as follows:

$$\begin{aligned} 179 \odot 254 &\cong \langle 4,3,8 \rangle \odot \langle 2,6,2 \rangle \\ &= \langle 4\odot 2, 3\odot 6, 8\odot 2 \rangle \end{aligned}$$

where the arithmetic operator $\odot$ can be $+$, $-$, and $\times$. For example:

$$\begin{aligned} 179+254 &\cong \langle 4,3,8 \rangle + \langle 2,6,2 \rangle \\ &= \langle 4+2, 3+6, 8+2 \rangle \\ &= \langle 6,9,10 \rangle \\ &= \langle 6,1,1 \rangle \end{aligned}$$

after executing the modulo operations $|6|_7$, $|9|_8$ & $|10|_9$. A reverse RNS-decimal conversion operation will then give the answer as $\langle 6,1,1 \rangle \cong 433$. Subtraction can be similarly executed with addition through using the two's complement of the subtrahend.

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.

B. BC format based modular arithmetic

The normal way to implement the modular arithmetic described above is based on BC encoded residues using standard binary arithmetic circuit. One inconvenience encountered in modular arithmetic for RNS encoded with conventional base-2 BC format is in the execution of the modulo operation on the intermediate value of residue. Fig. 2 shows the general way to perform the modular addition operation for general BC encoded residues (i.e. not of any specific types like those associate with 2^n and the likes). It consists of using two binary adders that in effect performing the modular addition operation for A & B using the two's complement of the modulus, $\tilde{m}$ to execute the modulo operation as described by the following equation:

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

The underlying adders can be based on the common ripple carry full adder which is slow but uses a simple logic structure,

Figure 2. Modular-m Adder for BC format residues

or the carry-look ahead full adder that is fast but required much high number of transistor counts.

This general approach to perform the BC modulo operation hence involves an addition, a subtraction and a comparison that add significant complexity to the circuit [2]. Furthermore, the carry bit propagation remains intact in the base-2 BC modular arithmetic since the addition is still based on conventional base-2 binary adder, albeit localized to the pair of residues involved in the addition.

In contrast, the proposed TC based approach will not have these complications as shall be described. In addition, the implementation of the TC based modular arithmetic operations such as addition and subtraction consists of multiplexers based shifter circuits with basic logic gates where the total number of transistor count is comparable to the conventional base-2 binary adders for same operation, but with much smaller gate-delay and hence can operate at higher clock rate.

III. TC FORMAT BASED MODULAR ARITHMETIC

Number that is encoded using the TC format number system is equivalent to a 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. 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 seven '1' bits and five '0' bits as 000111111_1 , 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 to 10, the decimal 7 would be encoded by using only 4 bits as 0111_2 . 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 3. Modular addition for TC based residues

By using TCRs, modular addition of two operands 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. 3 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 $t[7]$ 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 $r_a \bar{r}_b$, where $\bar{r}_b$ 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 r_a and r_b : $|5 + 4|_7 = 2$.

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. 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 shows an example of the subtraction operation of two integers $r_a = 5$ being the minuend and $r_b = 4$ 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 $r_a = 4 = 0001111_1$ being the minuend and $r_b = 5 = 0011111_1$ 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 $\bar{r}_b = 1100000_1$ followed by the concatenation with r_a to obtain $r_a \bar{r}_b$ and the modulo operation as shown below:

$$\begin{aligned} |0001111_1 + 1100000_1|_7 &= |00011111100000_1|_7 \\ &= |000111111_1|_7 \\ &= 0111111_1 \\ &= 6_{10} \end{aligned}$$

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 [3]. 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.

IV. CIRCUIT REALIZATION

As conceptually illustrated in Fig. 3, 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 r_a and r_b are applied to the circuit input. Instead of concatenating the two TCRs and then normalizing the intermediate sum by removing all the '0' bits in one of the operand, the modulo adder circuit pads appropriate number of '1' bits to the TCR operand r_b , based on the value of the other TCR r_a . This hence changes the number of '1' bit(s) in r_b based on the value of r_a , 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 [5] as shown in Fig. 6, 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 $k=0,1,2,\dots$, 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 $\lceil \log_2 m \rceil$ (E.g. for $m=7$ in Fig. 6, number of shifter groups needed =3).

The bit shifting is controlled by a shifter control logic circuit that decodes the number of '1' bits in r_a . As r_a 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 5. Modular adder for TCRs

Figure 6. Modulo adder for modulus-7 TCRs

$$S[1] = a[1] \oplus a[2] \oplus a[3] \oplus a[4] \oplus a[5] \oplus \dots$$

$$S[2] = a[2] \oplus a[4] \oplus a[6] \oplus a[8] \oplus \dots$$

$$S[4] = a[4] \oplus a[8] \oplus a[12] \oplus \dots$$

$$S[8] = a[8] \oplus a[16] \oplus \dots$$

Hence the shift control logic circuit can be implemented by combinatorial logic circuit such as the one shown in Fig. 7, which is to be used for the modulo-7 adder as shown in Fig. 6.

The log shifter circuit output, $c[1:2m-2]$ is then applied to the output multiplexers that perform the modulo operation by checking bit $c[m]$ corresponding to the modulus of the system (e.g., $c[7]$ in Fig. 6 for the modulus-7 system). This will hence select either the lower or upper $(m-1)$ bits as the final output, producing the correct residue for the operation.

V. CIRCUITS PERFORMANCE EVALUATION

This section compares the performance and complexity of the proposed TC based modular adder against the BC based modular adders in terms of critical path gate-delay and transistor counts. Using the moduli that vary between 5 and 9, the DR of their residues will range between 4 and 8 which can be represented with 3 or 4 binary bits for the BC encode residue but up to 8 bits for the TC encoded residue. These moduli together form a moduli set [5,7,8,9] that can provide a

Figure 7. Shifter control circuit for modulo-7 adder

Figure 8. Binary adders (a) Ripple carry (b) & (c) Carry-lookahead

DR of 2520, equivalent to a 11-bit binary system. Additional moduli such as 11 and 13 can also be used to further extend the DR if needed.

For the BC encoded residues, two binary adders are needed to implement the modular adder as shown in Fig. 2. In this comparison, two standard representative binary adders are used. These are the ripple carry adder and the carry-lookahead adder. The ripple carry adder is the most hardware efficient but slowest implementation of the binary adder, while the carry-lookahead adder is one of the fastest ways to implement binary adder with tradeoff in hardware circuit complexity. Note that special modular adders that are optimized for certain moduli set (e.g. 2^n and the likes) are not included in order to provide a fair comparison for the general modulus value considered here.

Fig. 8(a) shows the logic gate implementation for one bit of the ripple carry full adder [6]. A 3-bit or 4-bit ripple adder will hence use 3 or 4 of these circuits. For a 3-bit binary carry-lookahead adder, the circuit consists of the one shown in Fig. 8(a) for the 1st bit. For the 2nd and the 3rd bits, the portion of the circuit shown within the dotted box in Fig. 8(a) is replaced with those shown in Fig. 8(b) and Fig. 8(c) respectively.

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 is 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 7.

Gate count comparison between TC and BC based modular adders is complicated by the fact that practical circuit realization of the multiplexer is done by using transistor based circuits such as the 4-transistor based CMOS Transmission Gate or the 2-transistor based Pass-Transistor logic. Hence it is more appropriate to compare hardware complexity in terms of transistor count here, although this does not reflect the complexity involved in the wiring of the underlying circuit. In the comparison here, 6 transistors is used for each 2-input XOR logic gate, 4 transistors for all other 2-input logic gates, 2 transistors for each extra input pin and 2 transistors for one NOT gate. For the multiplexer, the 4-transistor based CMOS transmission gate is used as this is a fairly conservative design, with one NOT gate shared among all multiplexers to generate the internal complement shift control $\bar{S}(x)$.

Critical path gate-delay comparison is based on the longest path that a signal propagates through each circuit. For the BC

TABLE I. PERFORMANCE PARAMETRIC COMPARISON

Mod-m	BC-RCA		BC-CLA		TC-MUX	
	t-cnt	g-dly	t-cnt	g-dly	t-cnt	g-dly
m=5	158	16	210	12	122	6
m=7	158	16	210	12	174	8
m=8	158	16	210	12	200	8
m=9	210	20	322	12	288	8

based modular adder, this is equal to the delay through the two binary adders to generate the S'' value for the output multiplexer. Each ripple adder has a propagation delay equals to $(2n+2)$ due to the carry bit propagation [6]. For the carry-lookahead adder, the most optimum implementation will be 6 gate-delays independent of the number of bits [2], while in practice this may be longer due to the higher fan-in of higher bit logic gates. For the TC based adder, the critical path is the signal propagation path through the shift control circuit to produce the shift control signal $S[1]$, plus the delay through the last stage of the shifter circuit, including two NOT gates.

Based on the above, Table 1 summarizes the detail comparison of the three types of adders in term of transistor counts (t-cnt) and gate-delay (g-dly) for various values of modulo- n implementation. Results in Table 1 shows that compared to the BC ripple carry adder, the TC based modulo adder has similar order of transistor count at lower modulus while not excessive at higher modulus, but with much superior gate delay performance throughout. Compared to the BC carry-lookahead adder, the TC based modulo adder is superior in both aspects.

VI. CONCLUSION

A novel thermometer code based modular arithmetic technique has been described which provide a method to perform modular arithmetic in a highly efficient and high speed manner. Coupled with the new RNS ADC where the output is inherently presented in RNS representation encoded in TC format, the proposed techniques provide a mean to perform signal processing routine using the TC modular arithmetic approach in a very efficient manner.

REFERENCES

- [1] Vun, C. H.; Premkumar, A.B.; , "RNS Encoding Based Folding ADC," to be presented at *ISCAS 2012, IEEE International Symposium on Circuits & Systems*, Seoul, Korea 20-23 May 2012.
- [2] Omondi, A.; Premkumar, A.B.; , *Residue Number Systems, Theory and Implementation*, Imperial College Press, Singapore, 2007
- [3] Jullien, G.A.; , "Implementation of Multiplication, Modulo a Prime Number, with Applications to Number Theoretic Transforms," *Computers, IEEE Transactions on* , vol.C-29, no.10, pp.899-905, Oct 1980
- [4] White, S.A.; , "Applications of distributed arithmetic to digital signal processing: a tutorial review," *ASSP Magazine, IEEE*, vol.6, no.3, pp.4-19, Jul 1989
- [5] Atmel Corporation;, *Barrel shifter*, FPGA application note, doc0469, 1999
- [6] Mano, M.M.; Kime, C.R.; , *Logic and Computer Design Fundamentals*, Prentice Hall. USA, 1997