RN-coding of numbers: definition and some properties

Peter Kornerup
Dept. of Math. & Computer Science
University of Southern Denmark
Odense, Denmark
E-mail: kornerup@imada.sdu.dk

Jean-Michel Muller
CNRS-LIP-Arénaire
Ecole Normale Supérieure de Lyon
Lyon, France
E-mail: Jean-Michel.Muller@ens-lyon.fr

Abstract— **We define RN-codings as radix-β signed representations of numbers for which rounding to the nearest is always identical to truncation. After giving characterizations of such representations, we investigate some of their properties, and we suggest algorithms for conversion to and from these codings.**

Keywords— **Number systems, signed-digit representation, Booth recoding, balanced ternary system, roundings.**

I. INTRODUCTION

We are interested in representations or integer or real numbers that are particular cases of signed-digit representations. It is well-known that in radix $\beta \geq 2$, every integer (real number) can be represented by a finite (infinite) digit chain, with digits in the digit set $\{-a, -a+1, \ldots, +a-1, +a\}$, provided that $2a+1 \geq \beta$. If $2a+1 \geq \beta+1$ then some numbers can be represented in more than one way and the number system is said to be redundant. Redundancy makes it possible to perform fast, fully parallel, additions [Avi61], or to perform the four arithmetic operations on-line, i.e., digit-serially, most significant digit first [TE77], [Erc84].

And yet, signed-digit representations, either redundant or not, naturally appear in many other problems than parallel additions. In 1840, Cauchy suggested the use of digits -5 to $+5$ in radix 10 to simplify multiplications [Cau40]. The "balanced ternary" system (radix 3 with digits -1, 0 and 1) has some interesting properties. It is worth being noticed that computers using that system were actually built[1], at Moscow University, during the 60's. Knuth [Knu98] mentions that in that number system, the operation of rounding to the nearest is identical to truncation. When trying to design fast multipliers, it is sometimes interesting to recode one of the input binary operands so that its representations contains as many zeros as possible. This implies that we use, in radix 2, digits equal to -1, 0 or $+1$. A first attempt was suggested by Booth [Boo51]. His solution does not always increase the number of zero digits in the representation (what is actually used in modern multipliers, and frequently improperly named "Booth recoding" – "modified Booth recoding" is preferable – is a somewhat different method). See [PB04] for a recent presentation of this topic. Signed-digit representations also naturally appear as the output of fast digit-recurrence algorithms for division and square root [EL94].

[1]http://www.icfcst.kiev.ua/MUSEUM/PHOTOS/setun-1.html

In this paper we wish to investigate the positional, radix β, number systems that share with the balanced ternary system the property that truncating is equivalent to rounding to the nearest. Interestingly enough, the representations generated by Booth recoding algorithm satisfy that property. We will call such representations *RN-codings*, where "RN" stands for "Round to Nearest".

Most conventional representations are not RN-codings. When we truncate the conventional radix 10 representation of a given number x at some position j (i.e., of weight 10^j), the obtained number is not necessarily the multiple of 10^j that is closest to x.

Definition 1 (RN-codings): Let β be an integer greater than or equal to 2. The digit sequence

$$D = d_n d_{n-1} d_{n-2} d_{n-3} \ldots d_0 . d_{-1} d_{-2} \ldots$$

(with $-\beta+1 \leq d_i \leq \beta-1$) is an RN-coding, in radix β of x if

1) $x = \sum_{i=-\infty}^{n} d_i \beta^i$ (that is D is a radix β representation of x);
2) for any $j \leq n$,

$$\left| \sum_{i=-\infty}^{j-1} d_i \beta^i \right| \leq \frac{1}{2}\beta^j,$$

that is, if we truncate the digit sequence to the right at any position, the obtained sequence is always the number of the form $d_n d_{n-1} d_{n-2} d_{n-3} \ldots d_j$ that is closest to x.

Hence, truncating the RN-coding of a number at any position is equivalent to rounding it to the nearest.

For example, in radix 10, the RN-coding of π starts with

$$3.142\overline{4}\overline{1}3\overline{3}5\overline{4}4\overline{1}0\overline{2}\overline{1}3\cdots$$

where (as usual) $\overline{4}$ means -4.

II. Some Properties

In the following, we will consider radix β symmetric number systems (i.e.with a digit set of the form $\{-a, \dots +a\}$), that are not over-redundant (i.e., such that $a \leq \beta - 1$).

Theorem 1 (Characterizations of RN-codings):

- If β is odd, then the digit string $D = d_n d_{n-1} d_{n-2} d_{n-3} \dots d_0 . d_{-1} d_{-2} \dots$ is an RN-coding if and only if
$$\forall i, \frac{-\beta+1}{2} \leq d_i \leq \frac{\beta-1}{2};$$
- If $\beta = 2$ then the digit string $D = d_n d_{n-1} d_{n-2} d_{n-3} \dots d_0 . d_{-1} d_{-2} \dots$ (with $d_i \in \{-1, 0, 1\}$) is an RN-coding if and only if the non-zero digits have alternate signs (i.e., $d_i \neq 0$ implies that the largest $j < i$ such that $d_j \neq 0$ satisfies $d_i = -d_j$;
- If β is even and larger than 2 then $D = d_n d_{n-1} d_{n-2} d_{n-3} \dots d_0 . d_{-1} d_{-2} \dots$ is an RN-coding if and only if
  1) all digits have absolute value less than or equal to $\beta/2$;
  2) if $|d_i| = \beta/2$, then the first non-zero digit that follows on the right has an opposite sign, that is, the largest $j < i$ such that $d_j \neq 0$ satisfies $d_i \times d_j < 0$.

The proof is straightforward.

Example 1: For instance, in radix 3 with digits in $\{-1, 0, 1\}$, or in radix-5 with digits in $\{-2, -1, 0, 1, 2\}$, all representations are RN-codings. In these number systems, truncating is equivalent to rounding to the nearest. In radix 10 with digits $\{-5, \cdots, +5\}$, $450\bar{1}3$ is an RN-coding, whereas 45013 is not an RN-coding.

Another consequence of Theorem 1 is that, in radix 2, storing an n-digit RN-coding requires $n+1$ bits only: it suffices to store the sign of the first nonzero digit. The signs of all following nonzero digits – since these signs alternate – are immediately deduced from it, so that we can represent these digits by ones only.

Theorem 2 (Uniqueness of finite representations):

- If β is odd, then a finite RN-coding of x is unique (comes from Theorem 1: in that case, the number system is nonredundant);
- If β is even, then some numbers may have two finite representations. In that case, one has its rightmost nonzero digit equal to $-\frac{\beta}{2}$, the other one has its rightmost nonzero digit equal to $+\frac{\beta}{2}$.

Proof: If β is odd, the result is an immediate consequence of the fact that the number system is non redundant. If β is even, then consider two different RN-codings that represent the same number x, and consider the largest position j (that is, of weight 2^j) such that these RN-codings, truncated to the right at position j differ. Define x_a and x_b as the numbers represented by these digit chains. Obviously, $x_a - x_b \in \{-\beta^j, 0, +\beta^j\}$. Now, $x_a = x_b$ is impossible: since the digits of weight β^j of the considered digit chains differ, $x_a = x_b$ would imply that the two chains, truncated at position $j+1$ would differ, which is impossible (j is the first position such that the digit chains differ). Hence $x_a \neq x_b$. Without loss of generality, assume $x_b = x_a + \beta^{-j}$. This implies $x = x_a + \beta^{-j}/2 = x_b - \beta^{-j}/2$. Hence, the remaining digit chains (i.e., the parts that were truncated) are digit chains that start from position $j-1$, and that represent $\pm\beta^j/2$.

The only way of representing $\beta^j/2$ with an RN-coding and starting from position $j-1$ is
$$\left(\frac{\beta}{2}\right) 0000 \cdots 0.$$

This is easily shown: if the digit at position $j-1$ of a number is less than or equal to $\frac{\beta}{2} - 1$, then that number is less than
$$\left(\frac{\beta}{2} - 1\right) \beta^{j-1} + \left(\frac{\beta}{2}\right) \sum_{i=0}^{j-2} \beta^j < \beta^j/2$$
(since the largest allowed digit is $\beta/2$). Also, the digit at position $j-1$ of a RN code cannot be larger than or equal to $\frac{\beta}{2} + 1$. This ends the proof.

If β is even, then a number whose finite representation (by an RN-coding) has its last nonzero digit equal to $\beta/2$ has an alternative representation ending with $-\beta/2$ (just assume the last two digits are $d(\beta/2)$: since the representation is an RN-coding, $d < \beta/2$, hence if we replace these two digits by $(d+1)(-\beta/2)$ we still have an RN-coding of the same number). This has an interesting consequence: truncating off a number which is a tie will round either way, depending on which of the two possible representations the number happens to have. Hence, there is no bias in the rounding. ■

Example 2:

- In radix-7, with digits $\{-3, -2, -1, 0, 1, 2, 3\}$, all representations are RN-codings, and no number has more than one representation;
- In radix 10 with digits $\{-5, \dots, +5\}$, the number 15 has two RN-codings, namely 15 and $2\bar{5}$.

Theorem 3 (Uniqueness of infinite representations): We now consider *infinite* codings, i.e., codings that do not ultimately terminate with an infinite sequence of zeros.

- If β is odd, then some numbers may have two infinite RN-codings. In that case, one is eventually finishing with the infinite digit chain
$$\frac{\beta-1}{2}\frac{\beta-1}{2}\frac{\beta-1}{2}\frac{\beta-1}{2}\frac{\beta-1}{2}\frac{\beta-1}{2}\cdots$$
and the other one is eventually finishing with the infinite digit chain
$$\frac{-\beta+1}{2}\frac{-\beta+1}{2}\frac{-\beta+1}{2}\frac{-\beta+1}{2}\frac{-\beta+1}{2}\frac{-\beta+1}{2}\cdots;$$

- If β is even, then two different infinite RN-codings necessarily represent different numbers. As a consequence, a number that is not an integer multiple of an integral (positive or negative) power of β has a unique RN-coding.

Proof: If β is odd, the existence immediately comes from

$$
\begin{aligned}
1.\frac{-\beta+1}{2}\frac{-\beta+1}{2}\frac{-\beta+1}{2}\frac{-\beta+1}{2}\cdots \\
= 0.\frac{\beta-1}{2}\frac{\beta-1}{2}\frac{\beta-1}{2}\frac{\beta-1}{2}\cdots \\
= \frac{1}{2}
\end{aligned}
$$

Now, if for any β (odd or even) two different RN-codings represent the same number x, then consider them truncated to the right at some position j, such that the obtained digit chains differ. The obtained digit chains represent numbers x_a and x_b whose difference is 0 or $\pm\beta^j$ (a larger difference is impossible for obvious reasons).

First, consider the case where β is odd. The difference $x_b - x_a$ cannot be 0, otherwise these truncated digit chains would be the same from Theorem 2. So they differ by $\pm\beta^j$. Hence, from the definition of RN-codings, and assuming $x_a < x_b$, we have $x = x_a + \beta^j/2 = x_b - \beta^j/2$. Since β is odd, the only way of representing $\beta^j/2$ is with the infinite digit chain (that starts from position $j-1$)

$$\frac{\beta-1}{2}\frac{\beta-1}{2}\frac{\beta-1}{2}\frac{\beta-1}{2}\cdots$$

the result immediately follows.

Now, consider the case where β is even. Let us first show that $x_a = x_b$ is impossible. From Theorem 2, this would imply that one of the corresponding digit chains would terminate with the digit sequence $-\frac{\beta}{2}00\cdots00$, and the other one with the digit chain $+\frac{\beta}{2}00\cdots00$. But from Theorem 1, this would imply that the remaining (truncated) terms are positive in the first case, and positive in the second case, which would mean (since $x_a = x_b$ implies that they are equal) that they would both be zero, which is not compatible with the fact that the representations of x are assumed infinite. Hence $x_a \neq x_b$. Assume $x_a < x_b$, which implies $x_b = x_a + \beta^j$. We necessarily have $x = x_a + \beta^j/2 = x_b - \beta^j/2$. Although $\beta^j/2$ has several possible representations in a "general" signed-digit radix β system, the only way of representing it with an RN-coding is to put a digit $\beta/2$ at position $j-1$, no infinite representation is possible. ■

Example 3:

- In radix-7, with digits $\{-3,-2,-1,0,1,2,3\}$, the number $3/2$ has two infinite representations, namely
$$1.3333333333\cdots$$
and
$$1.\overline{3333333333}\cdots$$
- In radix 10 with digits $\{-5,\ldots,+5\}$, the RN-coding of π is unique.

III. CONVERSION ALGORITHMS

Here, we will be especially interested in conversions that can be done "in parallel". To be able to prove results, we must state more precisely what we understand by that. This is the purpose of the following definition.

Definition 2: A function that returns an output digit chain $\delta_{n+k}\delta_{n+k-1}\cdots\delta_j$ from an input digit-chain $d_n d_{n-1} d_{n-2}\cdots d_j$ (where j can be $-\infty$) is SW-computable (where "SW" stands for "sliding window") if there exists an integer k and a function ϕ such that

$$\delta_i = \phi(d_{i+k}d_{i+k-1}\cdots d_i\cdots d_{i-k}).$$

i.e., the output digits are deduced from a constant-sized "sliding window" of input digits.

If the output chain is SW-computable, then it can also be computed by a transducer, either most significant digit first or (in the case of finite representations), least significant digit first. The converse is not necessarily true (for instance, we will see that conversion from a radix β RN-coding to the conventional non redundant radix β representation is not SW-computable, although it is straightforwardly doable by a transducer, least significant digit first).

A. The particular case of radix 2

1) Conventional binary to radix 2 RN-coding:

In the special case of radix 2, converting a number from the conventional non-redundant binary system to an RN-coding is done very easily. Consider an input value:

$$x = d_n d_{n-1} d_{n-2}\cdots d_j$$

with $d_i = 0,1$ and $j < n$. Then the digit-sequence

$$\delta_{n+1}\delta_n\delta_{n-1}\cdots\delta_j$$

defined by

$$\delta_k = d_{k-1} - d_k \qquad (1)$$

(with by convention, for finite chains, $d_{j-1} = 0$) is an RN-coding of x. This is actually the well-known Booth recoding of x [Boo51].

Hence the conversion is SW-computable. This is illustrated by the following example, where the input is considered a 2's complement number.

input bits left-shifted one pos. d_k	1	0	0	1	0	1	1	0	1	**0**
input bits with sign-extension d_k	**1**	1	0	0	1	0	1	1	0	1
output digits δ_k	0	$\overline{1}$	0	1	$\overline{1}$	1	0	$\overline{1}$	1	$\overline{1}$

Note that the bit-pairs above each output digit δ_i forms an encoding of that digit.

Proof: First, the digit chain $\delta_{n+1}\delta_n\delta_{n-1}\cdots\delta_j$ is a representation of x. This is immediate by noticing that the algorithm actually computes $2x-x$. Second, consider two nonzero digits δ_k and δ_ℓ, with $k > \ell$, separated by zeros (i.e., $\delta_m = 0 for \ell < m < k$). Let us show that δ_k and δ_ℓ have opposite signs.

Assume $\delta_k = 1$ (the case $\delta_k = -1$ is very similar). From $\delta_k = d_{k-1} - d_k$ we deduce that $d_{k-1} = 1$ and $d_k = 0$. If $\ell < k-1$, then since $\delta_{k-1} = 0$, $d_{k-2} = d_{k-1} = 1$, and by induction that $d_m = 1$ for any m between $k-1$ and ℓ. This is also obviously true if $\ell = k-1$. From $\delta_\ell = d_{\ell-1} - d_\ell \neq 0$, we deduce that $d_{\ell-1} = 0$, hence $\delta_\ell = -1$. ■

2) Radix 2 RN-coding to conventional binary:

It is worth being noticed that the converse conversion is not SW-computable. This is easily understood by looking at the following example. The RN-coding

$$10000000000$$

represents the binary number

$$10000000000;$$

whereas the RN-coding

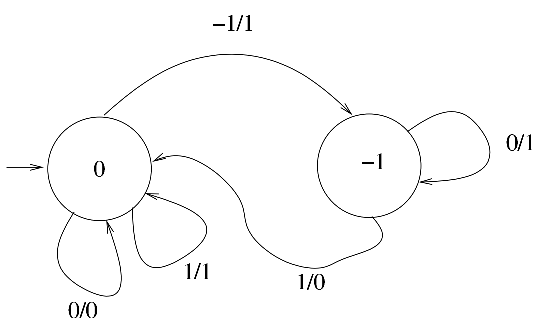
$$1000000000\bar{1}$$

represents the binary number

$$01111111111.$$

This easily generalizes to an arbitrarily long chain of input zeros. Hence, when we examine a chain of zeros, the information that is required to convert it can be arbitrarily far away.

The conversion can be done using a sequential, right-to-left process. This is obvious since radix 2 RN-codings are particular cases of signed-digit codings, so the usual conversion method applies. And yet, the fact that in the RN-codes the signs of the non-zero digits alternates allows a simpler algorithm, described by the following right-to-left transducer.



B. The particular case of radix 10

1) Conventional decimal to radix 10 RN-coding: Again, consider an input value

$$x = d_n d_{n-1} d_{n-2} \cdots d_j$$

with $0 \leq d_i \leq 9$, and let us define $d_{j-1} = 0$ for finite chains (d_{j-1} is added to simplify the presentation of the algorithm, otherwise we would have to treat the least significant digit as a special case). Notice that j can be $-\infty$ in case of an infinite input digit chain. We build an RN-coding

$$\delta_n \delta_{n-1} \cdots \delta_j$$

of x as follows. We scan in parallel all consecutive positions $d_k d_{k-1}$ and deduce δ_k by

$$\delta_k = \begin{cases} d_k & \text{if} \quad d_k < 5 \quad \text{and} \quad d_{k-1} < 5 \\ d_k + 1 & \text{if} \quad d_k < 5 \quad \text{and} \quad d_{k-1} \geq 5 \\ d_k - 10 & \text{if} \quad d_k \geq 5 \quad \text{and} \quad d_{k-1} < 5 \\ d_k - 9 & \text{if} \quad d_k \geq 5 \quad \text{and} \quad d_{k-1} \geq 5 \end{cases} \tag{2}$$

For instance, applied to the input digit chain 2718281828459, this algorithm returns $3\bar{3}2\bar{2}3\bar{2}2\bar{2}3\bar{2}5\bar{4}\bar{1}$, and on 9172 it returns $\bar{1}2\bar{3}2$ corresponding to the value -828 in 10's complement.

Proof:

1) **The obtained digit chain is an RN-coding.** It suffices to show that the digit sequence $\delta_{n+1}\delta_n\delta_{n-1}\cdots\delta_j$ satisfies the characterization property of Theorem 1. First the fact that $\forall k, |\delta_k| \leq 5$ comes obviously from the definition. Now, we must make sure that if $\delta_k = 5$, then the largest $\ell < k$ such that $\delta_\ell \neq 0$ is negative, and if $\delta_k = -5$, then the largest $\ell < k$ such that $\delta_\ell \neq 0$ is positive.
   - if $\delta_k = 5$,then from (2), we deduce that $d_k = 4$ and $d_{k-1} \geq 5$. But $d_{k-1} \geq 5$ implies that either $\delta_{k-1} < 0$, or $d_{k-1} = 9$ (which gives $\delta_{k-1} = 0$) and $d_{k-2} \geq 5$. Again, $d_{k-2} \geq 5$ implies that either $\delta_{k-2} < 0$ or $\delta_{k-2} = 0$ and $d_{k-3} \geq 5$: we continue by induction and deduce that the first nonzero δ_ℓ (if any) generated by that process will be negative;
   - $\delta_k = -5$, then from (2), we deduce that $d_k = 5$ and $d_{k-1} < 5$. Therefore δ_{k-1} will be equal to d_{k-1} or $d_{k-1} + 1$, it will be positive unless $d_{k-1} = 0$ and $d_{k-2} < 5$ (which gives $\delta_{k-1} = 0$). In that case δ_{k-2} will be equal to d_{k-2} or $d_{k-2} + 1$: we continue by induction and deduce that the first nonzero δ_ℓ (if any) generated by that process will be positive.
2) **The obtained digit chain represents the same number as the original digit chain.**
   Define variables c_k as

$$c_{k+1} = \begin{cases} 1 & \text{if} \quad d_k \geq 5 \\ 0 & \text{if} \quad d_k < 5 \end{cases} \tag{3}$$

With $c_j = 0$. It immediately follows from (2) and (3) that

$$\delta_k = d_k + c_k - 10c_{k+1}.$$

(By the way, this is another way of presenting the conversion algorithm, as a kind of "addition" algorithm, where the c_k play the role of carries) Therefore, since $c_{n+1} = c_j = 0$,

$$\begin{aligned} \sum_{k=j}^{n+1} \delta_k 10^k &= \sum_{k=j}^{n+1} (d_k + c_k - 10c_{k+1})\, 10^k \\ &= \sum_{k=j}^{n+1} d_k 10^k = x \end{aligned}$$

Hence the obtained digit chain represents the same number as the original one.

■

2) Radix 10 *RN-coding to conventional decimal:* Again, as in radix 2, conversion to conventional decimal is not SW-computable (the same example with a long string of zeros applies). As in radix 2, we can perform the conversion just by considering that an RN number is a particular case of a signed-digit number and applying the usual algorithms.

C. Odd radices: General case

In odd radices, the RN-codings are the (non redundant) signed digit representations that use digits $\frac{-\beta+1}{2}, \frac{-\beta+2}{2}, \ldots, \frac{\beta-1}{2}$. It is known that conversion to these systems cannot be done "in parallel": they are not SW-computable. We illustrate this with the following example. In radix 3, the input chain

$$1111111$$

must be converted into

$$1111111$$

(i.e., it is already an RN-coding), whereas the input chain

$$1111112$$

must be converted into

$$1\overline{1111111}.$$

This easily generalizes to an input chain of consecutive ones of arbitrary length: so if we are inside an input chain of ones, the information needed to perform the conversion can be arbitrarily far away.

Conversion from an RN-coding to conventional representation is done as usual.

D. Even radices: General case

The radix 10 algorithm easily generalizes to other even radices. Consider an input value

$$x = d_n d_{n-1} d_{n-2} \cdots d_j$$

with $0 \le d_i \le \beta - 1$, and define $d_{j-1} = 0$.

Define variables c_k as

$$c_{k+1} = \begin{cases} 1 & \text{if} \quad d_k \ge \beta/2 \\ 0 & \text{if} \quad d_k < \beta/2 \end{cases} \tag{4}$$

With $c_j = 0$. The digits δ_k of an RN-coding can be obtained using

$$\delta_k = d_k + c_k - \beta c_{k+1}.$$

It is worth being noticed that in radix 2, $c_{k+1} = d_k$, so that we find (1) again.

IV. Some Applications

A. Avoiding "double rounding"

Assume we use radix 10 arithmetic, with the conventional digit set. Consider the number

$$\gamma = 0.1234999000123\cdots$$

Assume that γ is computed and stored in a 6 decimal place format, with rounding to nearest. What is stored is

$$\gamma_1 = 0.123500$$

Now, assume we wish to re-use this value in a 3 decimal place format. We will convert γ_1 to that format, and we might get

$$\gamma_2 = 0.124,$$

which is not equal to γ rounded to the nearest 3-digit number. This phenomenon is called *double rounding.* For instance, it can occur (rarely) when a calculation is performed in the internal double-extended format of an Intel processor, and when the result is then converted to double precision. More generally, assuming the radix β is chosen, define

$$\circ_n(x)$$

as x rounded to the nearest n-digit radix β number. If $n_1 > n_2$, then sometimes

$$\circ_{n_2}(x) \neq \circ_{n_2}(\circ_{n_1}(x)).$$

A way to prevent this problem is to store the values RN-coded. In that case, rounding to nearest is equivalent to truncating, and we always have $\circ_{n_2}(x) = \circ_{n_2}(\circ_{n_1}(x))$.

B. Addition and multiplication of binary RN-coded operands

Although a binary RN-coded number has the same type of representation as a signed-digit redundant number, both using the digit set $\{-1, 0, 1\}$, it is a non-redundant representation. This implies that addition must take at least logarithmic time. Employing the encoding of digits as indicated in the conversion from 2's complement to RN-coding, a number X is represented as a pair of bit-vectors (X^+, X^-), where zero has two encodings, $(0, 0)$ and $(1, 1)$.

Preliminary investigations in [BM05] indicate that it is advantageous to restrict the digit encoding such that zero only has the $(0, 0)$ encoding. This simplifies addition by allowing an XOR gate to be substituted by an OR gate in the first-level addition, and similarly simplifies the multiplicand input to a multiplier. A complete multiplier is shown capable of forming products of RN-coded, as well as 2's complement operands, at very small overhead for RN-coded, compared to the 2's complement operands.

CONCLUSION AND FUTURE WORK

We have here shown some basic properties of RN codings, in particular on conversions to and from more standard radix-based number representations. But only preliminary results on the implementation on some of the basic arithmetic operations have so far been obtained, beyond the main advantage being the simple rounding by truncation.

REFERENCES

[Avi61] A. Avizienis. Signed-digit number representations for fast parallel arithmetic. *IRE Transactions on Electronic Computers*, EC-10:389–400, September 1961. Reprinted in [Swa90].

[Boo51] A.D. Booth. A Signed Binary Multiplication Technique. *Q. J. Mech. Appl. Math.*, 4:236–240, 1951. Reprinted in [Swa80].

[BM05] J.-L. Beuchat and J.-M. Muller. Multiplication Algorithms for Radix-2 RN-Codings and Two's Complement Numbers. Technical report, INRIA, February 2005. Available at: http://www.inria.fr/rrrt/rr-5511.html.

[Cau40] A. Cauchy. Sur les moyens d'éviter les erreurs dans les calculs numériques. *Comptes Rendus de l'Académie des Sciences, Paris*, 11:789–798, 1840. Republished in: Augustin Cauchy, oeuvres complètes, 1ère série, Tome V, pp 431-442, available at http://gallica.bnf.fr/scripts/ConsultationTout.exe?O=N090185.

[EL94] M.D. Ercegovac and T. Lang. *Division and Square Root: Digit-Recurrence Algorithms and Implementations*. Kluwer Academic Publishers, 1994.

[Erc84] M.D. Ercegovac. On-Line Arithmetic: An Overview. *Real Time Signal Processing VII*, SPIE-495:86–93, 1984.

[Knu98] D. E. Knuth. *Seminumerical Algorithms*, volume 2 of *The Art of Computer Programming*. Addison Wesley, 3 edition, 1998. 1. edition 1969 and 2. edition 1981.

[PB04] B. Phillips and N. Burgess. Minimal Weight Digit Set Conversions. *IEEE Transactions on Computers*, 53(6):666–677, June 2004.

[Swa80] Earl E. Swartzlander, editor. *Computer Arithmetic, Vol I*. Dowden, Hutchinson and Ross, Inc., 1980. Reprinted by IEEE Computer Society Press, 1990.

[Swa90] Earl E. Swartzlander, editor. *Computer Arithmetic, Vol II*. IEEE Computer Society Press, 1990.

[TE77] K.S. Trivedi and M.D. Ercegovac. On-line Algorithms for Division and Multiplication. *IEEE Transactions on Computers*, C-26(7):681–687, July 1977. Reprinted in [Swa90].