SIMPLE RADIX-4 DIVISION WITH DIVISOR SCALING

Milos D. Ercegovac
Tomas Lang

March 1987
CSD-870015

Simple Radix-4 Division with Divisor Scaling

Miloš D. Ercegovac and Tomas Lang

Computer Science Department

University of California, Los Angeles

Abstract

A radix-4 division algorithm with divisor scaling is proposed. The algorithm uses a recurrence with carry-save addition and combines simple scaling with a quotient-selection function that depends only on the estimate of the partial remainder and is independent of the divisor. The redundant quotient is converted on-the-fly into a conventional representation without carry-propagate addition. The scheme results in a significant speed-up with respect to both the radix-2 and radix-4 without scaling, with about the same hardware.

1. Introduction

Subtractive radix-4 division algorithms have been proposed that use a signed-digit representation of the quotient. This quotient representation reduces the complexity of generating the multiples of the divisor and allows the use of carry-save addition and limited precision comparisons. However, the quotient selection function is still complicated, resulting in a significantly larger step-time than that of a radix-2 implementation, which almost eliminates the advantage of using the higher radix [TAYL85,BUSH83].

To simplify the quotient selection function it is possible to prescale the divisor, as reported in [ERCE83, ERCE84, ERCE85]. However, the derivations performed there result in a range of the divisor that requires a complicated scaling. Here we discuss a variant that combines a simple selection with a simple scaling. The resulting scheme is significantly faster than both the radix-2 and the radix-4 without scaling.

The overall division algorithm consists of three steps: scaling of the divisor and the dividend, division recurrence, and conversion of the quotient (Figure 1). We now present each of these steps.

2. Divisor and Dividend Scaling

We want to transform the divisor X into the scaled divisor D such that

$$D = MX \quad \text{and} \quad 1-\alpha \leq |D| \leq 1+\beta$$

We choose α and β so as to have a simple scaling implementation and a simple quotient-selection function. Several alternatives were explored; we report here on the most attractive solution, which results in

$$\frac{63}{64} \leq |D| \leq \frac{9}{8}$$

The corresponding scaling to the specified range is described by the following table (for $D > 0$):

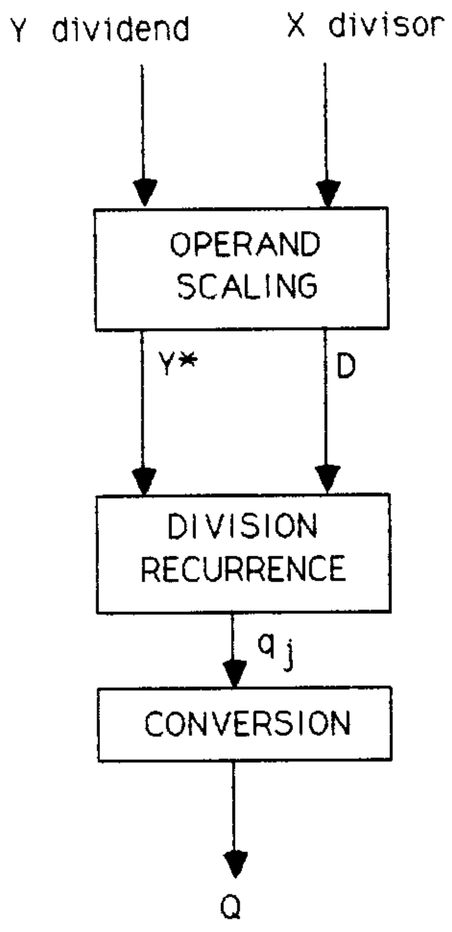

Figure 1. Division Scheme with Scaling

X	M	D
[14/16, 1)	9/8 = 1+1/8	[252/256, 288/256)
[13/16, 14/16)	10/8 = 1+1/4	[260/256, 280/256)
[12/16, 13/16)	11/8 = 1+1/4+1/8	[264/256, 286/256)
[11/16, 12/16)	12/8 = 1+1/2	[264/256, 288/256)
[10/16, 11/16)	13/8 = 1+1/2+1/8	[260/256, 286/256)
[9/16, 10/16)	14/8 = 1+1/2+1/4	[252/256, 280/256)
[8/16, 9/16)	16/8 = 1+1	[256/256, 288/256)

Table 1

The same scale factors apply for a negative divisor. The implementation requires a 3-input carry-save adder, with each input having a selection of two possible multiples of the divisor (dividend), as shown in Figure 2. The details of implementation are given in Appendix A. The same scaling operation has to be performed on the divisor X and on the dividend Y. To share the same hardware, these scaling operations can be performed in sequence. The carry-save adder can also be shared with the recurrence step. The output of the adder is the scaled divisor (dividend) in carry-save form. The scaled dividend is directly used as the first partial remainder. For the scaled divisor, there are two alternatives: to transform it to conventional form with a carry-propagate adder or to use it directly in carry-save form. The second alternative has the big advantage that it does not require a carry-propagate adder nor does it have the corresponding time overhead. However, it complicates the recurrence step. We explore these alternative further in the next section.

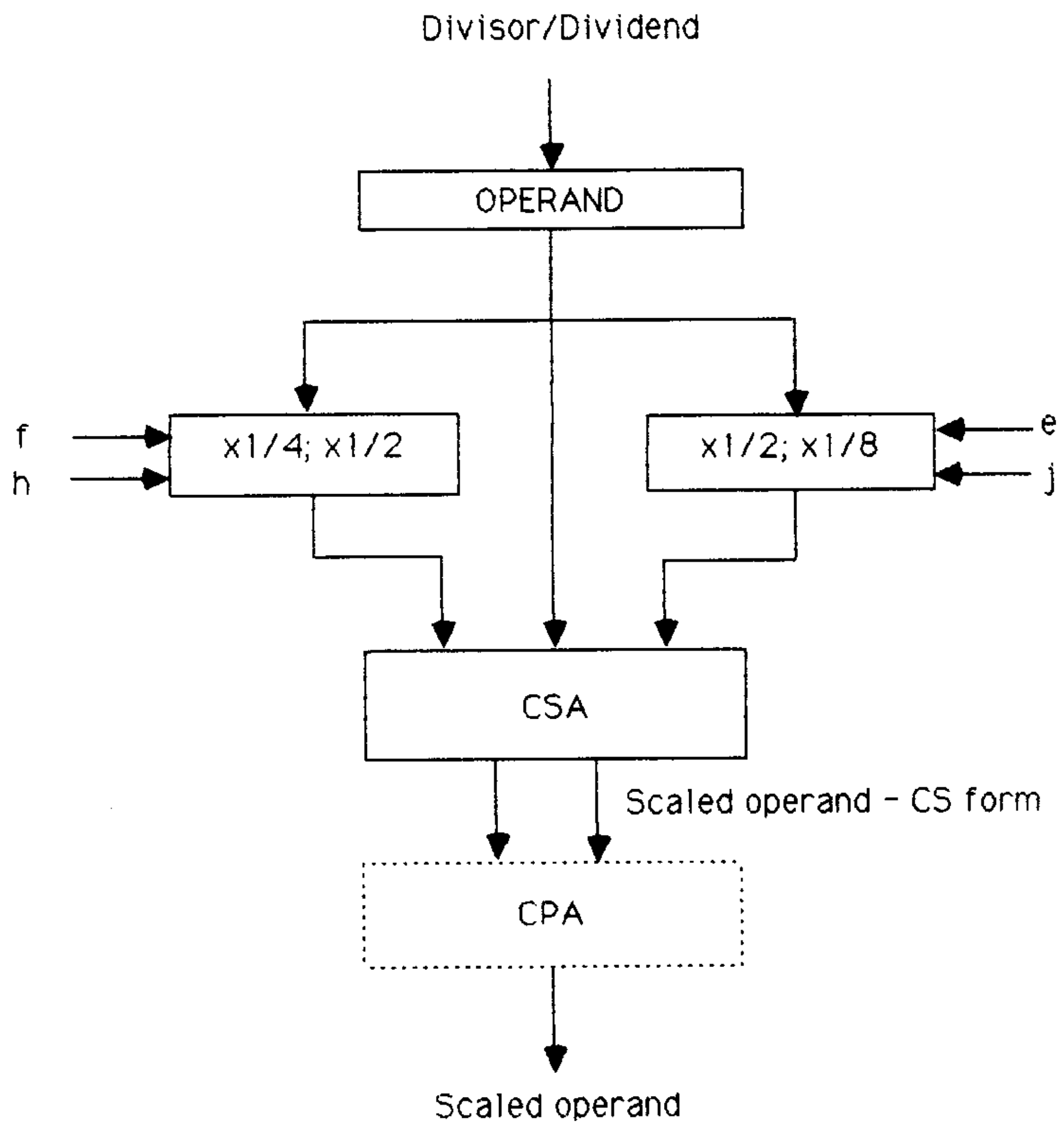

Figure 2. Operand Scaling

3. Division Recurrence

The recurrence we use is the standard radix-4 division algorithm with redundant quotient in the range [-2,2] [ROBE58, ATKI68]. The recurrence is

$$R[j] = 4R[j-1] - Dq_j$$

where the quotient is

$$Q = \sum_{j=1}^{n} q_j 4^{-j} \quad \text{with} \quad q_j \in [-2,2]$$

The bounds on the partial remainder are given by

$$-\frac{2}{3}D \le R[j] \le \frac{2}{3}D$$

and the selection interval for $q_j = k$ is

$$-\frac{2}{3}D + kD \le 4R[j-1] \le \frac{2}{3}D + kD$$

The P-D diagram (first quadrant only) of Figure 3 shows the selection intervals for the divisor in the standard range $1/2 \le D < 1$. As can be seen, the selection intervals overlap and this allows the selection to be performed using an estimate of the partial remainder and of the divisor. For a particular implementation, it is necessary to choose the precision of the partial remainder and the selection constants. The standard procedure [ATKI68] to determine these is as follows:

i) Divide the range of the divisor into subintervals so that in each of these subintervals the selection depends only on the estimate of the partial remainder.

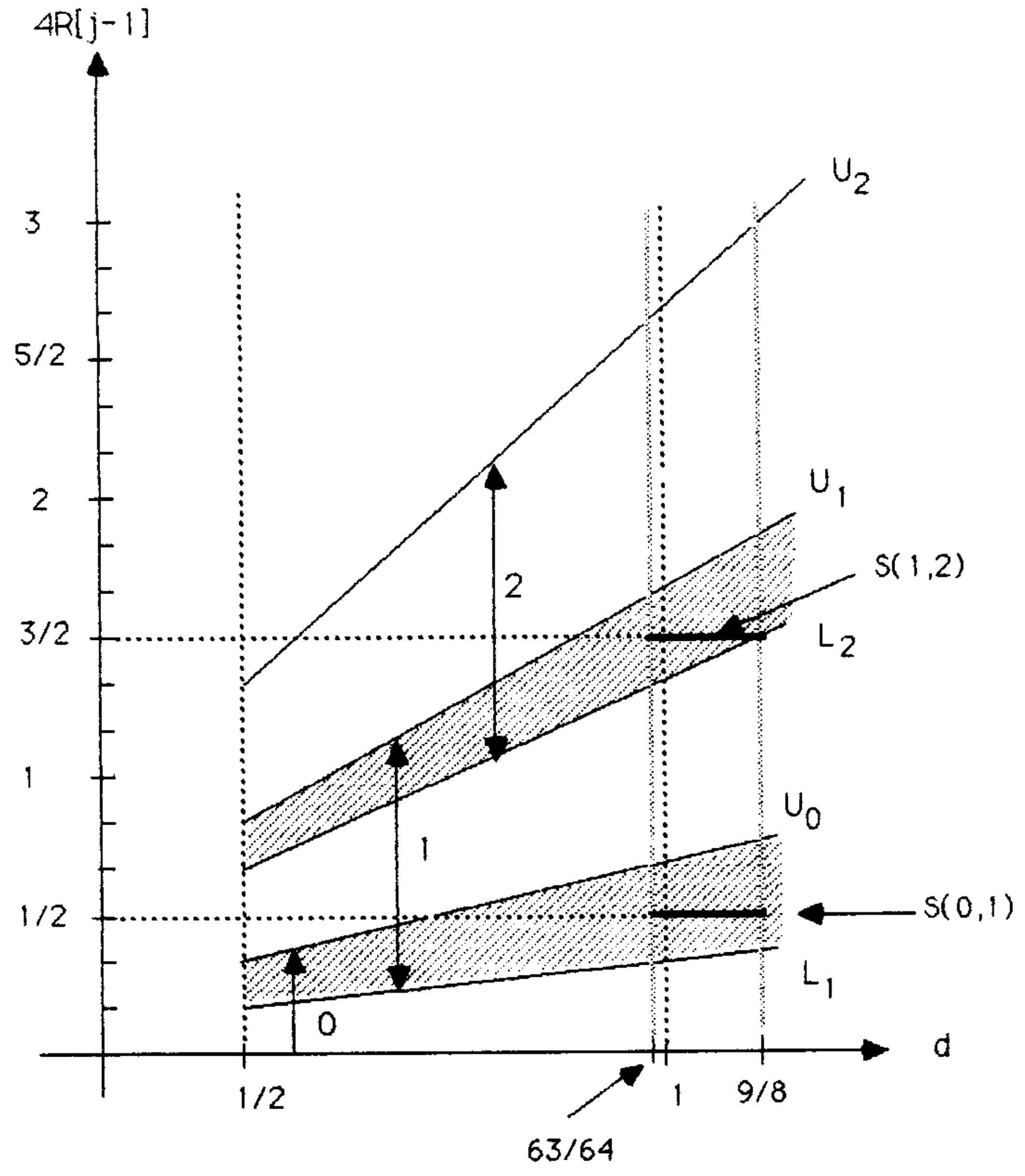

Figure 3: P-D Plot

ii) Select the subintervals in a way that a) reduces the number of bits of partial remainder that have to be assimilated to get the estimate, and b) reduces the number of bits of the divisor and partial remainder required for the quotient-selection function. Since these factors are contradictory, a compromise solution is required. Some solutions are given in [ATKI68, TAYL85].

Analysis of the indicated selection function [TAYL85] shows that it requires the assimilation of 8 bits of the partial remainder, and the use of 6 bits of the assimilated partial remainder and 4 bits of the divisor (Figure 4). The implementation of the corresponding function has a delay that increases significantly the recurrence step time, and consequently, the division time. For example, in the implementation described by Taylor [TAYL85] the radix-4 case has a step time roughly double that of the radix-2 case, which eliminates the advantage of using the higher radix.

The objective of scaling the divisor is to reduce the complexity of this selection function. This idea has been used in the algorithms presented in [ERCE83, ERCE85] where the selection function used is rounding the partial remainder, independently of the value of the divisor. The analysis done there requires that the scaling be to the range $[1-2^{-6}, 1+2^{-6}]$ which makes the transformation relatively complex [ERCE84, MAZE86]. In this paper we decided for a simpler scaling to the range [63/64,9/8] and will now determine a selection function that is suitable for this range.

As a first step in the determination of the selection function, we show that for the range of the transformed divisor (i.e., [63/64, 9/8]) it is possible to use a selection function that is independent of the divisor. For this, we now calculate the overlap between the selection regions

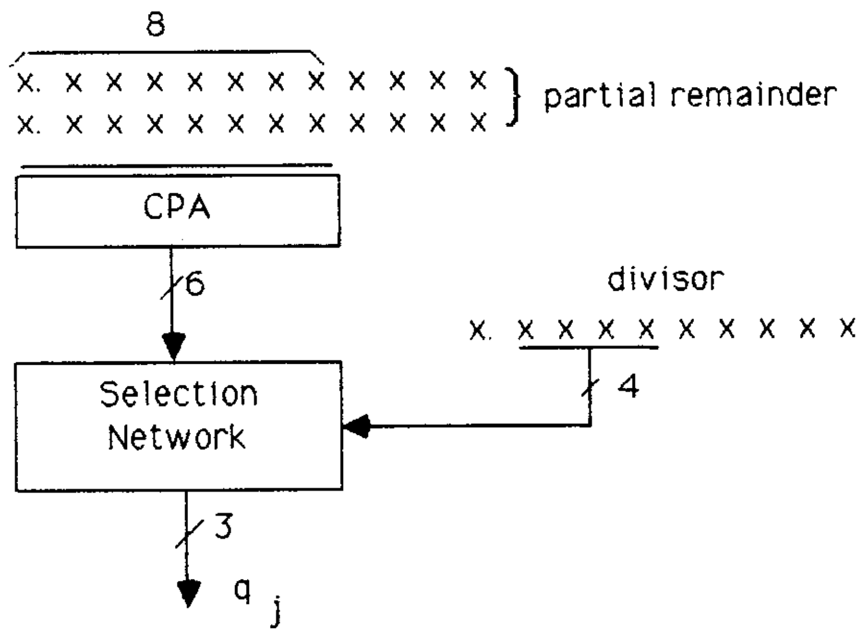

Figure 4: Quotient selection requirements for standard scheme

for $q_j = k-1$ and for $q_j = k$ in the range of the divisor [63/64,9/8]. This overlap is (see Figure 3)

$$\Delta(k-1,k) = \begin{cases} U_{k-1}[63/64] - L_k[9/8] & \text{for } k=1,2 \\ U_{k-1}[9/8] - L_k[63/64] & \text{for } k=0,-1 \end{cases}$$

where $U_k[D]$ and $L_k[D]$ are the upper and lower limits of the selection interval for $q_j = k$ as a function of the value of the divisor D. Since

$$U_{k-1}[63/64] = (2/3)(63/64) + (k-1)63/64 = -21/64 + (63/64)k$$

and

$$L_k[9/8] = -(2/3)(9/8) + (k)(9/8) = -3/4 + (9/8)k$$

we get

$$\Delta(k-1,k) = 27/64 - (9/64)k \quad \text{for } k = 1,2$$

Similarly,

$$U_{k-1}[9/8] = (2/3)(9/8) + (k-1)9/8 = -3/8 + (9/8)k$$

and

$$L_k[63/64] = -(2/3)(63/64) + (k)(63/64) = -(42/64) + (63/64)k$$

Consequently,

$$\Delta(k-1,k) = (18/64) + (9/64)k \quad \text{for } k = 0,-1$$

Introducing the possible values of k, we obtain

$$\Delta(1,2) = 9/64$$

$$\Delta(0,1) = 9/32$$

$$\Delta(-1,0) = 9/32$$

$$\Delta(-2,-1) = 9/64$$

Since the overlaps are positive, it is possible to obtain a single selection function (independent of the value of the divisor) for the whole range.

We now determine the selection function and the required precision of estimate of $4R[j]$. The estimate is computed by a carry-propagate adder of t fractional bits and an additional carry propagation up to fractional bit v (Figure 5). From the figure we see that

$$4R[j] = a \cdot 2^{-t} + b \cdot 2^{-m} + c \cdot 2^{-m}$$

and the estimate is

$$4\hat{R}[j] = a \cdot 2^{-t} + d \cdot 2^{-t}$$

where $d \epsilon \{0,1\}$ is the carry of $b \cdot 2^{-m}$ into fractional bit t.

Consequently, the error is

$$4R[j] - 4\hat{R}[j] = (b \cdot 2^{-m} - d \cdot 2^{-t}) + c \cdot 2^{-m}$$

Since

$$0 \le (b \cdot 2^{-m} - d \cdot 2^{-t}) < 2^{-t} \quad \textit{(by definition of } d\textit{)}$$

and

$$0 \le c \cdot 2^{-m} < 2^{-v}$$

we get

$$0 \le 4R[j] - 4\hat{R}[j] < 2^{-t} + 2^{-v}$$

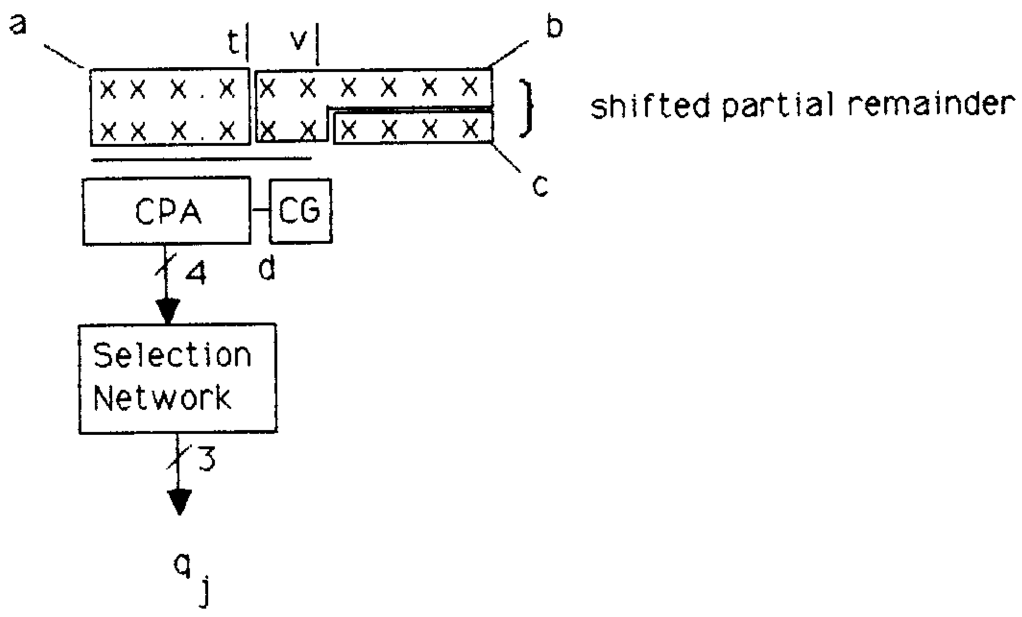

Figure 5. Quotient selection requirements for proposed scheme

That is,

$$4R[j] - \Delta < 4\hat{R}[j] \le 4R[j] \quad \Delta = (2^{-t} + 2^{-v})$$

Moreover, if $M(k)$ [$m(k)$] is the largest [smallest] value of $4\hat{R}[j]$ for which a quotient value of k is chosen, we have

$$m(k) \le 4\hat{R} \le M(k) \quad \rightarrow \quad q_{j+1} = k$$

$$m(k+1) = M(k) + 2^{-t} \quad (\textit{since } t \textit{ fractional bits are used in selection})$$

$$L(k) \le M(k) \le U(k) - \Delta$$

These relations are illustrated in Figure 6.

Consequently, the selection function is determined as follows. Select the smallest t such that,

$$L(k) \le M(k) = A(k)2^{-t} < U(k) - 2^{-t} \quad (A(k) \textit{ integer})$$

and

$$m(k+1) = (A(k)+1)2^{-t} \ge L(k+1)$$

Then, determine $v \ge t$ such that

$$U(k) - M(k) \ge 2^{-t} + 2^{-v}$$

Table 2 shows the selection for $t=1$ and the minimum value of 2^{-v}. We conclude that $v=3$.

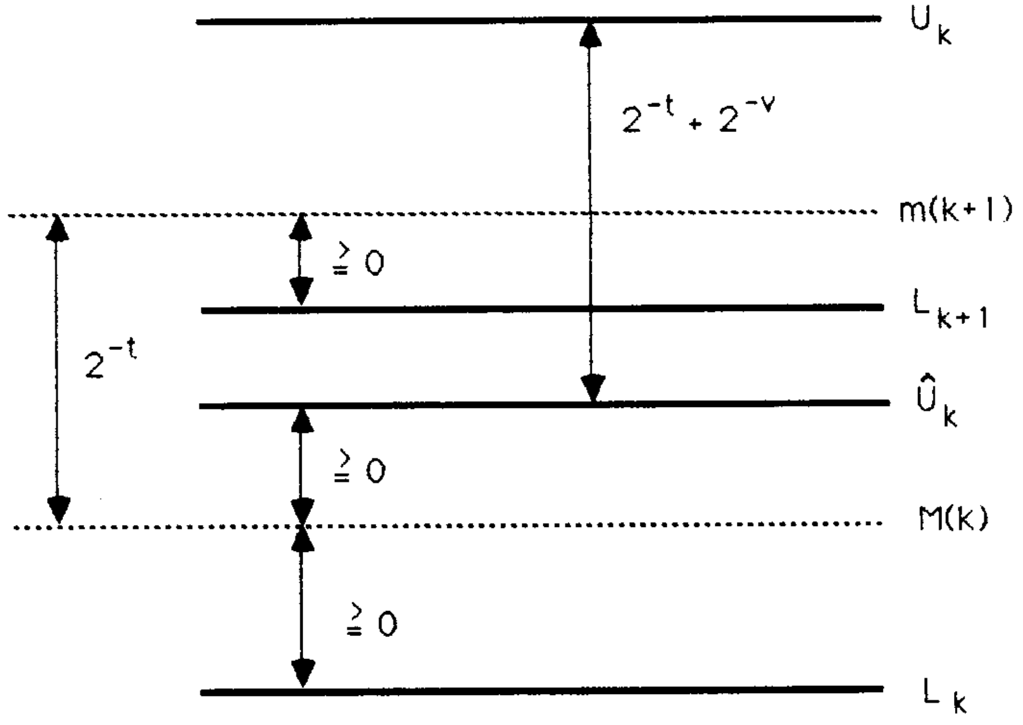

Figure 6. Selection Relations

L_k U_k	$M(k)$ $m(k)$ (for $t = 1$)	$2^{-v} = U_k - M(k) - 2^{-t}$
$L_2[9/8] = -(2/3)(9/8) + 2(9/8) = 3/2$	$m(2)=3/2$	
$U_1[63/64] = (2/3)(63/64) + 63/64 = 105/64$	$M(1)=1$	11/64
$L_1[9/8] = -(2/3)(9/8) + 9/8 = 3/8$	$m(1)=1/2$	
$U_0[63/64] = (2/3)(63/64) = 21/32$	$M(0)=0$	5/32
$L_0[63/64] = -(2/3)(63/64) = -21/32$	$m(0)=-1/2$	
$U_{-1}[9/8] = (2/3)(9/8)-(9/8) = -3/8$	$M(-1)=-1$	1/8
$L_{-1}[63/64] = -(2/3)(63/64) - (63/64) = -105/64$	$m(-1)=-3/2$	
$U_{-2}[9/8] = (2/3)(9/8) - 2(9/8) = -3/2$	$M(-2)=-2$	

Table 2

The binary-level specification of the quotient-selection function and the corresponding switching expressions are given in Appendix B.

We now summarize the values of the critical parameters for this scheme. They are

i) The selection function requires the assimilation of four bits of the shifted partial remainder.

ii) The carry of two additional bits of the partial remainder has to be propagated.

iii) The number of bits of the divisor required by the selection function. In our case the selection is independent of the value of the divisor (in the range after scaling).

The block diagram of the implementation showing the number of bits required is given in Figure 5. The comparison with the case without scaling, as shown in Figure 4, indicates a significant reduction in the complexity of the quotient selection function. The actual reduction in step time that this produces would depend on the specific implementation constraints. As an example, for the implementation described by Taylor [TAYL85] we estimate a speed-up of between 1.33 and 1.6 (2 or 3 fewer logic steps).

4. Quotient Conversion

Finally, the redundant representation of the quotient is transformed into the conventional radix-4 representation using the on-the-fly scheme presented in [ERCE85]. As the quotient digits are generated, the most significant digit first, the corresponding part of the conventional quotient is obtained without carry-propagate addition. Thus the conversion has no effect on the overall delay.

5. Overall Implementation and Timing

We now discuss the overall implementation of the division scheme. As indicated in Section 3, two alternative implementations are possible depending whether the scaled divisor is assimilated or not. These alternatives are shown in Figure 7(a) and 7(b), respectively. The assimilated version uses a 3-2 carry-save adder while the nonassimilated one requires a 4-2 reduction. On the other hand, the nonassimilated case does not require a carry-propagate addition for divisor scaling. To choose between these alternatives requires a detailed implementation, which is outside of the scope of this paper.

The timing of the divider consists of three parts: a) scaling of operands, b) division recurrence, and c) quotient conversion. As mentioned in previous sections, the times per quotient digit of parts b) and c) is significantly reduced with respect to the standard radix-4 scheme without scaling, and have estimated a speed-up of between 1.3 and 1.6 for the implementation constraints described by Taylor [TAYL85].

This speed-up is reduced by the overhead in the scaling operation. We estimate that this scaling operation would take three cycles: one to scale the dividend (without assimilation) and two to scale the divisor (with assimilation). For a 56-bit quotient the overhead would be of 3/28 cycles, that is, roughly 11%. Consequently, taking again as a reference the implementation described by Taylor [TAYL85], we estimate an overall speed-up of between 1.22 and 1.45 with respect to the scheme without scaling.

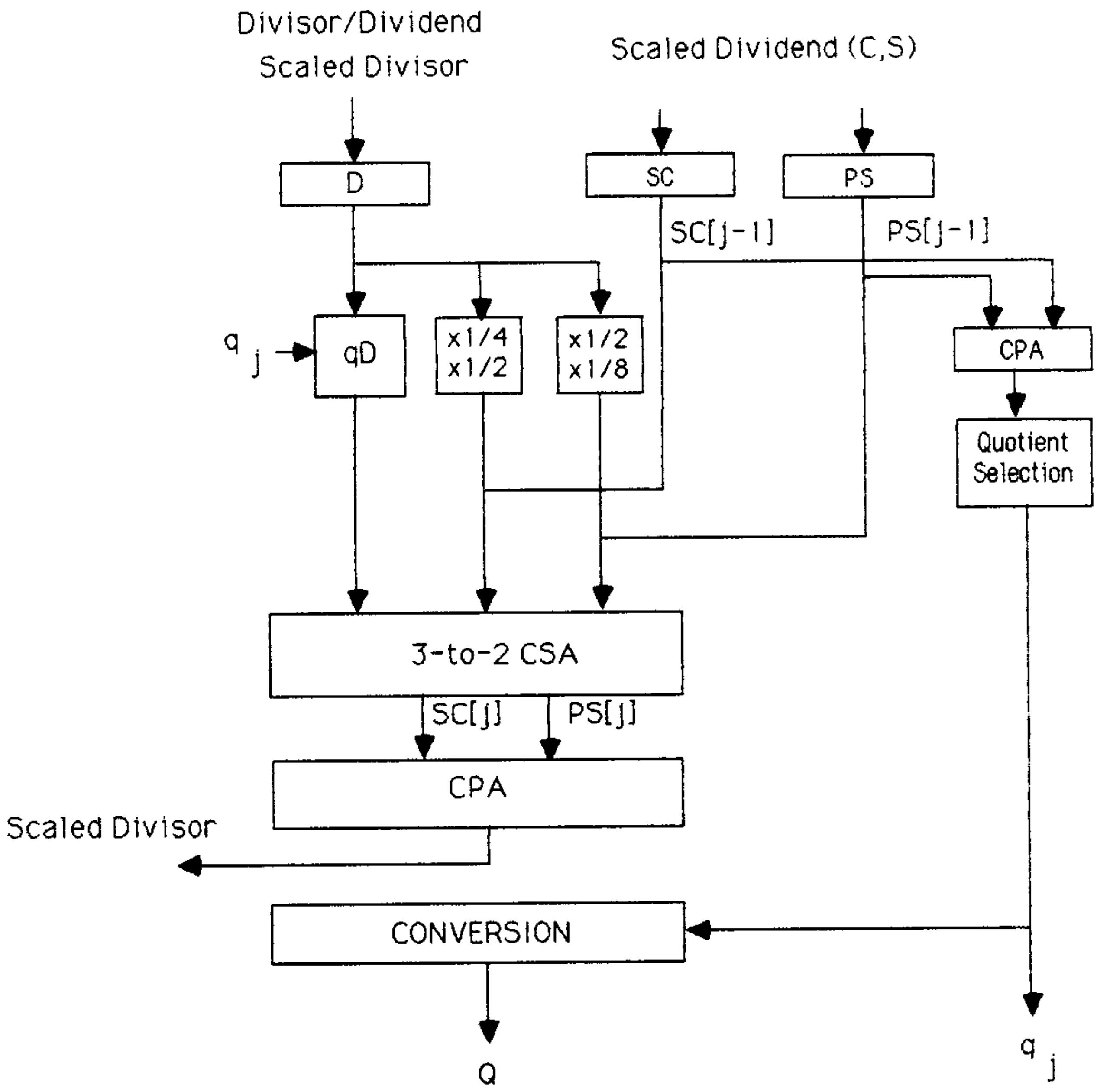

Figure 7(a): Division Scheme with Scaled Divisor and 3-to-2 CSA

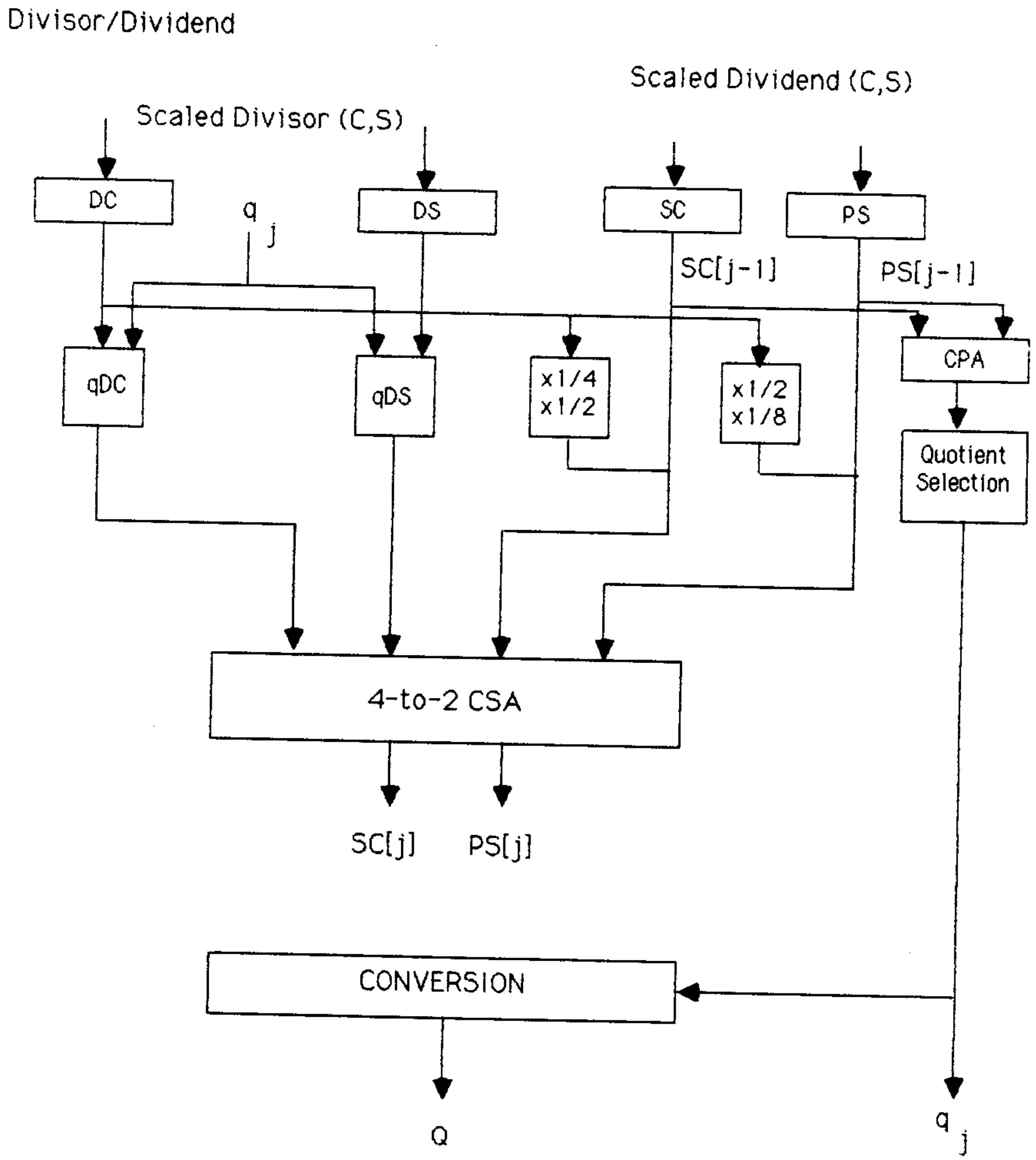

Figure7(b): Division Scheme with Redundant Scaled Divisor and 4-to-2 CSA

The proposed scheme would also produce a significant improvement in speed for higher radix dividers, such as the radix-16 divider with overlapped quotient selection stages proposed by Taylor [TAYL85].

Summary

A radix-4 division scheme is proposed having the following features:

(i) Divisor and dividend are scaled into a suitable range using a simple scaling scheme.

(ii) The recurrence uses a carry-save adder to produce partial remainders.

(iii) The quotient-selection function does not depend on the (scaled) divisor: it requires a 4-bit estimate of the partial remainder.

(iv) The conventional (2's complement) representation of the quotient is produced on the fly, without an extra carry-propagate addition.

The proposed scheme should result in a significantly faster implementation than the previously proposed radix-4 schemes.

Acknowledgements This work has been supported in part by the ONR Contract N00014-85-K-0159 and Hughes Research Laboratories. We appreciate interest and comments of Dr. J. G. Nash.

References

[ATKI68] D.E. Atkins, "Higher-Radix Division Using Estimates of the Divisor and Partial Remainders", IEEE Trans. on Computers, Vol.C-17, No. 10, October 1968, pp.925-934.

[BUSH83] L.B. Bushard, "A Minimum Table Size Results for Higher Radix Nonrestoring Division", IEEE Trans. on Computers, Vol. C-32, No. 6, June 1983, pp.521-526.

[ERCE83] M.D. Ercegovac, "A Higher Radix Division with Simple Selection of Quotient Digits", Proc. of 6-th IEEE Symposium on Computer Arithmetic, 1983, pp.94-98.

[ERCE84] M.D. Ercegovac and T. Lang, "Radix-4 Division with Range Transformation", UCLA CSD Internal Report, August 1984.

[ERCE85a] M.D. Ercegovac and T. Lang, "A Division Algorithm with Prediction of Quotient Digits", IEEE Proc. of 7th Symposium on Computer Arithmetic, 1985, pp. 51-56.

[ERCE85b] M.D. Ercegovac and T. Lang, "On-the-Fly Conversion of Redundant into Conventional Representations", UCLA CSD Internal Report, August 1985. (to appear in IEEE Trans. Computers, 1987)

[MAZE86] J. Mazerolle, "Simulation of a Fast Division Algorithm", UCLA CSD, Internal Report, August 1986.

[ROBE58] J.E. Robertson, "A New Class of Digital Division Methods", IRE Trans. on Electronic Computers, September 1958, pp.88-92.

[TAYL85] G.S. Taylor, "Radix 16 SRT Dividers with Overlapped Quotient Selection Stages", IEEE Proc. of 7th Symposium on Computer Arithmetic, 1985, pp. 64-73.

Appendix A

We determine here the control signals for the scaling of Figure 2. As required by Table 1, the scale factor is defined as

$$M = 1 + e \cdot 1/8 + f \cdot 1/4 + h \cdot 1/2 + j \cdot 1/2$$

>From the same Table 1, we determine that the binary variables e, f, h, and j are switching functions of bits x_0, x_2, x_3 and x_4 of the divisor X, as defined in the following table:

$x_0x_1x_2x_3x_4$	e	f	h	j
01110	1	0	0	0
01101	0	1	0	0
01100	1	1	0	0
01011	0	0	1	0
01010	1	0	1	0
01001	0	1	0	1
01000	0	0	1	1
10001	1	0	0	0
10010	0	1	0	0
10011	1	1	0	0
10100	0	0	1	0
10101	1	0	1	0
10110	0	1	0	1
10111	0	0	1	1

Let $w = x_2 \oplus x_0$, $y = x_3 \oplus x_0$, and $z = x_4 \oplus x_0$. Then the control signals for the scaling network are

$$e = (w + y)z' \qquad f = (w + z)y'$$

$$h = (y + z')w' \qquad j = w'y'$$

Appendix B

The switching expressions for the quotient digit selection network are obtained from the following table.

q_i	$z_{-2}z_{-1}z_0.z_1$
2	011.0
	010.1
	010.0
	001.1
1	001.0
	000.1
0	000.0
	111.1
–1	111.0
	110.1
–2	110.0
	101.1
	101.0

The quotient digit is represented by $(k_2, k_1, k_0, k_{-1}, k_{-2})$ as follows:

q_i	k_2	k_1	k_0	k_{-1}	k_{-2}
2	1	0	0	0	0
1	0	1	0	0	0
0	0	0	1	0	0
-1	0	0	0	1	0
-2	0	0	0	0	1

The corresponding switching expressions are:

$$k_2 = z_{-2}'z_{-1} + z_{-2}'z_0z_1$$

$$k_1 = z_{-2}'z_{-1}'z_0z_1' + z_{-2}'z_{-1}'z_0'z_1$$

$$k_{-1} = z_{-2}z_{-1}z_0'z_1 + z_{-2}z_{-1}z_0z_1'$$

$$k_{-2} = z_{-2}z_{-1}' + z_{-2}z_0'z_1'$$

$$k_0 = z_{-1}'z_0'z_1' + z_{-1}z_0z_1$$

The last expression need not be implemented since $k_0 = 1$ if all other $k_i = 0$.