A Division Algorithm with Prediction of Quotient Digits

M.D. Ercegovac and T. Lang

Computer Science Department
University of California, Los Angeles

Abstract

A division algorithm with a simple selection of quotient digits including prediction is possible if the divisor is restricted to a suitable range. The conditions that the divisor must satisfy to have the quotient digit q_{i+1} predicted while computing R_{i+1} are determined. Some implementation considerations are also given.

I. Introduction

Division algorithms have been extensively studied to satisfy the requirements of fast and efficient selection of quotient digits and computation of partial remainders, and compatibility with other frequent arithmetic operations such as multiplication [ROBE58, ATKI68, ATKI74, TAYL81]. A class of these algorithms is based on prescaling the divisor into a suitable range [SVOB63, KRIS70, ERCE77]. In this article a division scheme of this class, in which the quotient digits are obtained by rounding [ERCE83], is extended to allow prediction of the quotient digit. This results in a significant reduction of the iteration step.

The division algorithm presented in [ERCE83] has the following characteristics:

- It uses the recurrence

$$R[i+1] = r(R[i]-q_iX)$$

where X is the divisor, $R[0]$ is the dividend, and q_i is a digit of the quotient $Q = q_0.q_1q_2 \cdots q_m$.

- The quotient digit-set is redundant such that $-\rho \le q_i \le \rho$.

- The quotient digit is selected by the following simple function on an estimate of the partial remainder $\hat{R}[i]$:

$$q_i = \begin{cases} round(\hat{R}[i]) & \text{if } |\hat{R}[i]| \le \rho \\ \rho & \text{if } |\hat{R}[i]| > \rho \end{cases}$$

- The estimate $\hat{R}[i]$ is obtained by truncating the carry-save representation of $R[i]$ after the kth fractional digit. That is,

$$\hat{R}[i] \le R[i] \le \hat{R}[i] + 2^{-k+1}$$

- For this selection function to produce the correct quotient the divisor has to be in the range $1-\alpha \le X \le 1+\alpha$. Consequently, the divisor has to be transformed before the iteration into this range. The value of α is related to the error in the estimate by

$$\alpha \le \frac{1}{r}\left[1 - \frac{(r-1)(2^{-k+1} + 2^{-1})}{\rho}\right]$$

The transformed divisor is called X^* in [ERCE83]. To simplify the notation here we call it just X.

In this paper we extend the previous algorithm by introducing the prediction of the quotient digit. This permits the computation of q_{i+1} to be performed concurrently with the calculation of $R[i+1]$, and therefore reduces the execution time of the iteration step. This modification produces a reduction in the bound for α, so that more transformation steps are required.

II. Quotient Digit Prediction

The division process outlined in the introduction consists of a sequence of iterations, each of which is formed of four steps (see Figure 1):

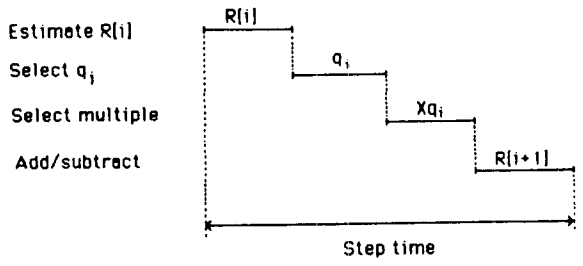


Figure 1: Basic Iteration

CH2146-9/85/0000/0051$01.00

- Determination of the remainder estimate $\hat{R}[i]$ in assimilated form.
- Determination of a quotient digit q_i (rounding).
- Selection of a divisor multiple q_iX.
- Subtraction to obtain the new partial remainder $R[i+1]$ in carry-save form.

The time of an iteration step is

$$T = t_a + t_q + t_s + t_{cs} + t_l$$

where

t_a = time for assimilation of $\hat{R}[i]$
t_q = time to round
t_s = time to select the divisor multiple
t_{cs} = time of subtraction in carry-save form
t_l = time to load the registers

To reduce the time of an iteration step it is possible to precompute the quotient digit in the previous iteration step. This results in an iteration step consisting of two parallel paths. In one the next partial remainder is obtained while in the other the next quotient digit is computed.

Using the quotient calculation procedure presented before, the quotient digit q_{i+1} depends on $\hat{R}[i+1]$. In order to predict this digit it is necessary to base the prediction on $\hat{R}[i]$ (and maybe X) since $\hat{R}[i+1]$ has not been computed yet. Since

$$R[i+1] = r(R[i]-q_iX)$$

it is possible to determine q_{i+1} by

$$q_{i+1} = round(R[i+1]) = round(r(R[i]-q_iX))$$

which could be approximated by

$$q_{i+1} = round(estimate(r(R[i]-q_iX))$$

(where $round(d)$ is ρ if $d \geq \rho$.)

However, this prediction does not produce a significant reduction in time since the path requires the same steps as the iterative step without prediction: selection of the multiple, subtraction, assimilation, and rounding (see Figure 2).

A more promising approach is to introduce an additional approximation and compute the quotient digit as

$$q_{i+1} = round(r(\hat{R}[i]-q_i))$$

which substracts q_i instead of q_iX.

This eliminates the step of selecting the multiple of the divisor and simplifies the subtraction, since q_i is an integer (see Figure 3). The time of a step is now

$$T = max(t_a + t_q + t_l\ ,\ t_s + t_{cs} + t_l)$$

where t_q now includes the subtraction of q_i and the rounding. This is reasonable since it is feasible to implement the function $round(r(\hat{R}[i]-q_i))$ in two logic levels. We refer to this approach as scheme A.

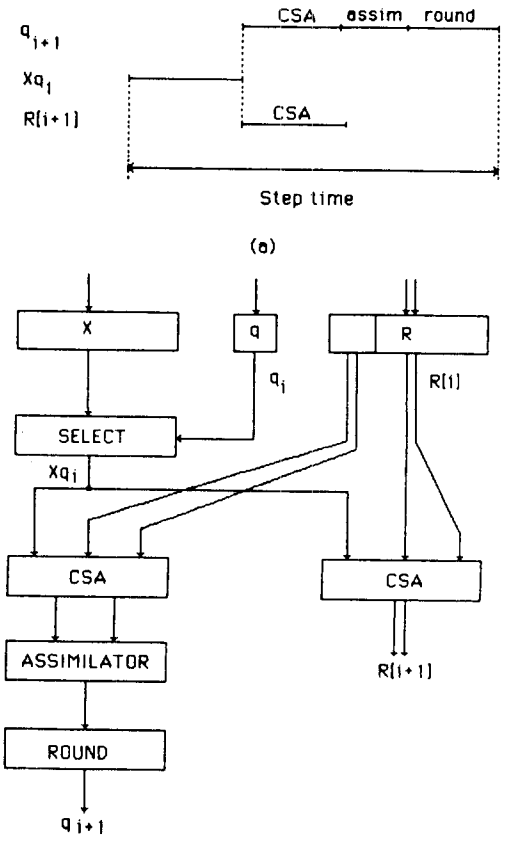


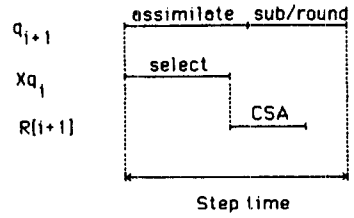
Figure 2: Prediction based R[i] and Xq_i

The smallest step time is obtained when both paths are balanced. If the path for calculating q_{i+1} is longer than that to compute $R[i+1]$, it is possible to balance the two paths by including the assimilation in the second path, as shown in Figure 4, and store $\hat{R}[i]$ (Scheme B). In addition, to reduce the critical path, it is possible to use faster circuts in the slice required to compute $\hat{R}$ (even duplicating this slice to reduce the complexity of the interconnection might be convenient). In this case the time is

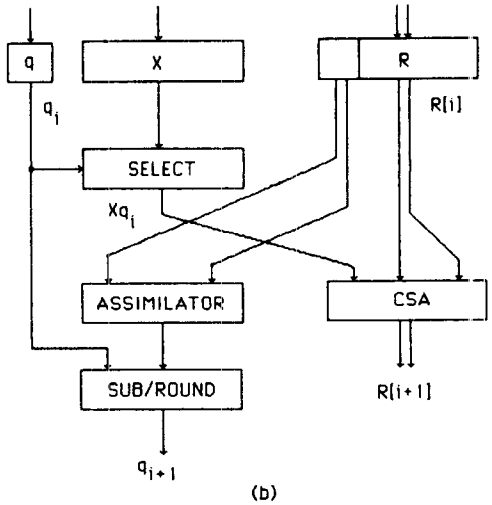
$$T = max(t_q + t_l\ ,\ t_s + t_{cs} + t_a + t_l)$$

As a third alternative, it is possible to decompose the assimilation step and compute part of it in each of the two paths. For example, if the addition is done using carry lookahead, the propagate and generate variables could be computed in the remainder path, while the computations of the carry and sum could be done in the quotient path (scheme C).

The actual step time depends on the radix, the technology, and some implementation decisions. For

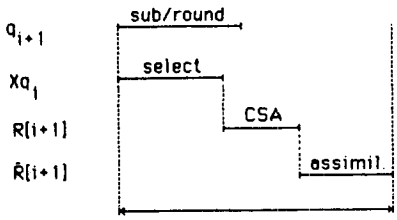


(a)

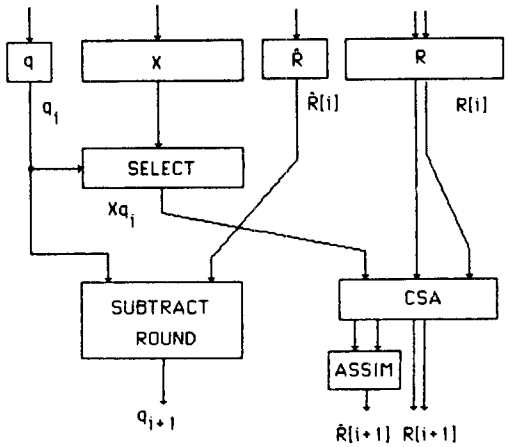


(b)

Figure 3: Prediction based on R[i] and q_i
(a) Step Time
(b) Scheme A



(a)



(b)

Figure 4: Prediction based on $\hat{R}$[i] and q_i
(a) Step Time
(b) Scheme B

comparison purposes, consider the radix-4 case. The number of bits over which the estimate of the remainder has to be assimilated depends on the range of the transformed divisor (value of α, see next section), and the choice for this is influenced by the time of transformation. A possible choice results in an assimilation over six bits as described in detail in [ERCE85]. Note that the rounding requires only four bits of the estimate (three integer bits and one fraction bit), so that the network for assimilation consists of a 4-bit CPA and a two-level network for the computation of the carry into the CPA (Figure 5). Since the actual times are very technology dependent, we limit our comparison to a description of the blocks required for the proposed approach and for the SRT radix-4 scheme as described in [TAYL81]. This comparison is given in Figure 5. It suggests a significant reduction in the iteration step.

III. Determination of α

Since this procedure of quotient digit prediction introduces an additional approximation (using q_i instead of q_iX) it requires an additional restriction in the range of X to produce the correct quotient. We now determine this restriction so that

$$R[i+2] \leq \rho+\beta$$

as required for the algorithm to produce the correct quotient [ERCE83].

The basic recurrence for $i+1$ can be written

$$R[i+2] = r(R[i+1] - q_{i+1}X)$$

Replacing $R[i+1]$ in terms of $R[i]$ we get

$$R[i+2] = r(r(R[i] - q_iX) - q_{i+1}X)$$

This can be transformed into

$$R[i+2] = r(r(R[i] - q_i) - q_{i+1} + r(1-X)q_i + (1-X)q_{i+1})$$

We now consider the two limiting cases, that is, those that might produce the largest value of $R[i+2]$.

CASE I: $|q_i| = |q_{i+1}| = \rho$

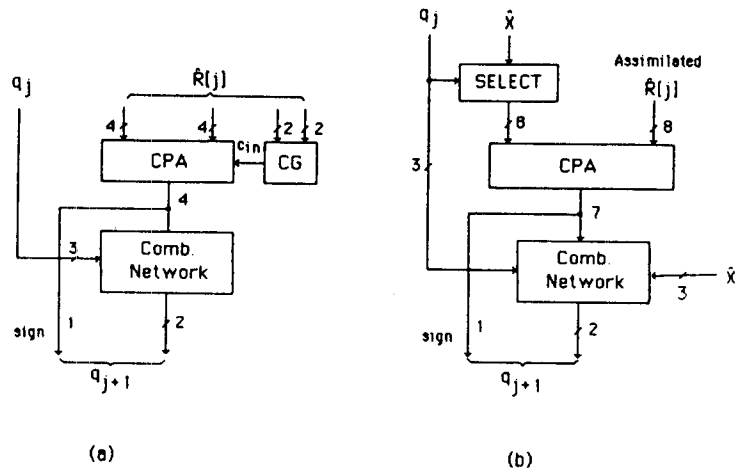


Figure 5. (a) Quotient Generation in the Proposed Scheme
(b) Quotient Generation in the SRT Radix-4 Scheme

Since the bound on the remainder is

$$|R[i]| \le \rho+\beta$$

we obtain that

$$|R[i]-q_i| \le \beta$$

and

$$|r(R[i] - q_i) - q_{i+1}| \le r\beta - \rho$$

Therefore,

$$R[i+2] \le r[(r\beta-\rho) + r\alpha\rho + \alpha\rho]$$

Consequently, to have $R[i+2] \le \rho+\beta$ we need that

$$\rho+\beta \ge r^2\beta - r\rho + r(r+1)\alpha\rho$$

From this we obtain

$$\alpha \le \frac{\rho + \beta - r^2\beta + r\rho}{r(r+1)\rho}$$

That is,

$$\alpha \le \frac{\rho(1+r) + \beta(1-r^2)}{r(r+1)\rho}$$

which results in

$$\alpha \le \frac{\rho - \beta(r-1)}{r\rho} = \frac{1}{r}[1 - \frac{\beta(r-1)}{\rho}]$$

This is the same value as without prediction.

CASE II: $q_i = \rho$ and $q_{i+1} = \rho-1$.

Since in this case the prediction is done by rounding $r(\hat{R}[i]-q_i)$ we obtain

$$r(R[i]-q_i) \le 1/2 + \delta$$

where δ is the error produced in the estimate of the remainder (that is $\hat{R}[i] \le R[i] \le \hat{R}[i]+\delta$).

Consequently

$$R[i+2] \le r(1/2 + \delta + r\alpha\rho + \alpha(\rho-1))$$

Introducing the bound on $R[i+2]$ we get

$$\rho+\beta \ge r(1/2 + \delta + r\alpha\rho + \alpha(\rho-1))$$

This results in

$$\alpha \le \frac{\rho + \beta - r/2 - r\delta}{r(r\rho + \rho - 1)}$$

which is roughly $1/r$ times the value without prediction.

IV. Range Transformation of the Divisor

The previous algorithm requires the divisor $X[0]$ to be transformed into X in the range $(1-\alpha) \le X \le (1+\alpha)$. Several alternative algorithms for this transformation have been examined. The objective is to have a fast transformation that utilizes the blocks required by the division step.

In [ERCE83] a transformation based on the continued product normalization algorithm [ERCE73] is given. The implementation is complex and does not use effectively the division blocks. Another possibility is to use the recurrence

$$X[i+1] = X[i] + s_{i+1}X[0]2^{-(i+1)}$$

with

$$1/2 \le X[0] < 1$$

and

$$1-2^{-p} < X[p]=X < 1+2^{-p}$$

The selection of s_{i+1} is done by

$$s_{i+1} = \begin{cases} 1 & \text{if } X_0[i]=0 \text{ and } X_{i+1}[i]=0 \\ -1 & \text{if } X_0[i]=1 \text{ and } X_{i+1}[i]=1 \\ 0 & \text{otherwise} \end{cases}$$

where X[i] is represented by the bit vector $X_0[i].X_1[i].....$

The resulting transformation requires p steps, where $2^{-p-1} \le \alpha$. As for division, the redundancy in the representation of s can be used to utilize a carry-save addition and limited assimilation for the selection of s_i. Since the transformed divisor is required in assimilated form, a carry-propagate addition is required at the end. We explored the possibility of using the divisor in carry-save form; however this would complicate significantly the remainder calculation path.

This transformation approach has the disadvantage of being radix-2 and, therefore, it is relatively slow and cannot use effectively the radix-4 network of the division recurrence step. Its generalization to higher radix results in a complex selection function, whose implementation would have a large delay.

A third approach for the transformation is to use a series expansion of the reciprocal $1/X[0]$. That is,

$$M = 1/X[0] + \epsilon$$

$$X = X[0]M = 1 + \epsilon X[0]$$

The value of ϵ has to be chosen so that X satisfies the range requirements. That is, $\epsilon \le \alpha$ since $X[0] < 1$.

We use McLaurin's expansion of the reciprocal of $D = 2X[0]$, to have $1 \le D \le 2$. Decomposing D into two terms, such that

$$D = D_1 + 2^{-k}D_2$$

results in

$$R = \frac{1}{D} = \frac{1}{D_1 + 2^{-k}D_2}$$

$$= \frac{1}{D_1(1 + 2^{-k}D_2\frac{1}{D_1})}$$

$$= R_1\frac{1}{1 + 2^{-k}D_2R_1}$$

$$= R_1[1 - 2^{-k}D_2R_1 + 2^{-2k}(D_2R_1)^2 - ...]$$

Therefore, an approximation to $1/X[0]$ is

$$\hat{R} = \hat{R}_1 - 2^{-k}\hat{R}_1^2\hat{D}_2$$

where $\hat{R}_1$, $\hat{R}_1^2$, and $\hat{D}_2$ correspond to truncated versions of R_1, R_1^2, and D_2, respectively. The number of bits required for each of these quantities, as well as the value k are determined so that the error is bounded as indicated previously.

The implementation of this scheme requires a network to generate $\hat{R}_1$ and $\hat{R}_1^2$, the multiplication of $\hat{R}_1^2$ by $\hat{D}_2$, and the subtraction (Figure 6). The resulting network is quite simple and fast.

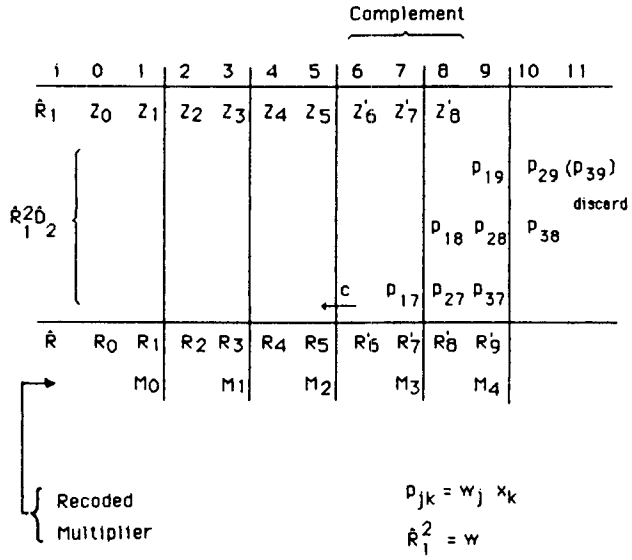


Figure 6 : Scheme for Approximation of 1/X[0]

The resulting $\hat{R}$ has to be multiplied by $X[0]$ to determine X. This multiplication can use the carry-save adder of the division recurrence. In order to use the left shift capability available there, the multiplication should be done most significant digit of $\hat{R}$ first. Usually, this type of multiplication requires an adder of increasing precision. However, in this case, the most significant bits of X are 1.00000 or 0.111111, so that a few extra bits are sufficient to determine X. Note also that, due to the redundant nature of the recoded $\hat{R}$, its most significant digits can be computed without waiting for the carry of the subtraction to propagate. This allows the multiplication to begin without the additional delay of the subtraction. Implementation details are given in [ERCE85].

V. Conclusion

A scheme for higher radix division has been presented. It consists of a transformation of the divisor and the dividend into a range which allows use of a higher radix division recurrence with a simple predictive quotient selection method. A detailed derivation of the conditions required for the quotient digit prediction and the comparison of several implementation alternatives have been described. The implementation details and the performance are discussed elsewhere [ERCE85].

Acknowledgements The authors are gratefull to G. Taylor of University of California, Berkeley, for incisive comments and to J.G. Nash of Hughes Research Laboratories, Malibu, for implementation contributions. This research has been supported in part by the Office of Naval Research under Grant N00014-83-K-0493.

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.

[ATKI70] D. E. Atkins, "Design of the Arithmetic Units of ILLIAC III: Use of Redundancy and Higher Radix Methods", IEEE Trans. on Computers, August 1970, pp.720-733.

[ATKI74] D. E. Atkins and U. Kalaycioglu, "Concurrency in Generalized Radix Non-Restoring Division", Proc. 12th Allerton Conference on Circuit and Switching Theory, pp.628-640, October, 1974.

[ERCE73] M. D. Ercegovac, "Radix-16 Evaluation of Certain Elementary Functions", IEEE Trans. on Computers, June 1973, pp.561-566.

[ERCE77] M. D. Ercegovac, "A General Hardware-Oriented Method for Evaluation of Functions and Computations in a Digital Computer", IEEE Trans. on Computers Vol. C-26, No.7, July 1977, pp. 667-680.

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

[ERCE85] M.D. Ercegovac, T. Lang, and G. Nash, "A VLSI Implementation of a Radix-4 Division Algorithm", in preparation, 1985.

[KRIS70] E.V. Krishnamurthy, "On Range-Transformation Techniques for Division", IEEE Transactions on Computers, Vol.C-19, No.3, March 1970, pp.227-231.

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

[SVOB63] A. Svoboda, "An Algorithm for Division", Information Processing Machines, Vol.9, 1963, pp.25-32.

[TAYL81a] G. S. Taylor, "Compatible Hardware for Division and Square Root", Proc. 5th Symposium on Computer Arithmetic, 1981, pp.127-134.