A Formal Model and Efficient Traversal Algorithm for Generating Testbenches for Verification of IEEE Standard Floating Point Division

David W. Matula, Lee D. McFearin
Department of Computer Science and Engineering
Southern Methodist University
Dallas, TX 75275
email : {matula,mcfearin}@engr.smu.edu

Abstract

We utilize a formal model of division for determining a testbench of p-bit (dividend, divisor) pairs whose output 2p-bit quotients have properties characterizing these instances as the most challenging for verifying any division algorithm design and implementation. Specifically, our test suites yield 2p-bit quotients where the leading p-bits traverse all or a pseudo-random sample of leading bit combinations, and the next p-bits comprise a round bit followed by (p-1) identical bits. These values are proven to be closest to the p-bit quotient rounding boundaries and shown to possess other desirable coverage properties. We introduce an efficient method of generating these testbenches. We also describe applications of these testbenches at the design simulation stage and the product evaluation stage.

1. Introduction

The IEEE floating point standard [IEEE] requires all conforming implementations of division and square root to correctly round the result as if the infinitely precise result had been calculated and then rounded to the target precision. As processors become larger and these implementations become more complex, the chances for expensive flaws increase [SB94]. Single precision unary functions such as square root and square root reciprocal have only 16 million distinct inputs and may be exhaustively tested. However, division has two arguments generating 64 trillion distinct normalized single precision input cases, and almost a quintillion times as many distinct double precision input cases.

Since division can not be exhaustively tested in a reasonable amount of time, selective, statistical test suites are typically used for verification. One such type of test suite focuses on those instances which yield infinitely precise quotients extremely close to the rounding boundaries[Ka87,Pa99, MM02, MM03]. These instances are the most sensitive to calculation errors as a very small approximation error may cause the result to cross a rounding boundary and round incorrectly.

Note that the problem of determining extremal rounding boundary cases for transcendentals has no known conveniently scalable solution for higher precisions and has been termed the "table maker's dilemma" [Mu97,LM98,LM04,Zi91,DE05]. While a body of excellent research on search techniques has evolved for finding extremal "hardest-to-round" cases for double precision and some double extended precision transcendental functions, it appears to be very hard to obtain a general characterization.

In contrast, the problem of determining extremal rounding boundary cases for division has a firm number theoretic foundation that is conveniently scalable to higher precisions. Our solution herein improves the efficiency of generating benchmark sets for this problem and is instructive in that it comprises fundamental elementary results from three distinct number types as follows.

i. Radix Arithmetic: the problem of determining extremal rounding boundary cases for floating point division is initially cast as a problem with binary radix floating point inputs and outputs.

ii. Rational Arithmetic: The extremal rounding boundary cases are recharacterized employing rational arithmetic where Farey fractions and continued fractions conveniently identify the desired extremal cases.

iii. Residue Arithmetic: The core computational steps are simplified by utilizing properties of the multiplicative inverse modulo 2^k. This is the principle addition herein to our previously published methodology.

3-9810801-0-6/DATE06 © 2006 EDAA

2. Background

The size of a floating point division operation is characterized by the size of the significand of the arguments. A positive *$p-$bit number* is a binary rational of the form $2^e i$, where $1 \le i \le 2^p - 1$ and i is odd. Thus, the *precision* $p \ge 1$ provides a measure of the size of the p-bit number's significand bit string in the normalized floating point factored format $z = 2^{e'}(1.b_1 b_2 \cdots b_{p-1})$.

A *p×p bit fraction* denotes a fraction $\frac{n}{d}$ where the numerator and denominator are positive p-bit integers[MM00,MM03]. The rational value $q = \frac{n}{d}$ is termed the infinitely precise *p×p bit quotient*. For our purposes herein a *normalized p×p* bit fraction has the denominator in the range $1 \le d \le 2^p - 1$, and the numerator in the range $d \le n < 2d$, so then the *normalized p×p* bit quotient is in the standard binade $1 \le q = \frac{n}{d} = 1.b_1 b_2 \cdots < 2$. Note that if this normalization yields a numerator greater than 2^p, it must be even to be a p-bit number.

Following the literature on continued fractions and Farey series, two fractions are *adjacent* if their cross product is unity [HW79,MK85]. With this background, the *extremal rounding boundary instances* of $p \times p$ bit fractions $\frac{n}{d}$ for *round-to-nearest* floating point division have been characterized by three distinct, and provably equivalent definitions[MM00, MM02, MM03]:

(i) [Distance form] the distance of $\frac{n}{d}$ from a closest p-bit midpoint $\frac{i}{2^p}$ satisfies $| \frac{n}{d} - \frac{i}{2^p} | < \frac{1}{2^{2p-1}}$.

(ii) [Binary expansion form] $\frac{n}{d}$ has the standard binary expansion $\frac{n}{d} = 1.b_1 b_2 \cdots b_{p-1} b_p \cdots$ where the tail "rounding critical" $b_p b_{p+1} \cdots b_{2p}$ is either 100...01 or 011..10 containing the maximum length run of $p-1$ like valued bits opposite to the round bit b_p.

(iii) [Best rational approximation form] $\frac{n}{d}$ is the best rational approximation of a p-bit midpoint, and is given by the closest continued fraction convergent to a p-bit midpoint $\frac{i}{2^p}$.

The set of $p \times p$ bit fractions equivalently defined by these characterizations for a given p is denoted by RN_p. For example, the set RN_5 of extremal midpoint rounding boundary 5×5 bit fractions has ten members and can be developed employing the multiplicative inverse as illustrated in Table 1. For any odd integer a with $1 \le a \le 31$, let a^{-1} be the unique odd integer with $1 \le a^{-1} \le 31$ and $aa^{-1} \equiv 1$ (mod 32). Traversing the rows for $a = 1, 3, 5, \ldots, 31$ in Table 1, the fractions $\frac{32+a}{32}$ traverse the midpoints of the five bit intervals $[\frac{i}{16}, \frac{i+1}{16})$ for $16 \le i \le 31$.

Let n and n' be determined by $(32 + a)a^{-1} = 32n + 1$ and $(32 + a)(32 - a^{-1}) = 32n' - 1$. This assures that $\frac{32+a}{32} - \frac{n}{a^{-1}} = \frac{1}{32a^{-1}}$ and $\frac{n'}{32-a^{-1}} - \frac{32+a}{32} = \frac{1}{32(32-a^{-1})}$. Then $\frac{n'}{32-a^{-1}} \in RN_5$ when $1 \le a^{-1} \le 15$ and n' is a 5 bit number (i.e. either $n' < 32$ or n' is even). Similarly $\frac{n}{a^{-1}} \in RN_5$ when $17 \le a^{-1} \le 31$ and n is a 5 bit number.

a	$\frac{n}{a^{-1}}$	$\frac{32+a}{32}$	$\frac{n'}{32-a^{-1}}$	**Extremal Value**
1	-	$\frac{33}{32}$	$\frac{32}{31}$	1.0000 10000 10000
3	-	$\frac{35}{32}$	$\frac{23}{21}$	1.0001 10000 11000
5	-	$\frac{37}{32}$	$\frac{22}{19}$	1.0010 10000 11010
7	$\frac{28}{23}$	$\frac{39}{32}$	-	1.0011 01111 01001
9	$\frac{32}{25}$	$\frac{41}{32}$	-	1.0100 01111 01011
11	-	$\frac{43}{32}$	-	
13	-	$\frac{45}{32}$	$\frac{38}{27}$	1.0110 10000 10010
15	-	$\frac{47}{32}$	$\frac{25}{17}$	1.0111 10000 11110
17	$\frac{26}{17}$	$\frac{49}{32}$	-	1.1000 01111 00001
19	-	$\frac{51}{32}$	-	
21	$\frac{48}{29}$	$\frac{53}{32}$	-	1.1010 01111 01110
23	-	$\frac{55}{32}$	-	
25	-	$\frac{57}{32}$	-	
27	-	$\frac{59}{32}$	-	
29	$\frac{40}{21}$	$\frac{61}{32}$	-	1.1110 01111 00111
31	-	$\frac{63}{32}$	-	

Table 1: 5×5 bit Extremal Midpoint Rounding Boundary Instances

3. Enumeration of Extremal Rounding Boundary Instances

For any precision $p \ge 2$, the set of extremal midpoint rounding boundary $p \times p$ bit fractions can be determined as follows.

Observation 1 *Let a be a p-bit odd integer and a^{-1} its p-bit multiplicative inverse. Then*
(i) if $2^{p-1} + 1 \le a^{-1} \le 2^p - 1$ let $n = \frac{(2^p+a)a^{-1}-1}{2^p}$. If n is even

or $n < 2^p$, then $\frac{n}{a^{-1}} \in RN_p$,

(ii) if $1 \le a^{-1} \le 2^{p-1} - 1$ let $n' = \frac{(2^p+a)(2^p-a^{-1})+1}{2^p}$. If n' is even or $n' < 2^p$, then $\frac{n'}{2^p-a^{-1}} \in RN_p$.

Note that the multiplicative inverse pairs a, a^{-1} are distinct odd value pairs except for the four cases $a = 1, 2^{p-1} - 1, 2^{p-1} + 1$, and $2^p - 1$, where a is its own multiplicative inverse.

Once a multiplicative inverse pair a,a^{-1} is identified, n or n' may be calculated via multiplication giving an extremal instance in most cases. On average 11 total extremal rounding boundary instances for round-to-nearest and directed rounding may be calculated from $\frac{n}{a^{-1}}$ (or $\frac{n'}{2^p-a^{-1}}$) using addition via the symmetric properties discussed in [MM01] and demonstrated in Example 1.

Example 1 *Given a multiplicative inverse pair modulo 32 $(a_k, a_k^{-1}) = (3, 11)$, the multiplication step described earlier, produces $n' = 23$. From Observation 1, $\frac{n}{a^{-1}} = \frac{23}{21} \in RN_p$. Furthermore, from [MM01], $\frac{40}{21}, \frac{22}{19}, \frac{38}{27}, \frac{48}{29} \in RN_p$, and $\frac{25}{21}, \frac{38}{21}, \frac{25}{19}, \frac{32}{27}, \frac{32}{19} \in RD_p$.*

After the multiplicative inverse pair a, a^{-1} and n (or n^{-1}) has been determined, the computational cost for each group of extremal rounding boundary instances as found in Example 1 is 22 additions. Past approaches have generated these multiplicative inverse pairs by using the Euclidean GCD Algorithm for each pair. The key to the creation of an efficient algorithm based on Observation 1 lies in the systematic modular generation of the $2^{p-2} + 2$ multiplicative inverse pairs.

Observation 2 *For $p \ge 3$, let $a_k \equiv 3^k \pmod{2^p}$ and $a_k^{-1} \equiv 3^{2^{p-2}-k} \pmod{2^p}$ be standard residues modulo 2^p for $k = 0, 1, 2, \ldots, 2^{p-3}$. Then the $2^{p-2} + 2$ multiplicative inverse pairs $(a_k,\ a_{k^{-1}})$ and $(2^p - a_k,\ 2^p - a_{k^{-1}})$ for $0 \le k \le 2^{p-3}$ contain all 2^{p-1} odd integers in $[1, 2^p - 1]$.*

Tables 2 and 3 illustrate the use of exponential residues $3^k(mod 2^p)$ in generating the multiplicative inverse pairs a, a^{-1} for $p = 5$ employing Observation 2.

	Exponential	**Residues**
k	$3^k(mod32)$	$3^{8-k}(mod32)$
0	1	1
1	3	11
2	9	25
3	27	19
4	17	17

Table 2: The 10 multiplicative inverse residues modulo 32.

	Multiplicative	**Inverse Pairs**
k	(a_k, a_k^{-1})	$(32 - a_k, 32 - a_k)$
0	(1, 1)	(31, 31)
1	(3, 11)	(29, 21)
2	(9, 25)	(23, 7)
3	(27, 19)	(5, 13)
4	(17, 17)	(15, 15)

Table 3: The 10 multiplicative inverse pairs modulo 32.

This method requires only a multiplication with two shift-and-add operations for determining a multiplicative inverse pair along with an n or n'. Using this computational cost along with the cost of finding the symmetries in [MM01] an average cost can be determined.

Lemma 1 *The average cost of 11 extremal rounding boundary instances is 1 multiplication and 24 additions providing a cost of slightly over 2 additions per extremal rounding boundary instance.*

Using Observation 2, we may determine all the extremal rounding boundary instances for $p{\times}p$ bit binary floating point division. The following steps outline the process.

- Create a storage array containing 2^{p-3} elements for $a \equiv 3^k(mod 2^p)$ as ARRAY; and a similar array for $a^{-1} \equiv 3^{2^{p-3}-k}(mod 2^p)$ as INV_ARRAY.
- Compute ARRAY[1] as $3^k \pmod{2^p}$, where k is the starting value, possibly 1.
- Compute ARRAY[$n + 1$] = 3 × ARRAY[n] (mod 2^p). Note this can be computed as ARRAY[$n + 1$] = (ARRAY[n] + (ARRAY[n] << 1)) (mod 2^p) where << is a binary shift operation.
- The last element of ARRAY is the last element of INV_ARRAY.

- Compute INV_ARRAY$[n-1] = 3 \times$ INV_ARRAY$[n]$ (mod 2^p).

Once ARRAY and INV_ARRAY have been computed, each ARRAY$[m]$ and INV_ARRAY$[m]$ represent an a, a^{-1} pair. Additionally $2^p - a$, $2^p - a^{-1}$ is a second pair. Note that the full set of $2^{p-2} + 2$ multiplictive inverse pairs can be generated employing only 2^{p-2} shift-and-add operations and $2^{p-2} + 2$ complement operations. The complement operation is a 1's complement since the values are odd integers. With at least one of these complementary pairs, extremal rounding boundary instances can be calculated with a multiplication as mentioned in Observation 1. Using all k as discussed in Observation 2 generates the entire set RN_p. The benchmark set RN_{24} for testing IEEE standard single precision division implementations is enumerated and available at the site [MM04]. Using this procedure the number of extremal midpoint rounding boundary cases $|RN_p|$ was computed for $3 \le p \le 28$ and is shown in Table 4.

p	$\|RN_p\|$	p	$\|RN_p\|$
3	3	16	22710
4	6	17	45393
5	10	18	90920
6	24	19	181620
7	40	20	363536
8	87	21	726476
9	173	22	1453890
10	359	23	2906902
11	703	24	5815346
12	1424	25	11628333
13	2832	26	23259306
14	5695	27	46515099
15	11319	28	93035551

Table 4: Number of Extremal Midpoint Rounding Boundary Cases for Precisions 3 to 28

To compute a psuedo random subset of RN_p with this method, a subsequence of the enumeration may be selected at random. Specifically, a first element may be picked at random for ARRAY and the "turning point" last element of INV_ARRAY may be obtained with the extended Euclidean algorithm from the "turning point" last element of ARRAY. It is possible to avoid the need for the Euclidean algorithm by restricting the turning points to elements a for which a^{-1} is computable by a single addition. Observation 3 provides a set of such multiplicative inverse pairs that are uniformly spaced throughout the sequence a_k, a_k^{-1} for $a \equiv 3^k$ $(mod 2^p)$ as k traverses the range $k = 0, 1, \ldots, 2^{p-2}$.

Observation 3 Let $j \ge 3$ and n be odd satisfying $1 \le n \le 2^{j+3} - 1$. Then with

$$a = n2^j + 1,$$
$$a^{-1} = |(2^j - n)2^j + 1|_{2^{2j+3}},$$

we obtain that a, a^{-1} is a multiplicative inverse pair modulo 2^{2j+3}.

IEEE standard double precision has $p = 53$, which would lead to $2^{51} + 2$ multiplicative inverse pairs in the full enumeration as given for $p = 5$ in Table 2. Employing Observation 3 the full sequence can be partitioned into subsequences of size 2^{24} pairs selectable by a 27-bit random integer seed q. Thus our enumeration procedure is initiated at $a = q2^{26} + 1$, and the turning point occurs after 2^{23} shift-and-adds when $a' = |3^{2^{23}}(q2^{26} + 1)|_{2^{53}} = q'2^{25} + 1$ with q' odd. This provides a reasonable size psuedo random sample of about 16 million multiplicative inverse pairs using only the addition operation, which would provide a psuedo random subset of RN_{53} of size about 22 million.

4. Applications

The sets RN_{24} and RD_{24} along with a C++ program for generating RN_p for $3 \le p \le 28$ are provided by the link [MM04]. The sets RN_{24} and RD_{24} were developed for and used in the design verification stage to test the floating point divide implementation on the one watt x86 compatible Geode processor developed by National Semiconductor in 2001 and now available from AMD. The set was also employed in an after-the-fact product evaluation stage to search for division implementation errors in the 1993 Pentium Processor having the well known fdiv flaw [SB94]. The test suites found the erroneous fdiv quotients with a frequency much higher than that of random testing, surprisingly some 10,000 times (4 orders of magnitude) more frequent for single precision. For evidence that finding these single precision erroneous quotients was not a fortuitous "accident", further testing showed that these distinct sets for the 15 precisions $22 \le p \le 36$, uncovered the flaw at each precision at a much higher frequency than random testing[MM02].

5. Conclusions

The extremal rounding boundary instances provide a natural testbench for design verification of any implementation of IEEE standard floating point division. They provide a large number of test cases, which can be calculated with relative ease. As the size of the division problems grow, a simple modular enumeration test suite can be carefully controlled statistically and as a sample still maintain a general coverage over the quotient, divisor, and dividend ranges. As improved formal models reveal more structured approaches to testing division, fewer flaws are likely to appear in future designs.

6. Acknowledgments

We thank Jason Moore for his assistance in the development of this paper.

This work was supported in part by the Semiconductor Research Corporation under contract RID 1289.

References

[DE05] F. de Dinechin, A. V. Ershov, and N. Gast, "Towards the post-ultimate libm," *Proc. 17th IEEE Symposium on Computer Arithmetic*. IEEE, 2005.

[HW79] C. H. Hardy and E. M. Wright, *An Introduction to the Theory of Numbers,* 5th ed. London, England: Oxford University Press, 1979.

[IEEE] *IEEE Standard 754 for Binary Floating Point Arithmetic*, ANSI/IEEE Standard No. 754, American National Standards Institute, Washington DC, 1988.

[Ka87] W. Kahan, *Checking Whether Floating Point Division is Correctly Rounded*, monograph, April 11, 1987, (see home page http://www.cs.berkeley.edu/~wkahan).

[LM98] V. Lefevre, Jean-Michel Muller, and A. Tisserand, "The Table Maker's Dilemma," *IEEE Transactions on Computers* http://perso.ens-lyon.fr/jean-michel.muller/Intro-to-TMD.htm, 1998

[LM04] V. Lefevre, and Jean-Michel Muller "Worst cases for correct rounding of the elementary functions in double precision. " http://perso.ens-lyon.fr/jean-michel.muller/Intro-to-TMD.htm, 2004

[MK85] D. W. Matula, P. Kornerup "Finite Precision Rational Arithmetic: Slash Number Systems." *IEEE Transactions on Computers*, Vol. C-34 No. 1, January 1985.

[MM00] D. W. Matula, L. D. McFearin "Number Theoretic Foundations of Binary Floating Point Division with Rounding" *Proceedings: Fourth Real Numbers and Computers*, April 2000, pp. 39-60.

[MM01] L. D. McFearin and D. W. Matula, "Generation and Analysis of Hard to Round Cases for Binary Floating Point Division." *Proc. 15th IEEE Symposium on Computer Arithmetic*. IEEE, 2001. pp.119-126.

[MM01a] L. D. McFearin and D. W. Matula, Selecting Test Suites for IEEE Standard Floating Point Division." *Proc.* 19*th IEEE International Conference on Computer Design*. IEEE, 2001.

[MM02] L. D. McFearin, "A p-Bit Model of Binary Floating Point Division and Square Root with Emphasis on Extremal Rounding Boundaries.", Ph. D. Dissertation, Southern Methodist University, Dallas, Texas, May 2002.

[MM03] D. W. Matula, L. D. McFearin "A pxp Bit Fraction Model of Binary Floating Point Division and Extremal Rounding Cases." *Journal of Theoretical Computer Science*, 291:159-182, 2003.

[MM04] D. W. Matula, See home page http://www.engr.smu.edu/~matula/extremal.html

[Mu97] J.-M. Muller, *Elementary Functions, Algorithms and Implementations.* Birkhauser, Boston, 1997.

[Pa99] M. Parks, "Number-theoretic Test Generation for Directed Rounding," *Proc. 14th IEEE Symposium on Computer Arithmetic*. IEEE, 1999.

[SB94] H.P. Sharangpani, M. L. Barton, "Statistical Analysis of Floating Point Flaw in the Pentium Processor", Intel Corporation, 1994.

[Zi91] A. Ziv. Fast evaluation of elementary mathematical functions with correctly rounded last bit. *ACM Transactions on Mathematical Software,* 17(3):410-423, Sept. 1991.