HAL
open science

Multiplication Algorithms for Radix-2 RN-Codings and Two's Complement Numbers

Jean-Luc Beuchat, Jean-Michel Muller

▶ **To cite this version:**

Jean-Luc Beuchat, Jean-Michel Muller. Multiplication Algorithms for Radix-2 RN-Codings and Two's Complement Numbers. [Research Report] RR-5511, LIP RR-2005-05, INRIA, LIP. 2005, pp.13. inria-00070495

HAL Id: inria-00070495

https://inria.hal.science/inria-00070495

Submitted on 19 May 2006

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Multiplication Algorithms for Radix-2 RN-Codings and Two's Complement Numbers

Jean-Luc Beuchat and Jean-Michel Muller

N° 5511

February 2005

Thème SYM

Rapport de recherche

ISSN 0249-6399 ISRN INRIA/RR--5511--FR+ENG

Multiplication Algorithms for Radix-2 RN-Codings and Two's Complement Numbers

Jean-Luc Beuchat and Jean-Michel Muller

Thème SYM — Systèmes symboliques
Projet Arénaire

Rapport de recherche n° 5511 — February 2005 — 13 pages

Abstract: The RN-codings are particular cases of signed-digit representations, for which rounding to the nearest is always identical to truncation. In radix 2, Booth recoding is an RN-coding. In this paper, we suggest several multiplication algorithms able to handle RN-codings, and we analyze their properties.

Key-words: Computer arithmetic, RN-codings, (modified) Booth recoding, rounding to the nearest

This text is also available as a research report of the Laboratoire de l'Informatique du Parallélisme `http://www.ens-lyon.fr/LIP`.

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe, 38334 Montbonnot Saint Ismier (France)
Téléphone : +33 4 76 61 52 00 — Télécopie +33 4 76 61 52 52

Algorithmes de multiplication pour RN-codes en base deux et nombres en complément à deux

Résumé : Les RN-codes sont des représentations à chiffres signés particulières pour lesquelles arrondir au plus près est toujours équivalent à tronquer. En base deux, le recodage de Booth est un RN-code. Dans cet article, nous présentons plusieurs algorithmes de multiplication destinés aux RN-codes et discutons leurs propriétés.

Mots-clés : Arithmétique des ordinateurs, RN-codes, recodage de Booth (modifié), arrondi au plus près

1 Introduction

Kornerup and Muller recently introduced the notion of "RN-coding" [6]. An RN-coding is a radix-β signed-digit representation for which truncating is always equivalent to rounding to the nearest. For example, in radix 10, the RN-coding of π starts with

$$3.142\bar{4}\bar{1}33\bar{5}\bar{5}44\bar{1}0\bar{2}\bar{1}3\ldots$$

where (as usually) $\bar{4}$ means -4. An interesting property of these recodings is that when we use them, there is no phenomenon of "double rounding". In this paper, we wish to present some arithmetic operators for these RN codings. More precisely, our goal is to add the smallest amount of hardware to a standard arithmetic and logic unit (ALU) so that it can efficiently handle both two's complement numbers and RN-codings. Before going further, let us give some definitions.

Definition 1. Let X be a radix-β number. We define $\circ_n(X)$ as X rounded to the nearest n-digit radix-β number.

Example 1. *Consider the radix-10 number $X = 12.2934524$. We have*

$$\circ_8(X) = 12.293452, \quad \circ_6(X) = 12.2935,$$
$$\circ_4(X) = 12.29, \qquad \circ_3(X) = 12.3.$$

Definition 2 (RN-codings [6]). 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 $-\beta+1 \le d_i \le \beta-1$) is a radix-$\beta$ RN-coding of X if $X = \sum_{i=-\infty}^{n} d_i\beta^i$ and $\left|\sum_{i=-\infty}^{j-1} d_i\beta^i\right| \le \frac{1}{2}\beta^j$ for any j.

Theorem 1 (Characterizations of RN-codings [6]). *Let β be an integer greater than or equal to 2. If β is odd, then $D = d_n d_{n-1} d_{n-2} d_{n-3} \ldots d_0.d_{-1}d_{-2}\ldots$ is an RN-coding if and only if all digits have absolute value less than or equal to $(\beta-1)/2$.*
If β is even then $D = d_n d_{n-1} d_{n-2} d_{n-3} \ldots d_0.d_{-1}d_{-2}\ldots$ 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 $d_i d_j < 0$.*

Example 2. *The well-known Booth recoding [3] is a radix-2 RN-coding. The number is written on the digit set $\{\bar{1}, 0, 1\}$ and the rightmost non-zero digit is $\bar{1}$. Each number has therefore a single finite representation [6]. Algorithm 1 summarizes the conversion of a two's complement number, where each output digit is deduced from a constant-sized sliding window of input digits.*

Consider the $(n+1)$-bit two's complement number $X = 2^{n-1} = (010\ldots0)_2$. Algorithm 1 returns the $(n+1)$-digit number $Y = 2^n - 2^{n-1} = (1\bar{1}0\ldots0)_2$, whereas the n-digit number $Z = 2^{n-1} = (10\ldots0)_2$ is also an RN-coding of X. If the $(n+1)$-bit numbers we have to convert belong to the set $\{-2^{n-1}, \ldots, 2^{n-1}\}$, it is therefore preferable to apply Algorithm 2, which guarantees that the result is an n-digit RN-coding [2].

Algorithm 1 Booth recoding of a two's complement number.

Require: An $(n+1)$-bit two's complement number $X \in \{-2^n, \ldots, 2^n - 1\}$ with $x_{-1} = 0$
Ensure: An $(n+1)$-digit radix-2 RN-coding Y such that $X = Y$

```
for i = 1 to n do
    y_i ← x_{i-1} − x_i;
end for
```

Algorithm 2 Two's complement to radix-2 RN-coding conversion.

Require: An $(n+1)$-bit two's complement number $X \in \{-2^{n-1}, \ldots, 2^{n-1}\}$ with $x_{-1} = 0$
Ensure: An n-digit radix-2 RN-coding Y such that $X = Y$
1: **for** $i = 1$ to $n-2$ **do**
2: $\quad y_i \leftarrow x_{i-1} - x_i$;
3: **end for**
4: $y^+_{n-1} \leftarrow \bar{x}_n(x_{n-1} \vee x_{n-2})$; $y^-_{n-1} \leftarrow x_n x_{n-1} \bar{x}_{n-2}$;

In the following, we will represent a radix-2 RN-coding X with a modified borrow-save encoding [1]. We define two bit-strings X^+ and X^- such that

$$\begin{cases} X = X^+ - X^-, \\ \text{if } x_i^{\pm} = 1 \text{ then } x_i^{\mp} = 0. \end{cases} \tag{1}$$

This paper devoted to the multiplication of radix-2 RN-codings is organized as follows: Section 2 describes an improvement of a multiplication algorithm we have introduced previously. Its major drawback is that it does not share many common resources with a conventional multiplier. In Section 3, we show how to very slightly adapt a conventional multiplier so that it can also handle RN-codings.

2 Improvement of the Multiplication Algorithm Proposed in [2]

Our first multiplication algorithm, published in [2], consists in computing $XY = X^+Y^+ + X^-Y^- - X^+Y^- - X^-Y^+$ in two's complement and in converting the result to radix-2 RN-coding. According to Definition 2 and Equation (1), neither X^+ nor X^- contain a string of ones whose length is greater than one (i.e. if $x_i^{\pm} = 1$, then $x_{i+1}^{\pm} = x_{i-1}^{\pm} = 0$). This property reduces the number of partial products involved in the computation of X^+Y^+, X^-Y^-, X^+Y^-, and X^-Y^+ by half at the price of OR gates (Figure 1a).

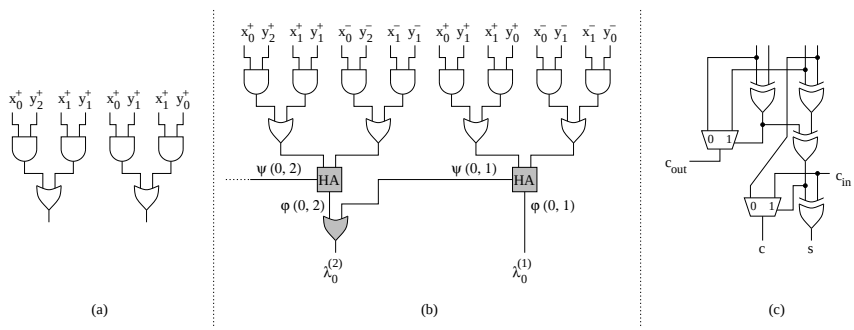

Figure 1: (a) Partial product generation for X^+Y^+. (b) Partial product generation for $(X^+Y^+ + X^-Y^-)$. (c) $(4,2)$-compressor (reprinted from [7]).

Example 3. *Assume that X and Y are two n-digit radix-2 RN-codings. The product X^+Y^+ can for instance be performed by adding $\lceil n/2 \rceil$ partial products:*

$$X^+Y^+ = \sum_{i=0}^{n-1} 2^i x_i^+ Y^+ = \sum_{i=0}^{\lceil \frac{n}{2} \rceil - 1} 2^{2i}(2x_{2i+1}^+ + x_{2i}^+)Y^+ = \sum_{i=0}^{\lceil \frac{n}{2} \rceil - 1} 2^{2i}(2x_{2i+1}^+ Y^+ \vee x_{2i}^+ Y^+),$$

where $x_n^+ = 0$.

Figure 2a depicts the architecture of the multiplier proposed in [2]. Four partial product generators taking advantage of the above-mentioned property and four carry-save adder trees compute in parallel X^+Y^+, X^-Y^-, X^+Y^-, and X^-Y^+. Then, two adders based on $(4,2)$-compressors (Figure 1c) and a subtracter generate the carry-save form of the product $P = XY$. The last step consists in converting this intermediate result to radix-2 RN-coding. Since the maximal absolute value of an n-digit radix-2 RN-coding X is 2^{n-1} [2], the product XY belongs to $\{-2^{2n-2}, \ldots, 2^{2n-2}\}$. We use a fast adder to compute the two's complement representation of P and apply Algorithm 2 to generate a $(2n-1)$-digit radix-2 RN-coding.

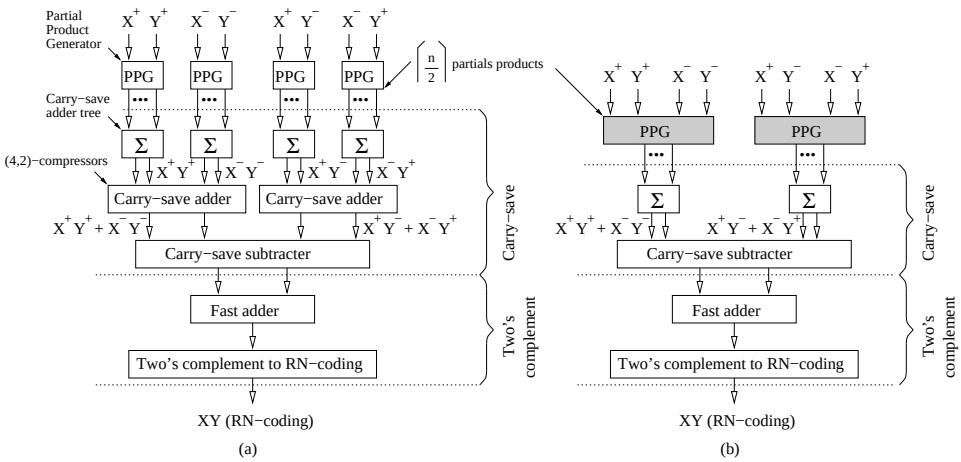

Figure 2: Multiplication of two RN-codings. (a) Architecture of the multiplier described in [2]. (b) Proposed improvement.

Consider now the addition of X^+Y^+ and X^-Y^- (Figure 3). The algorithm described in [2] computes $(x_0^+y_1^+ + x_1^+y_0^+) + (x_0^-y_1^- + x_1^-y_0^-) = (x_0^+y_1^+ \vee x_1^+y_0^+) + (x_0^-y_1^- \vee x_1^-y_0^-)$ with carry-save adders. This addition generates a carry iff $x_0^+ = y_1^+ = x_1^- = y_0^- = 1$ or $x_1^+ = y_0^+ = x_0^- = y_1^- = 1$. In both cases, the definition of the radix-2 RN-coding guarantees that $(x_0^+y_2^+ + x_1^+y_1^+) + (x_2^-y_0^- + x_1^-y_1^-) = 0$. Consequently, the jth partial products of X^+Y^+ and X^-Y^- can be added without carry propagation (Figure 1b). In the following, we respectively denote the carry bit and the sum bit of $(x_i^+y_j^+ + x_{i+1}^+y_{j-1}^+) + (x_i^-y_j^- + x_{i+1}^-y_{j-1}^-)$ by $\psi(i,j)$ and $\varphi(i,j)$. Applying this notation to our example, we obtain:

$$(x_0^+y_1^+ + x_1^+y_0^+) + (x_0^-y_1^- + x_1^-y_0^-) = 2\psi(0,1) + \varphi(0,1),$$
$$(x_0^+y_2^+ + x_1^+y_1^+) + (x_0^-y_2^- + x_1^-y_1^-) = 2\psi(0,2) + \varphi(0,2),$$

and $\psi(0,1) + \varphi(0,2) = \psi(0,1) \vee \varphi(0,2)$. The improvement proposed here is based on such properties: instead of adding X^+Y^+ to X^-Y^-, we combine their respective partial products in order to compute $(X^+Y^+ + X^-Y^-)$ with a single $\lceil n/2 \rceil$-operand carry-save adder. Theorem 2, whose proof is provided in Appendix A, and Algorithm 3 describe more precisely this partial product generation process. Compared to the scheme described in [2], our new approach only requires half-adder (HA) cells and OR gates, thus adding an XOR gate and an OR gate in the critical path.

Theorem 2. *Let X and Y be two radix-2 RN-codings. Consider two functions $\varphi(i,j)$ and $\psi(i,j)$ defined as follows:*

$$\varphi(i,j) = (x_i^+y_j^+ \vee x_{i+1}^+y_{j-1}^+) \oplus (x_i^-y_j^- \vee x_{i+1}^-y_{j-1}^-) \tag{2}$$

and

$$\begin{aligned}\psi(i,j) &= (x_i^+y_j^+ \vee x_{i+1}^+y_{j-1}^+) \cdot (x_i^-y_j^- \vee x_{i+1}^-y_{j-1}^-) \\ &= x_i^+y_j^+x_{i+1}^-y_{j-1}^- \vee x_i^-y_j^-x_{i+1}^+y_{j-1}^+,\end{aligned} \tag{3}$$

where $1 \leq j \leq n$ *and* $i + j < 2n - 2$. *Then,*

$$
\begin{aligned}
x_i^+ y_j^+ + x_i^- y_j^- &= x_i^+ y_j^+ \vee x_i^- y_j^-, && (4)\\
x_i^+ y_j^+ + x_i^- y_j^- + \psi(i-1,j) &= x_i^+ y_j^+ \vee x_i^- y_j^- \vee \psi(i-1,j), && (5)\\
x_i^+ y_j^+ + x_{i+1}^+ y_{j-1}^+ + x_i^- y_j^- + x_{i+1}^- y_{j-1}^- &= 2\psi(i,j) + \varphi(i,j) \in \{0,1,2\}, && (6)\\
\varphi(i,j) + \psi(i,j-1) &= \varphi(i,j) \vee \psi(i,j-1), && (7)\\
\varphi(i,j) + \psi(i-1,j) &= \varphi(i,j) \vee \psi(i-1,j). && (8)
\end{aligned}
$$

Example 4. *Let us consider two 6-digit RN-codings* X *and* Y. *We could compute the sum of the twelve partial products* $2^i x_i^+ Y^+$ *and* $2^i x_i^- Y^-$, $0 \leq i \leq 5$, *by means of a carry-save adder tree. However, Theorem 2 allows to perform the same task by adding only three terms (Figure 3). Let us denote the bit of weight* 2^i *of the* i*th partial product by* $\lambda_i^{(j)}$, *where* $0 \leq i \leq \lceil n/2 \rceil - 1$.

- *Equation (4) allows to replace an addition by an OR gate:* $\lambda_0^{(0)} = x_0^+ y_0^+ \vee x_0^- y_0^-$.
- *Since the computation of* $\lambda_0^{(0)}$ *does not generate a carry, we have* $\lambda_0^{(1)} = (x_0^+ y_1^+ + x_1^+ y_0^+ + x_0^- y_1^- + x_1^- y_0^-) \mod 2 = \varphi(0,1)$.
- *We respectively apply Equations (7) and (4) to compute* $\lambda_0^{(2)} = \varphi(0,2) \vee \psi(0,1)$ *and* $\lambda_1^{(2)} = x_2^+ y_0^+ \vee x_2^- y_0^-$.

					$x_0^+ y_5^+$ $x_0^- y_5^-$	$x_0^+ y_4^+$ $x_0^- y_4^-$	$x_0^+ y_3^+$ $x_0^- y_3^-$	$x_0^+ y_2^+$ $x_0^- y_2^-$	$x_0^+ y_1^+$ $x_0^- y_1^-$	$x_0^+ y_0^+$ $x_0^- y_0^-$
				$x_1^+ y_5^+$ $x_1^- y_5^-$	$x_1^+ y_4^+$ $x_1^- y_4^-$	$x_1^+ y_3^+$ $x_1^- y_3^-$	$x_1^+ y_2^+$ $x_1^- y_2^-$	$x_1^+ y_1^+$ $x_1^- y_1^-$	$x_1^+ y_0^+$ $x_1^- y_0^-$	
			$x_2^+ y_5^+$ $x_2^- y_5^-$	$x_2^+ y_4^+$ $x_2^- y_4^-$	$x_2^+ y_3^+$ $x_2^- y_3^-$	$x_2^+ y_2^+$ $x_2^- y_2^-$	$x_2^+ y_1^+$ $x_2^- y_1^-$	$x_2^+ y_0^+$ $x_2^- y_0^-$		
		$x_3^+ y_5^+$ $x_3^- y_5^-$	$x_3^+ y_4^+$ $x_3^- y_4^-$	$x_3^+ y_3^+$ $x_3^- y_3^-$	$x_3^+ y_2^+$ $x_3^- y_2^-$	$x_3^+ y_1^+$ $x_3^- y_1^-$	$x_3^+ y_0^+$ $x_3^- y_0^-$			
	$x_4^+ y_5^+$ $x_4^- y_5^-$	$x_4^+ y_4^+$ $x_4^- y_4^-$	$x_4^+ y_3^+$ $x_4^- y_3^-$	$x_4^+ y_2^+$ $x_4^- y_2^-$	$x_4^+ y_1^+$ $x_4^- y_1^-$	$x_4^+ y_0^+$ $x_4^- y_0^-$				
$x_5^+ y_5^+$ $x_5^- y_5^-$	$x_5^+ y_4^+$ $x_5^- y_4^-$	$x_5^+ y_3^+$ $x_5^- y_3^-$	$x_5^+ y_2^+$ $x_5^- y_2^-$	$x_5^+ y_1^+$ $x_5^- y_1^-$	$x_5^+ y_0^+$ $x_5^- y_0^-$					
$\psi(4,5)$	$\varphi(4,5)$ $\psi(3,5)$	$\varphi(3,5)$ $\psi(2,5)$	$\varphi(2,5)$ $\psi(1,5)$	$\varphi(1,5)$ $\psi(0,5)$	$\varphi(0,5)$ $\psi(0,4)$	$\varphi(0,4)$ $\psi(0,3)$	$\varphi(0,3)$ $\psi(0,2)$	$\varphi(0,2)$ $\psi(0,1)$	$\varphi(0,1)$	
			$\varphi(4,3)$	$\varphi(3,3)$	$\varphi(2,3)$	$\varphi(2,2)$	$\varphi(2,1)$			
		$\psi(4,3)$	$\psi(3,3)$	$\psi(2,3)$	$\psi(2,2)$	$\psi(2,1)$				
					$\varphi(4,1)$					
				$\psi(4,1)$						
$x_5^+ y_5^+$ $x_5^+ y_5^-$		$x_5^+ y_3^+$ $x_5^+ y_3^-$		$x_5^+ y_1^+$ $x_5^+ y_1^-$		$x_4^+ y_0^+$ $x_4^+ y_0^-$		$x_2^+ y_0^+$ $x_2^+ y_0^-$		$x_0^+ y_0^+$ $x_0^+ y_0^-$
$\lambda_0^{(10)}$	$\lambda_0^{(9)}$	$\lambda_0^{(8)}$ $\lambda_1^{(8)}$	$\lambda_0^{(7)}$ $\lambda_0^{(7)}$	$\lambda_0^{(6)}$ $\lambda_1^{(6)}$ $\lambda_2^{(6)}$	$\lambda_0^{(5)}$ $\lambda_1^{(5)}$ $\lambda_2^{(5)}$	$\lambda_0^{(4)}$ $\lambda_1^{(4)}$ $\lambda_2^{(4)}$	$\lambda_0^{(3)}$ $\lambda_1^{(3)}$	$\lambda_0^{(2)}$ $\lambda_1^{(2)}$	$\lambda_0^{(1)}$	$\lambda_0^{(0)}$

Figure 3: Computation of $X^+Y^+ + X^-Y^-$ when $n = 6$.

Figure 2b shows the general architecture of the multiplier. At the price of a more complex partial product generation, we save two carry-save adder trees and two carry-save adders based on $(4,2)$-compressors. It is worth being noticed that the critical path of our new partial product generator (two OR gates, one AND gate, and an XOR gate) is shorter than the one of a $(4,2)$-compressor (three XOR gates) (Figures 1b and 1c). Let $(U^{(c)}, U^{(s)})$ denote the carry-save form of $X^+Y^+ + X^-Y^-$. We have $X^+Y^+ + X^-Y^- = 2U^{(c)} + U^{(s)}$, where $u_i^{(c)} = 0$ and $u_i^{(s)} = \lambda_0^{(i)}$ for $i \in \{0, 1, 2n-3, 2n-2\}$ (Figure 4a). Algorithm 3 can also be applied to the computation of

Algorithm 3 Partial product generation for the computation of $X^+Y^+ + X^-Y^-$.

Require: Two n-digit RN-codings X and Y; two functions $\varphi(i,j)$ and $\psi(i,j)$ respectively defined by Equation (2) and Equation (3)

Ensure: $(2n-1)$ vectors $\Lambda^{(j)} = \lambda_{k-1}^{(j)} \dots \lambda_0^{(j)}$, where $1 \le k \le \lceil n/2 \rceil$

1: $\lambda_0^{(0)} \leftarrow x_0^+ y_0^+ \vee x_0^- y_0^-$; $\lambda_0^{(1)} \leftarrow \varphi(0,1)$; $\lambda_0^{(2n-2)} \leftarrow x_{n-1}^+ y_{n-1}^+ \vee x_{n-1}^- y_{n-1}^- \vee \psi(n-2, n-1)$;
2: **for** $i = 1$ to $\lfloor \frac{n}{2} - 1 \rfloor$ **do**
3: **for** $j = 0$ to $i-1$ **do**
4: $\lambda_j^{(2i+1)} \leftarrow \varphi(2j, 2i+1-2j) \vee \psi(2j, 2i-2j)$;
5: **end for**
6: $\lambda_i^{(2i+1)} \leftarrow \varphi(2i, 1)$;
7: **end for**
8: **for** $i = 1$ to $\lceil \frac{n}{2} - 1 \rceil$ **do**
9: **for** $j = 0$ to $i-1$ **do**
10: $\lambda_j^{(2i)} \leftarrow \varphi(2j, 2i-2j) \vee \psi(2j, 2i-2j-1)$;
11: **end for**
12: $\lambda_i^{(2i)} \leftarrow x_{2i}^+ y_0^+ \vee x_{2i}^- y_0^-$;
13: **end for**
14: **for** $i = \lfloor \frac{n+1}{2} \rfloor$ to $n-2$ **do**
15: **for** $j = 0$ to $n-i-2$ **do**
16: $\lambda_j^{(2i)} \leftarrow \varphi(2i+2j-n+1, n-2j-1) \vee \psi(2i+2j-n, n-2j-1)$;
17: **end for**
18: $\lambda_{n-i-1}^{(2i)} \leftarrow x_{n-1}^+ y_{2i-n+1}^+ \vee x_{n-1}^- y_{2i-n+1}^- \vee \psi(n-2, 2i-n+1)$;
19: **end for**
20: **for** $i = \lceil \frac{n-1}{2} \rceil$ to $n-2$ **do**
21: **for** $j = 0$ to $n-i-2$ **do**
22: $\lambda_j^{(2i+1)} \leftarrow \varphi(2i+2j-n+2, n-2j-1) \vee \psi(2i-2j-n+1, n-2j-1)$;
23: **end for**
24: **end for**

$(X^+Y^- + X^-Y^+)$. Thus, we obtain a carry-save number $(V^{(c)}, V^{(s)})$ such that $X^+Y^- + X^-Y^+ = 2V^{(c)} + V^{(s)}$. Note that the two's complement forms of $V^{(s)}$ and $V^{(c)}$ are respectively defined by:

$$2^{2n-1} - V^{(s)} = 1 + \sum_{i=0}^{2n-2} (1 - v_i^{(s)}) = 1 + \sum_{i=0}^{2n-2} \bar{v}_i^{(s)}$$

and

$$2^{2n-1} - V^{(c)} = 2^{2n-2} + 1 + \sum_{i=0}^{2} 2^i + \sum_{i=3}^{2n-3} (1 - v_{i-1}^{(c)}) = 2^{2n-2} + 8 + \sum_{i=3}^{2n-3} \bar{v}_{i-1}^{(c)}.$$

Thus, the carry-save form $P = (P^{(c)}, P^{(s)})$ of the product can be computed as follows:

$$\begin{aligned} 2P^{(c)} + P^{(s)} &= 2U^{(c)} + U^{(s)} - 2V^{(c)} - V^{(s)} \\ &= (u_{2n-2}^{(s)} + \bar{v}_{2n-2}^{(s)} + 1)2^{2n-2} + 8 + \sum_{i=3}^{2n-3} (u_i^{(s)} + u_{i-1}^{(c)} + \bar{v}_i^{(s)} + \bar{v}_{i-1}^{(c)})2^i + \\ &\quad 1 + \sum_{i=0}^{2} (u_i^{(s)} + \bar{v}_i^{(s)})2^i. \end{aligned}$$

The operator implementing this equation is mainly based on $(4,2)$-compressors (Figure 4b). P is finally converted to radix-2 RN-coding. Though this multiplier reduces the area and the delay compared to the one published in [2], it could only share a multioperand adder with a two's complement multiplier available in the ALU of a processor.

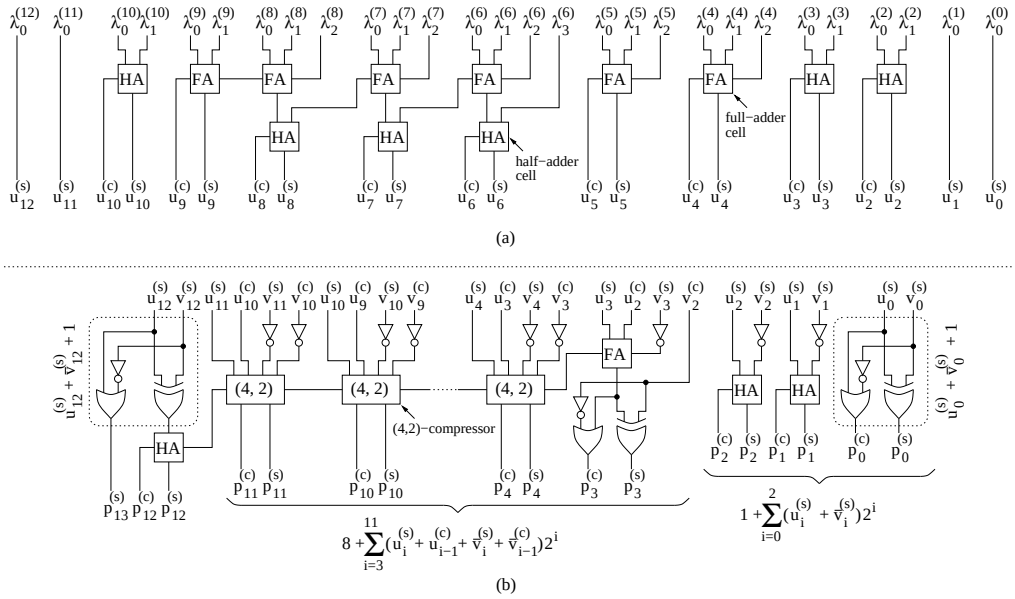

Figure 4: Multiplication of 7-digit radix-2 RN-codings. (a) Carry-save addition of the four partial products for $(X^+Y^+ + X^-Y^-)$. (b) Carry-save subtraction.

3 Modified Booth Recoding Revisited

Theorem 3 (Radix-β^k RN-codings). *Let $Y = y_{n-1} \dots y_0$ be an n-digit radix-β RN-coding of X. The radix-β^k number $Z = z_{\lceil n/k \rceil - 1} \dots z_0$, with $z_i = \sum_{j=0}^{k-1} y_{ki+j}\beta^i$, is also an RN-coding of X.*

Example 5 (Radix-4 modified Booth Recoding). *Trying to recode one of the binary operands so that its representation contains as many zeros as possible is a common way to design fast multipliers. The original Booth recoding was a first attempt. However, it sometimes increases the number of non-zero digits of the operand and is not implemented in modern ALUs: if we apply Algorithm 1 to $X = (101.0101)_2 = -2.6875$, we obtain $Y = (\bar{1}1\bar{1}.1\bar{1}1\bar{1})_4$. A common solution, known as radix-4 modified Booth recoding, consists in writing X on the digit set $\{\bar{2}, \dots, 2\}$ as follows:*

$$\begin{aligned} Z &= \sum_{i=-2}^{1} (-2x_{2i+1} + x_{2i} + x_{2i-1})4^i = \sum_{i=-2}^{1} (2(x_{2i} - x_{2i+1}) + (x_{2i-1} - x_{2i}))4^i \\ &= \sum_{i=-2}^{1} (y_{2i+1} + y_{2i})4^i = (\bar{1}1.11)_4 - 2.6875, \end{aligned} \tag{9}$$

where $x_{-5} = 0$ and $x_3 = 1$ (sign extension). We deduce from Equation (9) and from Theorem 3 that the radix-4 modified Booth recoding is an RN-coding of X. Daumas and Matula observed that "a digit 2 can only be followed by a negative digit possibly preceded by a string of zeros" [4] and that this special notation is not redundant. They applied this fact to design a circuit able to efficiently recode both binary and carry-save operands, but did not discover the rounding property of the representation.

3.1 Multiplication Algorithm Based on Radix-4 Modified Booth Recoding

A consequence of Theorem 3 is that a two's complement multiplier with radix-4 modified Booth recoding can easily be modified so that the recoded operand is either a two's complement number

or a radix-2 RN-coding. Partial product generation is often performed in two steps in state-of-the-art multipliers [5]: a Booth encoder is responsible of writing the operand X on the digit set $\{\bar{2}, \ldots, 2\}$; then, a Booth selector chooses a partial product among 0, $+Y$, $-Y$, $+2Y$, and $-2Y$ according to a digit of the recoded operand. Goto *et al.* showed that the area of the Booth selector highly depends on the encoding of the radix-4 digits [5] and proposed a solution with four bits (Table 1a): PL_i (positive), M_i (negative), $2R_i$ (doubled factor), and R_i (unchanged factor). Table 1b describes the computation of these control signals from two digits of a radix-2 RN-coding X. Note that several patterns never occur (we denote them by ϕ). According to the definition of X, we know for instance that $x_i^+ = 1 \Rightarrow x_i^- = 0$. Building a Karnaugh map allows to compute the logic equations defining PL_i, M_i, R_i, and $2R_i$:

$$\mathrm{PL}_i = x_{2i+1}^+ \vee x_{2i}^+ \overline{x_{2i+1}^-}, \quad M_i = x_{2i+1}^- \vee x_{2i}^- \overline{x_{2i+1}^+},$$
$$R_i = x_{2i}^+ \vee x_{2i}^-, \quad 2R_i = \overline{x_{2i}^+ \vee x_{2i}^-} = \bar{R}_i.$$

Figure 5 shows the implementation of the Booth encoder proposed by Goto *et al.* and our Booth encoder for radix-2 RN-codings. According to Goto *et al.*, the Booth encoder occupies 1.2% of the space in a 54×54-bit multiplier. Therefore, our modification does not increase significantly the area and the delay (a multiplexer added in the critical path) of the multiplier.

Table 1: Truth table for radix-4 modified Booth recoding.

x_{2i+1}	x_{2i}	x_{2i-1}	Func	R_i	$2R_i$	PL_i	M_i
0	0	0	0	0	1	0	0
0	0	1	$+Y$	1	0	1	0
0	1	0	$+Y$	1	0	1	0
0	1	1	$+2Y$	0	1	1	0
1	0	0	$-2Y$	0	1	0	1
1	0	1	$-Y$	1	0	0	1
1	1	0	$-Y$	1	0	0	1
1	1	1	0	0	1	0	0

(a) Two's complement operand (reprinted from [5]).

x_{2i+1}^+	x_{2i+1}^-	x_{2i}^+	x_{2i}^-	Func	R_i	$2R_i$	PL_i	M_i
0	0	0	0	0	0	1	0	0
0	0	0	1	$-Y$	1	0	0	1
0	0	1	0	$+Y$	1	0	1	0
0	0	1	1	–	ϕ	ϕ	ϕ	ϕ
0	1	0	0	$-2Y$	0	1	0	1
0	1	0	1	–	ϕ	ϕ	ϕ	ϕ
0	1	1	0	$-Y$	1	0	0	1
0	1	1	1	–	ϕ	ϕ	ϕ	ϕ
1	0	0	0	$+2Y$	0	1	1	0
1	0	0	1	$+Y$	1	0	1	0
1	0	1	0	–	ϕ	ϕ	ϕ	ϕ
1	0	1	1	–	ϕ	ϕ	ϕ	ϕ
1	1	0	0	–	ϕ	ϕ	ϕ	ϕ
1	1	0	1	–	ϕ	ϕ	ϕ	ϕ
1	1	1	0	–	ϕ	ϕ	ϕ	ϕ
1	1	1	1	–	ϕ	ϕ	ϕ	ϕ

(b) Radix-2 RN-coding.

This operator is a building block for our second multiplication algorithm. We suggest to compute $XY = XY^+ - XY^-$, where X is an n-digit radix-2 RN-coding, and Y^+ and Y^- are n-bit unsigned numbers. Up to this point, we assumed that X was either a two's complement number or a radix-2 RN-coding, and that the second operand was an n-bit two's complement number. Consequently, we need to perform a sign extension so that the second operand is an $(n+1)$-bit two's complement number. We define

$$\tilde{Y} = \begin{cases} 2^n y_{n-1} + Y & \text{if } Y \text{ is a two's complement number,} \\ Y & \text{otherwise.} \end{cases}$$

This implies a modification of the carry-save adder tree to handle an $(n+1)$-bit input. However, the number of partial products as well as the depth of the tree remain unchanged. Figure 5 describes an operator based on two such multiplier blocks, a subtracter, two fast adders, and multiplexers able to multiply two's complement numbers and RN-codings. Five control bits allow to select the desired operation (Table 2).

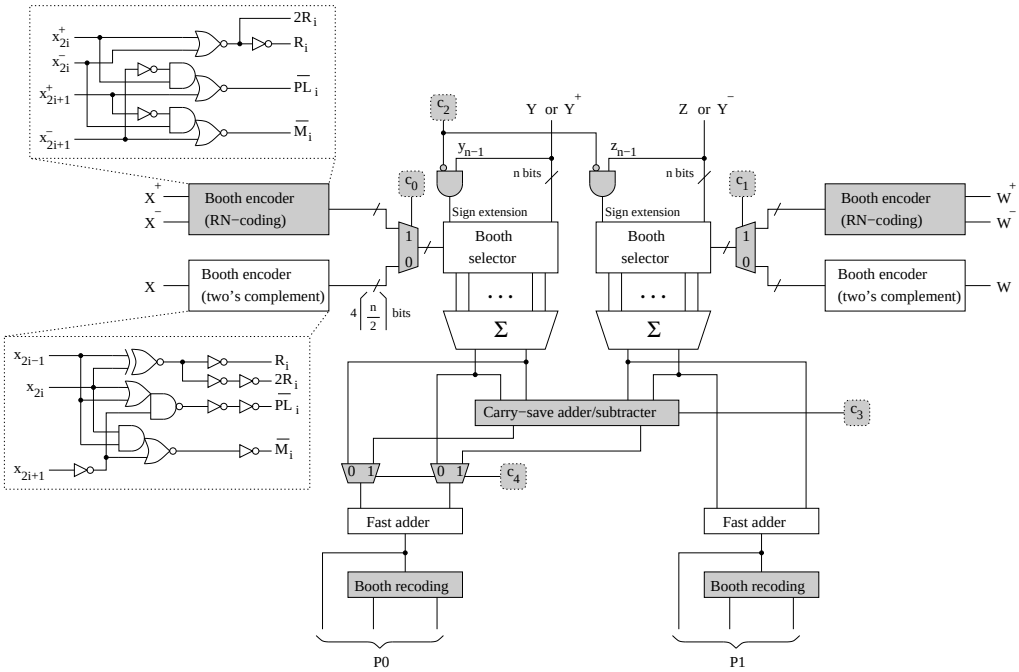

Figure 5: A multiplier handling two's complement numbers and radix-2 RN-codings. W, X, Y, and W are n-bit two's complement numbers. X^+, X^-, Y^+, and Y^- are n-bit unsigned numbers.

Table 2: Some operations implemented by the multiplier shown in Figure 5.

Operation		c_4	c_3	c_2	c_1	c_0
$P0 \leftarrow XY$	$P1 \leftarrow WZ$	0	0	0	0	0
$P0 \leftarrow XY + WZ$	$P1 \leftarrow WZ$	1	0	0	0	0
$P0 \leftarrow XY - WZ$	$P1 \leftarrow WZ$	1	1	0	0	0
$P0 \leftarrow (X^+ - X^-)Y$	$P1 \leftarrow WZ$	0	0	0	0	1
$P0 \leftarrow (X^+ - X^-)Y$	$P1 \leftarrow (X^+ - X^-)Z$	0	0	0	1	1
$P0 \leftarrow (X^+ - X^-)(Y^+ - Y^-)$	$P1 \leftarrow (X^+ - X^-)Y^-$	1	1	1	1	1

3.2 Combining Rounding to the Nearest and Partial Product Generation

Theorem 4 (Radix-2^k modified Booth recoding). *Let X be a two's complement number with n integer bits and m fractional bits. Choose two numbers b and k such that $k \geq 2$ and $bk < m$, assume that $x_j = x_{n-1}$, $\forall j \geq n$ (sign extension), and consider the radix-2^k number Y defined as follows:*

$$Y = \sum_{i=-b}^{\lceil n/k \rceil - 1} \left(-2^{k-1} x_{ki+k-1} + \sum_{j=0}^{k-2} 2^j x_{ki+j} + x_{ki-1} \right) 2^{ki}.$$

Y is the radix-2^k modified Booth recoding of $\circ_{n+bk}(X)$. If X is exactly between two representable numbers, then $Y = X + 2^{-bk-1}$.

Example 6. *Let $X = (1010.1010001011000)_2 = -5.3642578125$. We want to multiply a 12-bit two's complement number W by $\circ_{12}(X)$. The standard solution consists in rounding X to the nearest and in recoding $\circ_{12}(X)$ to select the partial products. We obtain $\circ_{12}(X) = (1010.10100011)_2$*

Algorithm 4 Combining rounding to the nearest and radix-2^k RN-coding conversion.

Require: A two's complement number with n integer bits and m fractional bits; two integers b and k with $bk < m$; $x_j = x_{n-1}$, $\forall j \geq n$ (sign extension)

Ensure: A radix-2^k RN-coding Y such that $Y = \circ_{n+bk}(X)$

1: **for** $i = -b$ to $\lceil n/k \rceil - 1$ **do**

2: $\quad y_i \leftarrow -2^{k-1}x_{ki+k-1} + \sum_{j=0}^{k-2} 2^j x_{ki+j} + x_{ki-1}$;

3: **end for**

and $Y = (\bar{1}\bar{1}.\bar{1}\bar{2}1\bar{1})_4 = -5.36328125$. *Theorem 4 allows to skip the first step: applied to* X *with* $k = 2$ *and* $b = 4$, *Algorithm 4 returns* $Y = (\bar{1}\bar{1}.\bar{1}\bar{2}1\bar{1})_4 = -5.36328125$.

To compute $\circ_{13}(X)$, *we choose* $b = k = 3$ *and obtain* $Y = (\bar{1}3.\bar{3}1\bar{2})_8 = -5.36328125$. *Note that* X *is exactly between two representable numbers. We check that* $X + 2^{-bk-1} = X + 2^{-10} = -5.36328125$.

We can for instance take advantage of Theorem 4 to evaluate a polynomial with Horner's rule. Instead of rounding intermediate results to the nearest (Figure 6a), we slightly modify the Booth selector of the multiplier so that it implements Algorithm 4 (Figure 6b). Assume that $k = 2$. In standard multiplier, the recoding cell responsible for the least significant digit is simpler than other cells in the sense that it only requires two bits (we assume that $x_{-bk-1} = 0$). To apply Algorithm 4, it suffices to add a third input bit to this cell so that it takes x_{-bk-1} into account.

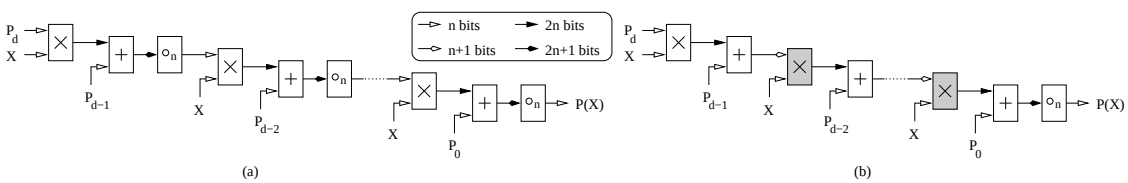

Figure 6: Polynomial evaluation with Horner's rule.

4 Conclusion

We have shown that very slightly modified arithmetic operators can efficiently handle both conventional binary representations and RN-codings. Since RN-codings can also be efficiently "compressed" for storage [6], we conclude that RN-codings are a good candidate for numerical computations. In a further study we will design dedicated division and square root algorithms, as well as an ALU able to handle RN-codings and conventional binary numbers.

References

[1] J.-C. Bajard, J. Duprat, S. Kla, and J.-M. Muller. Some operators for on-line radix-2 computations. *Journal of Parallel and Distributed Computing*, 22:336–345, 1994.

[2] Jean-Luc Beuchat and Jean-Michel Muller. RN-codes : algorithmes d'addition, de multiplication et d'élévation au carré. In *Actes de RenPar'16, CFSE'4 et SympAAA'2005*, 2005. To appear.

[3] A. D. Booth. A signed binary multiplication technique. *Quarterly Journal of Mechanics and Applied Mathematics*, 4(2):236–240, 1951.

[4] M. Daumas and D. W. Matula. Further reducing the redundancy of a notation over a minimally redundant digit set. *Journal of VLSI Signal Processing*, 33:7–18, 2003.

[5] G. Goto, A. Inoue, R. Ohe, S. Kashiwakura, S. Mitarai, T. Tsuru, and T. Izawa. A 4.1-ns compact 54×54-b multiplier utilizing sign-select Booth encoders. *IEEE Journal of Solid-State Circuits*, 32(11):1676–1682, November 1997.

[6] P. Kornerup and J.-M. Muller. RN-coding of numbers: definition and some properties. Technical Report 2004-43, Laboratoire de l'Informatique du Parallélisme, École Normale Supérieure de Lyon, 46 Allée d'Italie, 69364 Lyon Cedex 07, October 2004.

[7] R. Zimmermann. *Binary Adder Architectures for Cell-Based VLSI and their Synthesis.* PhD thesis, Swiss Federal Institute of Technology Zurich, Hartung–Gorre Verlag, 1997.

A Proofs

Proof of theorem 2. The definition of our radix-2 RN-coding guarantees that, if $x_i^{\pm} = 1$, then $x_{i+1}^{\pm} = x_i^{\mp} = x_{i-1}^{\pm} = 0$. Therefore,

$$x_i^+ y_j^+ + x_i^- y_j^- = x_i^+ y_j^+ \vee x_i^- y_j^-,$$

and

$$\begin{aligned} x_i^+ y_j^+ + x_i^- y_j^- + \psi(i-1,j) &= (x_i^+ y_j^+ \vee x_i^- y_j^-) + (x_{i-1}^+ y_j^+ x_i^- y_{j-1}^- \vee x_{i-1}^- y_j^- x_i^+ y_{j-1}^+) \\ &= (x_i^+ y_j^+ \vee x_i^- y_j^-) \vee (x_{i-1}^+ y_j^+ x_i^- y_{j-1}^- \vee x_{i-1}^- y_j^- x_i^+ y_{j-1}^+) \\ &= x_i^+ y_j^+ \vee x_i^- y_j^- \vee \psi(i-1,j). \end{aligned}$$

Let us build the addition table of $x_{i+1}^+ y_{j-1}^+$, $x_i^+ y_j^+$, $x_{i+1}^- y_{j-1}^-$, and $x_i^- y_j^-$ (Table 3). We apply again the definition of the RN-coding to remove all patterns that never occur. It is for instance impossible to have $x_i^+ y_j^+ = x_i^- y_j^- = 1$. The table indicates that the sum $S = 2s_1 + s_0$ belongs to $\{0,1,2\}$ and allows to check that:

$$\begin{aligned} s_0 &= (x_i^+ y_j^+ \vee x_{i+1}^+ y_{j-1}^+) \oplus (x_i^- y_j^- \vee x_{i+1}^- y_{j-1}^-) = \varphi(i,j), \\ s_1 &= (x_i^+ y_j^+ \vee x_{i+1}^+ y_{j-1}^+) \cdot (x_i^- y_j^- \vee x_{i+1}^- y_{j-1}^-) \\ &= \underbrace{x_i^+ y_j^+ x_i^- y_j^-}_{=0} \vee x_i^+ y_j^+ x_{i+1}^- y_{j-1}^- \vee x_{i+1}^+ y_{j-1}^+ x_i^- y_j^- \vee \underbrace{x_{i+1}^+ y_{j-1}^+ x_{i+1}^- y_{j-1}^-}_{=0} \\ &= x_i^+ y_j^+ x_{i+1}^- y_{j-1}^- \vee x_{i+1}^+ y_{j-1}^+ x_i^- y_j^- = \psi(i,j). \end{aligned}$$

Thus,

$$x_i^+ y_j^+ + x_{i+1}^+ y_{j-1}^+ + x_i^- y_j^- + x_{i+1}^- y_{j-1}^- = 2\psi(i,j) + \varphi(i,j).$$

Suppose now that $\varphi(i,j) = 1$ and remember that

$$\begin{aligned} \psi(i-1,j) &= x_{i-1}^+ y_j^+ x_i^- y_{j-1}^- \vee x_{i-1}^- y_j^- x_i^+ y_{j-1}^+, \\ \psi(i,j-1) &= x_i^+ y_{j-1}^+ x_{i+1}^- y_{j-2}^- \vee x_i^- y_{j-1}^- x_{i+1}^+ y_{j-2}^+. \end{aligned}$$

Table 4 summarizes the four cases for which $\varphi(i,j) = 1$. According to their definition, $\psi(i-1,j)$ and $\psi(i,j-1)$ are then equal to zero. Assume now that $\psi(i-1,j) = 1$, that is $x_{i-1}^+ = y_j^+ = x_i^- = y_{j-1}^- = 1$ or $x_{i-1}^- = y_j^- = x_i^+ = y_{j-1}^+ = 1$. Thus, $x_i^+ = y_j^- = x_{i+1}^- = y_{j-1}^+ = 0$ or $y_j^+ = x_i^- = y_{j-1}^- = x_{i+1}^+ = 0$, and $\varphi(i,j) = 0$. Consequently

$$\varphi(i,j) + \psi(i,j-1) = \varphi(i,j) \vee \psi(i,j-1).$$

Finally, if $\psi(i,j-1) = 1$, we have $x_i^+ = y_{j-1}^+ = x_{i+1}^- = y_{j-2}^- = 1$ or $x_i^- = y_{j-1}^- = x_{i+1}^+ = y_{j-2}^+ = 1$. Since $x_i^- = y_j^+ = x_{i+1}^+ = y_{j-1}^- = 0$ or $x_{i+1}^- = y_j^- = x_i^+ = y_{j-1}^+ = 0$, we obtain $\varphi(i,j) = 0$ and

$$\varphi(i,j) + \psi(i-1,j) = \varphi(i,j) \vee \psi(i-1,j).$$

□

Table 3: Addition of $x^+_{i+1}y^+_{j-1}$, $x^+_i y^+_j$, $x^-_{i+1}y^-_{j-1}$, and $x^-_i y^-_j$.

$x^+_{i+1}y^+_{j-1}$	$x^+_i y^+_j$	$x^-_{i+1}y^-_{j-1}$	$x^-_i y^-_j$	S
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	ϕ
0	1	0	0	1
0	1	0	1	ϕ
0	1	1	0	2
0	1	1	1	ϕ

$x^+_{i+1}y^+_{j-1}$	$x^+_i y^+_j$	$x^-_{i+1}y^-_{j-1}$	$x^-_i y^-_j$	S
1	0	0	0	1
1	0	0	1	2
1	0	1	0	ϕ
1	0	1	1	ϕ
1	1	0	0	ϕ
1	1	0	1	ϕ
1	1	1	0	ϕ
1	1	1	1	ϕ

Table 4: Computation of $\psi(i-1,j)$ and $\psi(i,j-1)$ when $\varphi(i,j)=1$.

x^+_{i+1}	y^+_{j-1}	x^+_i	y^+_j	x^-_{i+1}	y^-_{j-1}	x^-_i	y^-_j	$\psi(i-1,j)$	$\psi(i,j-1)$
1	1	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0
0	0	0	0	1	1	0	0	0	0
0	0	0	0	0	0	1	1	0	0

Proof of theorem 3. If β is odd, we have

$$|z_i| = \sum_{j=0}^{k-1} |y_{ki+j}|\beta^j \leq \frac{\beta-1}{2}\sum_{j=0}^{k-1}\beta^j.$$

Since

$$\sum_{j=0}^{k-1}\beta^j = \frac{\beta^k-1}{\beta-1},$$

$|z_i| \leq \frac{\beta^k-1}{2}$, and Z is therefore a radix-β^k RN-coding. Suppose now that β is even and that the most significant digit of the k-bit block $y_{(k+1)i-1}\dots y_{ki}$ is $\beta/2$ or $-\beta/2$. Theorem 1 guarantees that the first non-zero digit which follows on the right has an opposite sign. Thus, $-\beta/2 \leq y_{(k+1)i-1}\cdots y_{ki} \leq \beta/2$ and

$$|z_i| = \sum_{j=0}^{k-1} |y_{ki+j}|\beta^j \leq \frac{\beta}{2}\beta^{k-1} = \frac{\beta^k}{2}.$$

If $|z_i| = \beta^k/2$, we know that $|y_{(k+1)i-1}| = \beta/2$ and $y_{ki+j} = 0$, $\forall j \in \{0,\dots,k-2\}$. The most significant non-zero digit of $y_{ki-1}\dots y_0$, let us say y_q, is such that $y_{(k+1)i-1} \times y_q < 0$. Therefore, the most significant non-zero digit of $z_{i-1}\dots z_0$ has an opposite sign and Z is a radix-β^k RN-coding. □

Proof of theorem 4. Let Z be the radix-2^k modified Booth recoding of X. Algorithm 4 generates the $\lceil n/k \rceil$ integer digits and b fractional digits of Z (truncation). Since Z is an RN-coding (Theorem 3), $Y = \circ_{n+bk}(X)$. Assume now that X is exactly between two $(n+bk)$-bit numbers, that is $x_{-bk-1} = 1$ and $x_{-i} = 0$, $\forall i > bk+1$. The least significant digit of Y is

$$y_{-b} = -2^{k-1}x_{-kb+k-1} + \sum_{j=0}^{k-2} 2^j x_{-kb+j} + 1.$$

Therefore, $Y = X + 2^{-bk-1}$. □

Unité de recherche INRIA Rhône-Alpes
655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier (France)

Unité de recherche INRIA Futurs : Parc Club Orsay Université - ZAC des Vignes
4, rue Jacques Monod - 91893 ORSAY Cedex (France)
Unité de recherche INRIA Lorraine : LORIA, Technopôle de Nancy-Brabois - Campus scienti£que
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex (France)
Unité de recherche INRIA Rennes : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex (France)
Unité de recherche INRIA Rocquencourt : Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)
Unité de recherche INRIA Sophia Antipolis : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex (France)

Éditeur
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)
http://www.inria.fr
ISSN 0249-6399