RN-coding of Numbers: New Insights and Some Applications

Peter Kornerup

Dept. of Mathematics and Computer Science

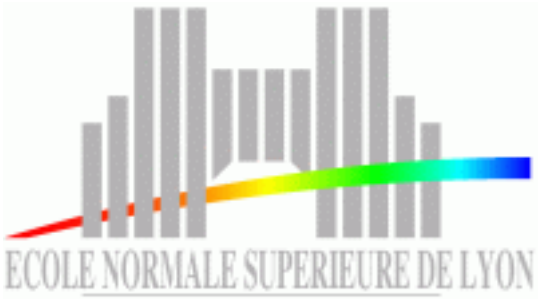
SDU, Odense, Denmark

&

Jean-Michel Muller

LIP/Arénaire (CRNS-ENS Lyon-INRIA-UCBL)

Lyon, France



RN-coding $\sim$ "Round-to-Nearest"

Certain radix representations have the property that truncation yields "round to nearest", e.g.,

Radix 3 with digit-set $\{-1, 0, 1\}$

- A computer based on this ternary system was actually built at Moscow University in the sixties.
- In 1840 Cauchy suggested the use of a decimal system with digit-set $\{-5, -4, -3, -2, -1, 0, 1, 2, 3, 4, 5\}$ to simplify arithmetic operations. With a simple restriction it has the "Round-to-Nearest" property of truncation.
- Binary Booth recoded numbers also have this property.

Definition of RN-Coding

Definition 1 *Let the radix $\beta \geq 2$ be an integer. The digit string*

$$D = d_n d_{n-1} \cdots d_0 . d_{-1} d_{-2} \cdots \quad \textit{with} \quad -\beta + 1 \leq d_i \leq \beta - 1$$

is an RN-coding of x iff

- *D is a radix-β representation of $x = \sum_{i=-\infty}^{n} d_i \beta^i$;*
- *For any $j \leq n$,*

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

i.e., if the digit sequence is truncated at any position, then the resulting sequence is always the number of the form $d_n d_{n-1} \cdots d_j$ which is closest to x.

We shall always assume that the digit-set is symmetric, $\{-a, \cdots, a\}$, $a \leq \beta - 1$.

Characterization of RN-codings

Theorem 2

- *If β is odd, then the digit-string $D = d_n d_{n-1} \cdots d_0 . d_{-1} d_{-2} \cdots$ is an RN-coding if and only if*

$$\forall i,\ \frac{-\beta+1}{2} \leq d_i \leq \frac{\beta-1}{2};$$

- *If β is even, then the digit-string $D = d_n d_{n-1} \cdots d_0 . d_{-1} d_{-2} \cdots$ is an RN-coding if and only if*
  - *all digits satisfy $|d_i| \leq \beta/2$;*
  - *if $|d_i| = \beta/2$, then the first non-zero digit following has the opposite sign.*
- *If $\beta = 2$, then the digit-string $D = d_n d_{n-1} \cdots d_0 . d_{-1} d_{-2} \cdots$, with $d_i \in \{-1, 0, 1\}$ is an RN-coding if and only if the non-zero digits have alternating signs.*

Uniqueness of Finite Representations

Theorem 3

- *If β is odd then a finite RN-coding is unique (the number system is non-redundant)*
- *If β is even then some numbers have two finite representations. In that case, one has its rightmost non-zero digit equal to $-\frac{\beta}{2}$, the other one has its rightmost non-zero digit equal to $+\frac{\beta}{2}$.*

Note: When β is even, if at truncation there is a tie, then this is resolved to round either way depending on the actual representation, hence the rounding by truncation is unbiased.

In particular, for $\beta = 2$, Booth Encoding has its non-zero digits from $\{-1, 1\}$ of alternating sign, hence the last non-zero digit may have either sign, e.g.:

$$\begin{array}{ll} 01 \rightarrow 1\bar{1} & \bar{1}1 \rightarrow 0\bar{1} \\ \text{and} & \\ 0\bar{1} \rightarrow \bar{1}1 & 1\bar{1} \rightarrow 01 \end{array}$$

The Problem of "Double Rounding"

If the value $\gamma = 0.123499900123\cdots$ has been calculated and stored in a 6-decimal place format with rounding to nearest, what is stored is $\gamma_1 = 0.123500$.

Assume that the value is to be re-used in a 3-decimal format, then from γ_1 we might get $\gamma_2 = 0.124$, which is not γ rounded into the nearest 3-decimal format.

Observation 4 *Let* $\mathrm{rn}_i(x)$ *be the function that rounds the value of* x *to nearest at position* i*. If* x *is represented in RN-coding, then for* $k > j$*:*

$$\mathrm{rn}_k(x) = \mathrm{rn}_k(\mathrm{rn}_j(x)).$$

Thus with RN-coding, problems with "double rounding" are avoided.

Parallel Calculations

Definition 5 *A function that returns an output digit string $\delta_{n+k}\delta_{n+k-1}\cdots\delta_j$ from an input string $d_n d_{n-1}\cdots d_j$ (where j can be $-\infty$) is SW-computable 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.

Note: SW-computable functions are of interest for digit-set conversion algorithms, as they allow such to be performed "in parallel" on all digits of a digit string.

From now on we will only deal with the radix-2 RN-representation employing the signed-digit (borrow-save) digit set $\{-1, 0, 1\}$.

Conversion from 2's Complement to RN-coding

Consider a 2's complement input value:

$$x = b_m b_{m-1} \cdots b_\ell = -b_m 2^m + \sum_{i=\ell}^{m-1} b_i 2^i$$

with $b_i \in \{0, 1\}$ and $\ell < m$, Then the digit string

$$\delta_m \delta_{m-1} \cdots \delta_\ell,$$

defined by $\delta_k = b_{k-1} - b_k \in \{-1, 0, 1\}$, is an RN-coding of x (by convention for finite strings $b_{j-1} = 0$). Note that it is SW-computable.

This is the Booth recoding, employing the digit encoding:

$$\begin{array}{rcl} -1 & \sim & (0,1) \\ 0 & \sim & (0,0) \text{ or } (1,1) \\ 1 & \sim & (1,0) \end{array}$$

The value of the digit is the difference between the first and second component.

Example

Let $x = 1$10100110010 be a sign-extended 2's complement number and write the digits of $2x$ above the digits of x:

$2x$	1	0	1	0	0	1	1	0	0	1	0	0
x	1	1	0	1	0	0	1	1	0	0	1	0
x in RN-Coding		$\bar{1}$	1	$\bar{1}$	0	1	0	$\bar{1}$	0	1	$\bar{1}$	0

where it is seen that in any column the two upper-most bits provide the encoding of the signed-digit below in the column, since $x = 2x - x$.

Since the digit in position $m+1$ will always be 0, there is no need to include the most significant position in the encoding.

Rounding Example

Rounding the value of x in the previous example by truncating off the two least significant digits we obtain

$\mathrm{rn}_2(2x)$	1	0	1	0	0	1	1	0	0	1
$\mathrm{rn}_2(x)$	1	1	0	1	0	0	1	1	0	0
$\mathrm{rn}_2(x)$ in RN-Coding		$\bar{1}$	1	$\bar{1}$	0	1	0	$\bar{1}$	0	1

where it is noted that the bit of value 1 in the upper rightmost corner acts as a round bit, assuring a round-up in cases there is a tie-situation as here.

Canonical Encoding

Definition 6 *Let the number x be given in 2's complement representation as the bit string $b_m \cdots b_{\ell+1} b_\ell$, such that $x = -b_m 2^m + \sum_{i=\ell}^{m-1} b_i 2^i$. Then the* canonical encoding *of the RN-Coded representation of x is defined as the pair*

$$x \sim (b_m b_{m-1} \cdots b_{\ell+1} b_\ell, r) \quad \text{where the round-bit is } r = 0$$

and after truncation at position k, for $m \geq k > \ell$

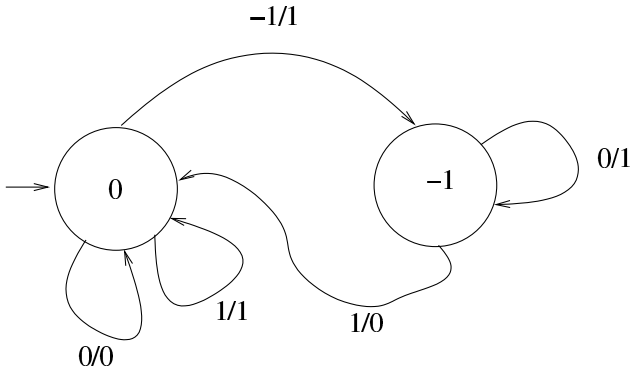
$$\mathrm{rn}_k(x) \sim (b_m b_{m-1} \cdots b_{k+1} b_k, r) \quad \text{with round-bit } r = b_{k-1}.$$

Note: This is a kind of "carry-save" in the sense that it contains a bit not yet added in. The same idea have previously been pursued by Matula et.al. in a floating-point setting, denoted "packet-forwarding".

Conversion from RN-coding to 2's Complement

Conversion this way is not SW-computable, e.g., the RN-coding 10000000 represents the binary number 10000000_2, whereas the RN-coding $1000000\bar{1}$ represents the binary number 0111111_2.

However, it is computable in a right-to-left conversion by a right-to-left transducer:



But obviously this is a linear-time algorithm.

Log-Time Conversion from Canonical Encoding to 2's Complement

If an RN-Coded number is in canonical encoding, conversion into 2's complement may require a non-zero round-bit to be added in. This is simply an incrementation for which efficient carry-propagate trees exist:

Let the operand in canonical encoding be

$$x \sim (x_m, x_{m-1} \cdots x_\ell, r),$$

then the propagate- and generate-bits for input to the trees are simply

$$p_i = x_i \quad \text{and} \quad g_i = 0 \quad \text{for} \quad i = \ell, \ell+1, \cdots, m.$$

By the composition rule $(g, p) \circ (g', p') = (g + pg', pp') = (0, pp')$, so all the generate-bits will be zero, and need not be calculated.

Building the Tree

The nodes of the parallel prefix tree for calculating the carries will consist of AND-operators, with input at the leaves $p_i = x_i$. Let the output of the i^{th} tree then be the combined propagate-bits

$$P_i = \bigwedge_{k=\ell}^{i} p_k \quad \text{for} \quad i = \ell, \ell+1, \cdots, m.$$

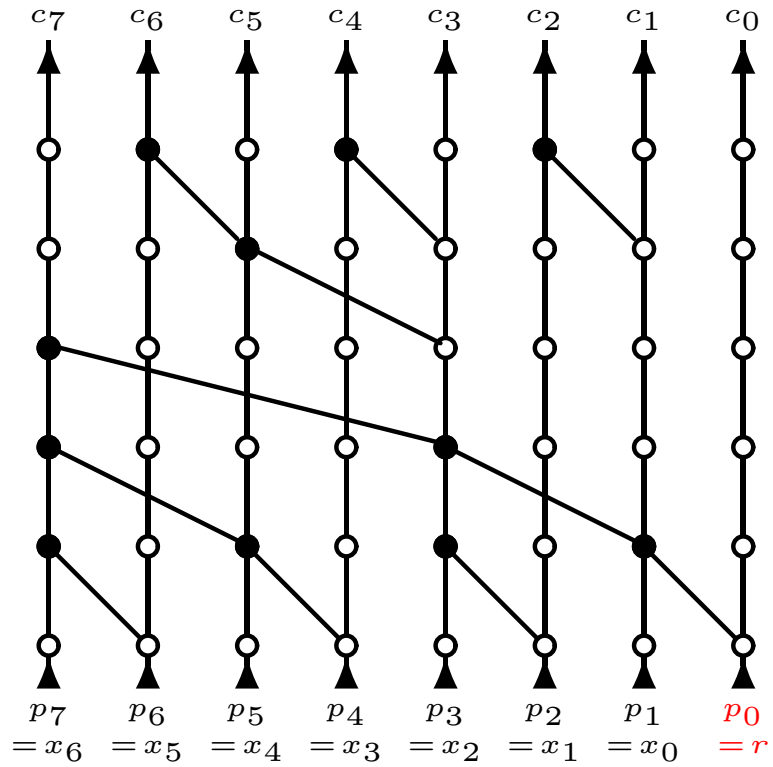
The actual carries can then be found as $c_\ell = r$ and $c_i = r\, P_{i-1}$ for $i > \ell$. However, this will require a broadcast of the round-bit to all positions.

This can be avoided by changing the input p_ℓ to $p_\ell = r$ and shifting the rest over so that they are defined as $p_i = x_{i-1}$ for $i > \ell$. The final output of the conversion is then found as

$$e_i = x_i \oplus c_i \quad \text{for} \quad i = \ell, \ell+1, \cdots, m.$$

A Parallel Prefix Tree

Choosing a Brent-Kung tree with AND-operators in the black nodes:



The final output of the conversion is then found as

$$e_i = x_i \oplus c_i \quad \text{for} \quad i = \ell, \ell+1, \cdots, m.$$

Some Standard Applications

Operands are assumed to be in canonical representation:

Addition: For two operands to a carry-completing adder, one round-bit may be used as carry-in, and the other as round-bit of the result.

Subtraction: The subtrahend is delivered inverted to the adder, with the inverted round-bit as carry-in.

Multi-Operand Addition: Redundant adders (like carry-save) apply with round-bits for each operand supplied as carry-in, the first being saved as final round-bit for result.

Multiplication: Any RN-coded multiplier can directly be recoded into higher radix like 4 or 8 simply by grouping digits. A round-bit of the multiplicand can be used as carry-in to the adders of the multiplier array or tree.

Applications in Signal Processing

Although RN-Coding applies equally well to floating-point representations incorporating non-absorbed round-bits, as also suggested by Matula et.al., let us here think of use in fixed-point digital signal processing applications.

Inner Products: Although double-precision accumulation of product terms can be performed exactly, for single precision accumulation it is essential that a fast and optimal rounding is available.

Polynomial Evaluation: For the Horner Scheme, let $f(x) = \sum_{i=0}^{n} a_i x^i$ be some polynomial approximation, then $f(x)$ is efficiently evaluated as

$$f(x) = (\cdots((a_n) * x + a_{n-1}) * x \cdots + a_1) * x + a_0,$$

where to avoid a growth in operand lengths, roundings are needed in each cycle of the algorithm.

An Extension to CORDIC Algorithms

Along the same ideas, although not based on RN-representations, we wish to decompose a number t as a sum

$$t = \sum_{i=0}^{\infty} b_i \arctan 2^{-i} \quad \text{for } b_i \in \{-1, 0, 1\}$$

(i.e., as the conventional CORDIC with double rotation – to handle the zeros), but with the additional constraint that:

- $b_n = -1 \Rightarrow$ the smallest $k > n$ such that $b_k \neq 0$ satisfies $b_k = +1$;
- $b_n = +1 \Rightarrow$ the smallest $k > n$ such that $b_k \neq 0$ satisfies $b_k = -1$.

The Idea

Let us denote

$$t_n = \sum_{i=0}^{n-1} b_i \arctan 2^{-i}.$$

Thanks to the additional constraint, the error bound when we approximate t by t_n, is now $\arctan 2^{-n}$, instead of the usual (approximately twice as large) bound $\sum_{k=n}^{\infty} \arctan 2^{-k}$.

This is the equivalent in the "base $\arctan 2^{-i}$", of the radix-2 (i.e., "base 2^{-i}") RN-coding.

To perform rotations, we can store the angle decomposed in that "base", this will lead to more accurate rotations, provided we can easily decompose.

Generating the Decomposition

We assume that

$$|t| \leq \arctan 2^0 = \pi/4,$$

and that we have computed $b_0, b_1, \ldots, b_{n-1}$, and $t_n = \sum_{i=0}^{n-1} b_i \arctan 2^{-i}$.

To find the next non-zero "digit", b_k, where $b_n = b_{n+1} = \cdots = b_{k-1} = 0$ for $n < k$, we have two cases:

First case: $t_n \leq t$. We can then prove that $t - t_n \leq \arctan 2^{-n}$.

Let k be the largest integer $\geq n$ such that $t_n + \arctan 2^{-k} \geq t$ and choose $b_k = 1$, implying that the next non-zero "digit" b_j, if any, will be -1.

Second case: $t_n \geq t$. Here $t_n - t \leq \arctan 2^{-n}$.

Let k be the largest integer $\geq n$ such that $t_n - \arctan 2^{-k} \leq t$ and choose $b_k = -1$, implying that the next non-zero "digit" b_j, if any, will be $+1$.

Conclusions

Concentrating on binary RN-Coded operands, with the feature of rounding by truncation, we have shown how a simple encoding, based on the ordinary 2's complement representation, allows:

- A trivial conversion from 2's complement representation to RN-Coding.
- A fast parallel prefix algorithm for conversion the other way.
- Operands in the canonical RN-Coding can be used at hardly any penalty in many standard calculations.
- Allowing rounding by a simple truncation.

The idea of exploiting coefficients of alternating signs may also be utilized in CORDIC algorithms, an idea to be exploited.

References

[BK82] R.P. Brent and H.T. Kung. A Regular Layout for Parallel Adders. *IEEE Transactions on Computers*, C-31(3):260–264, March 1982.

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

[NMLE00] A. Munk Nielsen, D.W. Matula, C.N. Lyu, and G. Even. An IEEE Compliant Floating-Point Adder that Conforms with the Pipelined Packet-Forwarding Paradigm. *IEEE Transactions on Computers*, 49(1):33–47, January 2000.

[Vol59] J.E. Volder. The CORDIC Trigonometric Computing Technique. *IRE Transactions on Electronic Computers*, EC-8:330–334, 1959.