# On Recoding in Arithmetic Algorithms

Miloš D. Ercegovac

Computer Science Department
University of California
Los Angeles, CA 90024

Tomás Lang

Dept. of Electrical and Computer Engineering
University of California
Irvine, CA 92717

## Abstract

Recoding (a transformation between digit sets and/or radices) plays a central role in the design of efficient arithmetic algorithms. A suitable recoding reduces both the cost and delay of implementation of arithmetic operations. We present a simplified basis for developing recodings which are useful in digit-recurrence and parallel algorithms for multiplication and division/square-root.

## 1 Introduction

Recoding of operands in arithmetic algorithms is a common practice. Early work on recoding includes the classic Booth's radix-2 multiplier recoding algorithm [2] and studies in [7, 12, 13, 9, 8]. The recoding of multipliers has been used in most processors, in the past as well as nowdays. Recoding of the multiplier into a higher radix is a well known technique for reducing the number of partial products in multiplication and thus reducing the time in sequential multipliers and time and cost in parallel (array) multipliers. The process of recoding has been often described as

- Extension of the Booth radix-2 recoding (often called "modified Booth recoding").
- String recoding, described with tables. For example, 360/91 multiplier recoded into radix-4 with the digit set {-2, -1, 0,1,2}.

The literature on this subject contains several papers concerned with proving correctness of the recoding algorithms [10, 14, 11]. The proofs presented are quite involved. Recently, [6] discusses problems arising from brute-force extension of Booth's algorithm to higher radices. In our opinion, if recoding is viewed as the transformation between radix-$r$ digit sets the rules of signed-digit arithmetic [1] apply, the proof is straightforward, and no anomalies take place. Here we present such a description of recoding as signed-digit conversion process. A generalization of digit set conversions is also discussed in [3, 5].

We begin with a general description of the recoding process and then apply it to obtain a representation with minimally redundant signed digits for the cases in which the operand is represented

- in the conventional 2's complement system.
- in carry-save in the 2's complement system.
- in radix-$r$ with maximally redundant signed-digit set.

## 2 General formulation of recoding

We consider the recoding function in which

- the operand $v$ has a radix-$r$ representation in the 2's complement system $(s, v_1, ..., v_m)$ such that $v = -s + \sum_{i=1}^{m} v_i r^{-i}$ and a digit set $V_{min} \le v_i \le V_{Max}$. Note that $s = 0$ for a sign-and-magnitude or signed-digit representation.
- the result $p$ has a signed-digit radix-$r$ representation $(p_0, ..., p_m)$ such that $p = \sum_{i=0}^{m} p_i r^{-i}$ and a digit set $P_{min} \le p_i \le P_{max}$.

The recoding process can be formulated by the following steps:

1. Compute a transfer digit $t_{i-1}$ and an intermediate value $w_i$ such that
$$v_i = r t_{i-1} + w_i \quad (1)$$
with $P_{min} - T_{min} \le w_i \le P_{max} - T_{max}$, where $T_{min} \le t_i \le T_{max}$.
2. Compute
$$p_i = w_i + t_i. \quad (2)$$

The design of the recoding algorithm consists then in finding the set of values of $t_i$ to satisfy the condition of Step 1. The following conditions have to be satisfied:

- Since $V_{max} \le r T_{max} + (P_{max} - T_{max})$ we obtain
$$T_{max} \ge \frac{V_{max} - P_{max}}{r - 1}$$
Similarly,
$$T_{min} \le \frac{V_{min} - P_{min}}{r - 1}$$



1058-6393/95 $4.00 

- If $|p_i| < r$ then two possible choices exist for $w_i$, namely ($v_i \ mod \ r$) and ($v_i \ mod \ r - r$). Consequently, to make Step 1 possible, either

$$v_i \ mod \ r \leq P_{max} - T_{max} \quad (3)$$

or

$$v_i \ mod \ r - r \geq P_{min} - T_{min} \quad (4)$$

If for some $v_i$ the last condition is not satisfied, $w_i$ cannot be selected independent of $t_i$. In such cases, it is necessary to have at least some partial information about $t_i$ to determine $w_i$. The recoding for carry-save representation, shown below, is an example of such a case.

After the recoding algorithm is designed as described, its implementation consists in coding the variables $w_i$ and $t_i$ on bit-vectors and designing the corresponding combinational network.

# 3 Recoding examples

## 3.1 Operand in conventional 2's complement system

Let us consider recoding of a binary, 2's complement operand $y$ to a radix $r = 2^k$ result $p$ with a minimally redundant digit set $D_{min} = \{-\frac{r}{2}, \ldots, -1, 0, 1, \ldots, \frac{r}{2}\}$.

Let

$$y = -y_0 + \sum_{i=1}^{n} y_i 2^{-i} \quad (5)$$

be represented by

$$y_0.y_1 y_2 \cdots y_n$$

where $y_i \in \{0, 1\}$, $1 \leq i \leq n$, and $y_0 \in \{-1, 0\}$. The result $p$ is

$$p = \sum_{i=0}^{m} p_i r^{-i} \quad p_i \in D_{min} \quad (6)$$

represented by

$$p_0.p_1 \cdots p_m$$

Before recoding the radix-2 bit-vector is extended with 0s to the right so that $n \bmod k = 0$. (In the case of an integer operand, sign is extended to the left to make $n$ divisible by $k$.) Partitioning into $k$-bit groups has an effect on the number of steps and position of the final product - these can lead to anomalies discussed in [6].

The radix-$r$ representation of the operand is

$$y = -y_0 + \sum_{i=1}^{m} v_i r^{-i} \quad (7)$$

or the digit-vector $y_0.v_1 v_2 \ldots v_m$.

In terms of the digit $v_i$ the recoding process is described by the following steps:

1. Compute a transfer digit $t_{i-1}$ and an intermediate value $w_i$ such that

   $$v_i = r t_{i-1} + w_i$$

   where $-r/2 \leq w_i \leq r/2 - 1$ and $t_i \in \{0, 1\}$.

   It can be shown that this is achieved for

   $$t_{i-1} = \begin{cases} 1 & \textbf{if} \quad v \geq r/2 \\ 0 & \textbf{otherwise} \end{cases}$$

2. Compute the final recoded digit as

   $$p_i = w_i + t_i$$

Now let us consider the implementation at the binary level. The radix-$r$ digit $v_i$ is represented by the bit-vector

$$(v_{i,k-1}, \ldots, v_{i,j} \ldots, v_{i,0})$$

such that

$$v_i = \sum_{j=0}^{k-1} v_{i,j} 2^j$$

and $v_{i,j} \in \{0, 1\}$. Note that these bits of $v_i$ correspond to the bits of $y$.

From the high-level algorithm given it can be seen that

$$t_{i-1} = v_{i,k-1}$$

and

$$w_i = v_i - r t_{i-1} = \sum_{j=0}^{k-1} v_{i,j} 2^j - r v_{i,k-1}$$

$$= -v_{i,k-1} 2^{k-1} + \sum_{j=0}^{k-2} v_{i,j} 2^j$$

Consequently, the bit vector $(v_{i,k-1}, \ldots, v_{i,0})$ represents $w_i$ in the 2's complement system.

This corresponds to the "splitting" step used in the totally-parallel signed-digit conversion [1].

To get the recoded digits $p_i's$, we perform the "adding" step of the signed-digit conversion:

$$p_i = w_i + v_{i+1,k-1}, \quad i = 1, \ldots, m, \quad v_{m+1,k-1} = 0 \quad (8)$$

To get the most significant digit, we perform

$$p_0 = y_0 + w_{1,k-1} \quad (9)$$

This recoding can be done in parallel for all digits. Figure 1 illustrates two recoding examples and Figure 2 describes a generic implementation.

Figure 3 illustrates a gate-level design of the radix-4 recoder and multiple generator.

| | | | | |
|---|---|---|---|---|
| $y(r=2)$ | 1 | 110 | 011 | 101 |
| $v_i(r=8)$ | 1 | 6 | 3 | 5 |
| $w_i$ | 11 | 110 | 011 | 101 |
| | $\overline{1}$ | $\overline{2}$ | 3 | $\overline{3}$ |
| $t_i = v_{i+1,k-1}$ | 1 | 0 | 1 | 0 |
| $p_i$ | 0 | 1111 | 0100 | 1101 |
| | 0 | $\overline{1}$ | 4 | $\overline{3}$ |
| $y(r=2)$ | 0 | 1001 | 1100 | 1010 |
| $v_i(r=16)$ | 0 | 9 | 12 | 10 |
| $w_i$ | 0 | $\overline{7}$ | $\overline{4}$ | $\overline{6}$ |
| $t_i$ | 1 | 1 | 1 | 0 |
| $p_i$ | 1 | $\overline{6}$ | $\overline{3}$ | $\overline{6}$ |

Figure 1: Recoding examples for $r = 8$ and $r = 16$.

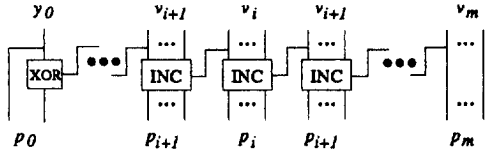


Figure 2: Implementation of recoding.

### 3.2 Operand in carry-save form

Let us now consider the case when the operand to be recoded is given in the standard carry-save form (sum and carry bit-vectors). This recoding can be used, for example, when the multiplier is obtained as a result of a previous addition or multiplication; this avoids the time-consuming carry-propagate step. In this case, the value of digit $i$ of the radix-2 representation is $y_i \in \{0, 1, 2\}$. The corresponding radix-$r$ digit $v_i$ is in the range $0 \leq v_i \leq 2(r-1)$.

As before, the first step in the recoding process consists in determining $t_{i-1}$ and $w_i$ such that

$$v_i = rt_{i-1} + w_i$$

and $-r/2 \leq w_i \leq r/2 - T_{max}$ (since $T_{min} = 0$).

Because of the range of $v_i$, this requires that $t_i \in \{0, 1, 2\}$. In this case, conditions (3,4) are not satisfied when $v_i$ has values $r/2 - 1$ and $(3/2)r - 1$ since the two possible values of $w_i$ are $r/2 - 1$ and $-(r/2 + 1)$, which are both outside of the allowable range. However, $w_i = r/2 - 1$ can be used when $t_i \neq 2$ and

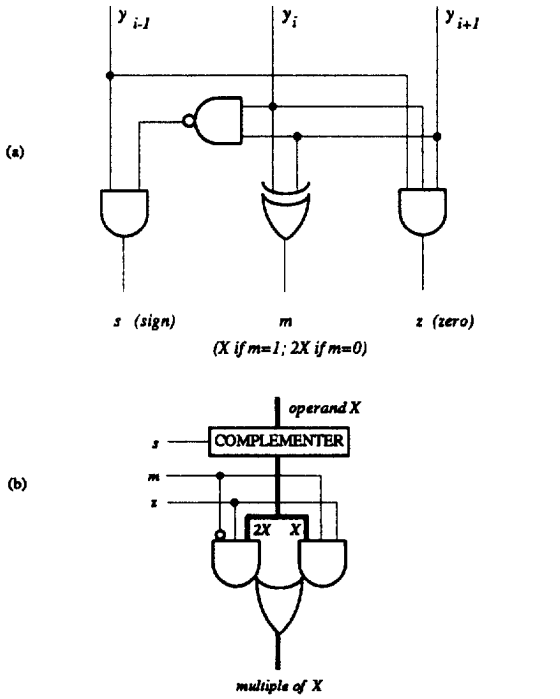


Figure 3: Implementation of radix-4 recoder (a) and multiple generator (b).

$w_i = -(r/2 + 1)$ when $t_i \neq 0$. Consequently, some knowledge from digit $i+1$ is required to obtain a suitable value of $w_i$. This can be achieved by splitting the transfer digit so that $t_i = g_i + h_i$, $0 \leq g_i, h_i \leq 1$ and making $h_{i-1}$ depend only on $v_i$, whereas $g_{i-1}$ depends both on $v_i$ and $h_i$. This is illustrated for a particular case below.

As a specific case, we consider that the output corresponds to a radix-4 digit vector with $-2 \leq p_i \leq 2$. Such a scheme has been applied to recoding of quotient digits in a high-radix division scheme [4].

Consider the $i-th$ radix-4 digit of the carry-save representation, as shown in Figure 4. Let us call $v_i$ the value of this digit so that $0 \leq v_i \leq 6$. To perform the recoding we define two transfer digits $h_i \in \{0, 1\}$ and $g_i \in \{0, 1\}$. Consequently, the recoded value is obtained as

$$p_i = v_i - 4(h_{i-1} + g_{i-1}) + h_i + g_i$$

The process is then

1. $w_i = v_i - 4(h_{i-1} + g_{i-1})$
2. $p_i = w_i + h_i + g_i$

| | | | |
|---|---|---|---|
| ... | $a$ | $b$ | ... |
| ... | $c$ | $d$ | ... |
| ... | | $v_i$ | ... |
| $h_{i-1}$ | | $h_i$ | ... |
| $g_{i-1}$ | | $g_i$ | ... |
| $w_{i-1}$ | | $w_i$ | ... |
| $p_{i-1}$ | | $p_i$ | ... |

Figure 4: Recoding of Carry-Save Form.

The $h$ and $g$ functions are selected so that $-2 \leq p_i \leq 2$. Moreover, to avoid a transfer-propagate chain we make $h_{i-1}$ dependent only on $v_i$, and $g_{i-1}$ dependent on $v_i$ and $h_i$, as shown in Figure 5.

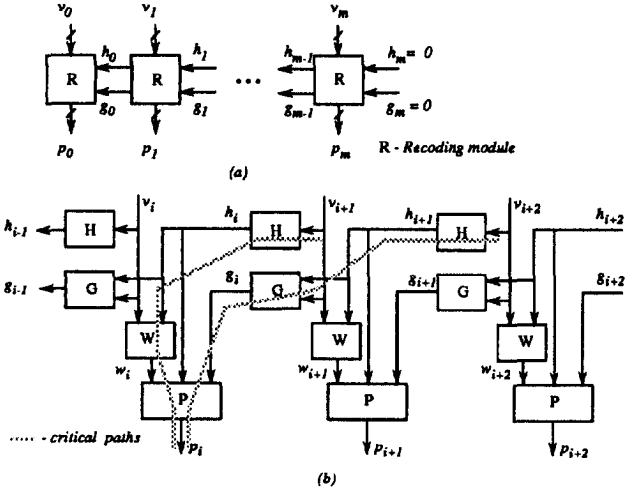


Figure 5: Recoding of in carry-save form into radix-4 digits: (a) Network. (b) Critical paths.

Table 1 describes possible functions $t_i$, $h_{i-1}$ and $g_{i-1}$ that result in the desired range of $p_i$ and prevent the carry-propagation chain.

We now give a specific implementation of this recoder. We encode $-3 \leq w_i \leq 1$ in the 5-bit vector $(wm3, wm2, wm1, w0, w1)$ so that $wk = 1$ when $w_i$ has the positive value $k$ and $wmk = 1$ when $w_i$ has the negative value $-k$. Switching expressions representing these functions are ($a, b, c, d$ are not indexed for notation simplicity);

$$\begin{aligned} \alpha &= a \oplus c \\ \beta &= b \oplus d \end{aligned}$$

Table 1: Recoding of Carry-Save Representation.

| $v_i$ | $h_i = 0$ | | | $h_i = 1$ | | |
|---|---|---|---|---|---|---|
| | $h_{i-1}$ | $g_{i-1}$ | $w_i$ | $h_{i-1}$ | $g_{i-1}$ | $w_i$ |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | $\overline{3}$ |
| 2 | 1 | 0 | $\overline{2}$ | 1 | 0 | $\overline{2}$ |
| 3 | 1 | 0 | $\overline{1}$ | 1 | 0 | $\overline{1}$ |
| 4 | 1 | 0 | 0 | 1 | 0 | 0 |
| 5 | 1 | 0 | 1 | 1 | 1 | $\overline{3}$ |
| 6 | 1 | 1 | $\overline{2}$ | 1 | 1 | $\overline{2}$ |

Table 2: Code for $p_i$

| $p_i$ | $pz$ | $ps$ | $p1$ |
|---|---|---|---|
| -2 | 0 | 0 | 0 |
| -1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 |
| 2 | 0 | 1 | 0 |

$$\begin{aligned} h_{i-1} &= a + c + bd \\ g_{i-1} &= abcd + \alpha'\beta h_i \\ wm3 &= \alpha'\beta h_i \\ wm2 &= abcd + \alpha b'd' + a'c'bd \\ wm1 &= \alpha\beta \\ w0 &= \alpha' b'd' + \alpha bd \\ w1 &= \alpha'\beta h_i' \end{aligned}$$

Note that for the left-most digit it is necessary to assure that $h_0 = g_0 = 0$. For this, the recoding is changed to $wm3 = wm2 = wm1 = 0$, $w0 = b'$ and $w1 = b$.

Finally, from (8) we obtain for $p_i$, represented in the code of Table 2,

$$\begin{aligned} pz &= w0h_i'g_i' + wm1(h_i \oplus g_i) + wm2h_ig_i \\ ps &= w1h_i'g_i' + (w0 + w1)(h_i \oplus g_i) \\ &\quad +(wm1 + w0)h_ig_i \\ p1 &= (wm1 + w1)h_i'g_i' + (wm2 + w0)(h_i \oplus g_i) \\ &\quad +(wm3 + wm1)h_ig_i \end{aligned}$$

These expressions can be implemented by MUXes using $h_i$ and $g_i$ as select inputs.

### 3.3 Operand in maximally redundant signed-digit form

We now describe the recoding from a maximally redundant signed-digit representation to a minimally

| $v_i$ | 0 | $\overline{3}$ | 1 | 2 | 3 |
|---|---|---|---|---|---|
| $w_i$ | 0 | 1 | 1 | $\overline{2}$ | $\overline{1}$ |
| $t_i$ | $\overline{1}$ | 0 | 1 | 1 | 0 |
| $p_i$ | $\overline{1}$ | 1 | 2 | $\overline{1}$ | $\overline{1}$ |

Figure 6: Recoding example for $r = 4$ signed digit.

redundant representation. From expressions (1), the range of the transfer digit is $-1 \leq t_i \leq 1$. Moreover, the pair $(t_{i-1}, w_i)$ is computed as

$$t_{i-1}, w_i = \begin{cases} -1, (v_i + r) & \text{if } v_i \leq -r/2 \\ 0, v_i & \text{if } |v_i| \leq r/2 - 1 \\ 1, (v_i - r) & \text{if } v_i \geq r/2 \end{cases}$$

Figure 6 illustrates a radix-4 recoding example and Figure 7 shows a generic implementation.

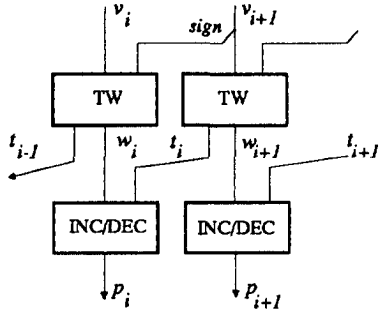


Figure 7: Implementation of recoding for signed-digit operand.

## References


[1] A.A. Avizienis, "Signed-Digit Number Representations for Fast Parallel Arithmetic," *IRE Trans. Electron. Comput.*, Vol. EC-10, no. 9, pp.389–400, Sept. 1961.

[2] A.D. Booth, "A Signed Binary Multiplication Technique," *Quartely J. Mechan. Appl. Math.*, Vol. IV, Part 2, pp.236-240, 1951.

[3] T. M. Carter and J.E. Robertson, "The Set Theory of Arithmetic Decomposition," *IEEE Transactions on Computers*, Vol.39, No.8, pp. 993–1005, August 1990.

[4] M.D. Ercegovac, T. Lang and P. Montuschi, "Very High Radix Division with Prescaling and Selection by Rounding," *Trans. on Computers*, Vol.43, No. 8, pp. 909–918, August 1994.

[5] P. Kornerup, "Digit Set Conversions: Generalizations and Applications," *IEEE Transactions on Computers* Vol.43, No.5, pp. 622–629, May 1994,

[6] P.E. Madrid, B. Millar, and E.E. Swartzlander, Jr., "Modified Booth Algorithm for High Radix Fixed-Point Multiplication," *IEEE Trans. on VLSI Systems*, Vol.1, No.2, pp. 164–167, June 1993.

[7] J.O. Penhollow, "A Study of Arithmetic Recoding with Applications to Multiplication adn Division," Ph.D. Dissertation, Department. of Computer Science, University of Illinois, Urbana, Report No. 128, September 1962.

[8] J.E. Robertson, "The Correspondence between Methods of Digital Division and Multiplier Recoding Procedures," *IEEE Transactions on Computers*, Vol.C-19, pp.692–701, 1970.

[9] F.A. Rohatsch, "A Study of Transformations Applicable to the Development of Limited Carry-Borrow Propagation Adders," Ph.D. Dissertation, Department of Computer Science, University of Illinois, Urbana, Report No. 226, June 1, 1967.

[10] L.P. Rubinfield, "A Proof of Modified Booth's Algorithm for Multiplication," *IEEE Transactions on Computers*, pp. 1014–1015, October 1975.

[11] H. Sam and A. Gupta, "A Generalized Multibit Recoding of Two's Complement Binary Numbers and Its Proof with Application in Multiplier Implementations," *IEEE Transactions on Computers*, Vol.39, No.8, pp. 1006–1015, August 1990.

[12] O.L. MacSorley, "High Speed Arithmetic in Binary Computers," *Proc. IRE*, January 1961.

[13] G. Reitwiesner, "Binary Arithmetic," in *Advances in Computers*, Vol.1, NY: Academic Press, pp. 232–308, 1960.

[14] S. Vassiliadis, E.M. Schwarz, and D.J. Hanrahan, "A General Proof for Overlapped Multiple-Bit Scanning Multiplications," *IEEE Transactions on Computers*, February, 1989.