MEASURING THE ACCURACY OF ROM RECIPROCAL TABLES*

Debjit Das Sarma and David W. Matula
Department of Computer Science & Engineering
Southern Methodist University
Dallas, Texas 75275

Abstract

We prove that a convenient ROM reciprocal table construction algorithm generates tables that minimize the relative error. The worst case relative errors realized for such optimally computed k-bits-in, m-bits-out ROM reciprocal tables are then determined for all table sizes $3 \leq k, m \leq 12$. We next prove the table construction algorithm always generates a k-bits-in, k-bits-out table with relative errors never any greater than $\frac{3}{4}2^{-k}$ for any k, and more generally with g guard bits that for $(k+g)$-bits-out the relative error is never any greater than $2^{-(k+1)}(1 + \frac{1}{2^{g+1}})$. To provide for determining test data without prior construction of a full ROM reciprocal table, we describe a procedure that requires generation and searching of only a small portion of such a table to determine regions containing input data yielding the worst case relative errors.

1 Introduction and Summary

With the density of transistors that can be realized in an integrated circuit on a single chip rising so rapidly, it has become increasingly common for arithmetic circuits to include a ROM reciprocal table to either assist or replace the division instruction [FW 71], [AC 74], [SB 86], [DG 89], [FS 89], [Co 90], [BM 91], [Ka 91], [OL 91], [WF 91].

For low precision arithmetic computation, direct use of a suitably large ROM reciprocal table [OL 91], or applying interpolation in a moderate size such table, provides an easy to implement efficient alternative to a division instruction. ROM reciprocal tables are also becoming more common in the high performance implementation of IEEE standard 754 double or double extended floating point chips for the platforms of workstations and PC's [DG 89], [Co 90], [BM 91]. The trend in the design of these chips is to include a fast multiplier, and then often to realize divide and square root by incorporating reciprocal refinement algorithms such as Newton Raphson or convergence [Go 64] in the microcode. These procedures benefit greatly from a good initial reciprocal approximation such as can be rapidly provided by a ROM reciprocal look-up table.

ROM reciprocal tables are generally constructed by assuming that the argument is normalized $1 \leq x < 2$ and truncated to k bits to the right of the radix point, $trunc(x) = 1.b_1b_2...b_k$. These k bits are used to index a table providing m output bits which are taken as the m bits after the leading bit in the m+1 bit fraction reciprocal approximation $recip(x) = 0.1b'_1b'_2...b'_m$. Such a table will be termed a k-bits-in m-bits-out reciprocal table of size 2^km bits.

The maximum relative error for any k-bits-in m-bits-out ROM reciprocal table denotes the supremum of the relative errors obtained between $\frac{1}{x}$ and the table value for the reciprocal of x for $1 \leq x < 2$. The precision in bits of the table is the negative base two logarithm of this supremum. A table precision of α bits then denotes that the approximation of $\frac{1}{x}$ by the table value will always yield a relative error of at most one part in 2^α.

In Section 2 we describe a method that has been used [FS 89] for constructing ROM reciprocal tables. The procedure is based on a known result for optimal reciprocal approximation by reals [Fe 67], but the applicability to approximation by finite precision reciprocal approximations requires a proof of optimality. Interestingly, we first demonstrate that the method can fail to find the optimal m-bit reciprocal approximation for certain input intervals. Our first principal result is then that the method restricted to the intervals occuring in k-bits-in, m-bits-out ROM reciprocal tables indeed always generates reciprocal approximations minimizing the relative error and guaranteeing the highest precision for each table. Employing this algorithm we compute this precision guarantee by exhaustive search of the constructed tables and tabulate these results for k-bits-in, m-bits-out ROM reciprocal tables for all sizes $3 \leq k, m \leq 12$.

It is convenient to parameterize our reciprocal tables by employing g as the number of output guard bits, implying then a k-bits-in, $(k+g)$-bits-out ROM reciprocal table. Our second principal result is given in Section 3 where we prove it is always possible to construct

*Supported in part by a grant from Cyrix Corporation and by the Danish Natural Science Research Council grant no. 11-8243.

a k-bits-in, $(k+g)$-bits-out ROM reciprocal table with relative error of the table at most $2^{-(k+1)}(1+\frac{1}{2^{g+1}})$ for any $g \geq 0$ and any $k \geq 1$. In particular this confirms that a k-bits-in, k-bits-out ROM reciprocal table will guarantee precision of at least $k + 0.415$ bits, and with one, two and three guard bits on the output we are guaranteed precisions of at least $k + 0.678$ bits, $k + 0.830$ bits, and $k + 0.912$ bits respectively. These resuts are useful in determining whether adding a few guard bits might provide sufficient additional accuracy in place of the more costly step of increasing k to $k+1$, which results in more than doubling the table size. Note that the inclusion of three guard bits adds about one half bit of precision. If the initial approximation is to be refined by three Newton Raphson iterations as part of a division process, the resulting doubling of precision in each iteration then provides that the three guard bit initial enhancement implicitly contributes four more bits of precision to the final result.

Finally in Section 4 we determine and tabulate the input intervals containing the worst case relative error input for all cases of k-bits-in, k-bits-out, and $(k+1)$-bits-out, $5 \leq k \leq 15$. Such intervals are useful in determining input values for testing the correctness of certain hardware division implementations, particularly when these ROM reciprocal table values are further refined by a Newton Raphson or convergence process to achieve a target accuracy level [AC 74], [SB86], [DG 89], [Co 90], [BM 91], [Ka 91]. Applications where the use of a guard bit on output is explicitly mentioned include a 13-bits-in, 14-bits out ROM reciprocal table in [WF 91], and an 8-bits-in, 9-bits-out table in [Co 90].

For larger ROM reciprocal tables it can be useful in advance to find input intervals yielding the largest relative error without having to construct and search the whole table. In Section 4 we also provide a method for determining such worst case input intervals that involves constructing and searching only a small portion of the respective tables.

2 Constructing Optimal ROM Reciprocal Tables

An optimal (k,m) bit table denotes a k-bits-in, m-bits-out ROM reciprocal table as described in the preceding section where the maximum relative error for each entry in the table is the minimum possible for that entry. Observe that each of the 2^k indices into a k-bits-in ROM reciprocal table corresponds to an input interval $[\frac{i}{2^k}, \frac{i+1}{2^k})$ for $0 \leq i \leq 2^k - 1$. To construct an optimal (k,m) bit table it is necessary to choose a reciprocal approximation table entry minimizing the error for each such interval. For motivation we note the following standard result valid for recirocal approximation by reals [Fe 67].

Lemma 1: For x any value from the interval $[a,b]$ with $0 < a \leq b$, the minimum value of the maximum relative error in an approximation of $\frac{1}{x}$ is $\frac{b-a}{b+a}$ and is realized by the reciprocal approximation $\frac{2}{a+b}$, *i.e.* by the reciprocal of the mid-point of the interval containing x.

Proof: The exact reciprocal will fall in the interval $[\frac{1}{b}, \frac{1}{a}]$, and the worst case error will occur for input being one or the other of the endpoints. These endpoint relative errors for the reciprocal approximation $\frac{2}{a+b}$ are $(\frac{1}{a} - \frac{2}{a+b})/\frac{1}{a} = \frac{b-a}{a+b}$, and $(\frac{2}{a+b} - \frac{1}{b})/\frac{1}{b} = \frac{b-a}{a+b}$. Any reciprocal approximation other than $\frac{2}{a+b}$ would result in one of these relative errors being larger. □

We must now extend Lemma 1 by considering the consequence of limiting our reciprocal approximation choice to m-bit values. Let RU denote ***round-up*** towards $+\infty$ and RD denote *round-down* towards $-\infty$ to the prescribed binary format.

Corollary 1.1: For x any value from the interval $[a,b]$ with $1 \leq a \leq b < 2$, the minimum value of the maximum relative error in an $m+1$ bit approximation $0.1b'_1b'_2......b'_m$ to $\frac{1}{x}$ is realized by choosing as the prescribed approximation that one of the two values $RU(\frac{2}{a+b})$ and $RD(\frac{2}{a+b})$ that yields the least relative error.

Proof: As in the proof of Lemma 1 we obtain that a value larger than $RU(\frac{2}{a+b})$ yields worse relative error in approximating $\frac{1}{x} = \frac{1}{b}$ than does $RU(\frac{2}{a+b})$. Similarly a value smaller than $RD(\frac{2}{a+b})$ yields worse relative error in approximating $\frac{1}{x} = \frac{1}{a}$ than does $RD(\frac{2}{a+b})$. So the better of these two approximations provides the minimum relative error approximation. □

Letting RN denote *round-to-nearest*(even), it is tempting to conjecture that $RN(\frac{2}{a+b})$ is the value that yields the minimum relative error in Corollary 1.1. But this is not true for arbitrary input intervals $[a,b]$ with $1 \leq a \leq b < 2$. In particular consider choosing the best 3 bit reciprocal approximation $0.1b_1b_2$ for x ranging over the interval $[1, \frac{15}{8}]$. The best infinitely precise reciprocal approximation is $\frac{16}{23} = 0.10110010...$ for which we obtain $RN(\frac{16}{23}) = 0.110 = \frac{3}{4}$. It is readily checked that the worst case relative error with reciprocal approximation $\frac{3}{4}$ is $\frac{13}{32}$, whereas $RD(\frac{16}{23}) = \frac{5}{8}$ has worst case relative error of only $\frac{3}{8}$, so that $\frac{5}{8}$ is the preferred reciprocal approximation.

For ROM reciprocal tables we need only concern ourselves with input intervals of the form $[\frac{i}{2^k}, \frac{i+1}{2^k})$ for $2^k \leq i \leq 2^{k+1} - 1$. Fortunately, for these input intervals we obtain the following satisfying result.

Theorem 2: For x any value from the interval $[\frac{i}{2^k}, \frac{i+1}{2^k})$ with i an integer in the range $2^k \leq i \leq 2^{k+1} - 1$, the minimum of the maximum relative error in approximating $\frac{1}{x}$ to $m+1$ bits is obtained by choosing the reciprocal approximation to be $RN(\frac{2^k}{i+\frac{1}{2}})$ to $m+1$ bits.

Proof: First we claim that the interval $I = (\frac{2^k}{i+\frac{1}{2}} - (\frac{i+1}{2i+1})2^{-(m+1)}, \frac{2^k}{i+\frac{1}{2}} + (\frac{i}{2i+1})2^{-(m+1)}]$ contains a unique $m+1$ bit reciprocal approximation $\frac{j}{2^{m+1}} = 0.1b'_1b'_2.....b'_m$ minimizing the maximum relative error over all $m+1$ bit approximations to $\frac{1}{x}$ for $\frac{1}{x}$ in the range $(\frac{2^k}{i+1}, \frac{2^k}{i}]$.

To justify the claim note that the left-open right-closed interval I has width $2^{-(m+1)}$ and therefore contains exactly one value $\frac{j}{2^{m+1}}$ where j is an integer. Approximating any value $\frac{1}{x}$ in the interval $(\frac{2^k}{i+1}, \frac{2^k}{i}]$ by $\frac{j}{2^{m+1}}$ yields a relative error no larger than the maximum of the two relative errors determined by :
1) approximating $\frac{2^k}{i}$ by $\frac{2^k}{i+\frac{1}{2}} - (\frac{i+1}{2i+1})2^{-(m+1)}$,
2) approximating $\frac{2^k}{i+1}$ by $\frac{2^k}{i+\frac{1}{2}} + (\frac{i}{2i+1})2^{-(m+1)}$.
But the relative errors for both (*1*) and (*2*) are the same value $r = \frac{1}{(2i+1)}(1 + i(i+1)2^{-(m+k+1)})$.
It follows that approximating any $\frac{1}{x}$ in the interval $(\frac{2^k}{i+1}, \frac{2^k}{i}]$ by $\frac{j}{2^{m+1}}$ yields relative error at most r. Furthermore, for any $\frac{j'}{2^{m+1}}$ outside interval I, it is readily seen that the process of approximating some value of $\frac{1}{x}$ in $(\frac{2^k}{i+1}, \frac{2^k}{i}]$ by $\frac{j'}{2^{m+1}}$ will yield a larger relative error than r, proving the claim.

Now we must show that the $m+1$ bit round-to-nearest value of $\frac{2^k}{i+\frac{1}{2}}$ falls in the interval I.
To prove this we assert a second claim that the interval $I' = (\frac{2^k}{i+\frac{1}{2}} - (\frac{i+1}{2i+1})2^{-(m+1)}, \frac{2^k}{i+\frac{1}{2}} - \frac{1}{2}2^{-(m+1)}]$ cannot contain any $m+1$ bit value $\frac{j}{2^{m+1}}$. Assuming the second claim false we would have for some integer j that $\frac{2^k}{i+\frac{1}{2}} - (\frac{i+1}{2i+1})2^{-(m+1)} < \frac{j}{2^{m+1}} \le \frac{2^k}{i+\frac{1}{2}} - \frac{1}{2}2^{-(m+1)}$. But then multiplying by $(2i+1)2^{m+1}$ yields $2^{k+m+2} - i - 1 < j(2i+1) \le 2^{k+m+2} - i - \frac{1}{2}$. The right hand inequality can be sharpened since $j(2i+1)$ is an integer to yield $2^{k+m+2} - i - 1 < j(2i+1) \le 2^{k+m+2} - i - 1$, a contradiction verifying the second claim.
Thus the unique $m+1$ bit value $\frac{j}{2^{m+1}}$ of the first claim must fall in the interval $I - I' = (\frac{2^k}{i+\frac{1}{2}} - \frac{1}{2}2^{-(m+1)}, \frac{2^k}{i+\frac{1}{2}} + (\frac{i}{2i+1})2^{-(m+1)}]$ and must then be the $m+1$ bit round-to-nearest value of $\frac{2^k}{i+\frac{1}{2}}$. □

Algorithm 1
[ROM Reciprocal Table Construction]

```
Stimulus: Integers $k \ge 1$ and $m \ge 1$

Response: A $k$-bits-in $m$-bits-out ROM reciprocal
          table with the relative error
          interval given for each table entry

Method:   for $i = 2^k$ to $2^{k+1} - 1$ step 1
          <for each input interval $[\frac{i}{2^k}, \frac{i+1}{2^k})$>
          begin
L1:           $table(i) := RN(\frac{2^k}{i+\frac{1}{2}})$
          <$m$-bit reciprocal approximation is $\frac{j}{2^{m+1}}$>
L2:           $r.e.int(i) := (1 - \frac{(i+1)j}{2^{k+m+1}}, 1 - \frac{ij}{2^{k+m+1}}]$
          <relative error interval>
          end
```

Theorem 3: For any $k \ge 1$ and $m \ge 1$, Algorithm 1 generates an optimal (k, m) bit table.
Proof: It is immediate from Theorem 2 that each table entry has the maximum relative error minimized by the m+1 bit reciprocal approximation specified in Algorithm 1. □

We apply Algorithm 1 to generate optimal (5,5) and (5,6) bit reciprocal tables and illustrate the resuts in Table 1. Consider for example line nine of Table 1. Line nine is indexed by the 5 bit string 01000 arising from input truncated to the value 1.01000 . This designates that the input value x for which we wish a lookup table reciprocal value was in the range [40/32,41/32). The reciprocal of the mid-point (being 32/41.5) is first given with several decimal places as 50.568/64 in the table so the reader may note the additional rounding error introduced in determining the 5-bit table value written as a fraction 51/64. The relative error interval for inputs approximated by 51/64 is then computed and shown to be $(\frac{-43}{2048}, \frac{8}{2048}]$. Note that this input interval yields the largest relative errors of any in the table, with only the input interval corresponding to 1.10010 yielding comparably large relative errors in the reciprocal approximation. The fact that line nine yields the largest relative error is at least partly associated with the observation that the mid-point reciprocal 50.568/64 suffers a comparatively large rounding error to become the m+1 bit value 51/64. It is further instructive to note that this relative error interval is not well centered around zero compared to others in the early part of the table. Finally note that with one more output bit the resulting relative error interval will be well centered for this entry, so the worst case input must occur elsewhere in the (5 ,6) bit table.
The precision of the optimal (5,5) bit reciprocal table is then $-\log_2(43/2048) = 5.573...$ .It is worth noting that the worst case relative error in the optimal (5,6) bit reciprocal table occurs only for input in the interval [35/32,36/32), which results in the precision of that table being $-\log_2(71/4096)$= 5.850... . The precision of such optimal (k, m) bit reciprocal tables for all $3 \le k, m \le 12$ is given in Table 2, which by itself provides a useful reference if one needs to employ a ROM reciprocal table for testing or implementing a

5 bits in			5 bits out		6 bits out	
Chopped Input	Input Interval	Middle point Reciprocal ×(1/64)	Rounded Reciprocal	Relative Error Interval ×(1/2048)	Rounded Reciprocal	Relative Error Interval ×(1/4096)
1.00000	[32/32,33/32)	63.015	63/64	(-31,32]	126/128	(-62,64]
1.00001	[33/32,34/32)	61.134	61/64	(-26,35]	122/128	(-52,70]
1.00010	[34/32,35/32)	59.362	59/64	(-17,42]	119/128	(-69,50]
1.00011	**[35/32,36/32)**	57.690	58/64	(-40,18]	115/128	(-44,**71**]
1.00100	[36/32,37/32)	56.110	56/64	(-24,32]	112/128	(-48,64]
1.00101	[37/32,38/32)	54.613	55/64	(-42,13]	109/128	(-46,63]
1.00110	[38/32,39/32)	53.195	53/64	(-19,34]	106/128	(-38,68]
1.00111	[39/32,40/32)	51.848	52/64	(-32,20]	104/128	(-64,40]
1.01000	**[40/32,41/32)**	50.568	51/64	(**-43**, 8]	101/128	(-45,56]
1.01001	[41/32,42/32)	49.349	49/64	(-10,39]	99/128	(-62,37]
1.01010	[42/32,43/32)	48.188	48/64	(-16,32]	96/128	(-32,64]
1.01011	[43/32,44/32)	47.080	47/64	(-20,27]	94/128	(-40,54]
1.01100	[44/32,45/32)	46.022	46/64	(-22,24]	92/128	(-44,48]
1.01101	[45/32,46/32)	45.011	45/64	(-22,23]	90/128	(-44,46]
1.01110	[46/32,47/32)	44.043	44/64	(-20,24]	88/128	(-40,48]
1.01111	[47/32,48/32)	43.116	43/64	(-16,27]	86/128	(-32,54]
1.10000	[48/32,49/32)	42.227	42/64	(-10,32]	84/128	(-20,64]
1.10001	[49/32,50/32)	41.374	41/64	(-2,39]	83/128	(-54,29]
1.10010	**[50/32,51/32)**	40.554	41/64	(**-43**,-2]	81/128	(-35,46]
1.10011	[51/32,52/32)	39.767	40/64	(-32, 8]	80/128	(-64,16]
1.10100	[52/32,53/32)	39.010	39/64	(-19,20]	78/128	(-38,40]
1.10101	[53/32,54/32)	38.280	38/64	(-4,34]	77/128	(-62,15]
1.10110	[54/32,55/32)	37.578	38/64	(-42,-4]	75/128	(-29,46]
1.10111	[55/32,56/32)	36.901	37/64	(-24,13]	74/128	(-48,26]
1.11000	[56/32,57/32)	36.248	36/64	(-4,32]	72/128	(-8,64]
1.11001	[57/32,58/32)	35.617	36/64	(-40,-4]	71/128	(-22,49]
1.11010	[58/32,59/32)	35.009	35/64	(-17,18]	70/128	(-34,36]
1.11011	[59/32,60/32)	34.420	34/64	(8,42]	69/128	(-44,25]
1.11100	[60/32,61/32)	**33.851**	34/64	(-26, 8]	68/128	(-52,16]
1.11101	[61/32,62/32)	33.301	33/64	(2,35]	67/128	(-58, 9]
1.11110	[62/32,63/32)	32.768	33/64	(-31, 2]	66/128	(-62, 4]
1.11111	[63/32,64/32)	32.252	32/64	(0,32]	65/128	(-64, 1]

Table 1: Optimal 5 bits in,5 bits out and 5 bits in,6 bits out Reciprocal Tables

bits in /bits out	3	4	5	6	7	8	9	10	11	12
3	**3.540**	4.000	4.000	4.000	4.081	4.081	4.081	4.081	4.087	4.087
4	4.000	**4.678**	4.752	5.000	5.000	5.000	5.042	5.042	5.042	5.042
5	4.000	4.752	**5.573**	5.850	5.891	6.000	6.000	6.000	6.022	6.022
6	4.000	5.000	5.850	**6.476**	6.790	6.907	6.950	7.000	7.000	7.000
7	4.081	5.000	5.891	6.790	**7.484**	7.775	7.888	7.948	7.976	8.000
8	4.081	5.000	6.000	6.907	7.775	**8.453**	8.719	8.886	8.944	8.974
9	4.081	5.042	6.000	6.950	7.888	8.719	**9.430**	9.725	9.852	9.942
10	4.081	5.042	6.000	7.000	7.948	8.886	9.725	**10.443**	10.693	10.858
11	4.087	5.042	6.022	7.000	7.976	8.944	9.582	10.693	**11.429**	11.701
12	4.087	5.042	6.022	7.000	8.000	8.974	9.942	10.858	11.701	**12.428**

Table 2: Minimum precisions of tables of different sizes

division instruction. The values in Table 2 were rounded down to provide true lower bounds on the precision attainable. Interestingly the table precision for optimal (k, m) and (m, k) reciprocals tables are the same for all k, m in the table. This can be proved to hold in general despite the great variance in the sizes $m2^k$ and $k2^m$ of total bits employed in the (k, m) and (m, k) tables. The result is tangential to our interests here and no proof is included.

The ROM reciprocal tables of greatest practical interest may be more aptly characterized as k-bits-in, $(k+g)$-bits-out where g indicates the number of guard bits on the output. The precisions of optimal $(k, k+g)$ reciprocal tables for $0 \leq g \leq 4$ and $k = 6, 8, ..., 16$ are given in Table 3 and suggest convergence of the fraction portion for fixed g with increasing k. This topic is further discussed in the next section.

k / g	6	8	10	12	14	16
0	6.476	8.453	10.443	12.428	14.422	16.418
1	6.790	8.719	10.693	12.687	14.682	16.679
2	6.907	8.886	10.858	12.844	14.834	16.833
3	6.950	8.944	10.924	12.918	14.915	16.914
4	7.000	8.974	10.970	12.963	14.959	16.956

Table 3: Minimum precisions of reciprocal tables with different guard bits

At this point we note that in many applications it can be required that the approximate reciprocal must be guaranteed higher(or lower) than the infinitely precise reciprocal [AC 74], [Ka 91], [WF 91]. A larger relative error is then necessary in the worst case reciprocal approximation to assure that the approximation is properly directed. By methods similar to our previous discussion we have computed the best precision attainable for similarly sized tables where the reciprocal approximation must be guaranteed either high or low, and these are shown in Table 4 for comparison with Table 3. The analysis of the worst case relative error for directed approximation of reciprocals is somewhat simpler than for reciprocal approximation in general. For brevity we do not further discuss herein the directed reciprocal approximation table construction process, and concentrate in the following on analysis of tables minimizing the maximum magnitude of the relative errors.

k / g	6	8	10	12	14	16
0	5.430	7.419	9.416	11.415	13.415	15.415
	5.565	7.482	9.447	11.429	13.421	15.417
1	5.705	7.685	9.680	11.678	13.678	15.678
	5.752	7.715	9.696	11.687	13.683	15.680
2	5.866	7.839	9.832	11.831	13.830	15.830
	5.921	7.875	9.853	11.841	13.835	15.832
3	5.953	7.923	9.915	11.913	13.913	15.912
	5.956	7.934	9.923	11.918	13.915	15.913
4	5.999	7.966	9.958	11.956	13.956	15.955
	6.000	7.978	9.967	11.961	13.958	15.957

Table 4: Precisions of tables with directed (high and low) reciprocal approximation

3 Error Bounds for Reciprocal Tables

For the reciprocal tables constructed by Algorithm 1, it is possible to derive an **upper bound** on the maximum relative error valid for any $(k, k+g)$ bit reciprocal table.

Theorem 4: The maximum relative error of an optimal reciprocal table with k-bits-in, $(k+g)$-bits-out is bounded above by $2^{-(k+1)}(1+\frac{1}{2^{g+1}})$.

Proof: First we claim that if $1 \leq y \leq 2$ and $g \geq 1$ where $y \in R^+$ and g is an integer, then
$y + \frac{2^{g+1}}{y}$ is maximum at $y = 1$, **(1a)** and
$y + \frac{2}{y}$ achieves the maximum at $y = 1$ and $y = 2$. **(1b)**
To justify the claim let $f(y) = y + \frac{2^{g+1}}{y}$ where $g \geq 1$. Then $\frac{d}{dy}(f(y)) = 1 - \frac{2^{g+1}}{y^2} \leq 0$ in $1 \leq y \leq 2$ for $g \geq 1$. So, $f(y)$ decreases monotonically in the interval $1 \leq y \leq 2$ and is maximum at $y = 1$.
Now let $f'(y) = y + \frac{2}{y}$. Then $\frac{d}{dy}(f'(y)) = 1 - \frac{2}{y^2}$ which is ≤ 0 in $1 \leq y \leq \sqrt{2}$ and > 0 in $\sqrt{2} < y \leq 2$. So $f'(y)$ monotonically decreases in the interval $1 \leq y \leq \sqrt{2}$, monotonically increases in the interval $\sqrt{2} < y \leq 2$, and achieves maximum value at $y = 1$ and $y = 2$, proving the claim.

Let $\frac{x}{2^k}$ be any midpoint of $[\frac{i}{2^k}, \frac{i+1}{2^k})$ for $2^k \leq i \leq 2^{k+1} - 1$, and let $\frac{x'}{2^{k+g+1}}$ be the true reciprocal of $\frac{x}{2^k}$. Let $n = 2k + g + 1$, so then $xx' = 2^n$.

Applying Theorem 2 and observing that the maximum rounding error x' can suffer is 0.5 ulp,
$|r.e|_{max} \leq \max(\frac{(x+1/2)(x'+1/2)-xx'}{2^n})$ where the max is over appropriate x.
$\leq \frac{1}{2}\frac{\max(x+x')+\frac{1}{2}}{2^n}$
$\leq \frac{1}{2}\frac{\max(x+\frac{2^n}{x})+\frac{1}{2}}{2^n}$
$\leq \frac{1}{2}\frac{\max(y+\frac{2^{g+1}}{y})+\frac{1}{2^{k+1}}}{2^{k+g+1}}$
where $y = \frac{x}{2^k}$ and the maximum may be taken over all y, $1 \leq y \leq 2$.
Applying $(1a)$ for $g \geq 1$ and $(1b)$ for $g = 0$,
$|r.e|_{max} \leq \frac{1}{2}\frac{1+2^{g+1}+\frac{1}{2^{k+1}}}{2^{k+g+1}}$
$\leq \frac{2^{k-1}+2^{k+g}+\frac{1}{4}}{2^n}$
But $|r.e|_{max}$ is of the form $\frac{j}{2^n}$ where j must be an integer, and since $2^{k-1} + 2^{k+g}$ is an integer, the inequality can be sharpened to yield
$|r.e|_{max} \leq \frac{2^{k-1}+2^{k+g}}{2^n}$
$\leq \frac{2^{k-1}+2^{k+g}}{2^{2k+g+1}}$
$\leq 2^{-(k+1)}(1+\frac{1}{2^{g+1}})$ □

From Theorem 4, we immediately obtain a **lower bound** on the precision guaranteed for any $(k, k+g)$ bit reciprocal table.

Observation: The precision of the k-bits-in, (k+g)-bits-out reciprocal table constructed by Algorithm 1 for any $k \geq 2$, $g \geq 0$ is at least $k + 1 - \log_2(1+\frac{1}{2^{g+1}})$.

Table 5 illustrates the value of extra guard bits for an optimal $(k, k+g)$ bit reciprocal table for any k. Note in particular that going from 0 to 3 guard bits brings about one half the benefit of increasing k by one bit. With respect to the cost of higher precision reciprocal approximation in terms of table size, the first few guard bits are proportionally much more beneficial than increasing k when k is large.

g	Precision
0	k+.415
1	k+.678
2	k+.830
3	k+.912
4	k+.955
5	k+.977

Table 5: Lower bounds on the precision of optimal $(k, k+g)$ bit reciprocal tables for any k

Comparing Table 3 with Table 5 demonstrates that the actual precision obtained in the constructed tables approaches that of the bound with increasing input bit size k. Table 6 shows the realized relative errors from table search as a percentage of the corresponding computed upper bound on relative error. The bound tightens with increasing k and appears to rapidly approach 100 %.

	k = 5		
	Realized	Upper Bound	% of Bound
g=0	$43/2^{11}$	$48/2^{11}$	89.58 %
g=1	$71/2^{12}$	$80/2^{12}$	88.75 %
g=2	$138/2^{13}$	$144/2^{13}$	95.83 %
	k = 10		
g=0	$1506/2^{21}$	$1536/2^{21}$	98.04 %
g=1	$2532/2^{22}$	$2560/2^{22}$	98.91 %
g=2	$4517/2^{23}$	$4608/2^{23}$	98.02 %
	k = 15		
g=0	$49058/2^{31}$	$49152/2^{31}$	99.81 %
g=1	$81616/2^{32}$	$81920/2^{32}$	99.63 %
g=2	$147154/2^{33}$	$147456/2^{33}$	99.79 %

Table 6: Comparison of the greatest realized relative error with the relative error upper bound from Theorem 4 for k-bits-in, $(k+g)$-bits-out reciprocal table

4 Efficient Determination of Worst Case Inputs to Reciprocal Tables

Using Algorithm 1 and Theorem 3, a reciprocal table can be constructed and searched exhaustively to compute the maximum relative error and precision of the table. The search also reveals particular input intervals that realize the worst case reciprocal approximation. Knowledge of such intervals is useful for testing iterative refinement division algorithms employing such tables. The rapid approach to the bounds with increasing k in Table 6 is a good indication that we might expect to encounter worst case errors near the theoretical limit of Theorem 4 in large tables. The methods determining the limits did not indicate where the realized worst case bound and the corresponding input occured. Table 7 indicates no pattern to the occurance of worst case input intervals as found by exhaustive search, other than the trend that the worst case error occurs comparatively earlier in the table as k increases.

	worst input for g=0	worst input for g=1
k=5	1.01000	1.00011
k=6	1.001011	1.000010
k=7	1.0000101	1.0000100
k=8	1.00010010	1.00000101
k=9	1.000001011	1.000001000
k=10	1.0000100100	1.0000001011
k=11	1.00000100111	1.00000010000
k=12	1.000001010101	1.000000010110
k=13	1.0000010010001	1.0000000110111
k=14	1.00000001000000	1.00000000101101
k=15	1.000000001011010	1.000000010001111

Table 7: Table input values generating the worst case relative errors

For larger tables *e.g.* $k \geq 10$ it is desirable to find the intervals that realize the worst case error without having to generate and search the whole table. Exhaustive search can be avoided if we can predict and locate the input regions which are guaranteed to suffer the maximum relative errors. Logically that should happen for input regions which suffer rounding errors for midpoint reciprocal rounding close to 0.5 ulp. Adjacent pairs of such input regions can be characterized by a switch in the midpoint reciprocal rounding from round-down to round-up in realizing round-to-nearest, defining a break point in the arithmetic progression of table values. Table 8 shows the initial arithmetic progression of table values for the (10,10) bit reciprocal table through the first break point, and further shows the next nine break points as characterized by the rounding direction switches. A few particularly bad input regions are highlighted. Note that the reciprocals of these inputs suffer rounding errors very close to 0.5 ulp. In fact $(\frac{1060}{1024}, \frac{1061}{1024}]$ contains the worst input for the (10,10) bit reciprocal table.

Lemma 5 locates the first break point of a (k, k) bit reciprocal table.

Lemma 5: For a (k, k) bit table, the first break point corresponds to the adjacent pair of input intervals $(2^k + \frac{1}{2}2^{\frac{k}{2}} - 1, 2^k + \frac{1}{2}2^{\frac{k}{2}}]$ and $(2^k + \frac{1}{2}2^{\frac{k}{2}}, 2^k + \frac{1}{2}2^{\frac{k}{2}} + 1]$.
Outline of proof: Consider the first input interval whose reciprocal suffers a rounding error of 0.5 ulp. Let the mid-point of such an interval be $2^k + i + \frac{1}{2}$. Then the reciprocal of the mid-point is clearly $2^{k+1} - 2i - \frac{1}{2} + \epsilon'$ where ϵ' is suitably small.

mid-pt	mid-pt recip	rounded recip	rel error interval
$\times(1/1024)$	$\times(1/2048)$	$\times(1/2048)$	$\times(1/2^{21})$
1024.5	2047.000	2047	(1023,1024]
1025.5	2045.004	2045	(1018,1027]
1026.5	2043.012	2043	(1009,1034]
1027.5	2041.024	2041	(996,1045]
1028.5	2039.039	2039	(979,1060]
1029.5	2037.059	2037	(958,1079]
1030.5	2035.082	2035	(933,1102]
1031.5	2033.109	2033	(904,1129]
1032.5	2031.140	2031	(871,1160]
1033.5	2029.175	2029	(834,1195]
1034.5	2027.213	2027	(793,1234]
1035.5	2025.255	2025	(748,1277]
1036.5	2023.301	2023	(699,1324]
1037.5	2021.351	2021	(646,1375]
1038.5	2019.405	2019	(589,1430]
1039.5	2017.462	2017	(528,1489]
1040.5	**2015.523**	2016	(**1504**, 512]
.	.	.	.
1051.5	1994.438	1994	(536,1458]
1052.5	1992.543	1993	(1477, 516]
.	.	.	.
1059.5	1979.379	1979	(588,1391]
1060.5	**1977.512**	1978	(**1506**, 472]
.	.	.	.
1066.5	1966.387	1966	(570,1396]
1067.5	1964.545	1965	(1468, 497]
.	.	.	.
1072.5	1955.386	1955	(563,1392]
1073.5	1953.565	1954	(1444, 510]
.	.	.	.
1077.5	1946.313	1946	(636,1310]
1078.5	**1944.508**	1945	(**1503**, 442]
.	.	.	.
1082.5	1937.323	1937	(619,1318]
1083.5	1935.535	1936	(1472, 464]
.	.	.	.
1087.5	1928.416	1928	(512,1416]
1088.5	1926.644	1927	(1351, 576]
.	.	.	.
1091.5	1921.349	1921	(580,1341]
1092.5	1919.590	1920	(1408, 512]
.	.	.	.
1095.5	1914.333	1914	(592,1322]
1096.5	1912.587	1913	(1322, 504]

Table 8: Inputs leading to the first break point and a first few worst case inputs to a (10,10) bit table

Then $(2^k+i+\frac{1}{2})(2^{k+1}-2i-\frac{1}{2}) = 2^{2k+1}$ ignoring lower order terms. Solving for i in $i^2+\frac{3}{4}i-\frac{1}{4}2^k+\frac{1}{8}=0$ we get $i=-\frac{3}{8}+\frac{1}{2}2^{\frac{k}{2}}+\epsilon$ where ϵ is sufficiently small.
So for the first break point which corresponds to inputs which suffer midpoint reciprocal rounding errors of nearly 0.5 ulp, we get $i=\lfloor -\frac{3}{8}+\frac{1}{2}2^{\frac{k}{2}}+\epsilon \rfloor$ or $i=\lceil -\frac{3}{8}+\frac{1}{2}2^{\frac{k}{2}}+\epsilon \rceil$ which implies $i=\frac{1}{2}2^{\frac{k}{2}}-1$ or $i=\frac{1}{2}2^{\frac{k}{2}}$. Thus the first break point corresponds to the pair of input intervals
$(2^k+\frac{1}{2}2^{\frac{k}{2}}-1, 2^k+\frac{1}{2}2^{\frac{k}{2}}]$, $(2^k+\frac{1}{2}2^{\frac{k}{2}}, 2^k+\frac{1}{2}2^{\frac{k}{2}}+1]$. □

Corollary 5.1: For a $(k, k+g)$ bit table, the first break point corresponds to the pair of input intervals $(2^k+\frac{1}{2}2^{\frac{k-g}{2}}-1, 2^k+\frac{1}{2}2^{\frac{k-g}{2}}]$ and $(2^k+\frac{1}{2}2^{\frac{k-g}{2}}, 2^k+\frac{1}{2}2^{\frac{k-g}{2}}+1]$.
Proof: Similar to the proof of lemma 5.

We now need to show that certain inputs in the first break point intervals are indeed good candidates for the worst case inputs. In particular, if all the preceding input intervals are guaranteed to have relative errors less than the relative errors of the input interval pair at the first break point, then we can branch directly to the first break point without missing any candidate for worst input interval. In the following lemma we show that the first break point intervals are effectively worse than all the preceding intervals.

Lemma 6: Let the two input intervals in the first break point be I_1 and I_2 respectively and let the relative error intervals of I_1 and I_2 be denoted by $(r_1, r_2]$ and $(r_1', r_2']$ respectively. Also let $(r_1^{(i)}, r_2^{(i)}]$ be the relative error interval of any preceding input interval $(i, i+1]$ where $i < \frac{1}{2}2^{\frac{k}{2}}-1$.
Then $r_2 > \max_i r_2^{(i)}$ and $r_1' > \max_i r_1^{(i)}$.
Outline of proof: $r_2^{(i)} = 2^{2k+1}-(2^k+i)(2^{k+1}-2i-1) = 2i^2+i+2^k$ and $r_2 = 2^{2k+1}-(2^k+j)(2^{k+1}-2j-1) = 2j^2+j+2^k$, where $j=\frac{1}{2}2^{\frac{k}{2}}-1$. Since $i<j$, therefore $r_2 > \max_i r_2(i)$. $r_1(i) = (2^k+i+1)(2^{k+1}-2i-1)-2^{2k+1} = 2^k-2i^2-3i-1 \le 2^k-1$. $r_1' = (2^k+l+1)(2^{k+1}-2l)-2^{2k+1} = 2^{k+1}-2l^2-2l$, where $l=\frac{1}{2}2^{\frac{k}{2}}$, which gives $r_1' = \frac{3}{2}2^k-2^{\frac{k}{2}}$ which is clearly $> 2^k-1$. Thus $r_1' > \max_i r_1^{(i)}$. □

The implications of Lemma 5 and Lemma 6 are of paramount importance. We know the first region of bad input and can locate that easily. Note that the first break point corresponds to an adjacent pair of input intervals whose midpoint reciprocals are in the neighbourhood of *xxx*.5 ulps, the first one slightly below and the second one slightly above *xxx*.5 ulps, each of them thus suffering rounding errors close to 0.5 ulp. In fact all the other bad regions will comprise a pair of adjacent input intervals of the same nature. If we can locate all the other break points to a high degree of approximation and all the other inputs between these break points are guaranteed to be no worse than the inputs corresponding to these break points, then our search for worst case inputs will be limited to a few branches to those regions, performing a small local search in those regions and identifying the worst case inputs.
The following lemma computes the approximate branching point to these regions of worst case input.

Lemma 7: For a $(k, k+g)$ reciprocal table, The j^{th} break point is in a small neighbourhood of the input

$2^k + \frac{\sqrt{2j-1}}{2} 2^{\frac{k-g}{2}}$ where $j \geq 1$.

Outline of proof: Consider the input $2^k + i + \frac{1}{2}$ to correspond to the j^{th} break point. If its reciprocal suffers a rounding error of exactly 0.5 ulps, the scaled reciprocal is $2^{k+g+1} - 2^{g+1}i - (2^g - 1) - \frac{1}{2} + (j-1)$. The product of these two terms should be exactly equal to $2^{k+k+g+1}$, so setting the equation and ignoring the lower order terms, we solve for i to obtain $i = \frac{\sqrt{2j-1}}{2} 2^{\frac{k-g}{2}}$. Since there is a small degree of approximation in the solution for i, the j^{th} break point is in a small neighbourhood of the input $2^k + \frac{\sqrt{2j-1}}{2} 2^{\frac{k-g}{2}}$ □

To devise our branch and bound search procedure, the only thing that needs to be proved is that the inputs falling in the interval between the break points need not be considered as candidates for the worst case inputs.

Lemma 8: For a $(k, k+g)$ bit reciprocal table the inputs falling between the j^{th} and $(j+1)^{th}$ break points have relative errors no worse than the maximum relative errors of the inputs corresponding to the $(j+1)^{th}$ break point.

Proof: Similar to the proof of Lemma 6.

Algorithm 2
[Search of Large Tables for Worst Case Inputs]

Stimulus: Integers $k \geq 10$ and $g \geq 0$

Response: The worst case inputs to the k-bits-in $(k+g)$-bits-out ROM reciprocal table

```
Method:  for $i = 1$ to 12 step 1
         <compute the first twelve break points>
         begin
L1:        Compute the $i^{th}$ break point as
           $break_i = 2^k + \frac{\sqrt{2i-1}}{2} 2^{\frac{k-g}{2}}$
L2:        Perform a small local search in the neighbourhood of $break_i$ and identify the two
           adjacent locally worst case inputs.
         end
```

Theorem 9: For a $(k, k+g)$ bit reciprocal table, Algorithm 2 generates the worst case inputs to the table.

Outline of proof: It is immediate from Lemma 7 and Lemma 8 that all the worst case inputs should be contained in the break point intervals which correspond to the bad input regions. For large tables it is desirable to bound the branching procedure described in Algorithm 2 to find the worst case inputs, so that the search is limited to only a small portion of the table. It can be proved that for $k \geq 10$, the search of the tables for worst case inputs can be bounded by the 12th break point such that the worst case input is guaranteed to fall within the first 12 such input interval pairs. The proof is beyond the scope of this paper and will be included in an expanded version. □

References

[AC 74] G. M. Amdahl and M. R. Clements, "Method And Apparatus For Division Employing Table-Lookup And Functional Iteration", in *United States Patent*, No. 3,828,175, 1974.

[BM 91] W. B. Briggs and D. W. Matula, "Method and Apparatus for Performing Division Using a Rectangular Aspect Ratio Multiplier", in *United States Patent*, No. 5,046,038, 1991.

[Co 90] D. Cocanougher, "Floating Point Division Method And Apparatus", in *European Patent Application*, Publication No. 0,377,992, 1990.

[DG 89] H. M. Darley and M. C. Gill,, "Floating Point/Integer Processor With Divide and Square Root Functions", in *United States Patent*, No. 4,878,190, 1989.

[Fe 67] D. Ferrari, "A Division Method Using a Parallel Multiplier", in *IEEE Trans. Electron. Comput.* , 1967, EC-16, pp 224-226.

[FS 89] D. L. Fowler and J. E. Smith, "An Accurate High Speed Implementation of Division by Reciprocal Approximation", in *Proc.* 9th *IEEE Symp. Comput. Arithmetic* , 1989, pp 60-67.

[FW 71] C.V. Freiman and C.C. Wang, "Division System And Method", in *United States Patent*, No. 3,591,787, 1971.

[Go 64] R. E. Goldschmidt, "Applications of division by convergence", in *M.S. thesis, Massachusetts Institute of Technology, Cambridge, Mass.*, June 1964.

[Ka 91] H. Kadota, "Apparatus For Executing Division By High Speed Convergence Processing", in *United States Patent*, No. 4,991,132, 1991.

[OL 91] W. J: Ooms, C. D. Leitch and R. M. Delgado, "Method And Apparatus For Obtaining The Quotient Of Two Numbers Within One Clock Cycle", in *United States Patent*, No. 5,020,017, 1991.

[SB 86] J. R. Schomburg and L. B. Bushard, "Apparatus For Performing Quadratic Convergence Division In A Large Data Processing System", in *United States Patent*, No. 4,594,680, 1986.

[WF 91] D. C. Wong and M. J. Flynn, "Fast Division Using Accurate Quotient Approximations to Reduce the Number of Iterations", in *Proc.* 10th *IEEE Symp. Comput. Arithmetic*, 1991, pp 191-201.