Channel Operations in the Residue Number System with Special Modulus on Hardware

Yinan KONG
Department of Physics and Engineering
Faculty of Science, Macquarie University
NSW 2109, Australia
ykong@ics.mq.edu.au

Abstract— **This paper presents fast hardware algorithms for channel operations in the Residue Number System (RNS) in terms of addition, subtraction and multiplication. These algorithms compare favorably with other popular algorithms due to the use of special moduli in the form of $2^n - 1$. They are particularly applicable for the largest channel or redundant channels in RNS.**

Keywords-The residue number system; modular operations; computer arithmetic; hardware algorithms

I. INTRODUCTION

The demand for high-speed computing continues to increase for Digital Signal Processing (DSP), multimedia and security applications, and appropriate research activity grows at a similar rate. The Residue Number System (RNS), as one of the efficient tools in meeting the demand, has long been a topic of interest due to its applicability to Very Large Scale Integration (VLSI) System design [1], [2]. The fast RNS operations, additions and multiplications, have proved to be most profitable to speed up computations in many fields.

Innovative techniques have been employed in the past by making use of the properties of RNS and also the number theoretical properties. Some architectures were designed with a very high degree of processing parallelism and communication parallelism tailored to the response time of adders and multipliers. [3] uses longer intermediate pseudo residues for look-up table implementations. [4] proposes a high speed pipelined Fast Fourier Transform (FFT) algorithm with relatively optimal VLSI complexity. Several adder and multiplier units were conceived with different characteristics, such as multiply-accumulate units (MAC) [5] and power-of-2 scaling units [6]. Furthermore, general purpose DSP chips were shown to achieve great higher data processing bandwidth when incorporated with RNS processors [7].

II. THE RESIDUE NUMBER SYSTEM (RNS)

A Residue Number System [8] is characterized by a set of N co-prime moduli $\{m_1,\ldots,m_N\}$. In the RNS a non-negative integer X is represented in N channels: $X = \{x_1, x_2, \ldots, x_N\}$, where x_i is the residue of X with respect to m_i, i.e. $x_i = \langle X \rangle_{m_i} = X \bmod m_i$. n, the length of m_i, is denoted as the channel width of the RNS. Within the RNS there is a unique representation of all integers in the range $0 \le X < D$ where $D = m_1 m_2 \ldots m_N$. D is therefore known as the dynamic range of the RNS.

If A, B and C have RNS representations given by $A = \{a_1, a_2, \ldots, a_N\}$, $B = \{b_1, b_2, \ldots, b_N\}$ and $C = \{c_1, c_2, \ldots, c_N\}$, then denoting $*$ to represent the operations $+$, $-$ or $\times$, the RNS version of $C = A * B$ satisfies

$$C = \{\langle a_1 * b_1 \rangle_{m_1}, \langle a_2 * b_2 \rangle_{m_2}, \ldots, \langle a_N * b_N \rangle_{m_N}\}. \quad (1)$$

Thus addition, subtraction and multiplication can be concurrently performed on the N residues within N parallel channels, and it is this high speed parallel operation that makes RNS attractive [9], [10]. This paper is trying to improve this operation to speed further up a RNS.

III. RNS CHANNEL OPERATIONS

RNS channel operations refer to those operations carried on within RNS channels. Note that regardless of whichever operation of the three (addition, subtraction and multiplication) is running within a RNS channel, it has a common characteristic: both operands, say a and b, are non-negative and less than the channel modulus m, because both a and b are resulted from previous operation modulo m.

Three new algorithms doing the three operations respectively are given below with the channel modulus m in the form of $m = 2^n - 1$.

A. RNS Channel Addition

$c = \langle a + b \rangle_m$ is computed in this section.

First compute $s = a + b$, and then $c = \langle a + b \rangle_m = \langle s \rangle_m$, where s satisfies,

$$0 \le s \le 2m - 2 < 2m \quad (2)$$

as $0 \le a \le m - 1$ and $0 \le b \le m - 1$. Because $m = 2^n - 1$, then

$$s = l \times (2^n - 1) + c \quad (3)$$
$$= l \times 2^n + (c - l) \quad (4)$$

978-1-4244-4994-1/09/$25.00 ©2009 IEEE

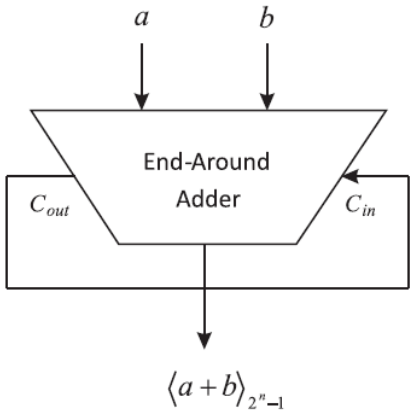

Figure 1. Architecture for RNS channel modular addition $\langle a+b\rangle_m$, where $m = 2^n - 1$ and $s = a + b$.

where $l = \left\lfloor \frac{c}{2^n - 1} \right\rfloor$ and the flooring function $\lfloor \cdot \rfloor$ represents the largest integer less than or equal to the input. It can be seen from (4) that l is the most significant bit of the $n+1$-bit number s. Reducing both sides of (4) modulo 2^n gives

$$c = \langle s \rangle_m = \langle s \rangle_{2^n} + l \tag{5}$$

where $\langle s \rangle_{2^n}$ is actually the least significant n bits of s. Thus, an addition modulo $2^n - 1$ can be conveniently performed using an end-around adder [11] in which the carry output l from a conventional adder is fed back into the carry input at the least significant end of the adder. The latency is only marginally longer than ordinary addition.

For example, $m = 7 = 2^3 - 1$, $s = 10 = (1010)_2$. Then, $\langle s \rangle_m = (s_2 s_1 s_0) + s_3 = (010)_2 + 1 = (11)_2 = 3$.

Figure 1 shows the scheme. The algorithm is described in Algorithm 1. Note that the end-around adder can be used in RNS channels directly, because the two inputs a and b are both less than $2^k - 1$, which again makes the output bounded by the modulus $2^k - 1$ and applied as an input to next end-around adders.

Algorithm 1 RNS Channel Addition

Require: $m = 2^n - 1$ {Channel modulus}
Require: $0 \le a \le m-1$ and $0 \le b \le m-1$
Ensure: $c = a + b \bmod m$
  $s = a + b$
  $c = \langle s \rangle_{2^n} + s_n$ {s_n is the most significant bit of s}

B. RNS Channel Subtraction

$c = \langle a - b \rangle_m$ is computed in this section.

We have $c = \langle a-b \rangle_m = \langle a-b+m \rangle_m = \langle a+(m-b) \rangle_m = \langle a+d \rangle_m$ where $d = m - b$. Then $1 \le d \le m$ as $0 \le b \le m - 1$. Because $0 \le a < m$, $1 \le a + d \le 2m - 1 < 2m$, which satisfies (2). Therefore, Algorithm 1 again can be applied to compute $c = \langle a + d \rangle_m$.

Because $m = 2^n - 1$, m is composed of '1's only, and therefore computing $d = m - b$ is equivalent to performing complementary operation on b, i.e. inverting every bit of b. The RNS channel subtraction algorithm is described in Algorithm 2. The latency is similar to a channel addition.

C. RNS Channel Multiplication

$c = \langle a \times b \rangle_m$ is computed in this section.

Doing a n-bit multiplication is equivalent to doing n-bit addition n times [12]. Here we perform multiplication interleaved with modular reduction because we've had a fast algorithm (Algorithm 1) doing modular addition within each iteration. Table I gives an example for this scheme, where a = 0110, b = 1011 and m = 1111.

First, set $c = 0$ and $d = a$. Then scanning b from right to left, compute $d = \langle d + d \rangle_m$ until the last second bit and compute $c = \langle c + d \rangle_m$ at '1's.

Algorithm 1 is used to compute $c = \langle c + d \rangle_m$ at b_i = 1. According to (5), $d = \langle d + d \rangle_m = \langle 2d \rangle_m$ can be derived from

$$\langle 2d \rangle_m = \langle 2d \rangle_{2^n} + (2d)_n, \tag{6}$$

where $(2d)_n$ denotes the most significant bit of $2d$. Note the least significant bit of $2d$ is definitely '0', so the addition in (6) is equivalent to left rotating d for 1 bit, as shown in Figure 2. The RNS channel multiplication algorithm is described in Algorithm 3.

Algorithm 2 RNS Channel Subtraction

Require: $m = 2^n - 1$ {Channel modulus}
Require: $0 \le a \le m-1$ and $0 \le b \le m-1$
Ensure: $c = a - b \bmod m$
  $d = \bar{b}$ {Invert all bits of b}
  $c = \langle a + d \rangle_m$ {Call Algorithm 1}

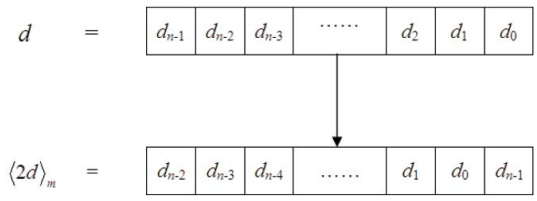

Figure 2. Computation of $\langle 2d \rangle_m$, where $m = 2^n - 1$ and $0 \le d \le m - 1$.

TABLE I

A EXAMPLE OF RNS CHANNEL MODULAR MULTIPLICATION

i	b_i	c	d	Computation of c	Computation of d
	Initialization	0	$\langle a\rangle_m = 0110$		
0	1	$\langle a\rangle_m = 0110$	$\langle 2a\rangle_m = 1100$	$\langle 0 + 0110\rangle_{1111} = 0110$	$\langle 2 \times 0110\rangle_{1111} = 1100$
1	1	$\langle 3a\rangle_m = 0011$	$\langle 4a\rangle_m = 1001$	$\langle 0110 + 1100\rangle_{1111} = 0011$	$\langle 2 \times 1100\rangle_{1111} = 1001$
2	0	$\langle 3a\rangle_m = 0011$	$\langle 8a\rangle_m = 0011$		$\langle 2 \times 1001\rangle_{1111} = 0011$
3	1	$\langle 11a\rangle_m = 0110$		$\langle 0011 + 0011\rangle_{1111} = 0110$	

D. Comparison with Other Modular Multiplication Algorithms

As can be seen from Table I and Algorithm 3, the computation of c and d in Algorithm 3 could be proceeded parallelly. The worst case is that all bits of b are '1's, which

Algorithm 3 RNS Channel Multiplication

Require: $m = 2^n - 1$ {Channel modulus}
Require: $0 \le a \le m-1$ and $0 \le b \le m-1$
Ensure: $c = a \times b \bmod m$
  $c = 0, d = a$ {Initialization}
  for $i = 0$ to $n-1$ **do**
    $d = (d_{n-2}d_{n-1}...d_2d_1)\&d_{n-1}$ {Rotate d left for 1 bit to compute $d = \langle 2d\rangle_m$}
    if $b_i = 1$ **then**
      $c = \langle c + d\rangle_m$ {Call Algorithm 1}
    end if
  end for

means $c = \langle c + d\rangle_m$ has to be done in each iteration. Since this specific modular addition is equivalent to a n-bit addition in terms of complexity, as shown in Section III-A, the complexity of Algorithm 3 is just n n-bit additions, which is only the complexity of a normal n-bit multiplication.

Montgomery Algorithm [13]–[15] and Improved Barrett Algorithm [16] are described in Algorithm 4 and Algorithm 5 respectively. The former needs at least $2n(n+1)$-bit additions and the latter needs $n(n+1)$-bit additions and $n(n+1)$-bit subtractions, apart from a certain amount of multiplications incurred by Quotient digit selection and other steps.

Algorithm 4 Montgomery Modular Multiplication

Require: $R = r^n$
Require: m_0^{-1} s.t. $m_0 \times m_0^{-1} \equiv 1 \bmod r$
Ensure: $C \equiv A \times B \times R^{-1} \bmod M$
  $C = 0$
  for $i = 0$ to $n-1$ **do**
    $C = C + a_iB$ {Partial product accumulation}
    $q_i = -c_0 \times m_0^{-1} \bmod M$ {Quotient digit selection}
    $C = (C + q_iM)/r$ {Reduction step}
  end for

Algorithm 5 Improved Barrett Modular Multiplication

Require: α, β {Pre-defined parameters}
Require: $K = \lfloor 2^{n+\alpha}/M \rfloor$ {A precomputed constant}
Ensure: $C \equiv A \times B \bmod M$
  $C = 0$
  for $i = n-1$ downto 0 **do**
    $C = C + a_iB$ {Partial product accumulation}
    $q_i = \lfloor (\lfloor C/2^{n+\beta} \rfloor \times K)/2^{\alpha-\beta} \rfloor$ {Quotient digit selection}
    $C = C - q_iM$ {Reduction step}
  end for

As a consequence, Algorithm 3 is at least twice as fast as Montgomery Algorithm and Improved Barrett Algorithm, which are generally used in RNS channels to do modular multiplications.

IV. CONCLUSIONS

We've shown that we could do a modular addition and a modular multiplication using special modulus at the speed of a normal addition and a normal multiplication respectively. In Section III-D, it's concluded that our new modular multiplication algorithm (Algorithm 3) is much faster than some algorithms generally used today. This is actually because the RNS channel modulus used in the new algorithm is in the special form of $m = 2^n - 1$. It is obviously not practical to find a set of moduli all in this special form.

However, one fact that obstructs the parallel nature of RNS is the different magnitudes between moduli lead to different speed in-between channels. The operation within the largest channel is usually slowest, e.g. $\langle a \times b\rangle_{37}$ is certainly slower than $\langle a \times b\rangle_7$. This is one of the areas where the new algorithms exhibit their advantages. Operations in the largest and slowest channel can be moved off the critical path by choosing the modulus to be in the form of $m = 2^n - 1$, where the new algorithms can be applied to speed up the whole RNS system. For example, in the RNS {19,23,29,31}, the operations in the largest channel $m_4 = 31 = 2^5 - 1$ is expected to be faster than channel $m_3 = 29$ and even channel $m_2 = 23$.

Another application is in the redundant channel. Some operations in RNS like scaling and base extension needs redundant channels to proceed. Most of the critical paths in architectures dealing with these operations lie on some of the redundant channels [17]–[19]. Therefore, applying the new algorithms in appropriate redundant channels will largely reduce the latency of those existing algorithms.

REFERENCES

[1] M. A. Soderstrand, W. Jenkins, and G. Jullien, "Residue number system arithmetic: Modern applications," *Digital Signal Processing*, 1996

[2] S. R. Barraclough, "The design and implementation of the ims a110 image and signal processor," in *Proc. IEEE CICC*, San Diego, May 1989, pp. 24.5/1–4.

[3] Parhami, "A note on digital filter implementation using hybrid rnsbinary arithmetic," *Signal Processing*, vol. 51, pp. 65–67, 1996.

[4] G. Alia, F. Barsi, and E. Martinelli, "Optimal vlsi complexity design for high speed pipeline fft using rns," *Computers & Electrical Engineering*, vol. 24, pp. 167–182, 1998.

[5] A. P. Preethy and D. Radhakrishnan, "A 36-bit balanced moduli mac architecture," in *Proc. IEEE Midwest Symposium On Circuits and Systems*, 1999, pp. 380–383.

[6] M. N. Mahesh and M. Mehendale, "Low power realization of residue number system based fir filters," in *13th Int. Conf. On VLSI Design*, Bangalore, India, Jan. 2000, pp. 30–33.

[7] M. Bhardwaj and B. Ljusanin, "The renaissance – a residue number system based vector co-processor," in *Proc. 32nd Asilomar Conference on Signals, Systems and Computers*, 1998, pp. 202–207.

[8] N. S. Szabo and R. H. Tanaka, Residue Arithmetic and its Applications to Computer Technology. New York: McGraw Hill, 1967.

[9] A. Mohan, Residue Number Systems: Algorithms and Architectures. Kluwer Academic Pub, 2002.

[10] A. Omondi and B. Premkumar, *Residue Number Systems – Theory and Implementation*, ser. Advances in Computer Science and Engineering: Texts. UK: Imperial College Press, 2007, vol. 2.

[11] B. Parhami, Computer Arithmetic – Algorithms and Hardware Designs. Oxford University Press, 2000.

[12] N. Weste and D. Harris, *CMOS VLSI Design – A Circuit and Systems Perspecitive*, 3rd ed. Addison Wesley, 2004.

[13] M. Shand and J. Vuillemin, "Fast implementations of RSA cryptography," in *Proc. 11th IEEE Symposium on Computer Arithmetic*, 1993, pp. 252–259.

[14] C. D. Walter, "Still faster modular multiplication," *Electronics Letters*, vol. 31, no. 4, pp. 263–264, Feb. 1995.

[15] H. Orup, "Simplifying quotient determination in high-radix modular multiplication," in *Proceedings of the 12th IEEE Symposium on Computer Arithmetic*, 1995, pp. 193–199.

[16] J.-F. Dhem, "Design of an efficient public-key cryptographic library for RISC based smart cards," Ph.D. dissertation, Universit´e Catholique de Louvain, May 1998.

[17] A. Shenoy and R. Kumaseran, "Fast base extension using a redundant modulus in rns," *IEEE Trans. Comput.*, vol. C-38, no. 2, pp. 292–297, Feb. 1989.

[18] ——, "A fast and accurate rns scaling technique for high speed signal processing," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 37, no. 6, pp. 929–937, June 1989.

[19] Y. Kong and B. J. Phillips, "Residue number system scaling schemes," in *SPIE International Symposium on Smart Materials, Nano-, and Micro-Smart Systems*. Sydney: SPIE, Dec. 2004.