Division with Speculation of Quotient Digits *

Jordi Cortadella

Dept. of Computer Architecture
Polytechnic University of Catalonia
Barcelona, Spain 08071

Tomás Lang

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

Abstract

The speed of SRT-type dividers is mainly determined by the complexity of the quotient-digit selection, so that implementations are limited to low-radix stages. We present a scheme in which the quotient-digit is speculated and, when this speculation is incorrect, a rollback or a partial advance is performed. This results in a division operation with a shorter cycle time and a variable number of cycles. We performed several designs and report results that show a radix-64 implementation that is 30% faster than the fastest conventional implementation (radix-8) at an increase of about 45% in area per quotient bit. Moreover, we show a radix-16 implementation that is about 10% faster than the radix-8 conventional one, with the additional advantage of requiring about 25% less area per quotient bit.

1 Introduction

Most implementations for the division operation are based on the SRT algorithm which involves a recurrence in which one digit of the quotient is produced per iteration [1]. Consequently, to reduce the number of iterations it is convenient to use a higher radix for the quotient digit. However, as the radix increases the added complexity of the quotient selection function increases the iteration delay and eliminates the advantage. Because of this, implementations have used radix 2 and radix 4 stages and higher radices are obtained by unfolding these stages [10, 6]. Moreover, reductions in time have been obtained by overlapping these unfolded stages [10, 12]. One way that has been proposed to reduce the complexity of the quotient selection function is to prescale the divisor (and the dividend) [9, 7, 4] or to multiply the estimate of the residual by an approximation of the reciprocal of the divisor [11, 8].

In this paper we present and evaluate the idea of using a simple function to speculate a probable value of the quotient digit and use this speculated value to continue with the algorithm. Another function determines whether the speculation is incorrect and, in that case, the algorithm rolls back, the digit is corrected and the process continues from there. In a variation of this scheme, we allow a partial advance of fewer bits than a full digit when an incorrect speculation is performed.

Because of the possible rollbacks and partial advances, the execution time is variable. Consequently, a division unit of this type is suitable when the rest of the system can make use of this variable time. In particular, if the unit forms part of a general-purpose processor with several functional units, it is necessary to have hardware support for control of dependencies. In the evaluation we determine the average execution time assuming a uniform distribution of operands.

The effectiveness of an implementation depends on a variety of factors, such as the time and cost of the speculation function, the probability of correct speculation, the time and cost of error detection, and the time for correction.

We develop the method and evaluate alternative possibilities. To evaluate its effectiveness we present some examples of implementations and compare with the implementation of conventional algorithms using the same technological constraints.

We now review very briefly the well-known division algorithm, mainly to establish the notation used. We use the standard recurrence

$$w[j+1] = rw[j] - q_{j+1}d \qquad w[0] = x \qquad (1)$$

where $w[j]$ is the residual after the j-th iteration, r is the radix, q_{j+1} is the new quotient digit, d is the divisor, and x is the dividend. We assume that x, d, and q are normalized fractions.

To have a fast iteration a redundant adder is used. In this paper we use a carry-save adder, although a similar development could be done for other redundant representations of $w[j]$.

The quotient digit is a signed digit $|q_{j+1}| \leq a$, with redundancy factor $\rho = a/(r-1)$. This requires that $w[0] \leq \rho d$, which is obtained by shifting the dividend. Moreover, for this case, the convergence of the algorithm requires the residual to be bounded so that

$$|w[j]| \leq \rho d \qquad (2)$$

The quotient digit q_{j+1} is determined by a quotient-digit selection function. This function depends on an estimate of the residual and on an estimate of the divisor, that is

$$q_{j+1} = sel(\hat{w}, \hat{d}) \qquad (3)$$

*Partially supported by the Ministry of Education of Spain (CICYT, TIC 91-1036)

The estimates are usually obtained by truncating the corresponding values. The number of bits of these estimates increases with the radix as shown in Table 1. This lengthens the delay of the function, so that practical implementations are limited to radix 2 and radix 4 stages. Examples of the number of bits required are shown in Table 1.

radix	2	4	8	16
a	1	2	7	10
ρ	1	2/3	1	2/3
# bits divisor	0	3	3	6
# bits residual	4	7	8	10

Table 1: Number of bits required for quotient-digit selection functions.

2 Scheme for Division with Speculation of Quotient Digits

We now describe the basic scheme (without partial advance) and then introduce the partial advance.

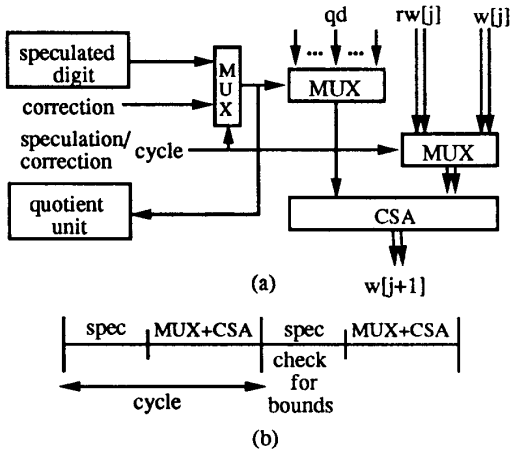


Figure 1: Basic scheme. (a) Block diagram, (b) Timing diagram

2.1 Basic Scheme

The basic scheme is shown in Figure 1. As mentioned in the introduction, the quotient digit is speculated (how this is done is described in the next section) and used in the recurrence to produce the speculated next residual as follows (speculation cycle):

$$w^s[j+1] = rw[j] - q^s_{j+1}d \qquad (4)$$

At the end of the iteration, we decide whether the speculated digit is correct by determining whether the residual is inside the allowed bounds (see expression 2). If it is inside the bounds,

$$w[j+1] = w^s[j+1] \qquad (5)$$

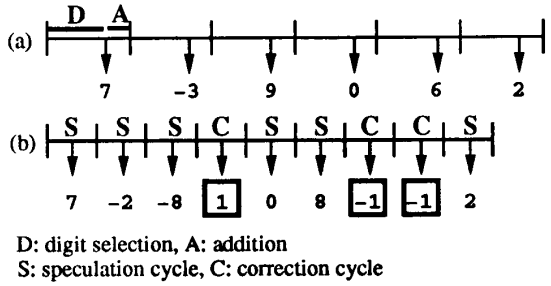


Figure 2: Timing diagram for (a) conventional division ($r = 16$) and (b) quotient-digit speculation

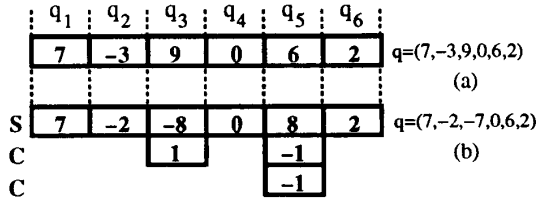


Figure 3: Quotient-digit generation

and the next iteration is performed. On the other hand, if the residual is out of bounds the quotient digit and the residual have to be corrected. A correction cycle is performed as follows:

$$w[j+1] = w^s[j+1] - q^c_{j+1}d \qquad (6)$$

In this case, the digit q^c_{j+1} has the same weight as the digit obtained in the speculation cycle and, thus, the correct digit is $q_{j+1} = q^s_{j+1} + q^c_{j+1}$ The amount of correction performed, q^c_j, is discussed in the next section. Consequently, when there is an error (incorrect speculation), more than one cycle is needed to obtain the correct quotient digit and the correct next residual. As indicated in Figure 1.b, the check for the bound can be overlapped with the speculation of the next quotient digit.

In Figure 2, we show the timing diagram of an example division and in Figure 3 the quotient digits produced. Case a) corresponds to a conventional division in which a correct quotient-digit is computed in each cycle. The second diagram corresponds to the basic scheme with speculation. As can be seen, the cycle time for this case is smaller than for the conventional case because the computation of the speculated digit is faster than the computation of the correct digit. In the example, the first two speculated digits are correct; the third digit is incorrect, producing a $w^s[3]$ which is out of bounds. This results in a correction of q_3 by $+1$ and the corresponding correction of $w^s[3]$.

This correction is sufficient, as indicated by the fact that the corrected residual is inside the bounds. Then, the next speculated digit is correct and after that another error is detected and corrected. In this case, two cycles are required for the correction, as the residual obtained after the first cycle of correction is still out of bounds.

2.2 Scheme with Partial Advance

In the basic scheme presented, two situations can occur: either we advance one digit of the quotient (when the speculation is correct) or we use the next cycle to correct the digit (no advance). So, at least one complete additional cycle is required to correct an incorrect speculation, even if the error is very small. The corresponding delay is reduced by the scheme with partial advance. In this scheme, when the error is small, a third situation is allowed, namely the advance of a number of bits which is less than a whole digit; we call this *partial advance*[1].

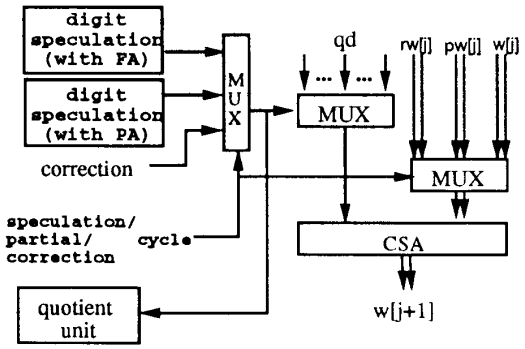


Figure 4: Scheme with partial advance

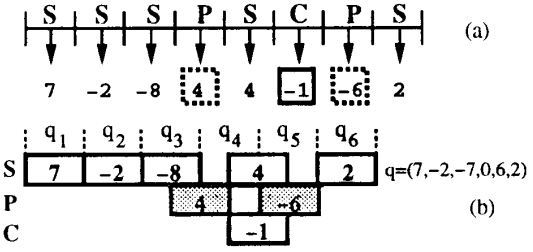
The amount of advance is selected so that the next residual is bounded. The partial-advance iteration is

$$w[j+1] = pw[j] - q^p_{j+1} d \qquad (7)$$

where $\log_2 p$ is the number of bits of the partial advance. In this case, the digit q^p_{j+1} has an overlap of $\log_2 \frac{r}{p}$ bits with the previous digit.

A block diagram for this scheme is shown in Figure 4. Note that two different speculations are performed depending whether there is a full advance ($\log_2 r$ bits) or a partial advance ($\log_2 p$ bits). The timing diagram of Figure 5.a and the quotient digit computations of Figure 5.b, show an example with partial advance. The first two digits are correct, whereas the third is incorrect. Consequently, it is not possible to advance a whole digit. Two possibilities exist: no advance or partial advance. In this case, the error is sufficiently small so that a partial advance can be made (we consider a partial of two bits, $log_2 p = 2$). The new speculated digit is 4, which overlaps with the

[1] A variation of this idea was suggested to us by Paolo Montuschi



S: speculation cycle, C: correction cycle, P: partial advance

Figure 5: Quotient-digit generation with partial advance (2 bits)

incorrect digit. This new digit is correct. Then a new speculation ($q = 4$) is incorrect and in this case no advance is possible (because the error is too large). A correction cycle is required, and after that, with still the residual out of bounds, another partial advance can be performed ($q = -6$). Finally, the last digit is correct.

3 Speculation of Quotient Digit and Error Detection/Correction

3.1 Speculation of Quotient Digit

As indicated in the previous section, the idea is to reduce the delay of the quotient-digit selection by using a simpler function which gives a correct value with high probability. To achieve this, we use

$$q^s_{j+1} = spec(\hat{w}^s, \hat{d}^s) \qquad (8)$$

where $\hat{w}^s$ and $\hat{d}^s$ have fewer bits than the $\hat{w}$ and $\hat{d}$ of expression (3), respectively.

The question is how to find such a speculation function, combining a low delay and a reasonable probability of speculating the correct value. There are many parameters in this choice of function, including characteristics of the implementation. Moreover, it involves a tradeoff between delay and cost. Therefore, it is not possible to search for an optimal solution, and we have to be content with finding one that is satisfactory. To simplify the process we perform it in two stages, namely, first we determine, for various number of bits of the estimates, the function that gives the highest probability of success and then we use this function in implementations.

For a given number of bits of the estimates, the function that produces the highest probability of correct speculation, can be determined theoretically or empirically. A possible theoretical approach is as follows (see Figure 6):

- For a particular value of $\hat{w}^s$ determine the range of values of $w[j]$. This defines a vertical strip in Figure 6.

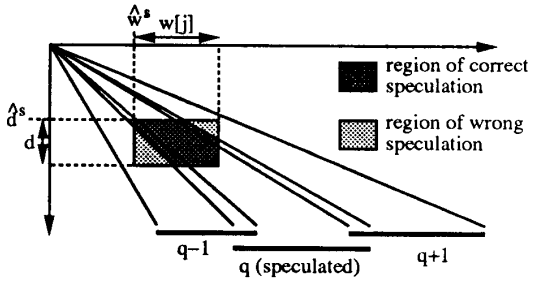


Figure 6: P-D diagram to determine the probability of correct speculation.

- For a particular value of $\hat{d}^s$ determine the range of values of d. This defines a horizontal strip in Figure 6.
- Determine the areas corresponding to the intersection of the rectangle formed by the two strips above and the selection intervals for different values of q_{j+1}. Select for q^s_{j+1} the value that corresponds to the largest area.

This approach has the disadvantage that it assumes that all values of the residual are equally likely. To avoid such an assumption, which might be even less true with speculation than with conventional division, we determined the speculation function empirically.

A possible empirical determination of the speculation function is to perform divisions using a conventional SRT algorithm. Then, for each pair of values $(\hat{w}^s, \hat{d}^s)$ select as q^s the value which occurs most frequently. However, this process is not accurate because the residual values obtained when executing a conventional algorithm are not the same as those produced when speculation is used. To eliminate the corresponding error, we determine the speculation function as follows:

- Build a speculation table (Table-spec) which, for each pair of values of $(\hat{w}^s, \hat{d}^s)$, has as entry the speculated quotient digit. Initialize this table with any value, for example all 0.
- Perform several iterations of the following process until a stable speculation table is obtained:
  1. Perform simulations of divisions in which the dividend and the divisor are selected at random.
  2. In these divisions, use as quotient digit the value obtained from Table-spec.
  3. Build a matrix with a row for each pair $(\hat{w}^s, \hat{d}^s)$ and a column for each value of q^s_{j+1}.
  4. For each residual and divisor obtained during the simulation, determine the possible correct values of the quotient digit and increment the corresponding entries in the matrix.
  5. At the end of the iteration, for each row of the matrix select for quotient-digit speculation the value that has the maximum entry in the row. Update the speculation table.

For each simulation, 10^5 divisons were executed; for each division, the dividend and divisor were randomly generated and a 54-bit quotient was calculated. Only few iterations (between 3 and 5) were required to obtain a stable speculation function. The technique of multiple independent repetitions was used to obtain results with a confidence level higher than 0.98.

Table 2 shows probabilities of success for the best function for several cases. Since $\hat{w}^s$ has a *carry-save* representation, different number of bits can be used from the *sum* and from the *carry*. When sum and carry bits are expressed in parenthesis, (s-c), the corresponding bits are first assimilated and then used by the speculation function. Since the probability of success is relatively high and the number of bits significantly smaller than those of Table 1, the approach looks promising.

radix	4	8	16
a	2	5	12
# bits $\hat{w}^s$ (s-c)	4-4	5-4	(6-5)
# bits $\hat{d}^s$	0	1	1
prob. of success	0.91	0.86	0.77

Table 2: Probability of success of the quotient-digit speculation function.

3.2 Error Detection

Since the speculation is not always correct it is necessary to detect the error. Two approaches for this seem possible:

1. Compute the correct value of the quotient digit and compare with the speculation. This scheme is not convenient because, for high radix, the exact quotient-digit selection is complex and slow and because there are cases in which more than one value is correct.
2. Determine whether the next residual is within the allowed bound.

We chose this second approach. We need to assure that the speculation is accepted only if

$$|w^s[j+1]| \leq \rho d \qquad (9)$$

To have a fast comparison, this determination has to be performed using a truncation of $w^s[j+1]$ and d. Let us call $\hat{w}^c$ and $\hat{d}^c$ these truncated values. We now determine the minimum number of bits that these estimates require. Let us call f the number of fractional bits of each estimate. Since a *carry-save* representation is used for w, $\hat{w}^c$ is calculated as the sum of the

most significant bits of the two bit vectors representing w, and therefore

$$w \in [\hat{w}^c, \hat{w}^c + 2^{-f+1}) \tag{10}$$

On the other hand, d is nonredundant so

$$d \in [\hat{d}^c, \hat{d}^c + 2^{-f}) \tag{11}$$

Consequently, relation (9) is satisfied if

$$-\rho\hat{d}^c \leq \hat{w}^c \leq \rho\hat{d}^c - 2^{-f+1} \tag{12}$$

Therefore, these are the comparisons that have to be performed[2].

In addition, the continuity condition has to be preserved. This requires that the difference between the upper and lower bound of the residual has to be at least equal to d. That is,

$$\rho\hat{d}^c - 2^{-f+1} + \rho\hat{d}^c \geq d \tag{13}$$

and since $d < \hat{d}^c + 2^{-f}$, it is sufficient that

$$2\rho\hat{d}^c - 2^{-f+1} \geq \hat{d}^c + 2^{-f} \tag{14}$$

and we get

$$(2\rho - 1)\hat{d}^c \geq 3 \times 2^{-f} \tag{15}$$

Since $\hat{d}^c \geq 1/2$, it is sufficient that

$$2^{-f} \leq \frac{1}{3}(\rho - \frac{1}{2}) \tag{16}$$

We now determine the number of integer bits of the estimates. Since intermediate residuals produced by speculative division might have values larger than ρd, additional integer bits are required. Let us call e the maximum error produced by a speculation, that is, e is the maximum difference between a speculated value and a correct quotient digit. Moreover call $\overline{M}$ and $\underline{M}$ the highest and lowest values of $w^s[j+1]$ obtainable when the recurrence is performed with a speculation having the maximum error. Since a correct residual is $w^s[j+1] \leq \rho d$, then

$$\overline{M} \leq \rho d + ed \tag{17}$$

and as $d < 1$,

$$\overline{M} < \rho + e \tag{18}$$

Similarly,

$$\underline{M} > -(\rho + e) \tag{19}$$

and thus, the number of integer bits (i) required to represent wrongly speculated residuals is

$$i = \lceil \log_2(\rho + e) \rceil + 1 \tag{20}$$

[2]Only the truncated value of $\rho\hat{d}^c$ is required for the implementation of the comparisons.

For $\rho \leq 1$ this reduces to

$$i = \lceil \log_2(e+1) \rceil + 1 \tag{21}$$

The value e is calculated as follows. Given a speculation function to obtain the quotient digit, for each rectangle R defined by $\hat{w}^s$ and $\hat{d}^s$ in the P-D diagram (see Figure 6) we calculate

$$e_R = \max_{q \in \{q_R\}} |q - spec(\hat{w}^s, \hat{d}^s)| \tag{22}$$

where $\{q_R\}$ is the smallest set of quotient digits that cover the rectangle. Finally, $e = \max e_R$. Table 3 shows several examples of the minimum number of bits required.

r	a	ρ	f
4	2	2/3	5
4	3	1	3
8	5	5/7	4
16	10	2/3	5

e	i
0	1
1	2
2...3	3
4...7	4

Table 3: Number of bits for comparison.

In summary, to determine whether there is an error it is necessary to have two comparators that perform the comparisons specified in (12). The estimate $\hat{w}^c$ has i integer bits and f fractional bits, whereas $\hat{d}^c$ has f fractional bits (the most significant being always 1).

Since the comparisons are done with truncated versions of $w^s[j+1]$ and d, there are cases in which the speculation is correct but the comparison (to be conservative) fails. Consequently, the probabilities of successful speculation are somewhat smaller than those presented in Section 3.1. Now these probabilities depend not only on the number of bits of $\hat{w}^s$ and $\hat{d}^s$ but also on f, and can be increased somewhat by using a larger value of f. An example is shown in Table 4.

f	4	5	6	7	≥ 8
prob. of success	0.73	0.75	0.76	0.76	0.77

Table 4: Variation of probability of success ($r = 16$, $a = 12$, (6-5) bits for $\hat{w}^s$ and 1 bit for $\hat{d}^s$)

3.2.1 Error Detection for Partial Advance

We now determine the range of error that permits a partial advance of $\log_2 p$ bits. As discussed above, we determine the error by looking at $w^s[j+1]$. In the basic scheme, if $w^s[j+1]$ is out of bounds, the speculated digit is corrected until a residual inside the bounds is obtained. However, if the error is small we can accept the speculated digit and make a partial advance of $log_2 p$ bits if

$$|pw^s[j+1]| \leq r\rho d \tag{23}$$

because this is inside the bounds for the next iteration. Consequently, we can advance $log_2 p$ bits if

$$|\frac{p}{r}w^s[j+1]| \leq \rho d \tag{24}$$

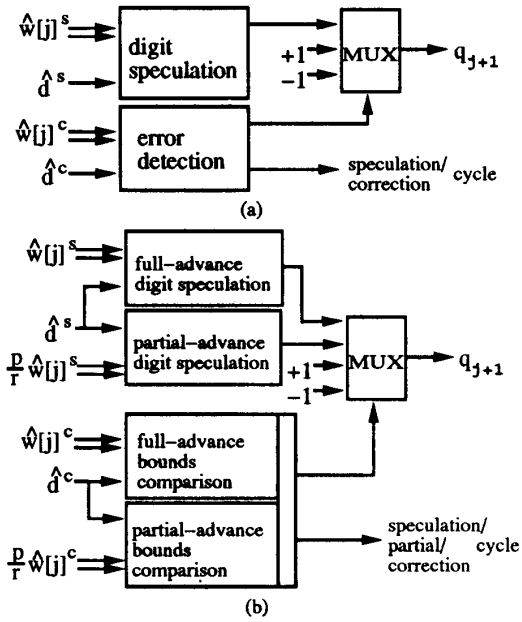


Figure 7: Units for digit speculation. (a) Basic scheme; (b) With partial advance (q_{j+1} denotes q^s_{j+1}, q^p_{j+1}, or q^c_{j+1} depending on the type of cycle).

These comparisons are done in a similar manner as those previously discussed, that is, using estimates of $w^s[j+1]$ and d.

3.3 Correction of Quotient Digit

Since with the detection scheme presented in the previous section, when there is an error the correct digit is not known, it is necessary to perform an incremental correction and to check again whether a correct digit is obtained. We have chosen to correct the quotient digit by $+1$ or -1, depending on the sign of the residual estimate. This method requires that in some cases more than one correction cycle be performed. However, this situation is infrequent so that the method is suitable.

Figure 7 depicts a block diagram of the circuit required for digit speculation, error detection, and digit correction. It consists of a multiplexor that selects between the speculated digit (q^s) and the correction (± 1) according to the result of the comparison with the bounds. In the scheme with partial advance, two speculation tables (for q^s and q^p) and two comparisons with the bounds (for full and partial advance) are required.

Skipping Quotient Digits

The correction performed because of incorrect speculation can also be used to avoid having to produce some divisor multiples that are difficult to generate. For example, it is possible to implement a radix-8 divider with a=5 using only one adder (and therefore only multiples that are powers of two) by avoiding to speculate the quotient-digit values 3 and 5. When these values are required, they are obtained by speculating some neighboring value and performing a correction cycle. We have performed some designs that show that this approach is beneficial because it reduces the critical path, while not increasing significantly the number of cycles.

4 Evaluation

To evaluate the speed advantage of the scheme we have described, we performed some implementations for 54-bit dividers (this value affects only the area estimates). Since there are many parameters that specify a particular implementation, we have not done a complete analysis of the solution space but performed some reasonable designs to evaluate and compare.

All designs have used the same technology and design tools. In particular, we have used a 1μm standard-cell CMOS library [5] (size of a 2-input NAND gate is $12.5 \times 47.5\mu m^2$, delay of an inverter is 0.15 ns). Some simple modules have been designed by hand (multiplexors and CSAs) while *misII* [2] has been used for the synthesis of the quotient-digit selection functions and the comparators. *MisII* has always been guided to optimize delay at the expense of increasing the area. Fan-in and fan-out capacitances (but no routing) have been considered for delay calculations. We only give the final results of the designs. More details can be found in [3].

4.1 Basic Scheme

We now evaluate the execution time of the basic scheme and compare with conventional SRT division units.

Since we do not specify the number of bits of the quotient, we compute the average execution time per quotient bit. This time is given by

$$T = \frac{C_d}{log_2 r} \times t_c \qquad (25)$$

where C_d is the average number of cycles per quotient digit (this includes the speculation cycles and the correction cycles) and t_c is the delay of one cycle.

In Table 5 we report the results of some of these designs. The delay does not include register loading, because the stages can be used in an implementation with unfolded iterations and even in a self-timed implementation [12]. Because of this possibility of unfolding, for our comparisons we normalize to delay and area per quotient bit. For the speculative approach, we show only cases that produce a faster implementation than the conventional approach and when increasing the radix produces an improvement in delay/bit.

We were surprised by the good performance obtained for the conventional radix-8 implementation; this is achieved by guiding *misII* to provide a different delay for both components forming the radix-8 quotient digit. We conclude that the fastest speculative implementation is about 10% faster than the

digit select.	conventional			speculative	
radix	2	4	8	8	16
a	1	2	7	5	12
# of CSAs	1	1	2	1	2
# bits $\hat{w}^s$	(4-4)	(7-7)	(8-8)	5-4	(6-5)
# bits $\hat{d}^s$	0	3	3	1	1
cmp: (i,f)	-	-	-	4,5	3,5
cycles/digit	1	1	1	1.27	1.30
cycle delay	6.3	8.4	11.3	8.5	10.7
cell area	500	840	2100	1800	2000
delay/bit	6.3	4.2	3.8	3.6	3.5
area/bit	500	420	680	580	500
speedup	1.00	1.50	1.66	1.75	1.80
area factor	1.00	0.84	1.36	1.16	1.00

Table 5: Characteristics of designs. (Area unit = Area of a 2-input NAND gate, delays are given in ns).

fastest conventional one, with the added advantage of reducing the area per bit by 25%.

4.2 Scheme with Partial Advance

To calculate the average number of cycles per quotient digit, we perform simulations that are similar to those discussed in Section 3.1. Now three types of cycles exist, namely, full cycles (with probability p_f), partial cycles (with probability p_p), and non-advance cycles (with probability p_n). The average number of cycles per digit is then calculated as

$$C_d = \frac{\log_2 r}{p_f \log_2 r + p_p \log_2 p} \tag{26}$$

Table 6 shows the number of cycles for several radices and several values of p.

Radix	4	8		16	
$\log_2 p$	1	2	1	3	2
p_f	0.90	0.64	0.73	0.69	0.69
p_p	0.10	0.32	0.27	0.24	0.31
p_n	0.00	0.04	0.00	0.07	0.00
C_d	1.05	1.17	1.22	1.15	1.18

Radix	32			64		
$\log_2 p$	4	3	2	5	4	3
p_f	0.63	0.66	0.61	0.47	0.49	0.53
p_p	0.27	0.34	0.39	0.30	0.48	0.47
p_n	0.10	0.00	0.00	0.23	0.03	0.00
C_d	1.18	1.16	1.31	1.39	1.23	1.32

Table 6: Speculative division with partial advance (the designs correspond to the speculative dividers in tables 5 and 7)

We have performed several designs chosing from Table 6 the values of p that produce the lowest average number of cycles per digit. Table 7 shows the characteristics for implementations that are faster than those described in Section 4.1. The results indicate that this method produces speedups with an increase in area;

radix	16	32	64
a	12	22	37
number of CSAs	2	2	2
# bits $\hat{w}^s$	(6-5)	(7-7)	(7-7)
# bits $\hat{d}^s$	1	2	3
cmp: (i,f)	3,5	4,5	4,6
partial adv. $(\log_2 p)$	3	3	4
cycles/digit	1.15	1.16	1.23
cycle delay	10.7	13.4	14.3
cell area	3120	3860	5910
delay/bit	3.1	3.1	2.9
area/bit	780	772	985
speedup	2.03	2.03	2.17
area factor	1.56	1.54	1.97

Table 7: Designs of speculative dividers with partial advance (Speedup and area factor are given w.r.t. radix 2).

this increase is mainly due to the duplication of the speculation function. The radix-64 implementation is 30% faster than the fastest conventional implementation (radix 8) with about 44% more cell area per quotient bit.

5 Design Example

We now present the implementation details of one design. For simplicity, a scheme without partial advance is explained. We have chosen the one that gives the smallest delay per quotient bit, namely, the speculative radix-16 divider shown in Table 5. The block diagram is shown in Figure 8.

The quotient digit generated by the speculation function is $q_h + q_l$, where $q_h \in \{-8,-4,0,4,8\}$ and $q_l \in \{-2,-1,0,1,2\}$. Therefore, the values $\{-12,-11,11,12\}$ are always obtained after corrections of the initially speculated digit. Although this increases somewhat the number of correction cycles, the use of $a = 12$ results in a larger overlap than $a = 10$, which makes the implementation of the speculation function simpler and faster. Moreover, the limited precision of the comparisons (error detection) reduces the range of "accepted" residuals and, consequently, diminishes the probability of requiring quotient digits near $\pm a$. Exploring several designs, we found $a = 12$ to be the best trade-off for radix 16.

Table 8 reports area and delay characteristics of this design. As q_h is the highest-weight component of the quotient digit, the speculation function is simpler and faster than for q_l. When synthesizing the speculation function, *misII* has been guided to reducing the delay of q_h which, in this case, is in the critical path.

The CSA has been designed as a radix-2 full adder. Its delay (2.2 ns) is determined by two cascaded XOR gates (1.1 ns each). However, the outputs of the $q_i d$ multiplexors have been connected to the last gate in order to reduce the critical path (the same optimization has been used for the conventional designs). This approach cannot be used with the residual, since the redundant representation requires two signals.

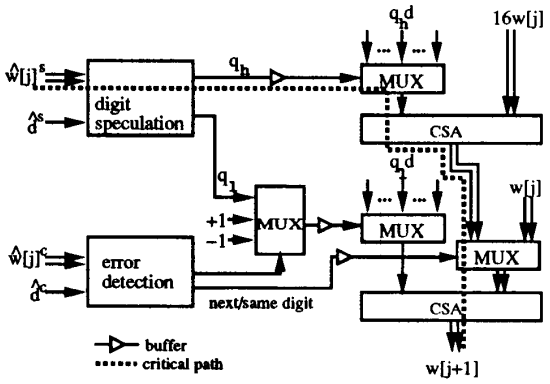


Figure 8: Block diagram for a speculative radix-16 divider.

Module	Area	Delay
digit speculation	270	4.8* (for q_h) 6.3 (for q_l)
error detection	210	5.3
MUX for $q_i d$	2×310	0.9*
MUX for $w[j]$	90	0.9*
MUX for q_l	4	0.9
CSA (56 bits)	2×380	2.2* (from residual) 1.1* (from $q_i d$)
Buffer	9×3	0.8*
Iteration (Total)	2000	10.7

Table 8: Area and delay for the speculative radix-16 divider (* indicate delays in the critical path)

6 Summary and Conclusions

The division method we have presented is based on the speculation of the quotient digit and on a roll-back when the speculation is incorrect. Because of the reduction in complexity of the quotient-digit selection, this can result in faster implementations with higher radices than for the conventional approach. Moreover, a reduction in the number of adders required is possible by speculating only a reduced set of quotient-digit values.

We have discussed the way to determine an optimal speculation for a given estimate of the residual and of the divisor. Moreover, we detect whether the speculation is correct by determining whether the next remainder is inside the required bound.

The designs performed show a radix-16 divider that is about 10% faster than the fastest conventional divider (radix 8). Moreover, the area per quotient bit is reduced by 25%.

The speed can be further improved by doing partial advance when the speculation is incorrect. We developed the condition required for partial advance and described an implementation that replicates the quotient-digit selection.

We performed designs which show that a radix-64 divider with speculation and partial advance of four bits is about 30% faster than the fastest conventional case with an area/bit increase of about 45%.

Acknowledgment

We thank Paolo Montuschi for his valuable discussions and comments.

References

[1] D.E. Atkins, "Higher-Radix Division Using Estimates of the Divisor and Partial Remainder," *IEEE Trans. Computezs*, Vol. C-17, pp. 925-934, Oct. 1968.

[2] R.K. Brayton, R. Rudell, A. Sangiovanni-Vincentelli, and A.R. Wang, "MIS: A multiple-level logic optimization system," *IEEE Trans. Computer-Aided Design*, vol. CAD-6, pp. 1062-1081, Nov. 1987.

[3] J. Cortadella and T. Lang, "Division with Speculation of Quotient Digits," *UPC/DAC Tech. Report*, 1993.

[4] M. Ercegovac and T. Lang, "Simple Radix-4 Division with Operands Scaling," *IEEE Trans. on Computers*, Vol. C-39, pp. 1204-1208, September 1990.

[5] European Silicon Structures, *ES2 ECPD10 Library Databook*, April 1991.

[6] J. Fandrianto, "Algorithm for High Speed Shared Radix 4 Division and Radix-4 Square Root," *Proc. 8th IEEE Symposium on Computer Arithmetic*, Como, Italy, pp. 73-79, May 1987.

[7] E.W. Krishnamurthy, "On Range-Transformation Techniques for Division," *IEEE Trans. on Computers*, Vol. C-19, pp. 227-231, March 1970.

[8] D.W. Matula, "Design of a Highly Parallel Floating Point Arithmetic Unit," *Symposium on Combinatorial Optimization Science and Technology (COST)*, April 1991.

[9] A. Svoboda, "An Algorithm for Division," Inf. Proc. Mach., Vol. 9, pp. 25-32, 1963.

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

[11] S. Waser and M.J. Flynn, *Introduction to Arithmetic for Digital System Designers*, Holr, Rinehart, and Winston, New York, 1985.

[12] T.E. Williams and M.A. Horowitz, "A 160ns 54bit CMOS Division Implementation Using Self-Timing and Symmetrically Overlapped SRT Stages," *Proc. 10th Symposium on Computer Arithmetic*, pp. 210-217, June 1991.