# Partial Compression and Recoding:
## An Arithmetic Unit Design Tool

Marc Daumas* Guy Even† David W. Matula‡

**Abstract**

Various single logic level carry generating elementary recodings of carry save and/or borrow save numbers are shown to be functionally equivalent to the output of a 3-2 adder (subtractor) where one argument is zero. Range compression properties for 3-2 adders and for the redundant representation recodings are summarized and a key proof is outlined. Applications showing the value of preprocessing carry save and/or borrow save numbers by from one to multiple recodings are shown in the following areas [i] maximal to minimal higher radix digit recoding (re: Booth recoding), [ii] leading zero estimation for normalization in parallel with subtraction, [iii] increment absorption in rounding.

## 1 Introduction and Summary

Conversions between redundant representations have been investigated directly in the literature such as by digit set conversions [6] and/or digit recodings [1]. These recodings/conversions have important properties and constitute an important design tool. Often instances of such recodings are buried in more complex arithmetic unit implementations and their distinct contributions are not identified. Our first purpose in this paper is to present recodings in a uniform and familiar context of simply special cases of 3:2-adders (subtractors) where one argument is set to zero. In particular consider the range of values of a carry save encoded digit $\{0, 1, 2\}$ and a borrow save encoded digit $\{-1, 0, 1\}$. Now consider the range of a column sum of a 3-2 adder $\{0, 1, 2, 3\}$ or 3-2 "subtractor", $\{-1, 0, 1, 2\}$. In all cases such digit values may be recoded as a positive (or negative) value carry and a positive (or negative) place digit. This is the basis for the P-recoding (positive carry) and N-recoding (negative carry) developed in [1].

Redundant numbers in carry save or borrow save form are employed as intermediate quantities in arithmetic unit design, such as between 3-2 adders in a Wallace tree. A carry save number has a fractional part in ulps at any position that falls in the range $[0, 2)$, and for borrow save this fraction range is $(-1, 1)$. Recodings by generating a carry out in each position have been shown to reduce these fraction ranges uniformly to $[0, 1\frac{1}{2})$ for carry save and $(-1, 1/2)$ for borrow save. Such recodings can be implemented using only a single logic level (i.e. a half-adder). Our purpose in this paper is to show how such recodings may be inserted along key logic paths to obtain simplified arithmetic unit results for a variety of operations.

*CNRS, Ecole Normale Supérieure de Lyon, France. E-mail: Marc.Daumas@ENS-Lyon.Fr.
†Dept. of Electrical Engineering Systems, Tel-Aviv University, Israel. E-mail: guy@eng.tau.ac.il.
‡SEAS, Southern Methodist University, Dallas, Texas. E-mail: matula@seas.smu.edu.

In Section 2 we (a) review 3:2-adders and subtractors, define three types of 3:2 recodings that correspond to the computation performed by 3:2-adders; (b) define 2:2 recodings based on modified 3:2-adders in which one of the arguments is set to zero; (c) formalize the notion of measuring fraction ranges in ulps; and (d) quantify the reduction in fraction range (termed partial compression) that is achieved by recodings.

Our principal result in Section 2 is a unified description of six types of 2:2 recodings using 3:2-adders. The partial compression properties of 3:2-adders directly imply the partial compression properties of 2:2 recodings. Moreover, we show that if a partially compressed carry save or borrow save number is added to a standard binary number in a 3-2 adder (subtractor), the output remains uniformly partially compressed. For example, suppose that a carry save value has been partially compressed so any tail sequence has a value in the range $[0, 1\frac{1}{2})$ ulps. Subtracting a binary value from this carry save value in a 3-2 subtractor generating a borrow save result by positive carries and negative place bits yields a borrow save value with all tail sequences in the range $(-1, 3/4)$ rather than the full $(-1, 1)$ range.

In Section 3 we show three useful applications of carry recodings. The first treats maximal to minimal redundant higher radix digit recoding, providing a key step in obtaining Booth recoding of carry save output without requiring an intermediate 2-1 addition. The second application allows leading zero estimation to proceed in parallel with subtraction. The third application to increment absorption in rounding shows how these recodings provide an alternative proof to certain rounding refinement techniques in the literature.

# 2 Recodings and Partial Compression of Fraction Ranges

## 2.1 Representation of numbers

In this section we describe the representation we use for redundant numbers and the notation that is used in the paper.

We represent numbers by arrays of bits. Every row of such an array is a binary encoded digit string. Every row is assigned a positive or negative sign. The value represented by such an array equals the signed sum of the values represented by the rows of the array. Hence, such an array represents a radix two polynomial over a redundant digit set.

We focus on two examples:

1. A carry-save (CS) encoded digit string is represented by an array with 2 rows. Both rows are assigned a positive sign.

2. A borrow-save (BS) encoded digit string is represented by an array with 2 rows. The first row is assigned a positive sign and the second row is assigned a negative sign. Each digit of the borrow-save encoded digit string is represented by a column of the array; the value of a digit is the difference between the bit with the positive sign and the bit with the negative sign.

We denote the row of an array of bits by lower case letters. For example, a row with a positive sign is denoted by $\mathbf{a}^+ = a^+_{k-1}a^+_{k-2}\cdots a^+_0$ and a row with a negative sign is denoted by $\mathbf{a}^- =$

$a^-_{k-1} a^-_{k-2} \cdots a^-_0$. The value represented by a row $\mathbf{a}$ equals

$$\|\mathbf{a}\| = \sum_{i=0}^{k-1} a_i \cdot 2^i$$

regardless of the sign of the row. The value represented by an array with 2 rows: a positive signed row $\mathbf{a}^+$ and a negative signed row $\mathbf{a}^-$ equals $\|\mathbf{a}^+\| - \|\mathbf{a}^-\|$.

We often denote positive rows by $\mathbf{p}$ and $\mathbf{q}$, and then omit the plus sign in the superscript. Similarly, we denote negative rows by $\mathbf{n}$ and $\mathbf{m}$, and then omit the minus sign in the superscript.

## 2.2 3:2-adders

In this section we define the three types of 3:2-adders used in this paper.

A 3:2-adder receives as input a $3 \times k$ array of bits and outputs a $2 \times (k+1)$ array that represents the same value as the value represented by the array that is input.

We distinguish between three types of 3:2-adders:

1. A full-adder (FA). The input to a full adder consists of three rows $\mathbf{x}, \mathbf{y}, \mathbf{z}$ with a positive sign. The output consists of two rows $\mathbf{s}, \mathbf{c}$ with positive sign. The outputs are defined by

$$\begin{aligned} s_i &= xor(x_i, y_i, z_i) \\ c_{i+1} &= \begin{cases} 1 & \text{if } x_i + y_i + z_i \geq 2 \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

   Note that if the columns of the output array are indexed from 0 to $k$, then $s_k = 0$ and $c_0 = 0$.

2. A PPM-adder. The input to a PPM-adder consists of two rows $\mathbf{p}^+, \mathbf{q}^+$ with a positive sign and one row $\mathbf{n}^-$ with a negative sign [1]. The output of a PPM-adder consists of two rows: $\mathbf{c}^+$ with a positive sign (a carry) and $\mathbf{s}^-$ with a negative sign (a sum). The outputs are defined by

$$\begin{aligned} s^-_i &= xor(p^+_i, q^+_i, n^-_i) \\ c^+_{i+1} &= \begin{cases} 1 & \text{if } p^+_i + q^+_i - n^-_i \geq 1 \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

   It is easy to verify that $2c^+_{i+1} - s^-_i = p^+_i + q^+_i - n^-_i$.

3. A MMP-adder. The input to an MMP-adder consists of two rows $\mathbf{n}^-, \mathbf{m}^-$ with a negative sign and one row $\mathbf{p}^+$ with a positive sign. The output of an MMP-adder consists of two rows: $\mathbf{c}^-$ with a negative sign (a negative carry) and $\mathbf{s}^+$ with a positive sign (a sum). The outputs are defined by

$$\begin{aligned} s^+_i &= xor(n^-_i, m^-_i, p^+_i) \\ c^-_{i+1} &= \begin{cases} 1 & \text{if } p^+_i - n^-_i - m^-_i \leq -1 \\ 0 & \text{otherwise} \end{cases} \end{aligned}$$

[1] The origin of the term PPM-adder is plus-plus-minus corresponding to the signs of the rows of the input.

It is easy to verify that $-2c^-_{i+1} + s^+_i = p^+_i - n^-_i - m^-_i$.

The 3:2-adders are depicted in Fig. 1; each adder is depicted by a "bit-diagram" and by a gate-level implementation. All three 3:2-adders have basically the latency of a full-adder. Note that one can also define a 3:2-adder that is fed three negative rows and outputs two negative rows. Such an adder would be functionally identical to a FA line just as PPM-cells and MMP-cells are identical. The reason that PPM-cells and MMP-cells are identical can be explained in general by applying a duality principle to describe the functionality and implementation of a 3:2-adders based on the number of positive and negative rows in the input.

### 2.3 Recodings via modified 3:2-adders

In this section we present a unified and simple description of recodings by considering modified 3:2-adders. We present $P, N, Q$-recodings from [1] and $HA$ recodings as well as new recodings called $R$ and $Q'$. We show how these recoding can be derived from 3:2-adders.

We consider each type of 3:2-adder separately:

1. HA and R recodings derived from a FA 3:2-adder.

   (a) HA (half-adder) recoding is obtained by feeding a FA 3:2-adder with an array in which the third row is set to all zeros. Formally,

   $$HA(\mathbf{x}, \mathbf{y}) = FA(\mathbf{x}, \mathbf{y}, 0^k)$$

   (b) In R recoding, the rows of the bit array are regarded as a 2's complement representation. Therefore, if a row of the array is all ones then it represents the value $-1$. An alternative way to add the value zero to $\mathbf{x}$ and $\mathbf{y}$ is to add a row of all ones (representing the value $-1$) and then use the empty slot of $c_0$ to add an additional ones. Hence, R recoding is defined by,

   $$R(\mathbf{x}, \mathbf{y}) = FA(\mathbf{x}, \mathbf{y}, 1^k) \quad \text{and } c_0 = 1.$$

2. P and Q recodings are derived from a PPM 3:2-adder.

   (a) P recoding is obtained by feeding a PPM 3:2-adder with an array in which one of the positive rows is set to all zeros. Formally,

   $$P(\mathbf{p}, \mathbf{n}) = PPM(\mathbf{p}, 0^k, \mathbf{n})$$

   (b) Q recoding is obtained by feeding a PPM 3:2-adder with an array in which the negative row is set to all zeros. Formally,

   $$Q(\mathbf{p}, \mathbf{q}) = PPM(\mathbf{p}, \mathbf{q}, 0^k)$$

3. N and Q' recodings are derived from an MMP 3:2-adder.

   (a) N recoding is obtained by feeding an MMP 3:2-adder with an array in which one of the negative rows is set to all zeros. Formally,

   $$P(\mathbf{n}, \mathbf{p}) = MMP(\mathbf{p}, 0^k, \mathbf{n})$$

Table 1: Summary of recodings and their properties (CS means Carry-Save, BS means Borrow-Save, and MM means two bits with negative signs)

| Recoding | Format | | Definition | Compression |
|---|---|---|---|---|
| | In | Out | | |
| HA | CS | CS | $\mathrm{FA}(\mathbf{x}, \mathbf{y}, 0^k)$ | $(\frac{\alpha}{2}, \frac{\beta}{2} + \frac{1}{2})$ |
| R | CS | CS | $\mathrm{FA}(\mathbf{x}, \mathbf{y}, 1^k)$ | $(\frac{\alpha}{2} + \frac{1}{2}, \frac{\beta}{2} + 1)$ |
| P | BS | BS | $\mathrm{PPM}(\mathbf{p}, 0^k, \mathbf{n})$ | $(-\frac{1}{2} + \frac{\alpha}{2}, \frac{\beta}{2})$ |
| Q | CS | BS | $\mathrm{PPM}(\mathbf{p}, \mathbf{q}, 0^k)$ | $(-\frac{1}{2} + \frac{\alpha}{2}, \frac{\beta}{2})$ |
| N | BS | BS | $\mathrm{MMP}(\mathbf{p}, 0^k, \mathbf{n})$ | $(\frac{\alpha}{2}, \frac{\beta}{2} + \frac{1}{2})$ |
| Q' | MM | BS | $\mathrm{MMP}(0^k, \mathbf{n}, \mathbf{m})$ | $(\frac{\alpha}{2}, \frac{\beta}{2} + \frac{1}{2})$ |

(b) Q' recoding is obtained by feeding a PPM 3:2-adder with an array in which the positive row is set to all zeros. Formally,

$$Q'(\mathbf{n}, \mathbf{m}) = MMP(0^k, \mathbf{n}, \mathbf{m})$$

The gate level description of a half-adder is simpler than that of a full-adder. In Fig. 2 implementations of the basic cells of the six types recoding based on half adders are suggested. Note that P-cells and N-cells are identical and that Q-cells and Q'-cells are identical; this follows from a duality principle that can be applied to PPM and MMP adders.

Table 1 summarizes the six types of recodings and presents for each recoding: the format of the input and the output, the definition in terms of a modified 3:2 adder, and the partial compression properties (described in Corollary 4).

## 2.4 Fraction range

In this section we define the fraction range of a radix polynomial. This is a key definition which enables quantifying the amount of redundancy employed in a redundant representation of a number. We later show that recodings partially compress redundant representations in the sense that they reduce the fraction range.

**Definition 1** *Let* $\mathbf{a} = a_{k-1}a_{k-2}\cdots a_0$ *denote the coefficients of a radix* 2 *polynomial. The* fraction value at position $j$, *denoted* $f_j(\mathbf{a})$, *is defined by*

$$\begin{aligned} f_j(a_{k-1}a_{k-2}\cdots a_0) &= \|0.a_{j-1}a_{j-2}\cdots a_0\| \\ &= a_{j-1}2^{-1} + a_{j-2}2^{-2} + \cdots + a_0 2^{-j} \end{aligned}$$

Recall that we represent redundant numbers by bit arrays. Such a bit array can be interpreted as a radix polynomial, and hence, the definition of the fraction value at position $j$ is well defined for such bit arrays. Note that one could also consider the fraction values for each row separately and then add the fraction values of the rows (taking into account the signs of the rows) to obtain the same fraction value.

**Definition 2** *The* fraction range *of a radix polynomial* $\mathbf{a}$ *is the interval* $[\alpha, \beta]$ *defined by*

$$\begin{aligned} \alpha &= \min_j f_j(a_{k-1}a_{k-2}\cdots a_0) \\ \beta &= \max_j f_j(a_{k-1}a_{k-2}\cdots a_0) \end{aligned}$$

We usually enclose the fraction range within an interval rather than actually measure the minimum and maximum fraction values. We provide some important examples: (a) The fraction range of every carry-save encoded digit string is included in the interval $[0, 2)$. (b) The fraction range of every borrow-save encoded digit string is included in the interval $(-1, 1)$. (c) The fraction range of every $b \times k$ bit array with $\alpha > 0$ rows with a negative sign and $\beta > 0$ rows with a positive sign is included in the interval $(-\alpha, \beta)$.

The width of the fraction range serves as a quantitative measure of the actual redundancy employed in representing the number. The width of the fraction range of a non-redundant binary representation of a number is less than 1. The width of the fraction range of a borrow-save encoded digit string is less than 2. A reduction of the width of the fraction range of a borrow-save encoded digit string below 2 is viewed as a *partial compression* of the digit string.

## 2.5 Partial Compression Theory

In this section we prove the partial compression properties of 3:2-adders and apply them to recodings which are modified 3:2-adders. The partial compression properties of 4:2-adders are presented as well. These results are a generalization of the partial compression properties proved by Daumas and Matula [1] on P,N and Q recodings and the partial compression properties of 3:2-adders and 4:2-adders stated in [8].

**Theorem 3 (Compression of 3:2-adders)**

1. *Let* $a_i = x_i + y_i + z_i$ *and* $b_i = c_i + s_i$*, where* $(\mathbf{c}, \mathbf{s}) = FA(\mathbf{x}, \mathbf{y}, \mathbf{z})$*. If the fraction range of* $\mathbf{a}$ *is in* $(\alpha, \beta)$*, then the fraction range of* $\mathbf{b}$ *is in* $(\frac{\alpha}{2}, \frac{1}{2} + \frac{\beta}{2})$*.*
2. *Let* $a_i = p_i + q_i - n_i$ *and* $b_i = c_i - s_i$*, where* $(\mathbf{c}, \mathbf{s}) = PPM(\mathbf{p}, \mathbf{q}, \mathbf{n})$*. If the fraction range of* $\mathbf{a}$ *is in* $(\alpha, \beta)$*, then the fraction range of* $\mathbf{b}$ *is in* $(-\frac{1}{2} + \frac{\alpha}{2}, \frac{\beta}{2})$*.*
3. *Let* $a_i = p_i - n_i - m_i$ *and* $b_i = -c_i + s_i$*, where* $(\mathbf{c}, \mathbf{s}) = MMP(\mathbf{p}, \mathbf{n}, \mathbf{m})$*. If the fraction range of* $\mathbf{a}$ *is in* $(\alpha, \beta)$*, then the fraction range of* $\mathbf{b}$ *is in* $(\frac{\alpha}{2}, \frac{1}{2} + \frac{\beta}{2})$*.*

The proof of Theorem 3 is presented in the Appendix. Since the six types of recodings are obtained by restricting one of the rows that is fed to a 3:2-adder, the partial compression properties are carried over to the recodings as well. The following corollary summarizes the partial compression properties of the six types of recodings.

**Corollary 4 (Compression of P, N, Q, R, HA recoders)**

1. *Let $a_i = x_i + y_i$ and $b_i = c_i + s_i$. Assume that the fraction range of* $\mathbf{a}$ *is in* $(\alpha, \beta)$. *Then*
   (a) *If* $(\mathbf{c}, \mathbf{s}) = HA(\mathbf{x}, \mathbf{y})$, *then the fraction range of* $\mathbf{b}$ *is in* $(\frac{\alpha}{2}, \frac{1}{2} + \frac{\beta}{2})$.
   (b) *If* $(\mathbf{c}, \mathbf{s}) = R(\mathbf{x}, \mathbf{y})$, *then the fraction range of* $\mathbf{b}$ *is in* $(\frac{\alpha}{2} + \frac{1}{2}, \frac{\beta}{2} + 1)$.
2. *Let $a_i = p_i - n_i$, $a' = p_i + q_i$, and $b_i = c_i - s_i$. Assume that the fraction ranges of* $\mathbf{a}$ *and* $\mathbf{a}'$ *are in* $(\alpha, \beta)$. *Then*
   (a) *If* $(\mathbf{c}, \mathbf{s}) = P(\mathbf{p}, \mathbf{n})$, *then the fraction range of* $\mathbf{b}$ *is in* $(-\frac{1}{2} + \frac{\alpha}{2}, +\frac{\beta}{2})$.
   (b) *If* $(\mathbf{c}, \mathbf{s}) = Q(\mathbf{p}, \mathbf{q})$, *then the fraction range of* $\mathbf{b}$ *is in* $(-\frac{1}{2} + \frac{\alpha}{2}, +\frac{\beta}{2})$.
3. *Let $a_i = p_i - n_i$, $a' = -n_i - m_i$, and $b_i = c_i - s_i$. Assume that the fraction ranges of* $\mathbf{a}$ *and* $\mathbf{a}'$ *are in* $(\alpha, \beta)$. *Then*
   (a) *If* $(\mathbf{c}, \mathbf{s}) = N(\mathbf{p}, \mathbf{n})$, *then the fraction range of* $\mathbf{b}$ *is in* $(\frac{\alpha}{2}, \frac{\beta}{2} + \frac{1}{2})$.
   (b) *If* $(\mathbf{c}, \mathbf{s}) = Q'(\mathbf{n}, \mathbf{m})$, *then the fraction range of* $\mathbf{b}$ *is in* $(\frac{\alpha}{2}, \frac{\beta}{2} + \frac{1}{2})$.

Finally, we consider two type of 4:2 adders: (a) A 4:2-adder in which all the input rows have a positive sign; we denote such a 4:2-adder by 4TO2_ADD. A 4TO2_ADD adder can be implemented by two lines of FA 3:2 adders. (b) A 4:2-adder in which two input rows have a positive sign and two input rows have a negative sign; we denote such a 4:2-adder by BS_ADD. A BS_ADD adder can be implemented by a line of PPM 3:2-adders followed by a line of MMP 3:2 adders. The partial compression properties of these two 4:2-adders are summarized in the following corollary.

**Corollary 5 (Compression of 4 to 2 and BS adders)**

1. *Let $a_i = x_i + y_i + z_i + w_i$ and $b_i = c_i + s_i$. Assume that the fraction range of* $\mathbf{a}$ *is in* $(\alpha, \beta)$. *If* $(\mathbf{c}, \mathbf{s}) = 4TO2\_ADD(\mathbf{x}, \mathbf{y}, \mathbf{z}, \mathbf{w})$, *then the fraction range of* $\mathbf{b}$ *is in* $(\frac{\alpha}{4}, 1 + \frac{\beta}{4})$.
2. *Let $a_i = p_i + q_i - n_i - m_i$ and $b_i = c_i - s_i$. Assume that the fraction range of* $\mathbf{a}$ *is in* $(\alpha, \beta)$. *If* $(\mathbf{c}, \mathbf{s}) = BS\_ADD(\mathbf{p}, \mathbf{q}, \mathbf{n}, \mathbf{m})$, *then the fraction range of* $\mathbf{b}$ *is in* $(-\frac{1}{2} + \frac{\alpha}{4}, \frac{1}{2} + \frac{\beta}{4})$.

# 3 Applications of Partial Compression

In this section we describe applications and examples in which recodings to obtain partial compression can and have been found useful in arithmetic unit design.

## 3.1 Converting a maximally redundant digit string to a minimally redundant digit string

In this application we utilize the partial compression properties of $P$ and $N$-recodings to recode a maximally redundant digit string into a minimally redundant one.

Consider an (infinite) borrow-save encoded string: $\mathbf{a} = \mathbf{a}^+ - \mathbf{a}^-$, namely $a_j = a_j^+ - a_j^-$, where $a_j^+, a_j^- \in \{0, 1\}$.

Let $A_j^k$ denote the block of digits: $a_{j+(k-1)}, \ldots, a_{j+1}, a_j$. Suppose that $\mathbf{a}$ is maximally redundant, namely:

$$\|A_j^k\| = \sum_{i=0}^{k-1} a_{j+i} \cdot 2^i \in [-(2^k - 1), 2^k - 1]$$

The following claim shows that $(\mathbf{b}^+, \mathbf{b}^-) = PN^k(\mathbf{a}^+, \mathbf{a}^-)$ constitutes a recoding of $\mathbf{a}$ that is minimally redundant in radix $2^k$ in a very strong sense: the $2^k$-radix digits that are guaranteed to be in the range $[-2^{k-1}, 2^{k-1}]$ can be obtained by partitioning the recoded string into blocks of $k$ consecutive borrow-save digits regardless of the choice of the offset of the block boundaries. Moreover the sign of a radix $2^k$ digit $B_j^k$ can be determined by $b_{j+k-1}$.

**Claim 6** *Let $(\mathbf{b}^+, \mathbf{b}^-) = PN^k(\mathbf{a}^+, \mathbf{a}^-)$ and let $b_j = b_j^+ - b_j^-$, then:*

1. *For every $j$: $\|B_j^k\| \in [-2^{k-1}, 2^{k-1}]$.*
2. *If $b_{j+k-1}^- = 1$, then $\|B_j^k\| \leq 0$ and if $b_{j+k-1}^- = 0$, then $\|B_j^k\| \geq 0$ and*

**Proof:** The fraction range of $N^k(\mathbf{a}^+, \mathbf{a}^-)$ equals $(-2^{-k}, 1)$, and the fraction range of $\mathbf{b}$ equals $(-\frac{1}{2} - 2^{-(k+1)}, \frac{1}{2})$. Consider a block $B_j^k$ followed in the right by a tail $t$, then

$$\begin{aligned} \|0.B_j^k t\| &\subseteq (-\frac{1}{2} - 2^{-(k+1)}, \frac{1}{2}) \\ \|0.t\| &\subseteq (-\frac{1}{2} - 2^{-(k+1)}, \frac{1}{2}) \end{aligned}$$

where $\|0.B_j^k t\|$ denotes the value represented by the binary string with a binary point followed the borrow save string $B_j^k$ and the borrow save digit string $t$. Therefore,

$$\|B_j^k\| = \|B_j^k.t\| - \|0.t\| \subseteq (-2^{k-1} - 1, 2^{k-1} + \frac{1}{2} + 2^{-(k+1)})$$

Since $\|B_j^k\|$ is an integer we conclude that

$$\|B_j^k\| \subseteq [-2^{k-1}, 2^{k-1}]$$

and the first part of the claim follows.

The second part of the claim is proved as follows: Consider the fraction range of $\|0.B_j^k t\|$. By plucking out $b_{j+k-1}^-$ and applying a reverse $P$ recoding, since we obtain an image of an $N^k$ recoding, we can conclude that

$$\frac{b_{j+k-1}^-}{2} + \|0.B_j^k t\| \subseteq (-2^{-k+1}, -\frac{1}{2})$$

Assume that $b_{j+k-1}^- = 1$. This implies that

$$\|B_j^k.t\| \subseteq (-2^{k-1} - \frac{1}{2}, 0)$$

Recall that

$$\|0.t\| \subseteq (-\frac{1}{2} - 2^{-(k+1)}, \frac{1}{2})$$

Therefore,

$$\|B_j^k\| = \|B_j^k.t\| - \|0.t\| \leq \frac{1}{2} + 2^{-(k+1)} < 1$$

Since $\|B_j^k\|$ is an integer it follows that $\|B_j^k\| \leq 0$. The case that $b^-_{j+k-1} = 0$ can be proved similarly. □

In [1] this technique was used to enable a direct feedback of a carry-save output of a multiplier back into the multiplier as a radix 4 Booth encoded multiplicand. The carry-save encoded product is first Q-recoded to a borrow-save encoded string. This borrow save encoded string, which is maximally redundant, is then $PN^2$-recoded so that blocks of two borrow-save digits are minimally redundant radix 4 digits, and can then be used as Booth digits in the multiplier.

### 3.2 Estimation of the number of leading zeros

Nielsen *et al.* [8] and later Seidel and Even [9] used the partial compression properties of PN-recoding to estimate the number of leading zeros in the binary representation of a difference of two binary numbers $a$ and $b$. In their setting, $a$ and $b$ were represented by binary strings, and the binary representation of the difference is not computed yet. The goal is to save latency by estimating the number of leading zeros in the binary representation of the difference while the difference is being computed so that the difference can be normalized right after the difference is computed.

The number of leading zeros in the difference $\|a\| - \|b\|$ is estimated using $PN$-recoding as follows. Let $\mathbf{d} = d_{i-1}d_{i-2}\dots d_0$ denote a borrow-save encoded string where $d_j = d_j^+ - d_j^-$, $d_j^+ = a_j$ and $d_j^- = b_j$. Let $\mathbf{f} = PN(\mathbf{d})$. Denote the digits of $\mathbf{f}$ by $f_if_{i-1}\dots f_0$. The following claim shows that a good estimate of the number of leading zeros in the binary representation of $\|a\| - \|b\|$ can be derived by counting the number of leading zeros in the binary string $\mathbf{x} = x_ix_{i-1}\dots x_0$, where $x_j = xor(f_j^+, f_j^-)$. Thus, an estimate of the number of leading zeros can be computed without computing the difference (in fact, this computation is performed in parallel with the subtraction).

**Claim 7** *Let* $\mathbf{x} = \text{bitwise-xor}(\mathbf{f})$, *where* $\mathbf{f} = PN(\mathbf{d})$. *Assume that* $\mathbf{x}$ *has* $k$ *leading zeros, namely,* $\mathbf{x} = 0^k1r$ *where* $r \in \{0,1\}^{i-k}$. *Let* $\sigma$ *denote the most significand non-zero digit of* $\mathbf{f}$. *Namely,* $\mathbf{f} = 0^k\sigma t$, *where* $\sigma \in \{-1,+1\}$ *and* $t \in \{-1,0,1\}^{i-k}$. *Let* $\delta = \|a\| - \|b\|$. *Then,*

1. *If* $\delta < 0$ *then* $\sigma = 1$ *and* $\|\sigma.t\| \in (\frac{1}{4}, 1)$.
2. *If* $\delta > 0$, *then* $\sigma = -1$ *and* $\|\sigma.t\| \in (-\frac{3}{2}, -\frac{1}{2})$.

**Proof:** Recall that $\mathbf{f} = PN(\mathbf{d})$, where $\|\mathbf{d}\| = \delta$. The fraction range of $\mathbf{f}$ is $(-\frac{3}{4}, \frac{1}{2})$. This implies that

$$\|0.t\| \text{ and } \|0.\sigma t\| \quad \in \quad (-\frac{3}{4}, \frac{1}{2})$$

Assume that $\sigma = 1$. Hence,

$$\begin{aligned} \|1.t\| &\in (-\frac{3}{2}, 1) \cap (1 - \frac{3}{4}, 1 + \frac{1}{2}) \\ &= (-\frac{1}{4}, 1) \end{aligned}$$

This completes the first part of the claim.

Assume that $\sigma = -1$. Hence,

$$\begin{aligned} \|-1.t\| &\in (-\frac{3}{2}, 1) \cap (-1 - \frac{3}{4}, -1 + \frac{1}{2}) \\ &= (-\frac{3}{2}, -\frac{1}{2}) \end{aligned}$$

This completes the proof of the claim. □

### 3.3 Increment absorption in rounding

We show how the partial compression properties of a $HA$-recoding can be used to prove that two successive increments do not generate a carry. This technique was used by Even and Seidel [3] when analyzing the rounding algorithm of Yu and Zyner [10].

Consider a carry save encoded digit string $\mathbf{a} = a_i a_{i-1} \dots a_0$, where $a_j \in \{0, 1, 2\}$ and $i \geq 2$. Let $\mathbf{b} = HA(\mathbf{a})$, and $\mathbf{b}' = b_i b_{i-1} \dots b_0$. Note that $b_j \in \{0, 1, 2\}$ and $b_0 \in \{0, 1\}$. Let $\mathbf{x} = x_{i+1} x_i \dots x_0$ denote the binary string that satisfies $\|x\| = \|\mathbf{b}'\| + 1$. We claim that $\|\mathbf{x}\| + 2 < 2^{i+2}$, namely, adding 1 or even 2 to $X$ will not generate a carry to position $i+2$.

This follows from the partial compression properties of HA-recoding. The fraction range of $\|0.b_i b_{i-1} \dots b_0\|$ is in $(0, \frac{3}{2})$. Therefore, the fraction range of $\|x_{i+1}.x_i x_{i-1} \dots x_0\|$ is in $(0, \frac{3}{2} + 2^{-(i+1)})$. Since $i \geq 2$, we can replace the upper bound of the fraction range by $(\frac{3}{2} + \frac{1}{8})$.

We now consider two cases: (a) If $x_{i+1} = 0$, then adding in an additional 1 or even 2 cannot generate a carry to position $i+2$, and we are done. (b) If $x_{i+1} = 1$, then $\|0.x_i x_{i-1}\| \dots \in (0, \frac{1}{2} + \frac{1}{8})$. Adding in an additional 1 or even 2 will increase the upper bound of the fraction range position by at most $\frac{1}{4}$, and hence a carry is not generated to position $i+2$, as required.

## References


[1] Daumas, M., Matula, D. W.: "Recoders for Partial Compression and Rounding"; Technical Report RR97-01, Ecole Normale Superieure de Lyon, LIP, available at http://www.ens-lyon.fr/LIP.

[2] Jean Duprat, Yvan Herreros, and Jean-Michel Muller, "Some Results about On-Line Computation of Functions"; Proc. 9th IEEE Symposium on Computer Arithmetic, pp. 112-118, September 1989.

[3] G. Even and P. M. Seidel, A comparison of three rounding algorithms for IEEE floating-point multiplication, TR EES1998-8, Aug., 1998. http://www.eng.tau.ac.il/Utils/reportlist/reports/repfram.html.

[4] Alain Guyot, Bertrand Hochet, and Jean-Michel Muller, "JANUS, an On-Line Multiplier/Divider for Manipulating Large Numbers"; Proc. 9th IEEE Symposium on Computer Arithmetic, pp. 106-111, September 1989.

[5] Israel Koren: "Computer Arithmetic Algorithms"; Prentice Hall, 1993.

[6] Peter Kornerup: "Digit-set conversion: generalizations and applications"; IEEE Transactions on Computers, 43(5), pp. 622-629, 1994.

[7] Matula, D. W., Nielsen, A. M.: "Pipelined Packet-Forwarding Floating Point: I. Foundations and a Rounder"; *13th IEEE Symposium on Computer Arithmetic*, Asilomar, California, July 6-9, 1997, pp. 140–147.

[8] A. M. Nielsen, D. W. Matula, C. N. Lyu, and G. Even, "Pipelined Packet-Forwarding Floating Point: II. An Adder", Proceedings of the 13th IEEE Symposium on Computer Arithmetic, 1997.

[9] P.M. Seidel and G. Even, "How many logic levels does floating-point addition require?" to appear International Conference on Computer Design (ICCD'98), 1998.

[10] R. K. Yu and G. B. Zyner, "167 MHz Radix-4 floating point multiplier, Proceedings 12th Symp. on Computer Arithmetic, 12:149-154, 1995.

**APPENDIX. Proof Outline for Theorem 3**

We present in this appendix a new technique to obtain results on compression. We prove part (2) of the theorem, but the proof can be easily modified to prove parts (1) and (3) on FA and MMP adders.

Recall that

$$\begin{aligned} 2c_{i+1} + s_i &= p_i + q_i - n_i \text{ for every } i \\ \|c\| + \|s\| &= \|p\| + \|q\| - \|n\| \end{aligned}$$

Recall that the strings $\mathbf{a}$ and $\mathbf{b}$ are defined by $a_i = p_i + q + i - n_i$ and $b_i = c_i - s_i$. Therefore,

$$\begin{aligned} f_j(\mathbf{b}) &= \|0.b_{j-1}b_{j-2}\ldots\| \\ &= \|0.a_{j-1}a_{j-2}\ldots\| - c_j \\ &= \frac{a_{j-1}}{2} + \frac{1}{2}\cdot\|0.a_{j-2}\ldots\| - c_j \\ &= \frac{1}{2}f_{j-1}(\mathbf{a}) - c_j + \frac{p_{j-1}+q_{j-1}-n_{j-1}}{2} \end{aligned}$$

We use exhaustive analysis on the value of $d_{j-1} = p_{j-1} + q_{j-1} - n_{j-1}$. We already know that $\frac{1}{2}f_{j-1}(\mathbf{a}) \in (\frac{a}{2}, \frac{b}{2})$.

- If $d_{j-1} = -1$ then $r_j = 0$ and $f_j(\mathbf{b}) \in (-\frac{1}{2} + \frac{a}{2}, -\frac{1}{2} + \frac{b}{2})$.
- If $d_{j-1} = 0$ then $r_j = 0$ and $f_j(\mathbf{b}) \in (\frac{a}{2}, \frac{b}{2})$.
- If $d_{j-1} = 1$ then $r_j = 1$ and $f_j(\mathbf{b}) \in (-\frac{1}{2} + \frac{a}{2}, -\frac{1}{2} + \frac{b}{2})$.
- If $d_{j-1} = 2$ then $r_j = 1$ and $f_j(\mathbf{b}) \in (\frac{a}{2}, \frac{b}{2})$.

It follows that the fraction range of $\mathbf{b}$ is in $(-\frac{1}{2} + \frac{a}{2}, \frac{b}{2})$. Q.E.D.

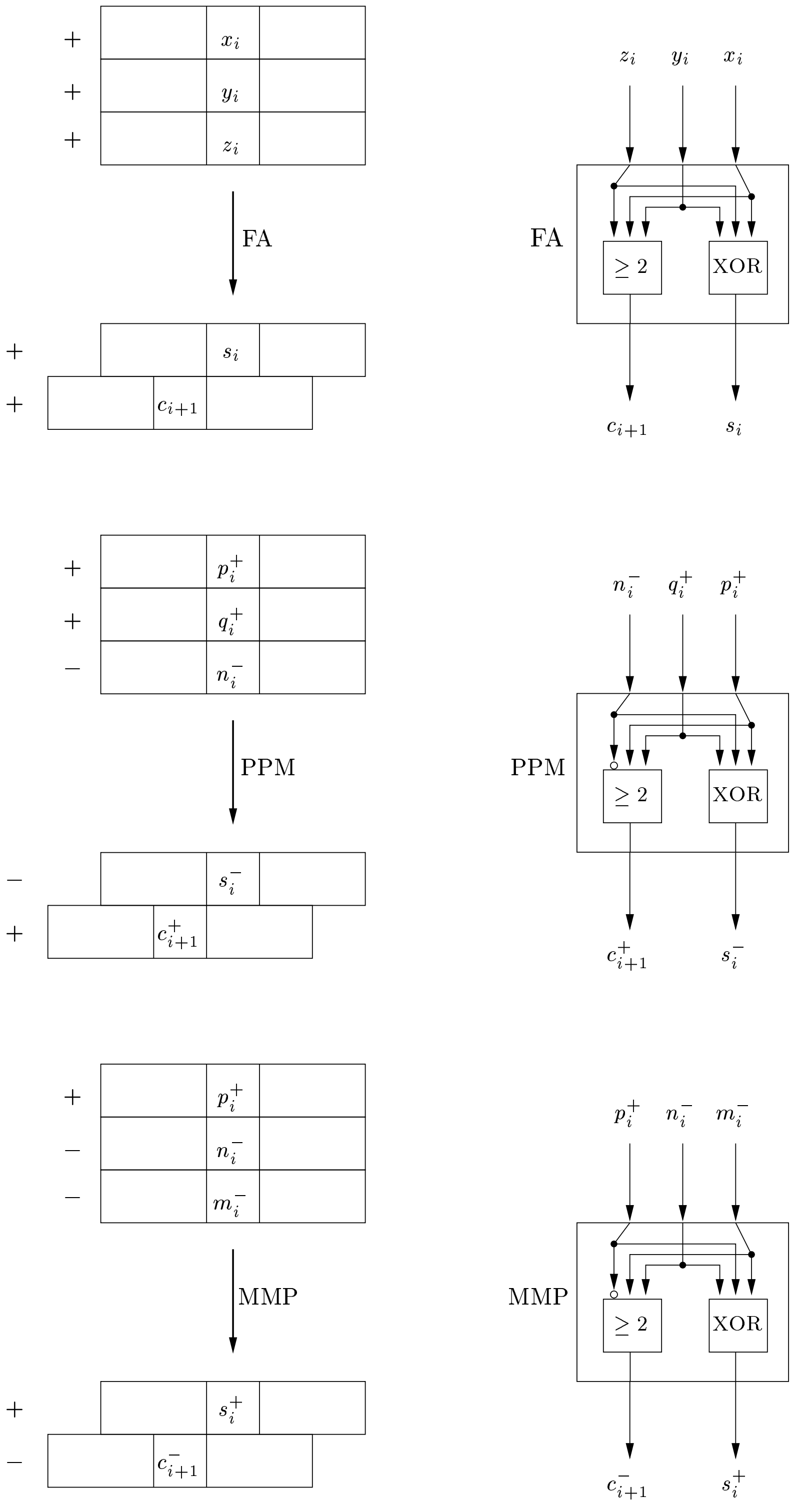


Figure 1: Bit-diagrams and gate-level descriptions of 3:2-adders

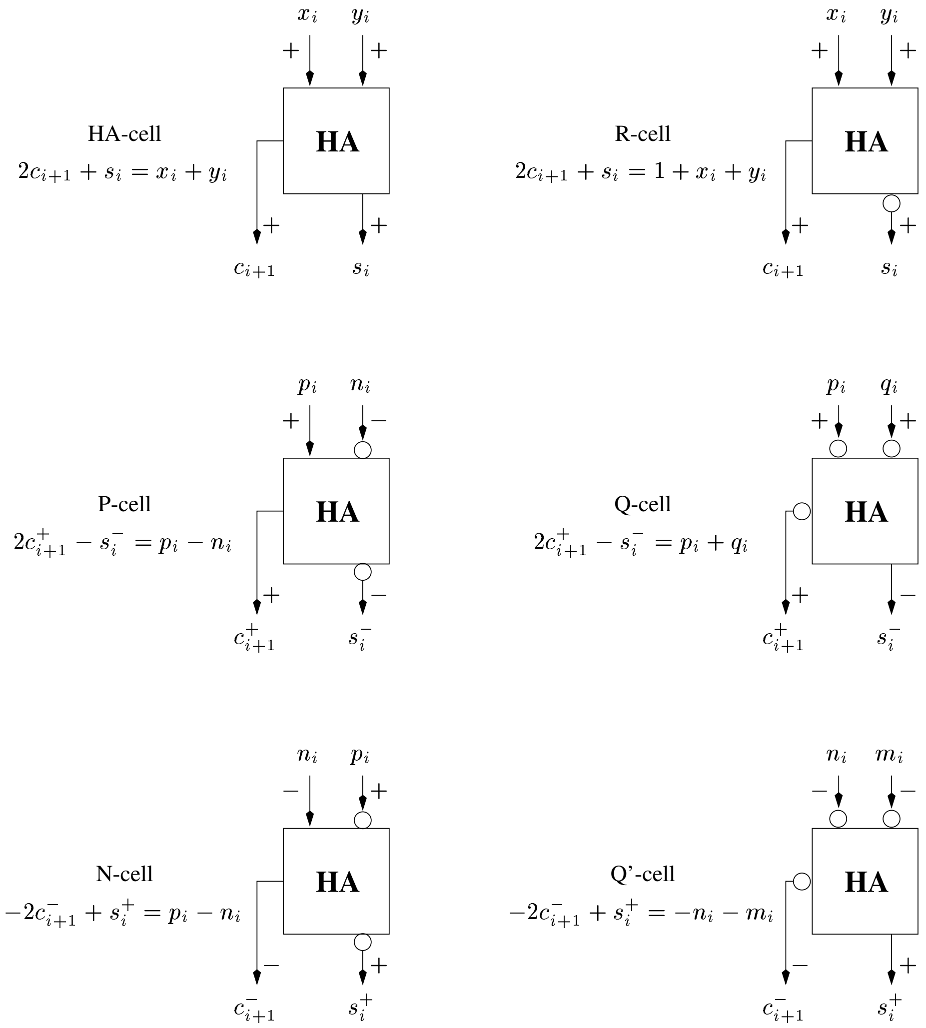


Figure 2: Gate-level descriptions of recodings