Recoders for partial compression and rounding.

Marc Daumas, David W. Matula

Recoders for Partial Compression and Rounding

M. Daumas and D. Matula

January 1997

Research Report N$^{\text{O}}$ 97-01

Recoders for Partial Compression and Rounding

M. Daumas and D. Matula

January 1997

Abstract

The purpose of this paper is to treat digit set conversions and digit recodings in terms of primitive recoding operations that have elementary implementations. The partial compressions and roundings are each associated with borrow-save or carry-save recodings implementable in one level of logic. Iterative utilization of recoding have application for:

i) reducing the range of truncated lower order digits of a redundant binary operand to intervals less than a 2 ulp range $(-1, 1)$ approaching 1 ulp,

ii) truncating strings of leading insignificant digits in a redundant binary operand,

iii) realizing Booth recoding for radices 2^k, $k \geq 2$, by realizing the symmetric minimal redundant digit set for 2^k for all k bit substring of a redundant binary operand.

Keywords: Computer Arithmetic, Floating Point Notation, Redundant Notation, Number Conversion

Résumé

Le but de ce travail est de traiter les conversions entre systèmes d'écriture des nombres, et les conversions entre systèmes d'écriture des chiffres en termes d'opérations primitives de recodage qui possèdent une implantation élémentaire. La compression partielle et l'arrondi partiel associés avec les systèmes de notation redondante *borrow-save* ou *carry-save* sont alors implantés en un seul niveau de cellules logiques. En utilisant plusieurs fois ces cellules de recodage, nous obtenons les applications suivantes :

i) on peut réduire le domaine de troncature des chiffres de poids faible d'un nombre redondant à un intervalle plus petit que l'intervalle usuel $]-1, 1[$, jusqu'à approcher un intervalle de largeur 1 ulp ;

ii) on peut supprimer les chiffres non significatifs de tête d'un nombre redondant sans modifier sa valeur ;

iii) le codage de Booth en base 2^k, $k \geq 2$, est réalisé en utilisant l'ensemble de chiffres minimal symétrique redondant pour 2^k à partir d'un nombre redondant quelconque.

Mots-clés: Arithmétique des Ordinateurs, Notation à Virgule Flottante, Notation Redondante, Conversion

Recoders for Partial Compression and Rounding[*]

Marc Daumas[1] and David W. Matula[2]

[1] Laboratoire de l'Informatique du Parallélisme
ENS Lyon - Lyon, France
Marc.Daumas@Lip.ENS-Lyon.Fr

[2] Southern Methodist University
SEAS - Dallas, Texas
Matula@SEAS.SMU.Edu

1 Introduction and Summary

Inputs and outputs of adder and multiplier components of an arithmetic logic unit (ALU) are traditionally described in terms of compressed binary formats (*eg.* two's complement). Micro-coded and hardware enhanced algorithms for division, transcendentals, and fused multiply-add involve internal feedback or forwarding of intermediate results. Output of the multiplier may be fed-back to the multiplier input or forwarded to the adder. The expensive 2-1 compression, that is an addition of the multiplier output can be avoided if the redundant output is forwarded directly. The problem we address here is how can the forwarded value be partially compressed so that the provision for forwarded input does not significantly degrade the adder or multiplier compared to optimisation for compressed binary input.

Previous research has shown the feasibility of multiplier and adder designs employing redundant binary operands [1, 2, 3, 4, 5, 6]. To avoid the general increase in hardware size entailed by redundant binary input, recent attention has been focused on limiting redundant input simply to the multiplier recoder input [7, 8, 9, 10, 11, 12]. Such recoders can add critical path delay for the more frequent case where compressed binary input is available.

Our purpose in this paper is to investigate the use of partial compression of redundant binary output for forwarding and feedback. The goal is that employing the forwarded value should be virtually transparent in performance compared to direct input of compressed binary values. We provide a methodology for partial compression and illustrate the realization of our goal in one important application by the logic design of a *precoder*. The precoder provides partial compression of a redundant binary value before feedback with output in a format that may be directly input to a standard radix 4 Booth recoder.

In Section 2, we describe various compressed binary and redundant binary formats and in-place recodings. In-place recoding of binary to redundant binary format involves only wire routing and possibly bitwise complementation. This establishes compressed binary input in a register

[*]This work has been partially supported by the Texas Advanced Technology program, by Cyrix Corporation and by the région Rhône-Alpes.

format that may alternately receive partially compressed forwarded redundant binary data in a performance-wise transparent manner.

In Section 3, we establish the foundation for partial compression by carry-recodings. Our methods apply to both carry-save and borrow-save format. They are most simply described for borrow-save. Borrow-save format has a positive bit p_i and a negative bit n_i, encoding the ith digit $d_i \in \{-1, 0, 1\}$ for all i. Truncating a fully redundant borrow-save format at the ith place deletes the fraction portion $d_{i-1}d_{i-2}...$ having value in the fraction range $(-1, 1)$ units in the last place (ulps). A P-carry recoding propagates a positive carry p'_{i+1} leaving a negative residual n'_i for all i. It reduces the fraction range $d'_{i-1}d'_{i-2}...$ to $\left(-1, \frac{1}{2}\right)$ ulps. An N-carry recoding propagates a negative carry n'_{i+1} with positive residual p'_i for all i with fraction range $\left(-\frac{1}{2}, 1\right)$ ulps. Thus a single P or N recoding compresses the 2 ulp width of redundant binary half way to the one ulp width of the compressed binary format at very little cost. Extending the terminology that 3-2 and 2-1 adders are also termed 3-2 and 2-1 compressors, we may call a single carry recoder a 2-1$\frac{1}{2}$ compressor.

Our principal result in Section 3 is the following theorem for a sequence of carry recodings. The compression theorem allows us to measure the compression obtained by recoding a borrow-save variable. Compression index k corresponds to a fraction width of at most $1 + 2^{-k}$ ulps. Note that a recoding obtaining compression index k is sufficient to provide a fully compressed binary output for a k digit borrow save input.

Theorem 1 (Partial Compression Theorem) *Given as input a borrow-save maximally redundant binary variable with fraction range of width 2 ulps, any sequence of k P- or N- recodings in any order compresses the width of the fraction range to $1 + 2^{-k}$ ulps.*

The basis of our precoder is the index 3 compression provided by the carry recoding $PN^2(\cdot)$. This recoding is shown sufficient to provide that every two digit output has a radix four value in $\{-2, -1, 0, 1, 2\}$. Further properties allow the recoded output to be directly input to a standard Booth recoder.

In Section 4, we show that a single carry recoding can be implemented with single logic level depth as described by a row of half adders. Our $PN^2(\cdot)$ precoder is comprised of three rows of half-adders. We describe a BiCMOS transistor implementation of the precoder indicating delay significantly less than a nanosecond. The precoder has delay only a small fraction of the delay of a 2-1 compressor which could require a full cycle at 400-500 MHz for compression of a 64 bit carry-save multiplier output. Conclusions and other applications of partial compressors are discussed in the Section 5.

2 Compressed and Redundant Binary Formats

Standard binary compressed representations suitable for storage include unsigned binary, sign-magnitude and two's complement. Redundant binary representations encountered internal to an ALU include carry-save, borrow-save and signed digits. Operations within the ALU can effectively recode, convert and round the internal formats.

2.1 Compressed Binary Formats

The basis for all compressed binary format representations is a bit vector and its associated weight vector which together determine a radix polynomial. Alternative interpretations of the leading bit are employed to distinguish between ranges that include or exclude negative values.

- An **unsigned binary** (UB) bit vector $b_{k-1}b_{k-2}...b_0$ with weight vector $\mathbf{W} = (2^{k-1}, 2^{k-2}...2^0)$ denotes the radix polynomial $\sum_{i=0}^{k-1} b_i 2^i$ with value in $[0, 2^k - 1]$.
- A **sign-magnitude** (SM) bit vector $sb_{k-2}...b_0$ denotes the sign by $(-1)^s$, yielding the signed radix polynomial $(-1)^s \sum_{i=0}^{k-2} b_i 2^i$ which has values in the balanced range $[-2^{k-1}+1, 2^{k-1}-1]$.
- A **two's complement** (2C) bit vector $b_{k-1}b_{k-2}...b_0$ with weight vector $\mathbf{W} = (-2^{k-1}, 2^{k-2}...2^0)$ determines negative values by having b_{k-1} denote a negative coefficient. The resulting radix polynomial $-b_{k-1}2^{k-1} + \sum_{i=0}^{k-2} b_i 2^i$ has values in the unbalanced range $[-2^{k-1}, 2^{k-1} - 1]$.

Observation 1 alerts us that the value zero requires special attention in sign-magnitude format, and the value -2^{k-1} requires special attention in the design of a two's complement arithmetic unit.

Observation 1 *The sign-magnitude bit vector $sb_{k-2}...b_0$ allows redundant representations of* 0 *but any other integer in its range is defined uniquely. A two's complement bit vector represents every integer uniquely in its range with the value -2^{k-1} being the only value for which its negation $-(-2^{k-1})$ is out of range.*

Conversion rules for the three compressed binary formats regarding leading insignificant bit deletion, higher radix digit conversion, and low order digit chopping are well known and follow similar but not identical rules. An appropriate number of leading bits (beyond the sign in sign-magnitude notation) may simply be deleted from a k-bit binary vector whose magnitude is at most 2^{j-1} for $j < k$. This holds for insignificant zeros or ones in two's complement in view of the following identity for b being 0 or 1.

$$-b2^{k-1} + \sum_{i=j}^{k-2} b2^i = -b2^j \tag{1}$$

A fixed point binary vector may be arbitrarily extended to the right of the radix point to $b_{k-1}...b_0b_{-1}b_{-2}...$ with the bit window $b_{-1}b_{-2}...$ denoting the radix polynomial $\sum_{i=1}^{+\infty} b_{-i}2^{-i}$ with value in the fraction range $[0, 1)$. For any finite or infinite bit vector $b_{k-1}b_{k-2}...$ the *fraction value at position* j, $f_j(\cdot)$ is the ratio of the term of order $j-1$ and below of the radix polynomial divided by the weight of the first term not counted to normalize to units in the last place. For $j < k$ this is defined along with the *fraction value function* $f(\cdot)$ by

$$f_j\left(b_{k-1}b_{k-2}...\right) = f\left(b_{j-1}b_{j-2}...\right) = \sum_{i>0} b_{j-i}2^{-i} \in [0, 1) \tag{2}$$

Chopping of the low order bits below position j results in subtracting the fractional value f_j. Note that this fractional range is always non negative for either unsigned or two's complement yielding a round down towards $-\infty$ (RD) rounded value for the bit vector $b_{k-1}b_{k-2}...b_j$. For sign-magnitude the chopped portion deleted inherits the sign of the bit vector which effects a round towards zero (RZ) with range $(-1, 1)$.

For any finite or infinite bit vector $b_{k-1}b_{k-2}...$ an *l-bit window* at position j denotes the substring $b_{j+l-1}b_{j+l-2}...b_j$. The value of the l-bit window is given by the integer ratio of the corresponding terms of the radix polynomial divided by the weight of the last term b_j.

$$d\left(b_{j+l-1}b_{j+l-2}...b_j\right) = \sum_{i=0}^{l-1} b_{j+i}2^i = d_l \in [0, 2^{l-1}] \tag{3}$$

In two's complement a window including the sign bit will have $d(\cdot)$ map into $[-2^{l-1}, 2^{l-1} - 1]$ by the appropriate modification. All this data are summarized in Table 1.

Table 1: Interpretation and properties of compressed binary vectors

Bit vector format	Radix polynomial	Range of values	Fraction range (ulps)	Rounding effected
Unsigned binary (UB)	$\sum_{i=0}^{k-1} b_i 2^i$	$[0, 2^k - 1]$	$[0, 1)$	RZ
Two's complement (2C)	$-b_{k-1}2^{k-1} + \sum_{i=0}^{k-2} b_i 2^i$	$[-2^{k-1}, 2^{k-1} - 1]$	$[0, 1)$	RD
Sign-magnitude (SM)	$(-1)^s + \sum_{i=0}^{k-2} b_i 2^i$	$[-2^{k-1} + 1, 2^{k-1} - 1]$	$(-1, 1)$	RZ

2.2 Redundant Binary Formats

A redundant binary format is a $2 \times k$ bit array $\{b_{j,i}\}$ with an associated $2 \times k$ weight array $\{w_{j,i}\}$ with $w_{j,i} \in \{2^i, -2^i, -2.2^i\}$ for all i. The bit array is equivalently interpreted by rows or columns:

(i) for $j = 1, 2$, the rows $\mathbf{b_j} = b_{j,k-1}b_{j,k-2}...b_{j,0}$ each comprise a binary vector of unsigned binary or two's complement format whose weighted values may be summed to provide the value of the representation,

(ii) the ith column $\begin{bmatrix} b_{1,i} \\ b_{2,i} \end{bmatrix}$ determines digit d_i and term $d_i 2^i = b_{1,i}w_{1,i} + b_{2,i}w_{2,i}$ of the radix polynomial $\sum_{i=0}^{k-1} d_i 2^i$ which provides the value of the representation.

Carry save, borrow save and signed digit redundant formats are differentiated simply by the assumed weight arrays associated with the $2 \times k$ bit array.

- A **carry save** (CS) bit array $\mathbf{C}$ employs the weight array $\mathbf{W}$ to denote the radix polynomial.

$$\mathbf{C} = \begin{bmatrix} p_{k-1} & p_{k-2} & \cdots & p_0 \\ q_{k-1} & q_{k-2} & \cdots & q_0 \end{bmatrix} \quad \mathbf{W} = \begin{bmatrix} -2^{k-1} & 2^{k-2} & 2^{k-3} & \cdots & 2^0 \\ -2^{k-1} & 2^{k-2} & 2^{k-3} & \cdots & 2^0 \end{bmatrix} \tag{4}$$

All digits $d_i = (p_i + q_i)$ are in the extended binary digit set $\{0, 1, 2\}$ for $0 \le i \le k-2$ with the sign digit $d_{k-1} = -(p_{k-1} + q_{k-1})$ in the negative binary extended digit set $\{-2, -1, 0\}$. The CS radix polynomial value is in the range $[-2^k, 2^k - 2]$. Equivalently the CS bit array has for rows the two's complement bit vectors $\mathbf{p} = p_{k-1}p_{k-2}...p_0$ and $\mathbf{q} = q_{k-1}q_{k-2}...q_0$ such that the sum $(\mathbf{p} + \mathbf{q})$ of their radix polynomials yields the CS radix polynomial.

$$\underbrace{-p_{k-1}2^{k-1} + \sum_{i=0}^{k-2} p_i 2^i}_{\text{Two's complement}} \quad + \quad \underbrace{-q_{k-1}2^{k-1} + \sum_{i=0}^{k-2} q_i 2^i}_{\text{Two's complement}} \quad = \quad \underbrace{-(p_{k-1} + q_{k-1})2^{k-1} + \sum_{i=0}^{k-2} (p_i + q_i) 2^i}_{\text{CS radix polynomial}}$$

- A **borrow save** (BS) bit array $\mathbf{B}$ employs the weight array $\mathbf{W}$ to denote the radix polynomial.

$$\mathbf{B} = \begin{bmatrix} p_{k-1} & p_{k-2} & \cdots & p_0 \\ n_{k-1} & n_{k-2} & \cdots & n_0 \end{bmatrix} \quad \mathbf{W} = \begin{bmatrix} 2^{k-1} & 2^{k-2} & \cdots & 2^0 \\ -2^{k-1} & -2^{k-2} & \cdots & -2^0 \end{bmatrix} \tag{5}$$

All digits $d_i = (p_i - n_i)$ are in the minimally redundant balanced digit set $\{-1, 0, 1\}$ and the BS radix polynomial value is in the balanced range $[-(2^k-1), 2^k-1]$. Equivalently the BS bit array has for rows the unsigned binary vectors $\mathbf{p} = p_{k-1}p_{k-2}...p_0$ and $\mathbf{n} = n_{k-1}n_{k-2}...n_0$ such that the difference $(\mathbf{p} - \mathbf{n})$ of their radix polynomials yields the BS radix polynomial.

$$\begin{array}{ccccc} \sum_{i=0}^{k-1} p_i 2^i & - & \sum_{i=0}^{k-1} q_i 2^i & = & \sum_{i=0}^{k-1} (p_i - q_i) 2^i \\ \text{Unsigned binary} & & \text{Unsigned binary} & & \text{BS radix polynomial} \end{array}$$

- A **signed digit** (SD) bit array $\mathbf{A}$ with $\begin{bmatrix} s_i \\ b_i \end{bmatrix} \neq \begin{bmatrix} 1 \\ 0 \end{bmatrix}$ for all i employs the weight array $\mathbf{W}$ to denote the SD radix polynomial $\sum_{i=0}^{k-1}(-2s_i + b_i)2^i = \sum_{i=0}^{k-1}(-1)^{s_i} b_i 2^i$ (see Observation 2).

$$\mathbf{A} = \begin{bmatrix} s_{k-1} & s_{k-2} & ... & s_0 \\ b_{k-1} & b_{k-2} & ... & b_0 \end{bmatrix} \quad \mathbf{W} = \begin{bmatrix} -2.2^{k-1} & -2.2^{k-2} & ... & -2.2^0 \\ 2^{k-1} & 2^{k-2} & ... & 2^0 \end{bmatrix} \tag{6}$$

All digits $d_i = (-2s_i + b_i)$ are in the minimally redundant balanced digit set $\{-1, 0, 1\}$ and the SD radix polynomial value is in the balanced range $[-(2^k-1), 2^k-1]$. Equivalently the SD bit array has for rows the unsigned binary vectors $\mathbf{s} = s_{k-1}s_{k-2}...s_0$ and $\mathbf{b} = b_{k-1}b_{k-2}...b_0$ such that the expression $(-2\mathbf{s} + \mathbf{b})$ of their radix polynomials yields the SD radix polynomial.

$$\begin{array}{ccccccc} -2 & \cdot & \sum_{i=0}^{k-1} s_i 2^i & + & \sum_{i=0}^{k-1} b_i 2^i & = & \sum_{i=0}^{k-1} (2s_i + b_i) 2^i \\ & & \text{Unsigned binary} & & \text{Unsigned binary} & & \text{SD radix polynomial} \end{array}$$

Observation 2 *The sign-magnitude 2-bit vector $sb_0 = b_1b_0$ denoting $(-1)^s b_0$ and two's complement 2-bit vector b_1b_0 denoting $-2b_1 + b_0$ satisfies*

$$(-1)^{b_1} b_0 = -2b_1 + b_0 \quad \textit{for} \quad b_1b_0 \in \{00, 01, 11\}. \tag{7}$$

Requiring $b_1b_0 \neq 10$ allows the sign bit s to be assigned the weight -2 in encoding the digit set $\{-1, 0, 1\}$

The alternative row and column evaluations of a carry-save format are illustrated in example 1. The $\mathbf{p}$ and $\mathbf{q}$ vectors evaluated as two's complement values yield 10 and -13, with their sum $10 - 13 = 3$ being the carry-save value. The columns yield the digit vector $(-1, 1, 0, 2, 1)$ with radix polynomial $-16 + 8 + 2 \cdot 2 + 1 = -3$.

Example 1

$$\begin{array}{cc} \mathbf{p} \\ \mathbf{q} \end{array} \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 1 \end{bmatrix} \begin{array}{cc} \longrightarrow & 10 \\ \longrightarrow & \text{-13} \end{array}$$

$$\text{Digits} \ \ (\ \text{-1} \ \ 1 \ \ 0 \ \ 2 \ \ 1\ ) \ \longrightarrow \ \text{-3}$$

For convenience we shall herein use $\mathbf{p}$ and $\mathbf{q}$ as the row vectors of the carry-save bit array. We then use $\mathbf{p}$ for the positive bit vector and $\mathbf{n}$ for the negative bit vector of the borrow save bit array, and $\mathbf{s}$ for the sign bit vector and $\mathbf{b}$ for the magnitude bit vector of the signed digit bit array.

A base β digit set is minimally redundant if it has $\beta + 1$ digit values and is balanced if $-d$ is an allowed digit whenever d is an allowed digit. The bitwise encoding of digits is a redundant encoding if some allowed digit is encoded by more than one bit vector. Table 2(a) illustrates that carry save provides redundant encoding of the extended binary digit set $\{0, 1, 2\}$ which is itself a

Table 2: Digit set encoding

(a) Carry Save

p_i	0	0	1	1
q_i	0	1	0	1
d_i	0	1	1	2

(b) Borrow Save

p_i	0	0	1	1
n_i	0	1	0	1
d_i	0	-1	1	0

(c) Signed Digit

s_i	0	0	1	1
b_i	0	1	0	1
d_i	0	1	NA	-1

minimally redundant base 2 digit set. Both borrow save and signed digit employ the minimally redundant balanced base 2 digit set $\{\overline{1}, 0, 1\}$ with borrow save providing a redundant encoding and signed digit a non redundant encoding as illustrated in Tables 2(b,c).

For any redundant binary format $2 \times k$ bit array, an *l-bit window at position j* denotes the $2 \times l$ subarray of the bits position j through $j + l - 1$. The value of the *l-bit window* is given by the integer valued *digit value function* $d(\cdot)$ separately extended to borrow save and carry save by:

$$\begin{aligned} d\begin{pmatrix} p_{j+l-1} & p_{j+l-2} & \cdots & p_j \\ n_{j+l-1} & n_{j+l-2} & \cdots & n_j \end{pmatrix} &= \sum_{i=0}^{l-1}(p_{i+j} - n_{i+j})2^i \in \{-(2^l-1), ..., 2^l - 1\} \\ d\begin{pmatrix} p_{j+l-1} & p_{j+l-2} & \cdots & p_j \\ q_{j+l-1} & q_{j+l-2} & \cdots & q_j \end{pmatrix} &= \sum_{i=0}^{l-1}(p_{i+j} + q_{i+j})2^i \in \{0, 1, ..., 2^{l+1} - 2\} \end{aligned} \tag{8}$$

The meaning and range of $d(\cdot)$ for carry save is appropriately modified when the window includes the sign bits p_k, q_k. The *fraction value at position j*, $f_j(\cdot)$ for $j < k$ and the *fraction value function* $f(\cdot)$ are defined for borrow save and carry save with value in ulps at position j by

$$\begin{aligned} f_j\begin{pmatrix} p_{k-1} & p_{k-2} & \cdots \\ n_{k-1} & n_{k-2} & \cdots \end{pmatrix} = f\begin{pmatrix} p_{j-1} & p_{j-2}\cdots \\ n_{j-1} & n_{j-2}\cdots \end{pmatrix} &= \sum_{i>0}(p_{j-i} - n_{j-i})2^{-i} \in (-1, 1) \\ f_j\begin{pmatrix} p_{k-1} & p_{k-2} & \cdots \\ q_{k-1} & q_{k-2} & \cdots \end{pmatrix} = f\begin{pmatrix} p_{j-1} & p_{j-2}\cdots \\ q_{j-1} & q_{j-2}\cdots \end{pmatrix} &= \sum_{i>0}(p_{j-i} + q_{j-i})2^{-i} \in [0, 2) \end{aligned} \tag{9}$$

The higher radix digit sets and fraction ranges for uncompressed redundant binary formats are both essentially twice as large as for compressed binary.

2.3 In-place Recoding

A recoding where (i) the input is one or two binary vectors or a redundant binary array and (ii) the output is a redundant binary array, is an *in-place recoding* whenever each output bit is determined by at most one input bit. In-place recodings are efficiently implemented in hardware by direct wiring with complementations as needed. Three classes of in-place recodings merit separate comments.

Compressed binary to redundant binary conversion In-place recodings from 2C or UB to redundant binary can be implemented so as to both convert the digit set of the higher radix digit windows and change the nature of the rounding obtained when low order digits are deleted. Table 3 illustrates five in-place recodings of 2C to redundant binary. The correctness of each is readily verified by reference to the defining weight vectors (4), (5), (6).

Table 3: In-place redundant binary recoding of 2C bit vectors

Recoding	Bit array	Length	Fraction range
2C to CS	$\begin{bmatrix} b_{k-1} & b_{k-2} & \dots & b_1 & b_0 \\ 0 & 0 & \dots & 0 & 0 \end{bmatrix}$	k	$[0,1)$ (directed)
	$\begin{bmatrix} b_{k-1} & b_{k-2} & b_{k-3} & \dots & b_0 & 1 \\ \overline{b}_{k-1} & \overline{b}_{k-1} & \overline{b}_{k-2} & \dots & \overline{b}_1 & \overline{b}_0 \end{bmatrix}$	$k+1$	$\left[\frac{1}{2},\frac{3}{2}\right)$ (centered)
2C to BS	$\begin{bmatrix} 0 & b_{k-2} & \dots & b_1 & b_0 \\ b_{k-1} & 0 & \dots & 0 & 0 \end{bmatrix}$	k	$[0,1)$ (directed)
	$\begin{bmatrix} b_{k-2} & b_{k-3} & \dots & b_0 & 0 \\ b_{k-1} & b_{k-2} & \dots & b_1 & b_0 \end{bmatrix}$	k	$\left(-\frac{1}{2},\frac{1}{2}\right]$ (centered)
2C to SD	$\begin{bmatrix} b_{k-1} & 0 & \dots & 0 & 0 \\ b_{k-1} & b_{k-2} & \dots & b_1 & b_0 \end{bmatrix}$	k	$[0,1)$ (directed)

Table 4: In-place redundant binary recoding of sum of binary vectors

Recoding	Sum	Bits array	Length	Fraction range
2C to CS	$a+b$	$\begin{bmatrix} a_{k-1} & a_{k-2} & \dots & a_1 & a_0 \\ b_{k-1} & b_{k-2} & \dots & b_1 & b_0 \end{bmatrix}$	k	$[0,2)$
	$a-b$	$\begin{bmatrix} a_{k-1} & a_{k-2} & \dots & a_1 & a_0 & . & 1 \\ \overline{b}_{k-1} & \overline{b}_{k-2} & \dots & \overline{b}_1 & \overline{b}_0 & & 1 \end{bmatrix}$	$k+1$	$[0,2)$
UB to BS	$a-b$	$\begin{bmatrix} a_{k-1} & a_{k-2} & \dots & a_1 & a_0 \\ b_{k-1} & b_{k-2} & \dots & b_1 & b_0 \end{bmatrix}$	k	$(-1,1)$

Observation 3 *For the 2C to BS (centered) recoding, an l-digit window of the output yields digit values in the minimally redundant 2^l digit set*

$$d\begin{pmatrix} b_{j+l-2} & \dots & b_j & b_{j-1} \\ b_{j+l-1} & \dots & b_{j+1} & b_j \end{pmatrix} \in \{-2^{l-1}, \dots 2^{l-1}\} \tag{10}$$

eg. $\{-2,-1,0,1,2\}$ *radix 4. Furthermore, low order digit chopping effects a round to nearest rounding (with mid-points towards $+\infty$) since*

$$f_j\begin{pmatrix} b_{k-2} & b_{k-3} & \dots \\ b_{k-1} & b_{k-2} & \dots \end{pmatrix} = f\begin{pmatrix} b_{j-2} & b_{j-3} & \dots \\ b_{j-1} & b_{j-2} & \dots \end{pmatrix} \in \left[-\frac{1}{2},\frac{1}{2}\right) ulp \tag{11}$$

Binary sums to redundant binary For the 2C binary vectors $a = a_{k-1}a_{k-2}\dots a_0$ and $b = b_{k-1}b_{k-2}\dots b_0$ the sum $\mathbf{a}+\mathbf{b}$ and the difference $a-b$ can be realized by a direct recoding of the result in redundant binary format. Table 4 illustrates three such summation and conversion operations realized by in-place recodings, the correctness is again readily verified by the defining weight vectors.

Note that for 2C to CS general summation $a-b$ and conversion the CS result was extended to the right one place with the added digit $(1+1)2^{-1} = 1$ ulp. This provides that the carry of

Table 5: In place recoding of redundant bit arrays

Recoding	Bit array	Length	Fraction range
BS to CS	$\left[\begin{array}{cccccccc} 0 & p_{k-1} & p_{k-2} & \cdots & p_1 & p_0 & . & 1 \\ 1 & \overline{n}_{k-1} & \overline{n}_{k-2} & \cdots & \overline{n}_1 & \overline{n}_0 & & 1 \end{array}\right]$	$k+1$	$[0,2)$
SB to BS	$\left[\begin{array}{cccccc} 0 & b_{k-1} & b_{k-2} & \cdots & b_1 & b_0 \\ s_{k-1} & s_{k-2} & s_{k-3} & \cdots & s_0 & 0 \end{array}\right]$	$k+1$	$(-\frac{1}{2},1)$

the two's complement operation on the vector b can be represented without the need for carry absorption logic.

Applications of binary sums to redundant binary conversion for more efficient complex number multiplication was discussed in [6], and for more efficient interpolation table look-up discussed in [13].

Observation 4 *Given the three compressed binary inputs a, b and c, the compound operation $(a \pm b) \times c$ can be implemented by a multiplier accepting its multiplier recoding input in redundant binary format.*

The usefulness of Observation 4 will depend on the latencies of the multiplier recoding for the cases of compressed and redundant binary inputs discussed in Section 3.

Redundant binary to redundant binary reformatting Table 5 illustrates that a BS (resp. SD) $2 \times k$ bit array can be recoded in-place to a CS $2 \times (k+2)$ (resp. BS $2 \times (k+1)$) bit array. When high and low bit array extensions are allowed, the carry save format is the most versatile of the output formats for in-place recoding to redundant binary.

For the recoding from SD to BS, it is important to note that the fraction range has been compressed from a width of two ulps to $1\frac{1}{2}$ ulps. It should be recognized that this compression is effectively realized at the cost that the 2×1 bit vector $\left[\begin{array}{c} s_i \\ b_i \end{array}\right] = \left[\begin{array}{c} 1 \\ 0 \end{array}\right]$ can not occur in the SD array.

3 Carry Recoding

Carry recoding maps a $2 \times k$ bit borrow save B or carry save C array into a $2 \times (k+1)$ bit borrow save array B' by generating an $(i+1)$th carry bit and an ith residual bit from each column i of the input array with no alteration to the actual value of the radix polynomial. The three recodings of interest herein are readily defined by logic formulas on the respective bits.

- $B' = P(B)$ denotes the P-carry recoding from BS to BS given by

  Carry $p'_{i+1} = p_i.\overline{n}_i$ **Residual** $n'_i = p_i \oplus n_i$

- $B' = N(B)$ denotes the N-carry recoding from BS to BS given by

  Carry $n'_{i+1} = \overline{p}_i.n_i$ **Residual** $p'_i = p_i \oplus n_i$

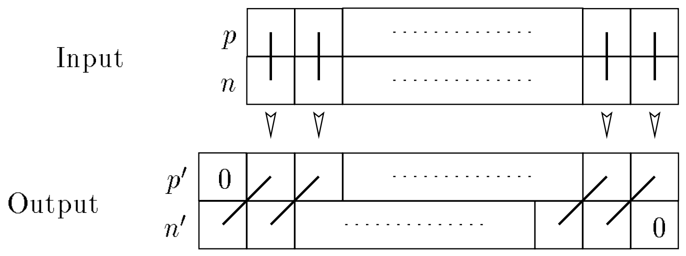


Figure 1: N-carry recoding

- $B' = Q(C)$ denotes the Q-carry recoding from CS to BS given for $0 \le i \le k-2$ by

  Carry $p'_{i+1} = p_i + q_i$ **Residual** $n'_i = p_i \oplus q_i$

  with the following pertaining to the sign digit input

  Carry $n'_{k+1} = p_k . q_k$ **Residual** $n'_k = p_k \oplus q_k$

Figure 1 illustrates how each output diagonal of the $2 \times (k+1)$ bit array is determined by an input column of the $2 \times k$ bit array for the N-carry recoding. A carry recoding compresses the fraction range of a borrow save bit array uniformly at each position i.

Lemma 1 *Let B be a borrow save bit array with fraction range $f_j(B) \in (a,b)$ for all j. Then*

$$f_j\left(P(B)\right) \in \left(-\frac{1}{2}+\frac{a}{2},\frac{b}{2}\right) \quad \textit{and} \quad f_j\left(N(B)\right) \in \left(\frac{a}{2},\frac{1}{2}+\frac{b}{2}\right) \quad \textit{for all } j \tag{12}$$

Proof With $B = \begin{bmatrix} p_{k-1} & p_{k-2} & \cdots \\ n_{k-1} & n_{k-2} & \cdots \end{bmatrix}$ and $P(B) = \begin{bmatrix} p'_k & p'_{k-1} & \cdots \\ 0 & n'_{k-1} & \cdots \end{bmatrix}$ we obtain by extracting a term from the radix polynomial of $P(B)$

$$\begin{aligned} f_j\left(P(B)\right) &= f\begin{pmatrix} p'_{j-1} & p'_{j-2} & \cdots \\ n'_{j-1} & n'_{j-2} & \cdots \end{pmatrix} = -n'_{j-1}2^{-1} + f\begin{pmatrix} p'_{j-1} & p'_{j-2} & \cdots \\ 0 & n'_{j-2} & \cdots \end{pmatrix} \\ &= -n'_{j-1}2^{-1} + f\begin{pmatrix} 0 & p_{j-2} & p_{j-3} & \cdots \\ 0 & n_{j-2} & n_{j-3} & \cdots \end{pmatrix} \\ &= -n'_{j-1}2^{-1} + \tfrac{1}{2} f\begin{pmatrix} p_{j-2} & p_{j-3} & \cdots \\ n_{j-2} & n_{j-3} & \cdots \end{pmatrix} \end{aligned}$$

Since $-n'_{j-1}2^{-1} \in \left\{-\frac{1}{2}, 0\right\}$ and $f\begin{pmatrix} p_{j-2} & p_{j-3} & \cdots \\ n_{j-2} & n_{j-3} & \cdots \end{pmatrix} \in [a,b)$ we obtain $f_j\left(P(B)\right) \in \left(-\frac{1}{2}+\frac{a}{2},\frac{b}{2}\right)$. The result for $f_j(N(B))$ follows similarly.

□

A sequence of carry recodings can be employed to determine a desired compression index for the fraction range approaching the width of 1 ulp. A λ *level carry recoding* is defined recursively for all $\lambda \ge 0$ on a borrow save bit array B as follows:

(i) $B^{[0]} = B$ is a 0-level carry recoding of B,

(ii) for $B^{[\lambda-1]}$ a $(\lambda-1)$-level carry recoding of B, $B^{[\lambda]} = P\left(B^{[\lambda-1]}\right)$ and $B^{[\lambda]} = N\left(B^{[\lambda-1]}\right)$ are λ-level carry recodings of B.

For example $P(N(B))$ or simply $PN(B)$ is a 2-level recoding of B and $P(N(N(B)))$ or $PN^2(B)$ is a 3-level recoding. From Lemma 1 we obtain the following providing a convenient measure of compression level.

Theorem 2 (Partial Compression Theorem) *Let B be a borrow-save bit array with the maximally redundant fraction range $f_i(B) \in (-1,1)$ for all i. Then the λ-level carry recoding $B^{[\lambda]}$ resulting from a sequence of λ P- or N- carry recodings in any order yields $f_i\left(B^{[\lambda]}\right) \in (a,b)$ for all i for some a and b with $b-a = 1+2^{-\lambda}$. This is best possible in that $f_i(B^{[\lambda]})$ is not restricted to any interval (a,b) where $b-a < 1+2^{-\lambda}$.*

Proof By induction the result holds for 0-level carry recoding $B^{[0]}$. Assume it holds for a $(\lambda-1)$-level carry recoding $B^{[\lambda-1]}$ so

$$f_i(B^{[\lambda-1]}) \in (a,b) \quad \text{with} \quad b-a = 1+2^{-(\lambda-1)}$$

Then, by lemma 1,

$$f_i(P(B^{[\lambda-1]})) = f_i(B^{[\lambda]}) \in (-\frac{1}{2}+\frac{a}{2}, \frac{b}{2}) \quad \text{where} \quad \frac{b}{2}-\frac{a}{2}+\frac{1}{2} = 1+2^{-\lambda}$$

Similarly $f_i(P(B^{[\lambda-1]}))$ has a width $1+2^{-\lambda}$, proving the bound. The fact that this interval is minimum width requires more detail and is omitted for brevity.

□

It is straightforward to show that the Q-carry recoding on a maximally redundant carry save bit array with fraction range $f_i(C) \in [0,2)$ has $f_i(Q(C)) \in [-\frac{1}{2},1)$. Thus the λ-level recoding $PNQ(C)$ will have the same fraction range as $PN^2(B)$ in the maximally redundant case with one exception that the lower bound is closed, *ie.* $[a,b)$ for $PNQ(C)$ and (a,b) for $PN^2(B)$.

Let the borrow save bit array B have *compression index* λ whenever $f_i(B) \in [a,b)$ with $b-a \leq 1+2^{-\lambda}$. We then obtain:

Corollary 1 *The λ-level carry recoding $B^{[\lambda]}$ of a borrow save bit array B and the λ-level carry recoding $B^{[\lambda-1]}(Q(C))$ of the carry save bit array C yields a borrow save bit array with compression index λ.*

Recall from Table 5 that a SD $2 \times k$ bit array can be recoded in-place to a BS $2 \times (k+1)$ bit array with fraction range $\left(-\frac{1}{2},1\right)$. Thus we may consider signed digit representation effectively as a form of borrow save representation with compression index one. In the following, we then restrict our focus to the carry save and borrow save formats.

The practical value of carry recodings is that just a few $(\lambda = 1,2,3)$ carry recodings provide partial compression sufficient to obtain nearly all the benefits of full compression while avoiding the high cost of a 2-1 compressor. The following observations are straightforward from the definition and support partial compression applications in rounding, leading insignificant digit deletion, and multiplier recoding.

Observation 5 *For a borrow save B or a carry save C bit array, truncating a low order part of the j-level carry recoding $PN^{j-1}(B)$ or $PN^{j-2}Q(C)$ effects a rounding with maximum error less than $\frac{1}{2} + 2^{-j}$ for any $j \geq 2$.*

Thus the 3-level rounding $PN^2(B)$ with fraction range $\left(-\frac{5}{8}, \frac{1}{2}\right)$ is sufficient to reduce rounding error below $\frac{5}{8}$ ulps. This can be quite useful in microcoded redundant binary designs for division, square root, and transcendentals. Partial compression also realizes virtually all the benefits of leading digit deletion.

Observation 6 *Let B be a borrow save $2 \times k$ bit array formed as the difference $a - b$ of two k-bit unsigned integers (see Table 4).*

(i) If $0 \leq a - b \leq 2^j - 1$, then the signed digit string $d_k d_{k-1} ... d_0$ determined by the carry recoding $N(B)$ must have $d_i = 0$ for all $i \geq j + 1$. Thus $N(B)$ may be truncated to a $2 \times (j+1)$ bit array.

(ii) If $|a - b| \leq 2^j - 1$, then the signed digit string $d_k d_{k-1} ... d_0$ determined by the 2-level carry recoding $PN(B)$ must have $d_i = 0$ for all $i \geq j + 2$. Thus $PN(B)$ may be truncated to a $2 \times (j+2)$ bit array.

Observation 6 has great applicability in extracting differences from a function table for performing interpolation [13]. Note that since case (ii) must provide for an effective sign bit, full compression would allow deletion of only one more leading digit than that provided by the partial compression for both cases (i) and (ii).

The following two observations support the factoring of multiplier recoding into a two step process. Partial compression is first applied to recode a redundant format so in a second step it may be passed through a standard Booth recoder.

Observation 7 *Let B be a borrow save bit array. The $l+1$ level recoding $PN^l(B)$ yields a borrow save bit array where any $2 \times l$ bit window has a value in the minimally redundant radix 2^l digit set $\{-2^{l-1}, -2^{l-1}+1, ..., 2^{l-1}\}$.*

Proof From Lemma 1, the fraction range of $PN^l(B)$ is $\left(-\frac{1}{2} + 2^{-(l+1)}, \frac{1}{2}\right)$. The value of any digit d' of a particular l bit window at a particular position j is given by

$$d' = f_{j+l}(B)2^l - f_j(B)$$

The integer d' for any j falls in the range

$$\begin{array}{ccccc} 2^l(-\frac{1}{2} + 2^{-(l+1)}) - \frac{1}{2} & < & d' & < & 2^l(\frac{1}{2}) + \frac{1}{2} - 2^{-(l+1)} \\ -2^{l-1} & \leq & d' & \leq & 2^{l-1} \end{array}$$

$\square$

Observation 8 *Let $PN^l(B)$ be the $l+1$ level recoding of the borrow save bit array B. Then the leading negative weight bit of any $2 \times l$ bit window of $PN^l(B)$ indicates the sign of the digit value of that window whenever that digit is non-zero.*

Table 6: Truth table of transformation cells

(a) P-cell

BS in	p_i	0	0	1	1
	n_i	0	1	0	1
BS out	p'_{i+1}	0	0	1	0
	n'_i	0	1	1	0
	Value	0	-1	1	0

(b) N-cell

BS in	p_i	0	0	1	1
	n_i	0	1	0	1
BS out	p'_i	0	1	1	0
	n'_{i+1}	0	1	0	0
	Value	0	-1	1	0

(c) Q-cell

CS in	p_i	0	0	1	1
	q_i	0	1	0	1
BS out	p'_{i+1}	0	1	1	1
	n'_i	0	1	1	0
	Value	0	1	1	2

Proof With $B' = N^l(B) = \begin{bmatrix} p'_{k-1} & p'_{k-2} & \cdots \\ n'_{k-1} & n'_{k-2} & \cdots \end{bmatrix}$ and $B'' = PN^l(B) = \begin{bmatrix} p''_k & p''_{k-1} & \cdots \\ 0 & n''_{k-1} & \cdots \end{bmatrix}$ we obtain any digit d' from the following equations. Extracting a term from the radix polynomial of B'' in $f_{j+l}(B'')$ allows us to continue.

$$\begin{aligned} d' &= f_{j+l}(B'')\,2^l - f_j(B'') = f\begin{pmatrix} p''_{j+l-1} & p''_{j+l-2} & \cdots \\ n''_{j+l-1} & n''_{j+l-2} & \cdots \end{pmatrix} 2^l - f_j(B'') \\ &= -n''_{j+l-1}2^{l-1} + f\begin{pmatrix} p''_{j+l-1} & p''_{j+l-2} & \cdots \\ 0 & n''_{j+l-2} & \cdots \end{pmatrix} 2^l - f_j(B'') \\ &= -n''_{j+l-1}2^{l-1} + f_{j+l-1}(B')\,2^{l-1} - f_j(B'') \end{aligned}$$

The observation falls from the fraction range of B' and B'', respectively equal to $\left(-2^{-l}, 1\right)$ and to $\left(-\frac{1}{2} - 2^{-(l+1)}, \frac{1}{2}\right)$.

□

4 Partial Compressors and Precoders

The implementation of a λ-level recoding involves only a logic depth of λ N- P- or Q- gates and is conveniently described by λ rows of transformed half adders obtained from the truth table of each cell (see Table 6).

4.1 Half-adder Logic

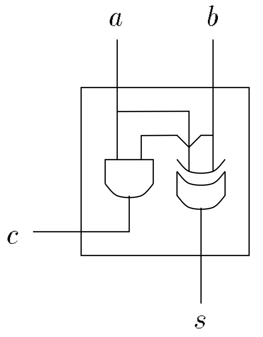
The half-adder cell is one of the basic cells of the computer arithmetic libraries. Functionally, it is implemented with an `exclusive or` gate and a `logical and` (see Fig 2-a). The arithmetic equation of the half-adder cell is given below with a, b, c and s all in $\{0, 1\}$.

$$a + b = 2c + s$$

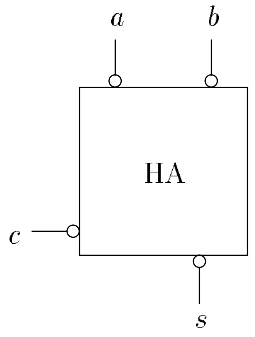
We can obtain the modified half-adder presented Figure 2-b from Moore's laws. Preferably, it is obtained from the arithmetic definition of the half-adder cell : a complemented value $\overline{a}$ acts as $(1 - a)$ and the equation are transformed as below.

$$\begin{aligned} (1-a) + (1-b) = 2(1-c) + (1-s) &\iff 2 - a - b = 3 - 2c - s \\ &\iff a + b + 1 = 2c + s \end{aligned}$$

In choosing the place of the inverters, we define the bit level cells corresponding to the P-, N- and Q- recodings (see Fig 3).

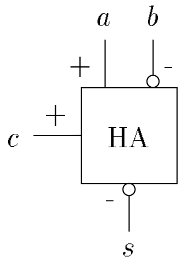


(a) Half Adder
$a + b = 2c + s$

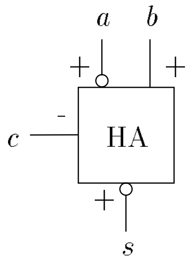


(b) Modified Half Adder
$a + b + 1 = 2c + s$

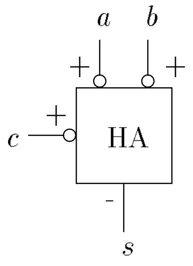
Figure 2: Half adders



(a) P-cell
$a - b = 2c - s$

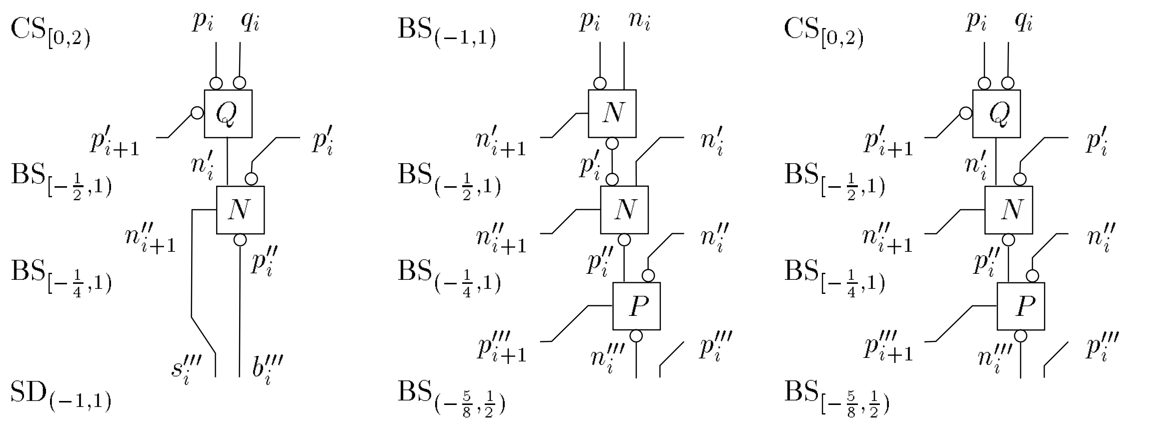


(b) N-cell
$a - b = -2c + s$



(c) Q-cell
$a + b = 2c - s$

Figure 3: Recoder cells



$\mathrm{CS}_{[0,2)}$
p_i
q_i
Q
p'_{i+1}
p'_i
n'_i
$\mathrm{BS}_{[-\frac{1}{2},1)}$
N
n''_{i+1}
p''_i
$\mathrm{BS}_{[-\frac{1}{4},1)}$
s'''_i
b'''_i
$\mathrm{SD}_{(-1,1)}$
$\mathrm{BS}_{(-1,1)}$
n_i
n'_{i+1}
$\mathrm{BS}_{(-\frac{1}{2},1)}$
$\mathrm{BS}_{(-\frac{1}{4},1)}$
P
p'''_{i+1}
n'''_i
p'''_i
$\mathrm{BS}_{(-\frac{5}{8},\frac{1}{2})}$
$\mathrm{BS}_{[-\frac{5}{8},\frac{1}{2})}$

(a) Translated NQ-recoding (b) PN^2-recoding (c) PNQ-recoding

Figure 4: Useful transformations

4.2 Precoder

Three useful recoder segments are presented Figure 4. To obtain the corresponding recoder, the desired segment is duplicated k according to the input. The digits on the first and the last row are applied an adapted transformation to avoid unnecessary generation of constant bits. This precoders allow transformation of CS to SD (Fig 4(a)) and BS or CS to radix 4 Booth recoding (Fig 4(b, c)) from observations 7. The format and the fraction range are indicated after each recoding.

With BiCMOS industry technology used in modern microprocessors, a row of modified half-adders delivers its output with a delay comprised between 100 ps and 200 ps depending on electrical properties. For all the cells compute a value and its complement in BiCMOS, the inverter do not had any additional logic or delay to the standard cells but merely consist on a wire reassignment.

4.3 Booth Recoder

Booth recoding involves the generation of radix-4 minimally redundant recoding of the multiplier $b = b_{k-1}b_{k-2}...$. The radix 4 recoded digit b'_j is best represented from three selection lines $p_{j,0}$, $p_{j,1}$ and $p_{j,2}$ (see Table 7) leading to the logic optimal specification of both Booth cells presented Figure 5. For each bit of the multiplicand $a = a_{k-1}a_{k-2}...$ the following operation are performed if necessary: complemented a_i to multiply by -1, shifted a_{i-1} in a_i to multiply by 2 or clear the result. The number obtained $a' = a'_{k+1}a'_k...$ represents the one's complement notation of $A \times b'j$.

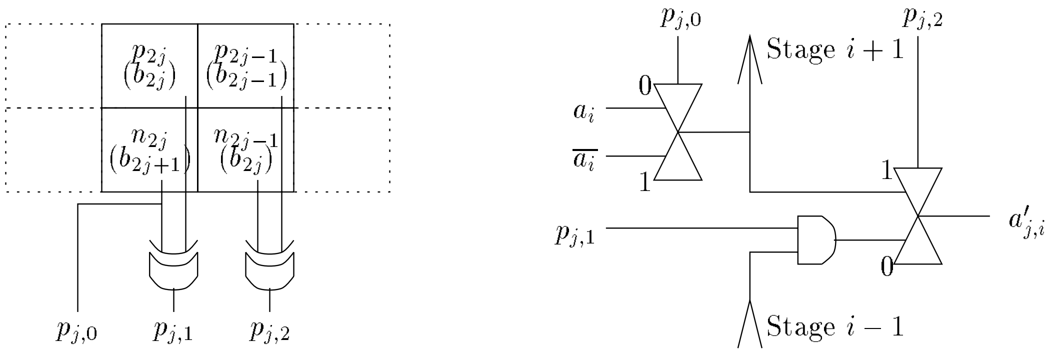
Due to replication, it is preferred in practical implementation to compute the following signals in the encoding part in order to reduce the number of transistors of the multiplexers.

$$\begin{aligned} \overline{s_{j,0}} &= p_{j,1} + p_{j,2} \\ s_{j,1} &= p_{j,1}.\overline{p_{j,2}} \end{aligned}$$

The Figure 5(a) presents the input of the encoding as a BS register. We show from observations 7 and 8 that the circuit used in usual implementation of Booth encoding works identically with a $PNQ(\cdot)$ precoded result or with the BS centered conversion of a UB bit vector. By the addition of a precoder that does not generate any delay to the non-redundant case, the Booth multiplier is modified to accept a redundant input in one of its operands (see Figure 6).

Table 7: Radix 4 Booth recoding

b_{2j+1}	b_{2j}	b_{2j-1}	b'_j	$p_{j,0}$	$p_{j,1}$	$p_{j,2}$
0	0	0	0	0	0	0
0	0	1	1	0	0	1
0	1	0	1	0	1	1
0	1	1	2	0	1	0
1	0	0	-2	1	1	0
1	0	1	-1	1	1	1
1	1	0	-1	1	0	1
1	1	1	0	1	0	0



(a) Encoding (b) Muliplexeurs

Figure 5: Logic optimal specification of Booth partial products

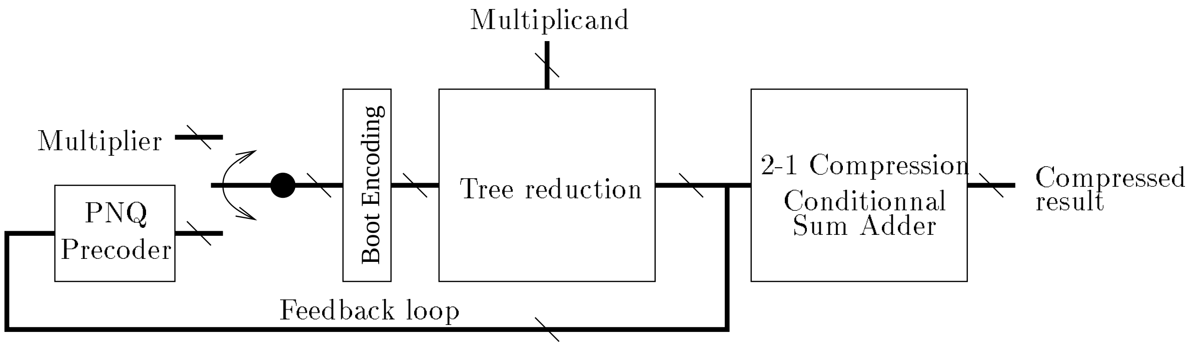


Figure 6: General purpose Booth multiplier with fast feedback capacities through a precoder

5 Conclusion

We have presented a general formalism for study of partial compression and roundings. This approach proves fruitful in reducing the redundancy of a borrow save or a carry save bit array to allow radix 2^k Booth recoding with minimal circuit. Reducing redundancy is useful in any application that do not allow any full range redundant number as input but do not require non redundant inputs. Other applications will certainly arise in the forwarding and feedback of number internal to an ALU.

References

[1] A. Avižienis, "Signed digit number representations for fast parallel arithmetic," *IRE Transaction on Electronic Computers*, vol. 10, pp. 389–400, 1961.

[2] D. W. Matula, *Applied Computation Theory: Analysis, Design, Modeling*, ch. Radix Arithmetic: Digital Algorithms for Computer Architecture, pp. 374–448. Prentice Hall, 1976.

[3] H. M. Darley *et al.*, "Floating point/integer processor with divide and square root functions," US Patent 4 878 190, US Patent Office, 1989.

[4] W. S. Briggs and D. W. Matula, "Rectangular array signed digit multiplier," US Patent 5 184 318, US Patent Office, 1993.

[5] P. Kornerup, "Digit-set conversion: Generalizations and applications," *IEEE Transactions on Computers*, vol. 43, no. 5, pp. 622–629, 1994.

[6] B. W. Y. Wei, H. Du, and H. Chen, "A complex number multiplier using radix 4 digits," in *Proceedings of the 12th Symposium on Computer Arithmetic* (S. Knowles and W. H. McAllister, eds.), (Bath, England), pp. 84–90, IEEE Computer Society Press, 1995.

[7] P. Chai *et al.*, "A 120 MFLOPS CMOS floating point processor," in *Proceedings of the 1991 Custom Integrated Circuits Conference*, (San Diego, California), pp. 15.1.1–15.1.4, IEEE Computer Society Press, 1991.

[8] H. Kabuo *et al.*, "Accurate rounding scheme for the newton raphson method using redundnat binary representation," *IEEE Transactions on Computers*, vol. 43, no. 1, pp. 43–51, 1994.

[9] S. M. Quek, L. Hu, J. P. Prabhu, and F. A. Ware, "Appartus for determiniung Booth recoder input control signals," US Patent 5 280 439, US Patent Office, 1994.

[10] N. Takagi, "Arithmetic unit based on a high speed multiplier with a redundant binary addition tree," in *Advanced Signal Processing Algorithms , Architectures and Implementation II*, vol. 1566 of *Proceedings of SPIE*, pp. 244–251, 1991.

[11] N. Takagi and S. Yajima, "A fast iterative multiplication method by recoding intermediate product," in *Proceedings of the 36th National Convention of Information Science*, (Kyoto, Japan), 1987.

[12] C. N. Lyu and D. W. Matula, "Redundant binary Booth recoding," in *Proceedings of the 12th Symposium on Computer Arithmetic* (S. Knowles and W. H. McAllister, eds.), (Bath, England), pp. 50–57, IEEE Computer Society Press, 1995.

[13] D. Das Sarma and D. W. Matula, "Hardware reciprocal table compression/decompression techniques," in *Scientific Computing and Validated Numerics* (G. Alefeld, A. Frommer, and B. Lang, eds.), (Wuppertal, Germany), pp. 11–17, Akademic Verlag, 1995.