ON THE FLY CONVERSION OF REDUNDANT INTO CONVENTIONAL REPRESENTATIONS

Milos Ercegovac
Tomas Lang

August 1985
Report No. CSD-850026

On-the-Fly Conversion of Redundant into Conventional Representations

Miloš D. Ercegovac and Tomas Lang
UCLA Computer Science Department
3732 Boelter Hall
University of California, Los Angeles

Abstract

An algorithm to convert redundant number representations into conventional representations is presented. The algorithm is performed concurrently with the digit-by-digit generation of redundant forms by schemes such as SRT division. It has a step delay roughly equivalent to delay of a carry-save adder and simple implementation. The conversion scheme is applicable in arithmetic algorithms such as nonrestoring division, square root, and on-line operations in which redundantly represented results are generated in a digit-by-digit manner, from most significant to least significant.

1. Introduction

We consider an algorithm and implementation to convert a signed integer from a redundant (signed digit) into a conventional range-complement representation. Such a conversion is necessary, for example, in SRT division or square root algorithms that produce this redundant result [1,2,3,4,5], or to convert results produced by on-line algorithms into the equivalent conventional forms [6,7,8]. The standard approach for the conversion is to separate the digit vector into two, one formed by the positive digits and the other by the negative ones, and add them in a carry-propagate adder. However, this adds the delay of the adder to the total operation time. To reduce this delay we propose here an algorithm that has the following characteristics:

• it performs the conversion on the fly, as the digits of the result (e.g., of the division or square root) are obtained in a serial fashion from most to least significant, and

• it has a delay which is compatible with one step of a fast division algorithm [4]. This delay is roughly of a carry-save adder.

We begin with a description of the algorithm for a radix-2 representation and then generalize to radix r.

2.Radix-2 Conversion

We want to convert the redundant digit-vector P (with p_i ε {-1,0,1}) representing the normalized fraction p into the digit-vector Q (with q_i ε {0,1}). That is,

$$p = \sum_{i=1}^{m} p_i 2^{-i}, \quad p_i \, \varepsilon \, \{-1,0,1\}$$

is to be converted into its conventional 2's complement form

$$q = -q_0 + \sum_{i=1}^{m} q_i 2^{-i}, \quad q_i \, \varepsilon \, \{0,1\}$$

As mentioned before, the transformation is to be performed as the digits of P are produced, from most significant to least significant. A simple algorithm would be to form $q[k]$ such that

$$q[k] = q[k-1] + p_k 2^{-k}$$

which results in $q = q[m]$.

However, this algorithm requires the propagation of a carry when $p_k = -1$. The propagation of this carry is up to the previous bit of Q having value 1. That is, if for example,

$$q_i, q_{i+1}, \ldots, q_{k-2}, q_{k-1} = 1,0\ldots0,0$$

then after adding -2^{-k} we get

$$q_i, q_{i+1}, \ldots, q_{k-2}, q_{k-1}, q_k = 0,1,\ldots,1,1,1$$

The approach we follow is to avoid the propagation of the carry by keeping two conditional forms, one ($A[k-1]$) expecting $p_k = 1$ or $p_k = 0$ and the other ($B[k-1]$) expecting $p_k = -1$, and selecting the correct one when p_k is known.

That is,

$$q[k] = \begin{cases} A[k-1] + 2^{-k} & \text{if } p_k=1 \\ A[k-1] & \text{if } p_k=0 \\ B[k-1] + 2^{-k} & \text{if } p_k=-1 \end{cases} \tag{1}$$

To obtain this result it is necessary that

$$A[k-1] = -q_0 + \sum_{i=1}^{k-1} q_i 2^{-i} = q[k-1] \tag{2}$$

and

$$B[k-1] = A[k-1] - 2^{-(k-1)} \qquad (3)$$

The final result is obtained as

$$q = A[m]$$

We now present the recurrence to compute $A[k]$ and $B[k]$ forms. Since p is normalized, the initial conditions are

$$A[1] = \begin{cases} +1/2 & (0.1) \text{ if } p>0 \quad (p_1=+1) \\ -1/2 & (1.1) \text{ if } p<0 \quad (p_1=-1) \end{cases}$$

$$B[1] = \begin{cases} +0 & (0.0) \text{ if } p>0 \quad (p_1=+1) \\ -1 & (1.0) \text{ if } p<0 \quad (p_1=-1) \end{cases}$$

For $k > 1$

$$A[k+1] = \begin{cases} A[k] + 2^{-(k+1)} & \text{if } p_{k+1} = 1 \qquad (4.1) \\ A[k] & \text{if } p_{k+1} = 0 \qquad (4.2) \\ B[k] + 2^{-(k+1)} & \text{if } p_{k+1} = -1 \qquad (4.3) \end{cases}$$

$$B[k+1] = \begin{cases} A[k] & \text{if } p_{k+1} = 1 \qquad (4.4) \\ B[k] + 2^{-(k+1)} & \text{if } p_{k+1} = 0 \qquad (4.5) \\ B[k] & \text{if } p_{k+1} = -1 \qquad (4.6) \end{cases}$$

Note that none of these requires a propagation of carries.

Proof

By induction. We assume that (2) and (3) are true for k and show that they are satisfied for $k+1$.

Basis.

Both (2) and (3) are satisfied for $k=1$ by the initial conditions.

Induction step. We consider all possible values of p_{k+1}.

For $p_{k+1} = 1$, we have

$$\begin{aligned} A[k+1] &= A[k] + 2^{-(k+1)} \quad \text{by (4.1)} \\ &= q[k] + p_{k+1} \quad \text{by induction hypothesis} \\ &= q[k+1] \end{aligned}$$

and

$$\begin{aligned} B[k+1] &= A[k] \quad \text{by (4.4)} \\ &= A[k+1] - 2^{-(k+1)} \quad \text{by (4.1)} \end{aligned}$$

For $p_{k+1} = 0$ we have

$$\begin{aligned} A[k+1] &= A[k] \quad \text{by (4.2)} \\ &= q[k] + p_{k+1} \quad \text{by induction hypothesis} \\ &= q[k+1] \end{aligned}$$

and

$$\begin{aligned} B[k+1] &= B[k] + 2^{-(k+1)} \quad \text{by (4.5)} \\ &= A[k] - 2^{-k} + 2^{-(k+1)} \quad \text{by induction hypothesis} \\ &= A[k+1] - 2^{-(k+1)} \quad \text{by (4.2)} \end{aligned}$$

For $p_{k+1} = -1$

$$\begin{aligned} A[k+1] &= B[k] + 2^{-(k+1)} \quad \text{by (4.3)} \\ &= A[k] - 2^{-k} + 2^{-(k+1)} \quad \text{by induction hypothesis} \\ &= A[k] - 2^{-(k+1)} \\ &= q[k] + p_{k+1} \quad \text{by induction hypothesis} \\ &= q[k+1] \end{aligned}$$

and

$$B[k+1] = B[k] \quad \text{by } (4.6)$$
$$= A[k+1] - 2^{-(k+1)} \quad \text{by } (4.3)$$

Consequently, the recurrence satisfies the conditions of the conversion. □

As an example consider the conversion of $P = 0.1101\bar{1}00\bar{1}1010$ where $\bar{1} = -1$.

k	p_k	$A[k]$	$B[k]$	$q[k]$
0	0			
1	1	0.1	0.1	0.1
2	1	0.11	0.10	0.11
3	0	0.110	0.101	0.110
4	1	0.1101	0.1100	0.1101
5	-1	0.11001	0.11000	0.11001
6	0	0.110010	0.110001	0.110010
7	0	0.1100100	0.1100011	0.1100100
8	-1	0.11000111	0.11000110	0.11000111
9	1	0.110001111	0.110001110	0.110001111
10	0	0.1100011110	0.1100011101	0.1100011110
11	1	0.11000111101	0.11000111100	0.11000111101
12	0	0.110001111010	0.110001111001	0.110001111010

The converted fraction is $q = 0.110001111010$.

Similarly, for a negative case such as $P = 0.\bar{1}0\bar{1}100\bar{1}1$ the conversion is

k	p_k	$A[k]$	$B[k]$	$q[k]$
0	0			
1	-1	1.1	1.0	1.1
2	0	1.10	1.01	1.10
3	-1	1.011	1.010	1.011
4	1	1.0111	1.0110	1.0111
5	0	1.01110	1.01101	1.01110
6	0	1.011100	1.011011	1.011100
7	-1	1.0110111	1.0110110	1.0110111
8	1	1.01101111	1.01101110	1.01101111

The converted fraction is $q = 1.01101111$.

3. Implementation of the Radix-2 Conversion

The implementation of the algorithm requires two registers to contain $A[k]$ and $B[k]$, respectively, These registers can be shifted one bit left with insertion in the least significant bit depending on the value of p_k. They also require parallel loading to load $A[k]$ with $B[k]$ and vic-eversa. This implementation is shown in Figure 1.

4. Radix-r Conversion

We now generalize the previous algorithm to the radix-r case. That is, we want to convert

$$p = \sum_{i=1}^{m} p_i r^{-i}, \;\; p_i \;\varepsilon\; \{-a, ..., 0, ..., a\} \;\; r/2 \leq |a| \leq r-1$$

into

$$q = -q_0 + \sum_{i=1}^{m} q_i r^{-i}, \;\; q_i \;\varepsilon\; \{0, ..., r-1\}$$

To keep the characteristics of the algorithm, the expression (1) generalizes to

$$q[k] = \begin{cases} A[k-1] + p_k r^{-k} & \textbf{if } p_k \geq 0 \\ B[k-1] + (r - |p_k|) r^{-k} & \textbf{if } p_k < 0 \end{cases}$$

Consequently, to obtain this we need that

$$A[k-1] = q[k-1]$$

and

$$B[k-1] = A[k-1] - r^{-(k-1)}$$

The initial conditions extend to

$$A[1] = \begin{cases} +p_1 r^{-1} \quad (0.p_1) & \text{if } p>0 \\ -|p_1| r^{-1} \quad (1.(r-|p_1|)) & \text{if } p<0 \end{cases}$$

$$B[1] = \begin{cases} +(p_1-1) r^{-1} \quad (0.(p_1-1)^{*} & \text{if } p>0 \\ -(|p_1|+1) r^{-1} \quad (1.(r-1-|p_1|)) & \text{if } p<0 \end{cases}$$

* $p_1>0$ since p is normalized.

The recurrence presented for the radix-2 case is transformed as follows. For $k>1$

$$A[k+1] = \begin{cases} A[k] + p_{k+1} r^{-(k+1)} & \text{if } p_{k+1} \geq 0 \\ B[k] + (r-|p_{k+1}|) r^{-(k+1)} & \text{if } p_{k+1}<0 \end{cases}$$

$$B[k+1] = \begin{cases} A[k] + (p_{k+1}-1) r^{-(k+1)} & \text{if } p_{k+1}>0 \\ B[k] + ((r-1)-|p_{k+1}|) r^{-(k+1)} & \text{if } p_{k+1} \leq 0 \end{cases}$$

The proof follows the same scheme as that for the radix-2 case.

The implementation is also similar. It requires a one-digit adder to compute $r-|p_{k+1}|$.

5. Conversion from Radix-4 Redundant to Conventional 2's Complement

As a detailed example, we now apply the conversion algorithm to the radix-4 case, compatible with the radix-4 division scheme [4]. The conversion rules from a radix-4 redundant representation into the conventional binary 2's complement are specified in Table 1. The digit appended to bit-vector $A[k]$ is $a = (a_1, a_0)$ with value $2a_1 + a_0 \varepsilon \{0,1,2,3\}$. Similarly, $b = (b_1, b_0)$ is appended to bit-vector $B[k]$.

Table 1: Radix-4 Conversion

p_{k+1}	$A[k+1] = (A[k], a_1, a_0)$	$B[k+1] = (B[k], b_1, b_0)$
0	$(A[k],0,0)$	$(B[k],1,1)$
1	$(A[k],0,1)$	$(A[k],0,0)$
-1	$(B[k],1,1)$	$(B[k],1,0)$
2	$(A[k],1,0)$	$(A[k],0,1)$
-2	$(B[k],1,0)$	$(B[k],0,1)$
3	$(A[k],1,1)$	$(A[k],1,0)$
-3	$(B[k],0,1)$	$(B[k],0,0)$

The implementation consists of two left-shift registers (A and B) that contain $A[k]$ and $B[k]$ respectively. The operations on these registers are

$$A \leftarrow \begin{cases} \textit{Shift A with insert } (a_1, a_0) & \text{if } C_{SHA}=1 \\ \textit{Shift B with insert } (a_1, a_0) & \text{if } C_{LDA}=1 \end{cases}$$

$$B \leftarrow \begin{cases} \textit{Shift B with insert } (b_1, b_0) & \text{if } C_{SHB}=1 \\ \textit{Shift A with insert } (b_1, b_0) & \text{if } C_{LDB}=1 \end{cases}$$

By definition, $C_{LDA} = C_{SHA}'$ and $C_{LDB} = C_{SHB}'$.

For the minimally redundant case $p_i \varepsilon \{-2,-1,0,1,2\}$ the radix-4 digit values are represented by the following sign-and-magnitude code:

p_i	s	m_1	m_0
0	0	0	0
1	0	0	1
2	0	1	0
-1	1	0	1
-2	1	1	0

where s is the sign bit, and m_1 and m_0 are the magnitude bits.

Using Table 1 we determine the following switching expressions for the rightmost radix-4 digits $a = (a_1, a_0)$ and $b = (b_1, b_0)$ where $a, b \in \{0,1,2,3\}$:

$$a_1 = s + m_1$$

$$a_0 = m_0$$

$$b_1 = m_o' m_1' + s m_1$$

$$b_0 = m_1 + m_0'$$

The shift and load control signals for the registers are

$$C_{LDA} = s$$

$$C_{SHA} = s'$$

$$C_{LDB} = (s + m_0' m_1')' = C_{SHB}'$$

$$C_{SHB} = s + m_0' m_1'$$

6. Summary

An algorithm for converting redundant forms into range complement conventional forms concurrently with digit-by-digit generation is presented. The algorithm has a simple implementation and a delay independent of the working precision, roughly equal to two logic levels plus a register shift/load time. It is applicable in nonrestoring division and square algorithms, and in producing conventional results in on-line algorithms.

Acknowledgements. This research has been supported in part by the Contract No. N00014-85-K-0159 of the Office of Naval Research We thank Dr. J.G. Nash of Hughes Research Laboratories for his interest and support.

References

[1] J.E. Robertson, "A New Class of Digital Division Methods", *IRE Trans. on Electronic Computers,* Vol.EC-7, September 1958, pp.218-222.

[2] D.E. Atkins, "Higher-Radix Division Using Estimates of the Divisor and Partial Remainders", *IEEE Trans. Computers,* Vol.C-17, October 1968, pp.925-934.

[3] G. Metze, "Minimal Square Rooting", *IEEE Trans. Electronic Computers,* Vol.EC-14, February 1965, pp.181-185.

[4] M.D. Ercegovac and T. Lang, "A Division Algorithm with Predicition of Quotient Digits", *Proc. 7th IEEE Symposium on Computer Arithmetic,* Urbana, Illinois, 1985, pp.51-56.

[5] G.S. Taylor, "Radix 16 SRT Dividers with Overlapped Quotient Selection Stages", Proc. 7th IEEE Symposium on Computer Arithmetic, Urbana, Illinois, 1985, pp. 64-71.

[6] M.D. Ercegovac, "A General Hardware-Oriented Method for Evaluation of Functions and Computations in a Digital Computer", *IEEE Trans. Computers,* Vol.C-26, July 1977, pp.667-680.

[7] K.S. Trivedi and M.D. Ercegovac, "On-Line Algorithms for Division and Multiplication", *IEEE Trans. Computers,* Vol.C-26, July 1977, pp.667-680.

[8] M.D. Ercegovac, "On-Line Arithmetic: An Overview", *Proc. SPIE 1984,* Vol.495 - Real Time Signal Processing VII, 1984, pp.86-93.

Keywords Conversion, redundant number representation, 2's complement, most significant digit first algorithms, SRT division, square root, on-line algorithms.

Captions

Figure 1: Implementation

Table 1: Radix-4 Conversion

Authors' Address and Affiliation

Professor Miloš D. Ercegovac
UCLA Computer Science Department
3732 Boelter Hall
UCLA
Los Angeles, CA 90024
(213) 825-5414 or -2660

Professor Tomas Lang
UCLA Computer Science Department
3732 Boelter Hall
UCLA
Los Angeles, CA 90024
(213) 825-6835 or -2660

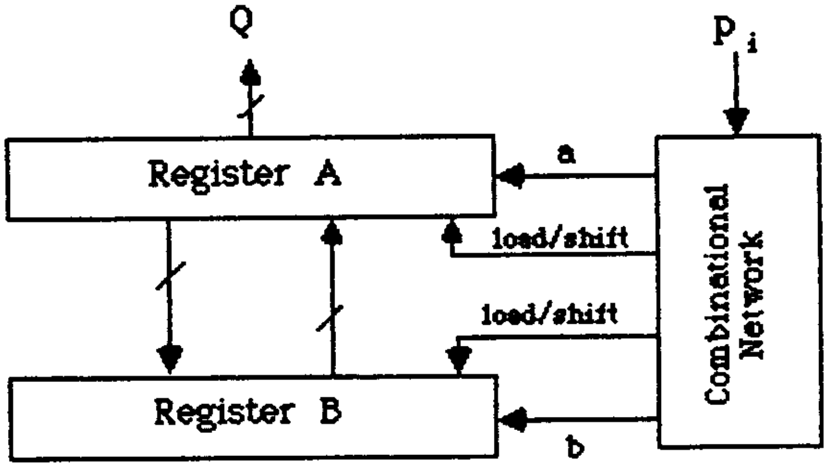

Figure 1. Implementation