It Takes Six Ones To Reach a Flaw

Tim Coe

Vitesse Semiconductor Corporation
741 Calle Plano
Camarillo, CA 93010

Ping Tak Peter Tang

Mathematics and Computer Science
Argonne National Laboratory
Argonne, IL 60439

Abstract

The intial release of the Pentium™ *processor has a flaw in its radix-4 SRT division implementation. It is widely-known that five entries were missing in the lookup table, yielding reduced-precision quotients occasionally. In this paper, we use mathematical techniques to analyze the divisors that can possibly cause failures. In particular, we show that Bits 5 through 10 (where Bit 0 is the MSB) of such divisors must be all ones. This result is useful in compiler-level software patches for systems with unreplaced chips; and we believe that the techniques used here are applicable in analyzing SRT division as well as other hardware algorithms for floating-point arithmetic.*

1 Introduction

SRT is a widely used algorithm for binary floating-point division [1, 2, 3]. Given a dividend/divisor pair p/d, $1 \leq p, d < 2$, the SRT algorithm generates a sequence of partial remainders and quotient digits, (p_i, q_i), $i = 0, 1, 2, \ldots$, satisfying

$$p_{i+1} = \text{radix}(p_i - q_i d), \quad |p_{i+1}/d| \leq \text{threshold}$$

where $p_0 \stackrel{\text{def}}{=} p$. The "threshold" is defined once we choose the two key design parameters of a particular SRT implementation, namely, "radix" and the set of allowable quotient digits q_i's.

A common implementation for radix $= 4$ is to allow $|q_i| \leq 2$, that is, $q_i \in \{-2, -1, 0, 1, 2\}$. This gives threshold $= 8/3$ (cf. [4], p. 134). Clearly, then, given any p_i, $|p_i| \leq \frac{8}{3}d$, we must choose $|q_i| \leq 2$ such that $|4(p_i - q_i d)| \leq \frac{8}{3}d$. The following check for legitimacy of quotient choices can be easily verified:

$$|q_i| \leq 2 \quad \text{OK if} \quad q_i - \frac{2}{3} \leq \frac{p_i}{d} \leq q_i + \frac{2}{3}. \qquad (1)$$

For example, $q_i = 0$ is legitimate when $-\frac{2}{3}d \leq p_i \leq \frac{2}{3}d$, and $q_i = 1$ is legitimate when $\frac{1}{3}d \leq p_i \leq \frac{5}{3}d$. The fact that some values of p_i allow for two legitimate choices of quotient digits is a well known advantage of SRT. The main reason is that q_i can now be decided based only on approximate values of p_i and d. For a specific case, we choose $P = \frac{k}{8}$ and $D = 1 + \frac{\ell}{16}$ to be integer multiples of $\frac{1}{8}$ and $\frac{1}{16}$ such that

Table 1: Quotient Selection P-D Table

q_i	when P lies in
2	$[\ \frac{4}{3}D_+\ ,\ \frac{8}{3}D_+\ )$
1	$[\ \frac{1}{3}D_+\ ,\ \frac{4}{3}D_+\ )$
0	$(\ -\frac{1}{4} - \frac{1}{3}D_+\ ,\ \frac{1}{3}D_+\ )$
-1	$(\ -\frac{1}{4} - \frac{4}{3}D_+\ ,\ -\frac{1}{4} - \frac{1}{3}D_+\ ]$
-2	$(\ -\frac{1}{4} - \frac{8}{3}D_+\ ,\ -\frac{1}{4} - \frac{4}{3}D_+\ ]$

$$P \leq p_i < P + \frac{1}{4} \stackrel{\text{def}}{=} P_+ \ \&\ D \leq d < D + \frac{1}{16} \stackrel{\text{def}}{=} D_+.$$

We illustrate how q_i can be decided by P and D alone. Clearly, we have $-\frac{1}{4} - \frac{8}{3}D_+ < P < \frac{8}{3}D_+$ and

$$\begin{array}{lllllll} P/D_+ & \leq & p_i/d & \leq & P_+/D & \text{for} & 0 \leq P \\ P/D & \leq & p_i/d & \leq & P_+/D_+ & \text{for} & 0 \geq P_+ \end{array}.$$

Combining these inequalities with (1) and using a convention that the quotient digit of larger magnitude is favored whenever two legitimate choices exist, we obtain a digit selection table (P-D table) exhibited in Table 1. To implement the quotient selection rule, one simply stores the quotient digits in a P-D table where

$$\begin{array}{lll} D & = & 1 + \frac{\ell}{16}, \quad \ell = 0, 1, \ldots, 15; \\ P & = & \frac{k}{8}, \qquad -\frac{1}{4} - \frac{8}{3}D_+ < \frac{k}{8} < \frac{8}{3}D_+. \end{array}$$

One common approach in implementing the division iterations is to keep the partial remainders p_i's in a carry-save format. Mathematically, p_i is represented as the sum of a truncated part P_i, a carry part C_i, and a sum part S_i:

$$p_i = \text{sum of} \begin{array}{ll} P_i & \frac{k}{8} \\ C_i & 0.000c_4c_5\ldots c_L \\ S_i & 0.000s_4s_5\ldots s_L \end{array}$$

1063-6889/95 $4.00 © 1995 IEEE

Figure 1: Carry-Save Implementation of SRT

$$
\begin{array}{c|c|l}
P_i & k/8 & \\
C_i & 0.000c_4c_5 & c_6c_7\ldots \\
S_i & 0.000s_4s_5 & s_6s_7\ldots \\
-q_id & e_3e_2e_1e_0.f_1\ldots f_5 & f_6f_7\ldots \\
\hline
P_{i+1} & 4\begin{pmatrix}\text{sum above }+\\ \text{carry}(c,s,f)_6\end{pmatrix} & \begin{array}{l} c'_4c'_5\ldots \\ s'_4s'_5\ldots \end{array}
\end{array}
$$

Figure 2: Compiler Patch S:

```
If d is in one of the 5 regions
    calculate (x * (15/16))/(y * (15/16))
else
    calculate x/y
End if
```

Figure 3: Compiler Patch F:

```
If d is in one of the 5 regions AND
d5 = d6 = ··· = d10 = 1
    calculate (x * (15/16))/(y * (15/16))
else
    calculate x/y
End if
```

Initially,

$$
\begin{aligned}
p_0 = p &= 1.p_1p_2\ldots p_L \\
P_0 &= 1.p_1p_2p_3, \\
S_0 &= 0.000p_4p_5\ldots p_L, \text{ and} \\
C_0 &= 0.00000\ldots 0.
\end{aligned}
$$

Calculation of $p_{i+1} = 4(p_i - q_id)$ is straightforward for $q_i \le 0$: Let

$$-q_id = e_3\,e_2\,e_1\,e_0\,.\,f_1\,f_2\,\ldots\,f_L.$$

The variables P_{i+1},C_{i+1}, and S_{i+1} are produced as in Figure 1 where carry(b_1, b_2, b_3) is 1 if the three input bits consist of two or more ones; and carry = 0 otherwise. When $q_i > 0$, one can form $-q_id$ as the one's complement of $q_id = e_1\,e_0\,.\,f_1\,f_2\,\ldots\,f_L$ plus 2^{-L}, that is $-q_id = \bar{e}_1\,\bar{e}_0\,.\,\bar{f}_1\,\bar{f}_2\,\ldots\,\bar{f}_L$ plus 2^{-L}. The value 2^{-L} can be put into the corresponding positions of C_i or of C_{i+1} (by deferring the action). One can verify that these positions are always zero before accepting 2^{-L}. The techniques discussed here are actually quite standard (cf. [5], pp. 268–270, or [6] Chapter 3).

2 Description of Problem

The previous section in fact functionally describes the SRT implementation on the Pentium™ processor. Due to a by-now famous mishap, in the processor's initial release, the five quotient digits stored in the P-D table for the five (P, D) pairs

$$
\begin{aligned}
D &= 1 + \tfrac{\ell}{16}, \quad \ell = 1, 4, 7, 10, 13 \\
P &= \tfrac{8}{3}D_+ - \tfrac{1}{8} \stackrel{\text{def}}{=} P_{\text{Bad}}
\end{aligned}
$$

were 0 when in fact they should have been 2. Consequently, for divisors d lying in the five corresponding regions of $[D, D_+)$, reduced-precision quotients (failure) are delivered occasionally (cf. Corollary 1). More precisely, for these divisors d, failure occurs whenever during a division process, at some i prior to completion, we encounter $P_i = P_{\text{Bad}}$. (According to empirical studies, divisors and dividends uniformly distributed in $[1, 2)$ give a probability of failure in the order of 10^{-9}.)

Since for any d lying in one of the five critical regions, $\frac{15}{16}d$ lie outside of them, a compiler-level patch can replace occurrences of x/y where $|y| = 2^nd$, $1 \le d < 2$, by Figure 2. However, executing (x*(15/16))/(y*(15/16)) not only involves extra multiplications but also requires the saving and restoring of several status variables (such as precision control) in order to ensure full IEEE compliance [7]. Thus, this replacement is considerably more expensive than it might seem. The patch in Figure 2 requires the slow substitute with probability 5/16 in general, degrading performance noticeably. Our main result is that in order for a failure to occur, the divisor $d = 1.d_1d_2\ldots d_L$ not only has to satisfy the obvious requirement that $1.d_1d_2d_3d_4 = 1 + \frac{\ell}{16}$ for $\ell = 1, 4, 7, 10, 13$, but that d_5 through d_{10} must all be ones. Thus the patch can be modified by a faster version given in Figure 3. Consequently, the slow substitute is invoked only with probability $2^{-6} \times 5/16$, rendering the performance degradation practically zero.

To derive our result, we analyze in the next section the (P_i, q_i) sequence just prior to the first reference to P_{Bad}. In the following section, we analyze the bit patterns of the corresponding carry and save vectors, (C_i, S_i). Inferences can then be drawn on d's bit pattern. Finally, we present two examples to illustrate our results.

3 Digit Sequence Analysis

In order to analyze the last few steps just prior to referencing the fatal P-D entries P_{Bad}, we first examine the evolution of the P_i's. From Figure 1, we see that

$$
\begin{aligned}
P_{i+1} &= 4(P_i - q_i\tilde{d}) + \frac{1}{8}\text{carry} + 0.0c_4c_5 + 0.0s_4s_5 \\
&\le 4(P_i - q_i\tilde{d}) + \frac{7}{8},
\end{aligned}
$$

Table 2: Upper Bounds on P_{i+1}

q_i	Bounds on P_{i+1}
2	$P_{i+1} < \frac{8}{3}D_+ + \frac{5}{4}$
1	$P_{i+1} < \frac{4}{3}D_+ + 1$
0	$P_{i+1} < \frac{4}{3}D_+ + \frac{7}{8}$
-1	$P_{i+1} \le \frac{8}{3}D_+ - \frac{1}{4}$
-2	$P_{i+1} \le \frac{8}{3}D_+ - \frac{1}{4}$

where $\tilde{d}$ is approximately d, taking into account the truncation and one's complement. For example,

$$\begin{array}{rcll} \tilde{d} & = & 1.d_1d_2d_3d_4d_5 & \text{for } q_i = -1, \text{ and} \\ \tilde{d} & = & 1.d_1d_2d_3d_4d_5d_6 + 2^{-6} & \text{for } q_i = 2. \end{array}$$

Combining the bounds for P_i in Table 1 and the inequality just obtained for P_{i+1}, we obtain a table of upper bounds, Table 2, on P_{i+1} for each of the five possible values of q_i.

Now consider a divisor d that belongs to one of the five critial regions of $[D, D_+)$. Let P_J be the first reference to $P_{\text{Bad}} = \frac{8}{3}D_+ - \frac{1}{8}$. Clearly, $J \ge 1$ because $P_0 \le p_0 < 2 < P_{\text{Bad}}$ for all five possible P_{Bad}'s. Our result in this section concerns the evolution of the few P_i's just prior to P_J.

Lemma 1. There is an integer $m \ge 1$ such that $J \ge m + 2$ and that the evolution of (P_i, q_i) from $i = J - m - 1$ to $i = J$ is given by

i	P_i	q_i
J	P_{Bad}	q_{bad}
$J-1$ to $J-m$	$P_{\text{Bad}} - \frac{1}{8}$	2
$J-m-1$	$-\frac{4}{3}D_+ - \frac{1}{4}$ or $-\frac{1}{3}D_+ - \frac{1}{4}$	-2 or -1

Proof of Lemma 1. From Table 2, in order for $P_J = P_{\text{Bad}}$, the only possible choice for q_{J-1} is 2. Since P_J is the first reference to P_{Bad}, we must have $P_{J-1} \le P_{\text{Bad}} - \frac{1}{8}$. Because the partial remainder p_J satisfies

$$P_J \le p_J = \frac{8}{3}d - \alpha, \quad \alpha \ge 0,$$

and that $p_i = p_{i+1}/4 + q_i d$, we have

$$p_{J-1} = \frac{8}{3}d - \frac{\alpha}{4} \ge p_J \ge P_{\text{Bad}}.$$

Consequently,

$$P_{J-1} > p_{J-1} - \frac{1}{4} \ge P_{\text{Bad}} - \frac{1}{4}.$$

The strict inequality and the fact that the P_i's are integer multiples of $\frac{1}{8}$ imply that in fact $P_{J-1} \ge P_{\text{Bad}} - \frac{1}{8}$. Moreover, we clearly have $J-1 > 0$ because $p_{J-1} \ge P_{J-1}$ which is bigger than 2 for all five possible values of $P_{\text{Bad}} - \frac{1}{8}$.

Examining Table 2 again tells us that q_{J-2} can only possibly be $2, -1$, or -2. Using the previous analysis, we see that if

$$q_{J-m} = q_{J-m+1} = \cdots = q_{J-1} = 2$$

for some $m > 1$, we must have

$$p_{J-m} \ge p_{J-m+1} \ge \cdots \ge p_{J-1},$$

forcing

$$P_{J-m} = P_{J-m+1} = \cdots = P_{J-1} = P_{\text{Bad}} - \frac{1}{8},$$

as well as $J - m > 0$. Consequently, there exist an integer $m \ge 1$ such that $J - m > 0$, $q_{J-m-1} \ne 2$, and

$$(P_i, q_i) = (P_{\text{Bad}} - \frac{1}{8}, 2), \quad i = J-m, J-m+1, \ldots, J-1.$$

Using Table 2 one more time, we must have $q_{J-m-1} = -2$ or -1. since $p_0 \ge 1$, we must have $q_0 > 0$ and thus $J-m-1 > 0$, that is, $J \ge m+2$. Finally, Table 2 shows that in order for $P_{J-m} = P_{\text{Bad}} - \frac{1}{8}$ with $q_{J-m-1} = -2$ or -1, we must have P_{J-m-1} to be the corresponding maximum possible values. Thus, P_{J-m-1} must be $-\frac{1}{4} - \frac{4}{3}D_+$ or $-\frac{1}{4} - \frac{1}{3}D_+$ for $q_{J-m-1} = -2$ or -1, respectively. This completes the proof of Lemma 1.

We further comment that in fact when $D = 1 + (4 \text{ or } 10)/16$, $-\frac{1}{3}D_+$ is not an integer multiple of $\frac{1}{8}$ and thus $q_i = -1$ implies $P_i < -\frac{1}{3}D_+ - \frac{1}{4}$. Consequently, $q_{J-m-1} = -1$ is possible only when D is one of the other three values.

4 Bit Pattern Analysis

We first show that the digit sequence established in the last section implies Bit 5 through 8 of d must be ones. This result in turns implies that $m = 1$, that is, $q_{J-2} = -1$ or -2, and that the carry and sum vectors at $J-2$, $(C, S)_{J-2}$, must each have at least 5 leading ones. This result then easily implies, in fact, that Bits 5 through 10 of d must all be ones.

Lemma 2. Bits 5 through 8 of d must be all ones and that C_{J-1} and S_{J-1} must each have at least three leading ones.

Figure 4: From $J-2$ to $J-1$

P_{J-2}	$k/8$	
C_{J-2}	$0.000\,c_4\,c_5$	$c_6\,c_7\,c_8\,\ldots$
S_{J-2}	$0.000\,s_4\,s_5$	$s_6\,s_7\,s_8\,\ldots$
$-q_{J-2}d$	$e_3\,e_2\,e_1\,e_0\,.\,f_1\,\ldots\,f_5$	$f_6\,f_7\,f_8\,\ldots$
P_{J-1}	$P_{\text{Bad}} - \frac{1}{8}$	$c'_4\,c'_5\,c'_6\,\ldots$ $s'_4\,s'_5\,s'_6\,\ldots$

Proof of Lemma 2. We consider the evolution of P_i from $i = J - m - 1$ through J. It is easy to see that because of the carry-save implementation, P_{J-m-1} together with the leading bits

$$\begin{aligned} \tilde{C}_{J-m-1} &\stackrel{\text{def}}{=} 0.000c_4c_5\ldots c_{6+3m}0\ldots0 \\ \tilde{S}_{J-m-1} &\stackrel{\text{def}}{=} 0.000s_4s_5\ldots s_{6+3m}0\ldots0 \end{aligned}$$

of C_{J-m-1} and S_{J-m-1} determine the evolution of the P_i's from $J-m-1$ through J. Thus, if we (re)initiate the division process at Step $J - m - 1$ with

$$\tilde{p}_{J-m-1} \stackrel{\text{def}}{=} (P + \tilde{C} + \tilde{S})_{J-m-1},$$

then, we still have

$$\begin{aligned} \tilde{P}_i &= P_{\text{Bad}} - \tfrac{1}{8}, \; i = J - m, \ldots, J \\ \tilde{P}_J &= P_{\text{Bad}}. \end{aligned}$$

Clearly,

$$\tilde{p}_{J-m-1} = P_{J-m-1} + \ell 2^{-(6+3m)},$$

where $0 \le \ell \le 2^{4+3m} - 2$. Now, consider the case $q_{J-m-1} = -1$. We have

$$\begin{aligned} \tilde{p}_{J-m} &= 4(\tilde{p}_{J-m-1} + d) \\ \tilde{p}_{i+1} &= 4(\tilde{p}_i - 2d), \; i = J - m, \ldots, J - 1, \end{aligned}$$

giving

$$\tilde{p}_J = 4^{m+1}\tilde{p}_{J-m-1} + \frac{1}{3}(4^{m+1} + 8)d.$$

Let $d = D_+ - \delta$, $\delta \ge 0$. Using the facts that

$$P_{J-m-1} = -\frac{1}{3}D_+ - \frac{1}{4}, \; \tilde{p}_J \ge P_{\text{Bad}} = \frac{8}{3}D_+ - \frac{1}{8},$$

we have

$$\begin{aligned} \frac{8}{3}D_+ - \frac{1}{8} \;\le\; & 4^{m+1}\left(-\frac{1}{3}D_+ - \frac{1}{4} + \ell 2^{-(6+3m)}\right) \\ & + \frac{1}{3}\left(4^{m+1} + 8\right)(D_+ - \delta). \end{aligned}$$

Using $\ell \le 2^{4+3m} - 2$, we arrive at

$$\delta \le \frac{3(1 - 2^{-m})}{8 + 4^{m+1}} 2^{-3}.$$

Thus $\delta \le 2^{-7}$ for $m = 1$ and $\delta \le 2^{-8}$ for $m > 1$. The bound $\delta \le 2^{-7}$ for all m clearly implies $d_5 = d_6 = d_7 = 1$.

Repeating the analysis for the case $q_{J-m-1} = -2$, that is, $P_{J-m-1} = -\frac{4}{3}D_+ - \frac{1}{4}$, gives $\delta \le 2^{-8}$ for all $m \ge 1$. Thus d_5 through d_8 are all ones. Therefore, at this point, we know that except for the case of $m = 1$ with $q_{J-2} = -1$ where we only know that we must have d_5 through d_7 to be ones, d_5 through d_8 must in fact be all ones for all other cases.

Using the fact that d_5 through d_7 are ones for all cases, we now show that C_{J-1} and S_{J-1} must each have at least three leading ones. This is derived by considering the generation of P_J. Refer to Figure 1 with $i = J-1$ and $i+1 = J$. Let (c_j, s_j), $j = 4, 5, 6$, be the three leading bits of $(C, S)_{J-1}$. Because $q_{J-1} = 2$,

$$\begin{aligned} P_J \;=\; & 4P_{J-1} - 8(D + d_5/32 + d_6/64) - \frac{1}{8} + \\ & 0.0c_4c_5 + 0.0s_4s_5 + \text{carry}(c_6, s_6, \bar{d}_7), \end{aligned}$$

where the $-\frac{1}{8}$ term is due to the one's complement. Because $d_5 = d_6 = d_7 = 1$, $P_J = P_{\text{Bad}} = P_{J-1} + \frac{1}{8}$, the equation simplifies to

$$\frac{7}{8} = 0.0c_4c_5 + 0.0s_4s_5 + \text{carry}(c_6, s_6, 0),$$

implying $c_j = s_j = 1$ for $j = 4, 5, 6$ as claimed. Note that this is true for all the possible choices of m's and q_{J-m-1}.

Finally, we reconsider the case of $m = 1$ with $q_{J-2} = -1$. Previously, we have only proved that d_5 through d_7 must be ones for this case. We now show that in fact $d_8 = 1$ also. Consider the generation of p_{J-1} from p_{J-2} as depicted in Figure 4. We have just established that $c'_j = s'_j = 1$ for $j = 4, 5, 6$. Clearly, then, we must have $f_8 = 1$. But $f_8 = d_8$ because $q_{J-2} = -1$. This completes the proof of Lemma 2.

Lemma 3. The quotient digit 2 just prior to q_J can occur only once, that is, in fact, $m = 1$ and $q_{J-2} = -1$ or -2. Moreover, C_{J-2} and S_{J-2} must each have at least five leading ones.

Proof of Lemma 3. We concentrate on the process

$$(P, C, S)_{J-2} \xrightarrow{q_{J-2}} (P, C, S)_{J-1}$$

as shown in Figure 4. We have already established that d_5 through d_8 to be ones and that $c'_j = s'_j = 1$ for $j = 4, 5, 6$. Consequently, we must have $c_j = s_j = f_j = 1$ for $j = 7, 8$. If $q_{J-2} \ge 0$, then $f_7 = \bar{d}_7$ or $\bar{d}_8$ implies $f_7 = 0$. Thus, we must have $q_{J-2} < 0$. This

Figure 5: From $J-3$ to $J-2$

P_{J-3}	$k/8$		
C_{J-3}	$0.000\,c_4\,c_5$	$c_6\,c_7\,\ldots$	$c_{10}\,\ldots$
S_{J-3}	$0.000\,s_4\,s_5$	$s_6\,s_7\,\ldots$	$s_{10}\,\ldots$
$-q_{J-3}d$	$e_3\,e_2\,e_1\,e_0\,.\,f_1\,\ldots\,f_5$	$f_6\,f_7\,\ldots$	$f_{10}\,\ldots$
P_{J-2}	$-\frac{1 \text{ or } 4}{3}D_+ - \frac{1}{4}$	$1\;1\;\ldots$ $1\;1\;\ldots$	$1\;\ldots$ $1\;\ldots$

fact, together with Lemma 1 forces $m = 1$, or in other words, $q_{J-2} = -1$ or -2. Now,

$$q_{J-2} = \left\{ \begin{array}{c} -2 \\ \text{or} \\ -1 \end{array} \right\}$$

and

$$P_{J-2}, P_{J-1} = \left\{ \begin{array}{c} -\frac{4}{3}D_+ - \frac{1}{4}, \frac{8}{3}D_+ - \frac{1}{4} \\ \text{or} \\ -\frac{1}{3}D_+ - \frac{1}{4}, \frac{8}{3}D_+ - \frac{1}{4} \end{array} \right\}$$

implies

$$0.0c_4c_5 + 0.0s_4s_5 = \frac{6}{8}, \text{ and carry}(c_6, s_6, f_6) = 1.$$

This, together with $s_4' = 1$ implies $c_j = s_j = 1$ for $j = 4, 5, 6$. Thus, $c_j = s_j = 1$ for $j = 4, 5, 6, 7, 8$ as claimed and the Lemma is proved.

Theorem 1. In order for the SRT to reference P_{Bad}, Bits 5 through 10 of the divisor d must all be ones. Moreover, $q_{J-3} < 0$.

Proof of Theorem 1. We concentrate on the process

$$(P, C, S)_{J-3} \xrightarrow{q_{J-3}} (P, C, S)_{J-2}$$

as depicted in Figure 5. It is clear that f_7 through f_{10} must be all ones. Consequently, we must have $q_{J-3} < 0$ for otherwise the fact that d_5 through d_8 are all ones would imply $f_7 = 0$ for any choice of $q_{J-3} \geq 0$. It follows immediately then that $q_{J-3} = -1$ or -2. In either case, we have

$$f_7 = \cdots = f_{10} = 1 \Longrightarrow d_9 = d_{10} = 1$$

and the theorem is established.

Corollary 1. $J \geq 8$, that is, the first 8 quotient digits generated are always correct despite the flaw P-D table.

Proof of Corollary 1. Initially, we have $C_0 = 0$. Therefore we can establish a lower bound on J by examining the earliest possible occurrence of an all-zero pattern in the sequence $C_J, C_{J-1}, C_{J-2}, \ldots$.

If we have L consecutive $(c_j, s_{j+1}) = 1$ patterns in $(C, S)_k$, we must have at least $L-1$ consecutive occurrence of such patterns in $(C, S)_{k-1}$. Since in $(C, S)_{J-2}$ we have 4 consecutive $(c_j, s_{j+1}) = 1$, we must have at least 3 such patterns in $J-3$; at least 2 in $J-4$; at least 1 in $J-5$; at least 1 non-zero carry bit in $J-6$. Thus, $J \geq 7$.

If in fact $J = 7$, then the above argument shows that indeed we can only have 3, *and no more*, such patterns in $J-3$, only 2 in $J-4$, and only 1 in $J-5$. Consider now $(C, S)_{J-2}$. Because $(c_j, s_{j+1}) = 1$ for $j = 4, 5, 6, 7$ and $c_8 = 1$, we must have $(c_j, s_j, f_j)_{J-3} = 1$ for $j = 7, 8, 9, 10$ in Step $J-3$. Moreover, at least 2 of (c_{11}, s_{11}, f_{11}) must be ones (in order to generate $c_8 = 1$ in $J-2$). This means that $c_{11} = f_{11} = 1$ and $s_{11} = 0$. Using the same argument, we conclude that in Step $J-4$, we must have $(c_j, s_j, f_j)_{J-4} = 1$ for $j = 10, 11, 12$ and $(c_j, s_j, f_j)_{J-4} = (1, 0, 1)$ for $j = 13, 14$ (in order to generate $c_{10} = c_{11} = 1$ in Step $J-3$).

Continuing this argument, we conclude that there must be a persistent five-consecutive-one pattern in the f's of Step $J-3, J-4, \ldots, J-7$. More precisely,

$$(f_j, f_{j+1}, \ldots, f_{j+4})_K = (1, 1, 1, 1, 1)$$

for

$$(j, K) = (7, J-3), (10, J-4), \ldots, (19, J-7).$$

Since $f_j = d_j, d_{j+1}, \bar{d}_j$, or $\bar{d}_{j+1}$, the overlapping consecutive ones forces q_{J-3}, q_{J-4}, up to q_{J-7} to be of the same sign. But $q_{J-3} < 0$ by Theorem 1. Thus $q_{J-7} < 0$, implying that it cannot be the first quotient digit after all. Thus $J \geq 8$ and the corollary is established.

5 Relative Error Analysis

In this section, we provide an upper bound for the relative error

$$\left| \frac{\text{absolute error}}{\text{correct quotient}} \right|$$

where absolute error is defined as

$$\text{abs. err.} = \text{correct quotient} - \text{computed quotient}.$$

Let

$$q_0, q_1, \ldots, q_{J-1}, q_J, q_{J+1}, \ldots$$

be the correct sequence of quotient digit generated had there been no flaw; and let

$$q_0, q_1, \ldots, q_{J-1}, \tilde{q}_J, \tilde{q}_{J+1}, \ldots$$

be the sequence of flawed digits (from J onwards). Note that, in particular, $\tilde{q}_J = 0$.

Lemma 4. The magnitude of the absolute error is bounded as

$\tilde{q}_{J+1}$	2	1	0	-1	-2
abs. err. bound	3.56 $\times 10^{-5}$	3.94 $\times 10^{-5}$	4.32 $\times 10^{-5}$	4.71 $\times 10^{-5}$	5.08 $\times 10^{-5}$

Proof of Lemma 4. The absolute error E is given by

$$E = \sum_{j=J}^{\infty} q_j/4^j - \sum_{j=J}^{\infty} \tilde{q}_j/4^j,$$

where $q_J = 2$, $\tilde{q}_J = 0$, and $J \geq 8$ (Corollary 1). Thus,

$$|E| \leq \sum_{j=8}^{\infty}\left(\frac{2}{4^j} + \frac{2}{4^{j+2}}\right) - \frac{\tilde{q}_{J+1}}{4^9}.$$

Substituting the various cases of $\tilde{q}_{J+1}$ yields the tabulated result.

An obvious way to obtain an upper bound for the relative error is to divide the maximum entry of the previous table by a lower bound on the correct quotient:

$$\text{correct quotient} \geq \frac{1}{\max D_+}.$$

The bound obtained in this manner is roughly 10^{-4}. We can reduce this bound by exploiting the correlation between $\tilde{q}_{J+1}$ and D_+. This is the task of the rest of this section.

Lemma 5. Let the carry and save vectors at Step J, C_J, SSJ be

$$\begin{aligned} C_J &= 0.000\,c_4\,c_5\,c_6 \ldots \\ S_J &= 0.000\,s_4\,s_5\,s_6 \ldots. \end{aligned}$$

Then

$$0.0c_4c_5 + 0.0s_4s_5 + \text{carry}(c_6, s_6, 0)/8 \leq 3/8.$$

Proof of Lemma 5. In the flawless situation, $q_J = 2$ and

$$\begin{aligned} P_{J+1} &= 4(P_J - 2(D + 2^{-5} + 2^{-6}) - 2^{-5}) \\ &\quad + 0.0c_4c_5 + 0.0s_4s_5 + \text{carry}(c_6, s_6, 0)/8 \end{aligned}$$

because $d_5 = d_6 = d_7 = 1$. Moreover $P_J = \frac{8}{3}D_+ - \frac{1}{8}$ and $P_{J+1} \leq \frac{8}{3}D_+ - \frac{1}{8}$ since this is the maximum P value possible. Putting these information to the previous equation yields the result immediately.

Because of the flaw, we have $\tilde{q}_J = 0$. Thus,

$$\tilde{P}_{J+1} = 4P_J + 0.0c_4c_5 + 0.0s_4s_5 + \text{carry}(c_6, s_6, 0)/8.$$

Lemma 5 shows that the leading bit pattern of $\tilde{P}_{J+1}$ is given exactly by $4P_J = 4(\frac{8}{3}D_+ - \frac{1}{8})$. Note that overflow of $4P_J$ leads $\tilde{P}_{J+1}$ tbe be interpreted as negative in some cases. Using the bit patterns of $\tilde{P}_{J+1}$ and Table 1, we derive the following table for $\tilde{q}_{J+1}$:

D_+	$1+\frac{2}{16}$	$1+\frac{5}{16}$	$1+\frac{8}{16}$	$1+\frac{11}{16}$	$1+\frac{14}{16}$
$\tilde{q}_{J+1}$	?	-2	0	1	2

When $D_+ = 1 + \frac{2}{16}$, $\tilde{P}_{J+1}$ is interpreted as between -7.5 and $-7.5 + 1/8$ since

$$\tilde{P}_{J+1} = 1000.1\text{XXX}\ldots.$$

This is clearly out of bound of the legitimate P values. As far as an error bound is concerned, we can take $\tilde{q}_{J+1}$ to be -2. Theorem 2 is now obvious.

Theorem 2. An upper bound of the relative error is 6.7×10^{-5}.

Proof of Theorem 2. The result is obtained by combining Lemma 5 and the previous table: The relative error is bounded by 10^{-5} times the maximum of

$$\frac{30}{16} \times 3.56,\ \frac{27}{16} \times 3.94,\ \frac{24}{16} \times 4.32,\ \frac{21}{16} \times 5.08.$$

This completes the proof.

6 Examples

We present two examples to show that both Theorem 1 and Corollary 1 are sharp. We scale the dividends and divisors so that they become integers and represent them in both decimal and hexadecimal forms.

Example 1.

$$\begin{aligned} \text{dividend} &= 1092\,49940\,73628 \quad \texttt{9EF AC64 141C} \\ \text{divisor} &= 103\,02563\,26687 \quad \texttt{EF E00F F81F} \\ q_0, \ldots, q_J &= +1, -1, \ldots, -2, +2, q_{14} = q_{\text{Bad}} \end{aligned}$$

Note that the divisor corresponds to

$$1.d_1d_2d_3d_4 = 1.1101,$$

with d_5 through d_{10} to be ones and that $d_{11} = 0$.

Example 2.

$$\begin{aligned} \text{dividend} &= 41\,95835 \quad \texttt{80 0BF6} \\ \text{divisor} &= 31\,45727 \quad \texttt{BF FFFC} \\ q_0, \ldots, q_J &= +1, -1, \ldots, -1, +2, q_8 = q_{\text{Bad}} \end{aligned}$$

Note that indeed the ninth quotient digit can be wrong. This case, however, is not associated with only six ones in the divisor.

7 Acknowledgement

The authors thank the referees and Alan Edelman for suggesting improvement on the presentation, especially for the latter in suggesting using a simple table of upper bounds on the P_i's to describe digit sequences. Edelman also suggested including the result given in Corollary 1. The authors thank Intel Corporation for providing sufficient details of their division implementation to make the analysis possible. The second author performed this work while on leave visiting the Mathematics Department of the Chinese University of Hong Kong. The generous support of the department is also acknowledged.

References

[1] O. L. MacSorley, High-speed arithmetic in binary computers, *Proc. of IRE,*, **49**, pp. 67–91, 1961.

[2] J. E. Robertson, A new class of digital division methods, *IRE Trans. on Electron. Computers,* **EC-7**, pp. 218–222, 1958.

[3] K. D. Tocher, Techniques of multiplication and division for automatic binary computers, *Quart. J. Mech. Appl. Math. XI,* **Part 3**, pp. 364–384, 1958.

[4] I. Koren, *Computer Arithmetic Algorithms,* Prentice Hall, Englewood Cliffs, 1993.

[5] Amos R. Omondi, *Computer Arithmetic Systems — Algorithms, Architecture and Implementations,* Prentice Hall Series in Computer Science, Englewood Cliffs, 1994.

[6] M. D. Ercegovac and T. Lang, *Division and Square Root: Digit Recurrence Algorithms and Implementations,* Kluwer, 1994.

[7] IEEE standard for binary floating-point arithmetic, ANSI/IEEE 754-1985, also in *Computer,* **14**, pp. 51–62, 1981.