How to Improve Division in Residue Number Systems

Jean-Claude Bajard Fabien Rico *

Abstract

This paper deals with conversion from Residue Number System (RNS) to Mixed Radix System (MRS) and application to division algorithms. We present a conversion algorithm most significant digits first in $O(\log(n))$. We apply this method to two different division algorithms, and we can , like this, verify the efficiency of our approach.

1 Introduction

If we refer to Knuth [Knu81], the first publications on Residue Number System in Computer science, date from 1955 [SV55] and 1959 [Gar59]. The most referenced book on the topic was published in 1967 by Szabo and Tanaka [ST67]. This representation is due to the Chinese Remainder Theorem, a number can be represented by its residues modulo a set of relatively prime numbers called the RNS base. Thus, an operation on large numbers can be distributed to small operators, one by modulo. This kind of representation is welcomed for huge number like in cryptography [BDK98], and also for Digital signal processing to decomposed operations in table lookup methods [KJM86].
These systems are very useful for addition and multiplication which can be done in parallel without any communication. The main drawbacks are comparison and division. RNS are not positional systems, "digits" don't give any idea of magnitude.
The literature on RNS division algorithm is numerous. Among the most recent we can cite Gamberger [Gam91], Lu and Chiang [LC92], Hitz and Kaltofen [HK95], Bajard, Didier and Muller [BDM98].
Gamberger proposed an algorithm where the denominator converges to 1, at each step he used conversion from an RNS base to another using MRS representation for intermediate computing. Lu and Chiang have adapted a classical binary algorithm where comparison are done using a parity test. The algorithm of Hitz and Kaltofen is based on the Newton iteration where as Gamberger, conversion to another RNS base is needed. As for Bajard, Didier and Muller, they use a high radix algorithm where comparison are made in a floating point like representation.
Whichever the original algorithm, the nerve center is conversion. So we focus our attention on this problem. We, first, give some definitions and properties of Residue Number Systems and Mixed radix Systems. Then we introduce our method to convert RNS to MRS.To see the interest of this approach, we show how we can adapted it to different division algorithms. As examples, we study the two most recent methods, cited before.

2 Some definitions and properties

We note $|X|_m$ the residue of the integer X modulo m.

2.1 Residue Number Systems

Let $(m_1, \ldots, m_n)$ a set of relatively prime numbers.
We suppose that: $m_1 < m_2 < \ldots < m_n$.

*LIRMM UMR-CNRS 5506, 161 rue ADA, 34392 Montpellier Cedex 5, FRANCE, e-m:bajard@lirmm.fr, rico@lirmm.fr

16th IMACS World Congress (© 2000 IMACS)

We note

$$M = \prod_{i=1}^{n} m_i \quad \text{and} \quad M^i = \prod_{k=1}^{i} m_k$$
$$M_j = \frac{M}{m_j} \quad \text{and} \quad M_j^i = \frac{M_i}{m_j}$$

Let X be an integer such that $0 \leq X < M$. We call **RNS representation of** X the set of residues $(x_1, \ldots, x_n)$ such that

$$\left\{ \begin{array}{lcl} x_1 & = & |X|_{m_1} \\ x_2 & = & |X|_{m_2} \\ & \cdots & \\ x_n & = & |X|_{m_n} \end{array} \right. \tag{1}$$

The set $(m_1, \ldots, m_n)$ is called **RNS base**, we note it $\mathcal{M}$. We note $\mathcal{M}_i$, the RNS base reduced to the set $(m_1, \ldots, m_i)$ with $i \leq n$.
We note $|M_i|_{m_i}^{-1}$ the **inverse of** M_i modulo m_i, $M_i \times |M_i|_{m_i}^{-1} \bmod m_i = 1$.
The Chinese Remainder Theorem assumes that all X, such that $0 \leq X < M$, have one and only one such representation, and we have:

$$X = (\sum_{i=1}^{n} x_i \times |M_i|_{m_i}^{-1} \times M_i) \bmod M \tag{2}$$

We can remark that this construction can be done in a logarithmic time with a tree of adders, but the size of the operands are close to the size of M. So it is not really implementable. But this equation can be used to approximate some value like in [BDM98] with the floating point like representation. It is also that we will do in our conversion algorithm.

2.2 Operations in RNS

Let X and Y two integers, such that $0 \leq X < M$ and $0 \leq Y < M$, we note $(x_1, x_2, \ldots, x_n)$ and $(y_1, y_2, \ldots, y_n)$ their RNS representations in the RNS base $\mathcal{M}$.
If, $X + Y < M$ then the RNS representation of $X + Y$ is

$$(|x_1 + y_1|_{m_1}, \ldots, |x_n + y_n|_{m_n})$$

If, $X \times Y < M$, then the RNS representation of $X \times Y$ is

$$(|x_1 \times y_1|_{m_1}, \ldots, |x_n \times y_n|_{m_n})$$

About the division, we have the following property: If X is a multiple of Y (ie $X \bmod Y = 0$) and if Y is invertible modulo M (ie $GCD(M, Y) = 1$, in other words, for each i, $1 \leq i \leq n$, we have $y_i \neq 0$), then the RNS representation of the inverse of Y modulo M is

$$(|y_1|_{m_1}^{-1}, \ldots, |y_n|_{m_n}^{-1})$$

and the RNS representation of the quotient $\frac{X}{Y}$ is

$$(|x_1 \times |y_1|_{m_1}^{-1}|_{m_1}, \ldots, |x_n \times |y_n|_{m_n}^{-1}|_{m_n})$$

2.3 Mixed Radix

Let X an integer, $0 \leq X < M$, and $(x_1, x_2, \ldots, x_n)$ its RNS representation in the RNS base $\mathcal{M}$. Another proof of the Chinese Remainder Theorem [Gar59] is to construct a set of residue $(x_1^{mr}, x_2^{mr}, \ldots, x_{n-1}^{mr}, x_n^{mr})$ such that:

$$\left|\begin{array}{l} x_1^{mr} = |x_1|_{m_1} \\ x_2^{mr} = \left|(x_2 - x_1^{mr}) \times |m_1|_{m_2}^{-1}\right|_{m_2} \\ x_3^{mr} = \left|(x_3 - x_1^{mr}) \times |m_1|_{m_3}^{-1} - x_2^{mr}) \times |m_2|_{m_3}^{-1}\right|_{m_3} \\ \vdots \\ x_n^{mr} = \left|(\cdots(x_n - x_1^{mr}) \times |m_1|_{m_n}^{-1} - x_2^{mr}) \times |m_2|_{m_n}^{-1})\cdots - x_{n-1}^{mr}) \times |m_{n-1}|_{m_n}^{-1}\right|_{m_n} \end{array}\right.$$

(3)

thus,

$$X = x_1^{mr} + x_2^{mr} \times m_1 + x_3^{mr} \times m_1 m_2 \cdots + x_n^{mr} \times m_1 m_2 \cdots m_{n-1} \tag{4}$$

The set $(x_1^{mr}, x_2^{mr}, \ldots, x_{n-1}^{mr}, x_n^{mr})$ is called **MRS representation** of X in the RNS base $\mathcal{M}$.
The conversion from RNS to MRS is done only with numbers smaller than the m_i, $i = 1 \ldots n$. The time needs is linear if we use n modular operators. As MRS is a positional representation, we can directly compare numbers without using large operators. The conversion from MRS to RNS is easy and is often used to pass from an RNS base to another or more generally to extend (to add moduli) an RNS base.
The division algorithms proposed by Gambeger [Gam91], Lu and Chiang [LC92] or Hitz and Kaltofen [HK95] use frequently this kind of conversion. So we propose an algorithm where the MRS digits are obtained in a logarithmic time, most significant digits first and we study the interest of this method for division.

3 A new Conversion Algorithm

3.1 Lemma

Let $X = (x_1, \cdots, x_n)_{\mathcal{M}}$ we want to compute $(x_1^{mr}, \cdots, x_n^{mr})$ the standard mixed radix representation associated to $\mathcal{M}$ i.e. such that

$$X = x_1^{mr} + x_2^{mr} m_i + \cdots + x_n^{mr} m_1 \cdots m_{n-1}.$$

Standard MRS conversion compute the least order digits first ([Knu81]). We want to compute the digits in every order using the Chinese remainder theorem and the values $X^i = (x_1, \cdots, x_i)_{\mathcal{M}_i}$. By definition (4) of the MRS, we have for each i, $1 \leq i \leq n$:

$$X^i = x_1^{mr} + x_2^{mr} m_i + \cdots + x_i^{mr} m_1 \cdots m_{i-1}.$$

Due to the construction (3) of the MRS digits from the RNS representation, the maximum order MRS digit of X^i is also the i^{th} MRS digit of X. Thus, if we are able to find this most significant digit in a logarithmic time, then we can compute the MRS representation of X in a logarithmic time. For that we compute in parallel this most significant digit for each X^i, $1 \leq i \leq n$.
So, we consider the Chinese remainder theorem (2) applies to X_i:

$$X^i = \sum_{k=1}^{i} \left| x_k \, |M_k|_{m_k}^{-1} \right|_{m_k} M_k \pmod{M^i}$$

Thus,

$$\frac{X^i}{M^i} = frac\left(\sum_{k=1}^{i} \left| x_k \, |M_k|_{m_k}^{-1} \right|_{m_k} \frac{1}{m_k} \right) \tag{5}$$

$$\text{and} \quad x_i^{mr} = \left\lfloor m_i \times \frac{X^i}{M^i} \right\rfloor \tag{6}$$

As for the floating point like notation in [BDM98], we don't compute the exact value of $\frac{X^i}{M^i}$ that require a big precision, but an approximation with p digits, p is an integer close to $\log(m_i)$. The value of p will be given in the following.

Our goal is to obtain the value of x_i^{mr}. For that we want to obtain $m_i \times \frac{X^i}{M^i}$ with an error less than $\frac{1}{4}$. Thus it will be possible to know x_i^{mr} up to one. To assume this precision, we need to compute $\frac{X^i}{M^i}$ with an error less than $2^{-\lceil \log m_i \rceil - 2}$.
We note R_i^p the value computed from the addition $\sum_{k=1}^{i} \left| x_k \left| M_k \right|_{m_k}^{-1} \right|_{m_k} \frac{1}{m_k}$ using a p digits adder where the integer part is not taken into account.
The values $\left| x_k \left| M_k \right|_{m_k}^{-1} \right|_{m_k} \times \frac{1}{m_k}$ are obtained with p digits in the fractional part from one exact modular multiplication, and one multiplication by an approximate value on p digits.
Thus for p sufficiently large, we will obtained R_i^p or $1 - R_i^p$ close to $\frac{X^i}{M^i}$. As we use a rounding near zero, we obtain:

$$0 \leq \frac{X^i}{M^i} - (R_i^p \text{ or } 1 - R_i^p) \leq 2^{-p+\lceil \log(i) \rceil} \tag{7}$$

The case $1 - R_i^p$ comes from that $\frac{X^i}{M^i}$ could be close to zero, so as we are rounding near zero R_i^p could be close to 1 (don't forget that the integer part is not taken into account).
Now if we suppose that $p \geq \lceil \log(i) \rceil + \lceil \log(m_i) \rceil + 2$ Then we obtain two cases:

$$\begin{array}{rcccl} 0 & \leq & m_i \frac{X}{M^i} - m_i R_i^p & \leq & \frac{1}{4} \\ & & \text{or} & & \\ 0 & \leq & m_i \frac{X}{M^i} - (m_i R_i^p + 1 - m_i) & \leq & \frac{1}{4} \end{array} \tag{8}$$

The last inequation corresponds to the specific case of inequation (7).
We generalize this result in the following lemma.

Lemma 3.1 *For $p = \lceil \log(n) \rceil + \left\lceil \log(\max_i(m_i)) \right\rceil + 2$, let $A_i^p = m_i R_i^p$. Let $\widetilde{x_i^{mr}} = \lfloor A_i^p \rfloor$, and $\widetilde{x_1^{mr}} = x_1^{mr} = x_1$. Then $\forall i > 1$,*

- *If $frac(A_i^p) \in [0, \frac{3}{4}]$ then $x_i^{mr} = \widetilde{x_i^{mr}}$,*
- *Else* $x_i^{mr} = \begin{cases} \widetilde{x_i^{mr}} & \text{if} \quad \left\lfloor \frac{3m_i - 1}{4} \right\rfloor \leq x_{i-1}^{mr} \leq m_{i-1} - 1 \\ \left| \widetilde{x_i^{mr}} + 1 \right|_{m_i} & \text{if} \quad 0 \leq x_{i-1}^{mr} \leq \lfloor \frac{m_i - 1}{4} \rfloor \end{cases}$

Proof

We have: $\frac{m_i X}{M_i} \in [A_i^p, A_i^p + \frac{1}{4}]$.
So, if $frac(A_i^p) \in [0, \frac{3}{4}]$, then $x_i^{mr} = \left\lfloor \frac{m_i X}{M_i} \right\rfloor = \lfloor A_i^p \rfloor = \widetilde{x_i^{mr}}$. We obtain like that, the exact value of x_i^{mr}.
Now, if $frac(A_i^p) \in [\frac{3}{4}, 1]$, as $m_i \frac{X}{M_i} \in [\widetilde{x_i^{mr}} + \frac{3}{4}, \widetilde{x_i^{mr}} + 1 + \frac{1}{4}]$, we may have two possibilities $x_i^{mr} = \widetilde{x_i^{mr}}$ or $x_i^{mr} = \widetilde{x_i^{mr}} + 1$.
This choice depends of $frac\left(m_i \frac{X}{M_i}\right)$, we have:

$$x_i^{mr} = \begin{cases} \widetilde{x_i^{mr}} & \text{if} \quad \frac{3}{4} \leq frac\left(m_i \frac{X}{M^i}\right) \leq 1 \\ \widetilde{x_i^{mr}} + 1 \pmod{m_i} & \text{if} \quad 0 \leq frac\left(m_i \frac{X}{M^i}\right) \leq \frac{1}{4} \end{cases}$$

In fact, we don't know $frac\left(m_i \frac{X}{M_i}\right)$, but, as we have

$$frac\left(m_i \frac{X}{M^i}\right) = \frac{X^{i-1}}{M^{i-1}} = \frac{x_1^{mr} + \cdots + m_1 \cdots m_{i-2} x_{i-1}^{mr}}{m_1 \cdots m_{i-1}}$$

we can conclude that:

- If $frac\left(\frac{X}{M^i}\right) \leq \frac{1}{4}$ then $x_{i-1}^{mr} \leq \lfloor \frac{m_i - 1}{4} \rfloor$,
- if $\frac{3}{4} \leq frac\left(\frac{X}{M^i}\right)$ then $\left\lfloor \frac{3m_i - 1}{4} \right\rfloor \leq x_{i-1}^{mr}$.

□

Unfortunately, this characterization of x_i^{mr} depends of the value x_{i-1}^{mr}. But, we just want to know if $\frac{X^{i-1}}{M^{i-1}} \leq \frac{1}{4}$ or $\frac{3}{4} \leq \frac{X^{i-1}}{M^{i-1}}$. Except when $\widetilde{x_{i-1}^{mr}} = m_{i-1} - 1$ (x_{i-1}^{mr} may be equal to $m_{i-1} - 1$ or to 0), knowing $\widetilde{x_{i-1}^{mr}}$ is sufficient to conclude.
Thus,

- If we know the value of x_{i-1}^{mr} or if $\widetilde{x_{i-1}^{mr}} \neq m_{i-1} - 1$ then we know the value of x_i^{mr},
- Else if we know the value of x_{i-2}^{mr} or if $\widetilde{x_{i-2}^{mr}} \neq m_{i-2} - 1$ then we know the value of x_i^{mr} and x_{i-1}^{mr},
- etc.

3.2 Property

Now, we suppose that we know all the $(\widetilde{x_i^{mr}})_{1 \leq i \leq n}$ and we define $(\varepsilon_i)_{1 \leq i \leq n}$ such that:

$$\varepsilon_i = \begin{cases} 0 & \text{if } x_i^{mr} = \widetilde{x_i^{mr}} \\ 1 & \text{if } x_i^{mr} = \widetilde{x_i^{mr}} \text{ or } \widetilde{x_i^{mr}} + 1 \pmod{m}_i \text{ depending of the value } x_{i-1}^{mr} \end{cases}$$

Property 3.2 *For all j, $2 \leq j \leq n$,*
If $\varepsilon_j = 1$ then
there exists a lowest $i \leq 1$ such that : $\varepsilon_{j-i} = 0$ or $\widetilde{x_{j-i}^{mr}} \neq m_{j-i} - 1$.
and,

- *If $0 \leq \widetilde{x_{j-i}^{mr}} \leq \lfloor \frac{m_{j-i}}{4} \rfloor$ then $x_{j-i+1}^{mr} = \cdots = x_{j-1}^{mr} = 0$ and $x_j^{mr} = \widetilde{x_j^{mr}} + 1$*
- *Else $x_{j-i+1}^{mr} = m_{j-i+1} - 1, \cdots x_{j-1}^{mr} = m_{j-1} - 1$ and, $x_j^{mr} = \widetilde{x_j^{mr}}$*

We can remark that as i is the lowest value such that $\varepsilon_{j-i} = 0$ or $\widetilde{x_{j-i}^{mr}} \neq m_{j-i} - 1$, we have $\varepsilon_{j-i+1} = \cdots = \varepsilon_j = 1$ and $\widetilde{x_{j-i+1}^{mr}} = m_{j-i+1} - 1, \cdots \widetilde{x_{j-1}^{mr}} = m_{j-1} - 1$. Furthermore, the value of $x_{j-i+1}^{mr}, \cdots x_j^{mr}$ are directly deduced from $\widetilde{x_{j-i}^{mr}}$.
Proof
The existence of i is easy to prove, as we know exactly x_1^{mr}, $\varepsilon_1 = 0$. i is the minimum number such that $\varepsilon_{j-i} = 0$ or $\widetilde{x_{j-k}^{mr}} \neq m_{j-k} - 1$.
In order to compute the result x_j^{mr}, we want to use the lemma 3.1. But, $\forall k$, $j - i < k < j$, we don't know if or not $0 \leq x_k^{mr} \leq \lfloor \frac{m_k}{4} \rfloor$, we only know that $\widetilde{x_k^{mr}} = m_k - 1$, in other words that $x_k^{mr} = 0$ or $m_k - 1$.
If $x_{j-i+1}^{mr} = 0$ then, according to the lemma 3.1, $0 \leq x_{j-i}^{mr} \leq \lfloor \frac{m_{j-i}}{4} \rfloor$.
Thus, either $0 \leq \widetilde{x_{j-i}^{mr}} \leq \lfloor \frac{m_{j-i}}{4} \rfloor$ or $\widetilde{x_{j-i}^{mr}} = m_{j-i} - 1$ and $x_{j-i}^{mr} = 0$. But by definition of i, this second case is not possible.
Now, if $x_{j-i+1}^{mr} = m_{j-i+1} - 1$ then $\left\lfloor \frac{3m_{j-i}}{4} \right\rfloor \leq x_{j-i}^{mr} \leq m_{j-i} - 1$.
As $m_{j-i} \geq 4$, we have either $\lfloor \frac{m_{j-i}}{2} \rfloor < \left\lfloor \frac{3m_{j-i}}{4} \right\rfloor - 1 \leq \widetilde{x_{j-i}^{mr}} < m_{j-i} - 1$, or, $\varepsilon_{j-i} = 0$ and $\widetilde{x_{j-i}^{mr}} = m_{j-i} - 1 = x_{j-i}^{mr}$.
By induction,we can deduce that: if $x_{j-i+1}^{mr} = 0$ then $x_{j-i+1}^{mr} = \cdots = x_{j-1}^{mr} = 0$ and $x_j^{mr} = \widetilde{x_j^{mr}} + 1$, else $x_{j-i+1}^{mr} = \widetilde{x_{j-i+1}^{mr}}, \cdots, x_j^{mr} = \widetilde{x_j^{mr}}$.
□

We present now, an $O(\log(n))$ time, conversion from RNS to MRS, with an $O(n^2)$ space complexity.

3.3 Conversion RNS-MRS

3.3.1 Approximate MRS

An (A_j^p) is obtained from equations (5) and (6), with j p-digits operators (addition-multiplication) in $O(\log(j))$ steps (simulation of a tree of adders).
Thus, with $1 + 2 + \cdots + n = \frac{n(n+1)}{2}$ p-digits operators, the $(A_j^p)_{1 \leq j \leq n}$ can be computed in parallel in $\lfloor \log(n) \rfloor$

3.3.2 Reduction

From (A_j^p), we can deduce the $\widetilde{x_j^{mr}}$ and ε_j such that: $\varepsilon_j = 0$ if we know that $\widetilde{x_j^{mr}} = x_j^{mr}$, and $\varepsilon_j = 1$ if we don't know, in other words:

$$\varepsilon_j \quad = \quad \begin{cases} 0 & \text{if} \quad frac\left(A_j^p\right) \in [0, \frac{3}{4}] \\ 1 & \text{else} \end{cases}$$

We call *group*, a sequence of consecutive values $\widetilde{x_j^{mr}}, \widetilde{x_{j+1}^{mr}}, \cdots, \widetilde{x_{j+i}^{mr}}$ such that :

- $\forall k < i$, $\varepsilon_{j+k} \geq \varepsilon_{j+k+1}$ (this means that the vector $(\varepsilon_j, \cdots, \varepsilon_{j+i})$ is of the form $(1, \cdots, 1, 0, \cdots 0)$),
- $\forall k < i$, if $\varepsilon_{j+k} = \varepsilon_{j+k+1} = 1$ then $\widetilde{x_{j+k}^{mr}} = m_{j+k} - 1$

Suppose that we have two consecutive *groups* of size $i+1$ and l respectively ($\widetilde{x_j^{mr}}, \cdots, \widetilde{x_{j+i}^{mr}}$ and $\widetilde{x_{j+i+1}^{mr}}, \cdots, \widetilde{x_{j+i+l}^{mr}}$). We can merge those *groups* to obtain one *group* of length $i+l+1$ using $O(i+l+1)$ operators :

- If $\varepsilon_{j+i+1} = 0$ then $\widetilde{x_j^{mr}}, \cdots, \widetilde{x_{j+i+l}^{mr}}$ is a *group*.
- If $\varepsilon_{j+i} = 1$ and $\widetilde{x_{j+i}^{mr}} = m_{j+i} - 1$ then, $\widetilde{x_j^{mr}}, \cdots, \widetilde{x_{j+i+l}^{mr}}$ is a *group*.
- Else
  - If $\widetilde{x_{j+i}^{mr}} \leq \left\lfloor \frac{m_{j+i}}{2} \right\rfloor$ then we can do in parallel $\forall k \leq l$,
    $\widetilde{x_{j+i+k}^{mr}} = \left| \widetilde{x_{j+i+k}^{mr}} + \varepsilon_{j+i+k} \right|_{m_{j+i+k}}$ and $\varepsilon_{j+i+k} = 0$.
  - else $\forall k \leq l$, $\varepsilon_{j+i+k} = 0$.

  By application of property 3.2, in both case, $\widetilde{x_j^{mr}}, \cdots, \widetilde{x_{j+i+l}^{mr}}$ is now a *group*.

3.3.3 Summary

Algorithm 1
Compute in parallel $(A_j^p)_{1 \leq j \leq n}$ *and deduce from those value* $(\widetilde{x_j^{mr}})$ *and* ε_j.
Let $k = n$, *each single value* $g_i = \widetilde{x_i^{mr}}$ *are* groups
while $k \neq 1$ *do*
  $\forall i$, $1 \leq i \leq \left\lfloor \frac{k}{2} \right\rfloor$ *merge the* groups g_{2i-1} *and* g_{2i} *to obtain* group g_i.
  let $k = \left\lceil \frac{k}{2} \right\rceil$.
end_ while.

This algorithm stops after $\lceil \log n \rceil$ steps, and we get only a *group* g_1 composed of the values $\widetilde{x_1^{mr}}, \cdots, \widetilde{x_n^{mr}}$. As we have $\varepsilon_1 = 0$ from the beginning, we obtain at the end of the evaluation, $\varepsilon_i = 0$ and $x_i^{mr} = \widetilde{x_i^{mr}}$, $\forall i$, $1 \leq i \leq n$,. This algorithm use $O(n^2)$ p-digits operators (the merging *groups* operation needs $O(n \log(n))$ operators).

4 Application to Division

We will study two examples of division. In the first one we directly use the previous algorithm so $O(n^2)$ p-digits operators are needed. While, in the second example, we just need to know the most significant MRS digits, so we have to use $O(n)$ operators.

4.1 Hitz and Kaltofen division

Let X and $Y \in \mathbb{N}$ two integers and $(x_1, \cdots x_n)$ and $(y_1, \cdots, y_n)$ their respective RNS representations in $\mathcal{M}$.
We consider an auxiliary RNS base $\mathcal{M}'$ such that , $M < M'$ and $gcd(M, M') = 1$.
We note Q and R the quotient and the remainder: $X = QY + R$ $(0 \leq R < Y)$.
The algorithm proposed by Hitz and Kaltofen [HK95] computes first the value $Y' = \left\lfloor \frac{M}{Y} \right\rfloor$ i.e. $M = YY' - \alpha$ $(0 \leq \alpha < Y)$.
Then, it evaluates the product $XY' = QM - Q\alpha + RY'$. As $R < Y$, we have $RY' < M$ and $Q\alpha < X < M$.
Thus, $\left\lfloor \frac{XY'}{M} \right\rfloor = Q$ or $Q - 1$.

Algorithm 2
Compute Y'
$Q = \left\lfloor \frac{XY'}{M} \right\rfloor$
$R = X - QY$
If $R < Y$ *then*
    return (Q, R)
else
    return $(Q+1, R-1)$

The division by M is done with the auxiliary RNS base $\mathcal{M}'$.
Y' and XY' are computed in the two RNS bases $\mathcal{M}$ and $\mathcal{M}'$, thus we obtain $XY' \bmod M$ in $\mathcal{M}$. We convert $XY' \bmod M$ in $\mathcal{M}'$, then we can compute $XY' - XY' \bmod M$ in $\mathcal{M}'$, the obtained value is a multiple of M, so, as we have seen in the presentation of the RNS $\frac{XY' - XY' \bmod M}{M} = (XY' - XY' \bmod M) \times |M|_{M'}^{-1}$ in the RNS base $\mathcal{M}'$.
All those RNS bases extensions can use our conversion algorithm and can be done in $O(\log(n))$ using $O(n^2)$ operators.
To compute $Y' = \left\lfloor \frac{M}{Y} \right\rfloor$ Hitz and Kaltofen use a Newton algorithm :

Algorithm 3
$Z_1 = 1$
$Z_2 = 2$
while $Z_1 \neq Z_2$ *do*
    $Z_1 = Z_2$
    $Z_2 = \left\lfloor \frac{Z_1 \times (2M - Z_1 Y)}{M} \right\rfloor$
end_while
if $(M - Y * Z_2 < Y)$ *then*
    return Z_1
else
    return $Z_2 + 1$

They prove that $O(n)$ iterations are needed if we initialize Z_2 to the value 2. But they also show that if we have a better starting point $\frac{M}{4Y} \leq Z_2 \leq \frac{M}{Y}$, we get the value $\left\lfloor \frac{M}{Y} \right\rfloor$ in only $O(\log(n))$ iterations.
The last step of the division is the exact comparison $R < Y$, so we need to compute the mixed radix representation of Y. We have,

$$\begin{aligned} Y &= y_1^{mr} + \cdots + y_k^{mr} m_1 \cdots m_{k-1} \\ y_k^{mr} &\neq 0 \end{aligned}$$

We can easily obtain a good approximation $\tilde{y}$ of $\frac{Y}{M^k}$. For example, we can choose $\tilde{y}$ as the rounding up value of $\frac{y_k^{mr} m_{k-1} + y_{k-1}^{mr}}{m_k m_{k-1}}$ $\lceil \log(\max_i(m_i)) \rceil$ digits.
Thus, if

$$Z_2 = \left\lfloor \frac{m_n}{\tilde{y}} \right\rfloor m_{k+1} \cdots m_{n-1}$$

then,

$$\begin{array}{rcccl} \frac{Y}{M^k} & \leq & \tilde{y} & \leq & \frac{Y}{M^k} + \frac{2}{m_k m_{k-1}} \\ \frac{M^k m_n}{Y} \frac{1}{1 + \frac{2}{m_k m_{k-1}}} - 1 & \leq & \left\lfloor \frac{m_n}{\tilde{y}} \right\rfloor & \leq & \frac{M^k m_n}{Y} \\ \frac{M^k m_n}{Y} \left(1 - \frac{4}{m_k m_{k-1}}\right) - 1 & \leq & \left\lfloor \frac{m_n}{\tilde{y}} \right\rfloor & \leq & \frac{M^k m_n}{Y} \\ \frac{M^k}{Y} \frac{M^n}{Y} - \frac{4M^n}{m_k m_{k-1} Y} - \frac{YM^n}{YM^k m_n} & \leq & Z_2 & \leq & \frac{M^n}{Y} \end{array}$$

And, if $\forall i,\ 4 \leq m_i$, then,

$$\frac{M^n}{4Y} \leq \frac{M^n}{Y} - \frac{M^n}{4Y} - \frac{M^n}{4Y} \leq Z_2 \leq \frac{M^n}{Y}$$

To conclude, only $O(\log(n))$ iterations are needed in order to found the final value Y'. The cost of each iteration is in $O(\log(n))$, thus the division algorithm is performed in $O(\log(n)^2)$.

4.2 Bajard, Didier and Muller division

Let X and Y be two RNS number, $0 \leq X \leq \frac{3M}{4}$.
We note k, $0 \leq k \leq n$, the integer such that

$$Y = \quad y_1^{mr} + \ldots + y_{k-1}^{mr} m_1 \cdots m_{k-2} + y_k m_1 \cdots m_{k-1} \quad \text{and } y_k \neq 0$$

We note $\tilde{y}$ an approximation of $\frac{Y}{M^k}$, such that $\frac{Y}{M^k} \leq \tilde{y} \leq \frac{Y}{M^k} + 2^{-e}$, with $e \geq 2 \times \lceil \log(\max_i(m_i)) \rceil + 3$.

4.2.1 One step of the division

We consider the $n - i + 1$ iteration of our algorithm. We suppose that $X \leq \frac{3}{4}M^i$ (ie the partial remainder during computing), and we want to found Q such that

$$\begin{array}{lll} 0 & \leq X - QY & \leq \frac{3}{4}M^{i-1} \\ \text{or } 0 & \leq X - QY & < 2Y \text{ (for the last iteration)} \end{array} \tag{9}$$

To find Q, we compute R_i^p an approximation of $\frac{X}{M^i}$. R_i^p is evaluated with $p = \lceil \log(n) \rceil + \lceil \log(\max_i(m_i)) \rceil + 2$ digits with our algorithm in $O(\log(i))$, with $O(i)$ operators.
As $0 \leq \frac{X}{M^i} \leq \frac{3}{4}$, we have $R_i^p \in [0, \frac{3}{4}] \cup [1 - 2^{-p+\lceil \log(n) \rceil}, 1]$. Thus we take into account in which intervalR_i^p is to select Q.

- If, $R_i^p \in [1 - 2^{-p+\lceil \log(n) \rceil}, 1]$, then

  $$\begin{array}{rcccl} 0 & \leq & \frac{X}{M^i} & \leq & \frac{1}{4m_i} \\ 0 & \leq & X & \leq & \frac{M^{i-1}}{4} \end{array}$$

  and $Q = 0$.

- Else, we have

  $$\frac{X}{M^i} - 2^{-e'} \quad \leq \quad R_i^p \quad \leq \quad \frac{X}{M^i}$$

  with, $e' = \lceil \log(\max_i(m_i)) \rceil + 2$.
  we evaluate q and $w(k+1, i-2)$ such that:

  $$q = \begin{cases} \left\lfloor \frac{R_i^p m_i m_{i-1}}{\tilde{y}} \right\rfloor & \text{if} \quad k \leq i-2 \\ \left\lfloor \frac{R_i^p m_i}{\tilde{y}} \right\rfloor & \text{if} \quad k = i-1 \\ \left\lfloor \frac{R_i^p}{\tilde{y}} \right\rfloor & \text{if} \quad k = i \end{cases}$$

  $$w(\alpha, \beta) = \begin{cases} m_\alpha \cdots m_\beta & \text{if} \quad \alpha \leq \beta \\ 1 & \text{if} \quad \alpha > \beta \end{cases}$$

  thus we obtain, $Q = q \times w(k+1, i-2)$

Now we verify that this value of Q satisfies inequation (9).
We remark that, $\forall x \in [0, \frac{1}{2}]$, we have $\frac{1}{1+2x} \geq (1 - 2x)$. So we obtain,

$$\begin{array}{rcccl} \frac{\frac{X}{M^i} - 2^{-e'}}{\frac{Y}{M^k} + 2^{-e}} & \leq & \frac{R_i^p}{\tilde{y}} & \leq & \frac{XM^k}{YM^i} \\ \frac{M^k}{Y}\left(\frac{X}{M^i} - 2^{-e'}\right)\left(1 - \frac{M^k}{Y}2^{-e+1}\right) & \leq & \frac{R_i^p}{\tilde{y}} & \leq & \frac{XM^k}{YM^i} \\ \frac{XM^k}{YM^i} - 2^{-e'}\frac{M^k}{Y} - \frac{XM^{k2}}{Y^2M^i}2^{-e+1} & \leq & \frac{R_i^p}{\tilde{y}} & \leq & \frac{XM^k}{YM^i} \end{array} \tag{10}$$

- If $k \leq i-2$, as $q = \left\lfloor \frac{R_i^p}{\tilde{y}} \right\rfloor$, inequation (10) becomes:

  $$\begin{array}{rcccl} \frac{XM^k}{YM^{i-2}} - 2^{-e'}\frac{M^k m_i m_{i-1}}{Y} - \frac{XM^{k2}}{Y^2M^{i-2}}2^{-e+1} - 1 & \leq & q & \leq & \frac{XM^k}{YM^{i-2}} \\ \frac{X}{Y} - 2^{-e'}\frac{M^i}{Y} - \frac{XM^{k2}}{Y^2M^k}2^{-e+1} - w(k+1, i-2) & \leq & Q & \leq & \frac{X}{Y} \\ X - 2^{-e'}M^i - \frac{XM^k}{Y}2^{-e+1} - Yw(k+1, i-2) & \leq & QY & \leq & X \\ X - \frac{M^{i-1}}{8} - \frac{M^{i-1}}{8} - \frac{M^{i-1}}{m_{i-1}} & \leq & QY & \leq & X \end{array}$$

then, as $\forall i,\ m_i \geq 2$,

$$0 \quad \leq \quad X - QY \quad \leq \quad \tfrac{3}{4}M^{i-1}$$

- If $k = i-1$, we obtain,

$$\begin{array}{rcccl}
\frac{XM^k}{YM^{i-1}} - 2^{-e'}\frac{M^k m_i}{Y} - \frac{XM^{k2}}{Y^2M^{i-1}}2^{-e+1} - 1 & \leq & q & \leq & \frac{XM^k}{YM^{i-1}} \\
\frac{X}{Y} - 2^{-e'}\frac{M^i}{Y} - \frac{XM^k}{Y^2}2^{-e+1} - 1 & \leq & q & \leq & \frac{X}{Y} \\
X - 2^{-e'}M^i - \frac{XM^k}{Y}2^{-e+1} - Y & \leq & qY & \leq & X \\
X - \frac{M^k}{8} - \frac{M^k}{8} - Y & \leq & QY & \leq & X
\end{array}$$

  We know that $M^{k-1} \leq Y < M^k$.

  - If $Y \leq \frac{M^k}{2}$ then
  $$0 \quad \leq \quad X - QY \quad \leq \quad \tfrac{3}{4}M^{i-1}$$
  - else
  $$0 \quad \leq \quad X - QY \quad \leq \quad \tfrac{1}{4}M^{i-1} + Y \quad < \quad 2Y$$

- And if $k = i$, we have,

$$\begin{array}{rcccl}
\frac{X}{Y} - 2^{-e'}\frac{M^k}{Y} - \frac{XM^k}{Y^2}2^{-e+1} - 1 & \leq & q & \leq & \frac{X}{Y} \\
X - 2^{-e'}M^k - \frac{XM^k}{Y}2^{-e+1} - Y & \leq & qY & \leq & X \\
X - \frac{M^{k-1}}{8} - \frac{M^{k-1}}{8} - Y & \leq & QY & \leq & X
\end{array}$$

  As $M^{k-1} \leq Y$, we obtain,
  $$0 \quad \leq \quad X - QY \quad \leq \quad \tfrac{5}{4}Y$$

So, we have defined a selection function of Q that satisfy (9). We propose the following algorithm.

4.2.2 The algorithm

Algorithm 4

compute $(y_1^{mr}, \cdots, y_n^{mr})$
compute $\tilde{y}$
$R = R_n^p(X)$
$i = n$
$Q_f = 0$
while $k < i$ *or* $(k = i$ *and* $\tilde{y} \leq R)$
  if $k = i$ *then*
    $q = \lfloor R \rfloor \tilde{y}$
  else if $k = i-1$ *then*
    $q = \lfloor Rm_i \rfloor \tilde{y}$
    else
      $q = \lfloor Rm_i m_{i-1} \rfloor \tilde{y}$
      $Q = qw(k+1, i-2)$
    end if
  end if
  $X = X - QY$
  $Q_f = Q_f + Q \qquad i = i - 1$
  $R = R_i^p$
end while
if $(X < Y)$ *then return* (Q_f, X)
else return $(Q_f + 1, Y - 1)$

The value $(y_1^{mr}, \cdots, y_n^{mr})$ can be computed with the usual algorithm presented in section 2.3 with n operators (multiplier and adder) in k steps (after the k^{th} step, we get 0 as result for all the operator). The "while" loop in the

algorithm takes $n-k+1$ iterations (because $k \leq i \leq n$). Each iteration needs to compute R_i^p using a logarithmic tree adder using n operators with $\lceil \log(\max_i(m_i)) \rceil + \lceil \log(n) \rceil + 2$ digits.
So the complexity is $O(n \log(n))$ with $O(n)$ operators. That is the same complexity than the original algorithm. But, we use now $2n$ operators of $\lceil \log(\max_i(m_i)) \rceil + \lceil \log(n) \rceil + 2$ digits
while the original division uses $2n$ operators of $2 \times \lceil \log(\max_i(m_i)) \rceil$ digits. Our method is better when $4n < \max_i(m_i)$. Another point is that we compute R_i^p with a sum of i terms and i decreases a each step, whereas previously the sum has always n terms for the approximation of X.

5 Conclusion

The RNS to MRS conversion algorithm that we have presented in this paper is in $O(\log(n))$. But it uses $O(n^2)$ operators. If we have a small number of moduli, that is the case in DSP, this method becomes efficient. The great advantage of this approach is that the MRS representation is obtained most significant digits first. Thus we can use it partially like in the adaptation to Bajard, Didier Muller algorithm.

References

[BDK98] J-C. Bajard, L-S. Didier, and P. Kornerup. An RNS Montgomery Modular Multiplication Algorithm. *ieee-tc*, 46(7):766–776, July 1998.

[BDM98] Jean-Claude Bajard, Laurent Sthéphane Didier, and Jean-Michel Muller. A new Euclidean division algorithm for residue number systems. *Journal of VLSI Signal Processing*, 19:167–178, 1998.

[Gam91] D. Gamberger. New Approach to Integer Division in Residue Number System. In P. Kornerup and D. W. Matula, editors, *Proceedings of the 10th IEEE Symposium on Computer Arithmetic*, pages 84–91, Grenoble, France, 1991. IEEE Computer Society Press, Los Alamitos, CA.

[Gar59] H. L. Garner. The residue number system. *IRE Transactions on Electronic Computers*, EC-8:140–147, 1959.

[HK95] Hitz and Kaltofen. Integer division in residue number systems. *IEEETC: IEEE Transactions on Computers*, 44, 1995.

[KJM86] R. Krishnan, G. A. Jullien, and W. C. Miller. Complex digital signal processing using quadratic residue number systems. *IEEE Trans. Acoustics, Speech and Signal Processing*, ASSP-34(1):166, 1986.

[Knu81] D. E. Knuth. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*. Second Edition, Addison-Wesley, Reading, 1981.

[LC92] M. Lu and J. S. Chiang. A novel division algorithm for the residue number system. *IEEE Transactions on Computers*, 41(8):1026–1032, August 1992.

[ST67] N. Szabo and R. I. Tanaka. *Residue Arithmetic and its application to Computer Technology*. McGraw-Hill, 1967.

[SV55] Svoboda and Valach. Operational circuits (czech). *STROJEZI: Stroje na Zpracovani Informaci, Nakl. CSAV, Praha*, 3, 1955.