A FEASIBILITY ANALYSIS OF BINARY FIXED-SLASH AND FLOATING-SLASH NUMBER SYSTEMS

David W. Matula
Peter Kornerup

Computer Science Department Southern Methodist University Dallas, Texas
Computer Science Department Aarhus University Aarhus, Denmark

Abstract

Design and analysis of finite precision rational number systems based on fixed-slash and floating-slash representation is pursued. Natural formats for binary fixed-slash and binary floating-slash number representation in computer words are described. Compatibility with standard integer representation is obtained. Redundancy in the representation is shown to be minimal. Arithmetic register requirements are considered. Worst case and average case rounding errors are determined, and the concept of adaptive variable precision in the rounding is developed.

Key Words and Phrases:

Rational arithmetic, Finite precision, Fixed-slash numbers, Floating-slash numbers, Binary numeric word format, Redundancy, Range of number system, Integer compatibility, Adjacent fractions, Farey fractions, Mediant roundings, Worst case and average case rounding error, Adaptive variable precision.

CR Categories:

5.11, 6.32, 3.15

I. Introduction and Summary

The purpose of this and a companion paper [1] is to explore the feasibility and merit of computer number systems composed of limited precision fractions. Two specific systems originally proposed in [2,3] which we shall investigate are characterized as follows:

(i) Fixed-slash: numerator and denominator size are independently bounded, e.g. N numerator and N denominator bits per word,

(ii) Floating-slash: the total number of digits in numerator and denominator is bounded, e.g. n_1 numerator bits, n_2 denominator bits, $n_1+n_2 \leq N$, and a slash position field per word.

The merits of these proposed finite precision fraction number systems will be assessed by addressing the following questions:

(1) Naturalness: How natural are the representable numbers of these fraction number systems? Are they sufficiently natural to suggest canonical implementations on computers made by different manufacturers that would enhance portability of numeric software in these systems?

(2) Cost-Effectiveness: How cost-effective is arithmetic unit design to realize the algorithms for standard arithmetic operations in these fraction number systems?

(3) Accuracy: How numerically accurate is scientific computation hosted in these fraction number systems?

In section II we address question (1) and describe natural computer word formats for binary fixed- and floating-slash number systems. The obvious redundancy due to representation of reducible as well as irreducible fractions is shown to cost less than one bit per machine word, and the normalization of floating-slash is shown to cost at most another bit per machine word. These storage efficiency costs are clearly negligible for standard number representations employing at least 32 bits. The fraction formats proposed are designed to be a natural extension of standard integer formats, allowing for the possibility that these fraction number systems might replace and extend standard integer arithmetic units, and possibly obviate the need for floating-point hardware on many machines. Range/accuracy tradeoffs are considered, and extended range floating-slash systems comparable in range/accuracy specifications to floating-point systems are shown to be readily obtainable.

Natural advantages of fraction number systems over fixed- and floating-point systems include:

(i) the ability to exactly represent all "simple" fractions (e.g. 1/3, 3/13, etc.),

(ii) the ability to generate exact intermediate results of the standard arithmetic operations, including division, in suitable "double length" registers before rounding,

(iii) the existence within the system of exact multiplicative inverses for the members of each such number system,

(iv) the independence of base in the characterization of fixed-slash (a feature of significant importance in investigations of the foundations of these number systems).

Question (2) on the cost-effectiveness of implementing approximate arithmetic in these limited precision fraction number systems reduces immediately to the design of an accurate and efficient rounding procedure. Feasibility of rounding for fixed-slash is described in detail in our companion paper [1], where the natural and theoretically desirable "mediant rounding" [3] is shown to be achievable at a cost comparable to a standard binary division instruction. Although more expensive than (the variety of) roundings employed in floating-point, the cost is certainly not prohibitive. Additional research [5] has shown comparable results for floating-slash number systems.

Section III of this paper addresses the preceding question (3). The concept of adjacent rationals is utilized to extend the theory of Farey fractions to provide a foundation for both floating- and fixed-slash arithmetic. The natural "mediant rounding" is defined. For a fixed-slash system composed of N numerator and N denominator bits, we show that although the variable gap size between successive representable fractions yields accuracy varying between N bits and 2N bits, the rounding error for a number uniform on [0,1] is sufficiently limited to allow an average accuracy of approximately $(2N-\lg N)$ bits.

For scientific computations the control of relative error throughout the number range is of more critical concern. The relative gap size for an N bit floating-slash fraction system similarly varies in relative accuracy between N/2 and N bits.

Rather than apologize for the variable precision feature and only utilize the conservative N/2 bit worst case bound, we suggest that this natural variable precision feature may represent a significant positive contribution to control of error accumulation in finite-precision computation for the following reason. The larger "rounding errors" are associated with roundings to simpler fractions. Thus the simpler fractions represent a subsystem of smaller precision, or conversely, the fractions with larger numerators and denominators represent a higher precision background approximation space for extended computations whose input and output can be suitably described by simpler fractions. We are thus led to conjecture that this natural adaptive variable precision feature may be a very desirable error control property allowing recovery of true intermediate or final results after an extensive accumulation of more moderate rounding errors. This is particularly plausible for special classes of problems such as rational matrix computations.

In summary, we assert that this and the companion paper [1] show the feasibility of implementation of binary fixed- and floating-slash arithmetic at a cost comparable in storage efficiency to that of equivalent fixed- and floating-point systems, and in execution time efficiency roughly 3 to 5 times the cost of comparable floating-point systems. This is not prohibitive for most numeric computations in view of currently attainable basic machine cycle times.

More comprehensive investigations of the foundations of finite precision fraction number sustems are currently in preparation [4,5]. The desirable properties of fraction representation and the potential benefits of adaptive variable precision in conjunction with the existence of microprogramable numeric architecture for implementation suggests to us that research in fraction systems will become quite vigorous.

II. Limited Precision Fractions: Normalization, Range and Redundancy

Fractions are quite simply represented as (numerator, denominator) integer pairs. A computer representation of a fraction must provide both compact efficient storage along with rapid accessibility to numerator and denominator portions. Furthermore, the set of representable fractions allowed by a particular computer word format (implicitly determining the precision limitation) must provide a set of representable real values that serve to control and minimize error accumulation for closed numeric computations in the system. In this paper we shall specifically investigate binary fixed-slash and binary floating-slash number representation in computer words and arithmetic units.

Let the <u>binary fixed-slash system</u> denote the set of fractions KX given by

$$KX=KX(2N)=\{(p,q)\,\big|\,0\le|p|\le 2^N-1,\ 0\le q\le 2^N-1\} \text{ for } N\ge 1. \quad (1)$$

Sign-magnitude binary representation of the set KX in a 2N+2 bit machine word is conveniently described in Figure 1.

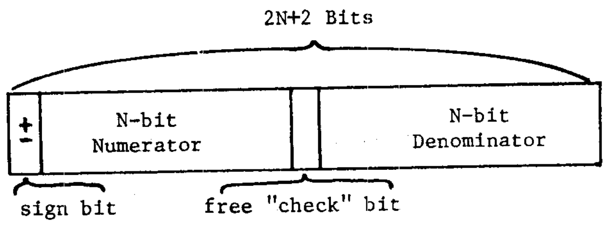


Figure 1. Binary Fixed-Slash Representation.

Note that the numerator and denominator can be right shifted in parallel until at least one of the low order bits is zero, this operation corresponds to canceling any factors of two in the greatest common divisor. The representation of Figure 1 is then termed <u>normalized</u> if at least one of the low order bits of the numerator and denominator is unity, and more strongly, is termed <u>reduced</u> if the greatest common divisor of the numerator and denominator is unity.

The following important results are immediate from the definitions.

<u>Observation 1. [Exact Inverses]</u>: For every fraction $(p,q) \in KX$, the additive inverse $(-p,q)$ and multiplicative inverse (q,p) are both in KX and are

normalized and reduced if the original fraction was normalized and reduced, respectively.

Comment: Note that the inverse operations are readily implemented by (1) complementing the sign bit and by (2) swapping the N bit numerator and denominator fields, respectively.

Observation 2. [Integer Compatibility]: The leading N+1 bits (half-word) of the reduced binary fixed-slash representation of a fraction is the standard integer representation if and only if the right most N bits contain the denominator value unity.

Comment: This justifies the binary fixed-slash representation as a convenient extension of binary integer representation where the integrality test is trivial for a reduced fraction. If it is not known whether or not the fraction is reduced, then application of well known binary gcd methods (discussed in our companion paper [1]) provides an efficient reduction.

Observation 3. [Independence of Base]: The set, KX, of numeric values represented is not influenced by the fact that binary radix representation is utilized to represent the numerator and denominator.

Comment: This means that the range limit (largest integer) in fixed-slash also implicitly determines all gaps between representable values, allowing for analysis of the finite precision structure in a more natural (base independent) manner.

Redundancy occurs in fixed-slash systems since reducible as well as irreducible fractions can be represented. The loss in representation efficiency is negligible, as the following classical number theoretic result of Cesàro shows (see [6] p. 314 for a proof).

Theorem 1: Let I_n be the number of finite valued irreducible fractions $\frac{p}{q}$, where $0 \le p \le q \le n-1$. Then

$$\lim_{n\to\infty} \frac{I_n}{n^2} = \frac{6}{\pi^2} = .6079\ldots \quad (2)$$

In binary representation of a class of objects by fixed length N-bit code words it is appropriate to measure redundancy and undefined code words in terms of bit loss defined as follows:

$$\text{Bit loss} = \begin{pmatrix}\#\text{ bits in}\\ \text{code words}\end{pmatrix} - \log_2\begin{pmatrix}\#\text{ distinct objects}\\ \text{represented}\end{pmatrix}$$

Taking irreducible fractions as the distinct objects we obtain the following:

Corollary 1.1: The binary fixed-slash representation of Figure 1 for sufficiently large N has a bit loss of approximately $1-\log_2(6/\pi^2)=1.718\ldots$ bits, of which one bit is the free (check) bit and 0.718... bits are lost due to redundancy.

Although Theorem 1 and Corollary 1.1 are valid only in the limit for large N, actual computation of the ratio I_n/n^2 for $n=2^N$ (e.g. N=2,3,4 below) suggests that this limiting value is approached rapidly and is suitable for typical word sizes (e.g. $8 \le N \le 120$).

$$\frac{I_8}{8^2} = \frac{36}{64} = .5625,$$

$$\frac{I_{16}}{16^2} = \frac{144}{256} = .5625,$$

$$\frac{I_{32}}{32^2} = \frac{616}{1024} = .601\ \ldots$$

That the 1.718... bits lost in fixed-slash is negligible is apparent in that the .718... bits lost to redundancy is less than the loss of the leading bit in normalized floating-point which is traditionally ignored. Of course it is not necessary to have the free (check) bit in Figure 1 since a 2N+1 bit word would be ample for the representation, but there are compelling architectural reasons to employ the representation of Figure 1.

(i) Machine words with an even number of bits are standard, and avoiding an imbalance between the number of numerator bits and denominator bits means the inverse operation will never overflow,

(ii) extraction of numerator or denominator and appending the sign yields a standard half-word integer in both cases,

(iii) the results of standard fraction arithmetic for $(p,q), (r,s) \in KX$ are given by

$$\frac{p}{q}+\frac{r}{s}=\frac{ps+qr}{qs}, \qquad \frac{p}{q}-\frac{r}{s}=\frac{ps-qr}{qs},$$
$$\frac{p}{q}\times\frac{r}{s}=\frac{pr}{qs}, \qquad \frac{p}{q}\div\frac{r}{s}=\frac{ps}{qr}. \quad (3)$$

It is evident from (3) that the maximum numerator magnitude of any arithmetic operation requires 2N+1 bits and the maximum denominator magnitude requires 2N bits. With the provision for a sign, a 2N+2 bit word length is then necessary for the numerator result as well as being adequate for the denominator result. Thus the presence of the free bit in the binary fixed-slash word representation of Figure 1 assures that all standard arithmetic operations give exact results representable in two words without overflow.

Fixed-slash representation has significant positive features for data scaled to fall in the unit interval [0,1], and clearly also provides a natural and desirable extension of integer arithmetic over the region $[1,2^N-1]$. Thus fixed-slash arithmetic is proposed as an attractive hardware/firmware realization and extension of arithmetic of type INTEGER, or more generally, of the scaled integer type FIXED-POINT as currently employed in programming languages. The evolution of user expectations for arithmetic of type REAL implicitly contain two other requirements for which fixed-slash is quite deficient. First, a much more substantial range for representation of values of type REAL is required than that provided by fixed-slash representation, and secondly, the gaps between representable values must serve to preserve the growth of relative error in extended computations of REAL type. These latter two goals lead us quite naturally to the concept of floating-slash [2,3] number systems.

Let the binary floating-slash system denote the set of fractions KL given for any $N \geq 2$ by

$$KL=KL(N)=\{(0,q)\,|\,1\leq q\leq 2^{N-1}-1\}$$
$$\bigcup\left\{(p,q)\,\middle|\,|p|\neq 0, q\geq 1, \lfloor \log_2|p|\rfloor+\lfloor \log_2 q\rfloor \leq N-2\right\}. \qquad (4)$$

The specification of KL is intended to include those fractions having the number of bits in the numerator plus the number of bits in the denominator sum to at most N. Note that the denominator must always have a leading bit of unity (for all finite fractions), and this fixed bit may implicitly be "attached" to the slash, gaining a bit of representation in binary floating-slash. A computer word format for the set KL will then also have to include a field of $\lceil \log_2(N-1)\rceil$ bits to specify the implicit slash position between the concatenated numerator and denominator bit sequences as well as a sign bit.

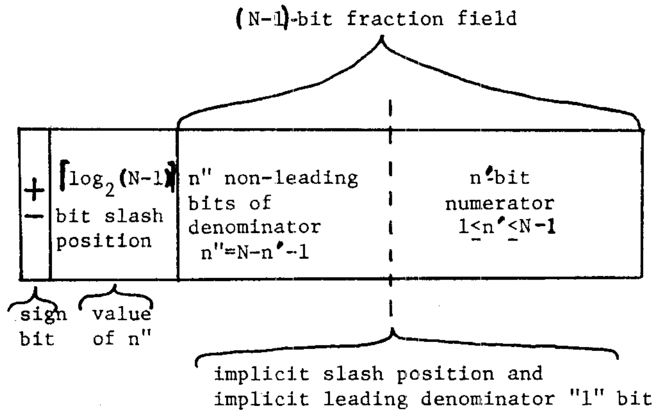

(N-1)-bit fraction field
+
−
⌈log2(N-1)⌉ bit slash position
n" non-leading bits of denominator n"=N-n'-1
n'-bit numerator 1≤n'≤N-1
sign bit
value of n"
implicit slash position and implicit leading denominator "1" bit

Figure 2: Binary Floating-Slash Representation

As in the fixed-slash case we say the representation is normalized if at least one of the low order bits of the numerator and denominator is unity, and is reduced if the gcd is unity (the low-order denominator bit will be the implicit bit if the fraction is an integer in reduced form).

The following important consequences should be noted:

Observation 1. [Exact Inverses]: For every fraction $(p,q) \in KL$ the additive inverse $(-p,q)$ is in KL, and for every non-zero fraction $(p,q) \in KL$, the multiplicative inverse (q,p) is in KL.

Comment: Computation of the additive inverse simply requires complementing the sign bit. Computation of the multiplicative inverse requires a search for the leading numerator bit position.

Observation 2. [Integer Compatibility]: If the slash position field contains the value zero, then the whole N-1 bit fraction field contains an integral valued numerator with implicit denominator unity. Thus binary floating-slash representation of reduced fractions is identical to standard sign-magnitude binary integer representation for all integers i with $|i|\leq 2^{N-1}-1$.

Comment: It is this desired integer compatibility property that dictates our choice for denominator left adjusted and numerator right adjusted fields in the fraction. The implicit denominator leading bit is significant here also.

Observation 3. [Numerator and Denominator Fetching]: The floating-slash representation of Figure 2 allows efficient numerator and/or denominator fetching by masking (rather than sequential shifting).

Comment: Further considerations lead us to suggest the architectural desirability that the denominator bits be in reverse order, e.g. lowest order bit left adjusted.

Binary floating-slash representation can result in some unallowed bit patterns thus generating a slight bit loss in storage utilization. We now show that this bit loss is negligible.

Theorem 2: Let $|KL(N)|$ denote the number of fractions, reducible or irreducible, in KL(N). Then

$$|KL(N)| = (N - \tfrac{3}{2})\times 2^N + 1 \qquad \text{for } N \geq 2. \qquad (5)$$

Proof: Every fraction of KL(N) is in precisely one P_i, $0 \leq i \leq N-1$, where

$$P_0=\{(0,q)\,|\,1\leq q\leq 2^{N-1}-1\},$$

$$P_i=\{(p,q)\,|\,2^{i-1}\leq|p|\leq 2^i-1, 1\leq q\leq 2^{N-i}-1\} \text{ for } 1\leq i\leq N-1.$$

Then $|P_i|=2\times 2^{i-1}\times(2^{N-i}-1)=2^N-2^i$, so for $N \geq 2$,

$$\sum_{i=1}^{N-1} |P_i|=(N-1)2^N - \sum_{i=1}^{N-1} 2^i$$
$$= (N-2)2^N+2,$$

and since $|P_0| = \frac{1}{2}(2^N) - 1$, equation(5) is verified. |

Corollary 2.1: The bit loss in representing members of the binary floating-slash system KL(N) by the format of Figure 2 is less than 1 bit.

Proof: The format of Figure 2 utilizes $N+\lceil\log_2(N-1)\rceil$ bits and represents the members of $KL(N)$, so

$$\text{Bit loss}=N+\lceil\log_2(N-1)\rceil - \log_2((N-\tfrac{3}{2})2^N+1)$$

$$\leq\lceil\log_2(N-1)\rceil - \log_2(N-\tfrac{3}{2})$$

$$\leq 1+\log_2(N-2) - \log_2(N-\tfrac{3}{2}) < 1. \quad |$$

Binary floating-slash representation also is redundant due to representation of reducible as well as irreducible fractions, but further considerations [4,5] show the resultant bit loss is of the order $-\log_2(6/\pi^2)= .718...$ bits as in fixed-slash. Thus the total bit loss in binary floating-slash representation is approximately between .7 and 1.7 bits, which is comparable (and often superior) to the bit lost in normalized binary floating-point representation that is traditionally ignored.

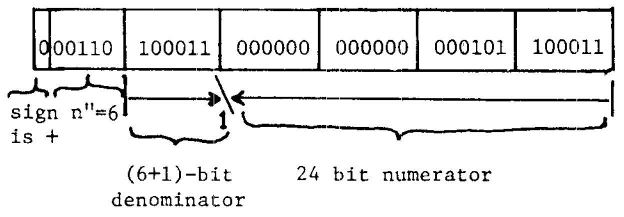
Binary floating-slash representation is more involved than fixed-slash and some examples will aid the development. Convenience of architectural design suggests the desirability that the sign and slash position field fit together in a particular machine "byte", and that $\log_2(N-1)$ be either equal to or just slightly less than an integer (to minimize bit loss). In the following examples, the limiting value $-\log_2(6/\pi^2)$ is employed to account for the bit loss due to redundancy, e.g. to account for the reducible fractions, in determining the total bit loss of each system.

Example I. [31 bit binary floating-slash]. Assume a 6 bit byte and a 36 bit (6 byte) word.

Fraction field:	30 bits (5 bytes), which is sufficient to represent 31 bit binary floating-slash numbers because of the implicit leading denominator bit.
Sign/slash field:	6 bits (1 byte). 1 bit for sign and 5 bits to represent the slash position value n", where $0\leq n''\leq 29$ are the allowed values for n".
Bit loss:	0.835...bits.

Numeric magnitude range: $[\frac{1}{2^{30}-1}, 2^{30}-1] \approx [10^{-9}, 10^{9}]$.

Consider the intrepretation of the following floating-slash number.



numerator = 101100011_2 = 355,

denominator = 1110001_2 = 113,

fraction = $\frac{355}{113}$ = 3.1415929... .

Note that the contents of the slash position field is the number 6, indicating the left most 6 bits of the fraction field are the successive low order bits of the denominator. Reversing these six bits and adding the implicit leading bit yields the denominator 1110001_2 = 113, and the remaining 24 bit numerator field is seen to contain the numerator 101100011_2 = 355. The resulting value $\frac{355}{113}$ = 3.1415929... is one of the better "convergents" to π = 3.1415926..., agreeing to seven decimal places although the numerator and denominator together employ only 16 bits.

Example II. [121 bit binary floating-slash]. Assume an 8 bit byte and a 128 bit 6 byte) word.

Fraction field:	120 bits (5 bytes), which is sufficient to represent 121 bit binary floating-slash numbers because of the implicit leading denominator bit.
Sign/slash field:	8 bits (1 byte). 1 bit for sign and 7 bits to represent the slash position value n", where $0\leq n''\leq 119$ are the allowed values for n".
Bit loss:	0.817... bits.

Numerical magnitude range: $[\frac{1}{2^{120}-1}, 2^{120}-1] \approx [10^{-36}, 10^{36}]$

Example III. [41 bit binary floating-slash]: Assume an 8 bit byte and a 48 bit (6 byte) word.

Fraction field:	40 bits (5 bytes), which is sufficient to represent 41 bit binary floating-slash numbers.
Sign/slash field:	8 bits (1 byte). 1 bit for sign and 7 bits to represent the slash position value n", where $0\leq n''\leq 39$ are the allowed values for n".
Bit loss:	2.414... bits.

Numeric magnitude range: $[\frac{1}{2^{40}-1}, 2^{40}-1] \approx [10^{-12}, 10^{12}]$.

Comment: This example has a bit loss of $7-\log_2 40$ = 1.6781 bits in the slash position specification field. The unnecessary extra slash field bit is a consequence of the desire to allow smaller word size than that of Example II while maintaining the 8 bit byte boundary between the sign/slash field and the fraction field.

Example IV. [7 digit hexadecimal floating-slash]. This hexadecimal example is technically outside the scope of the formulation of our binary floating-slash systems. It does not employ an implicit leading bit except at the extremes of numeric range, where either the numerator or denominator is assumed to be unity. This example is included since it gives a particularly compact representation for an architecture that is currently widely in use.

Assume an 8 bit byte and a 32 bit word, where a byte is assumed to hold two hexadecimal digits.

Fraction field:	28 bits (7 hex digits).
Sign/slash field:	4 bits. 1 bit for sign, and 3 bits to represent the slash position value n", $0 \leq n" \leq 7$.

Numeric magnitude range: $[\frac{1}{16^7-1}, 16^7-1] \approx [10^8, 10^{-8}]$

The fraction $\frac{355}{113}$ of Example 1 would here have the eight hexadecimal digit encoding 21700163, where then denominator = $71_{16} = 1110001_2 = 113$, and numerator = $163_{16} = 101100011_2 = 355$. Note: We have taken the denominator to have its hexadecimal digits left adjusted, low order digits to the left.

Quite often the specifications for floating-point type REAL require exponent ranges larger than $[10^{-300}, 10^{300}]$ with only moderate precision, such as 32 to 64 bits. To achieve greater range we may extend the N-bit binary floating-slash system described in Figure 2 by allowing the slash position field to hold values greater than (N-1) and less than zero, with the following interpretation:

$n" \geq N-1$:	the numerator is implicitly unity and the denominator is the scaled integer $1.X \times 2^{n"}$ where X is the N-1 bit binary fraction taken from the fraction field (reversing bit order),
$n" \leq -1$:	the denominator is implicitly unity and the numerator is the scaled integer $1.X \times 2^{N-2-n"}$ where X is the N-1 bit binary fraction taken from the fraction field.

Comments: (1) The interpretation in both cases is that the numerator (denominator) is an integer of magnitude at least 2^{N-1}, with the N leading bits composed of an implicit leading unit bit followed by the N-1 bits from the fraction field in left-to-right (right-to-left) order with an implicit denominator (numerator) of unity.

(2) The extended range N-bit binary floating-slash system as described represents precisely the same numbers as the standard N-bit binary floating-slash system over the range $[\frac{1}{2^N-1}, 2^N-1]$, with values in $[2^N, \infty]$ identical to those obtained in an N bit "radix fraction" floating-point system in the region $[2^N, \infty]$, and with values in $[0, \frac{1}{2^N}]$ the inverses of the just described floating-point numbers from $[2^N, \infty]$ (assuming a symmetric range constraint).

Example V. [49 bit extended range binary floating-slash].

Assume an 8 bit byte and a 64 bit (8 byte) word.

Fraction field:	48 bits (8 bytes), which is sufficient to represent 49 bit binary floating-slash numbers.
Sign/slash field:	16 bits (2 bytes). 1 bit for sign and 15 bits to represent the slash position value n", where $-(2^{14}-N+1) \leq n" \leq 2^{14}-1$ are the allowed values for n".

Numerical magnitude range: $\approx [\frac{1}{2^{2^{14}}}, 2^{2^{14}}]$

$$\approx [10^{-4932}, 10^{4932}]$$

For extended range binary floating-slash the bit loss over the standard floating-slash range is the 0.701 bits due to the reducible fractions represented, and in the extended range the bit loss is comparable to that in a floating-point system (actually better due to the implicit leading bit in floating-slash). Thus we state:

Conclusion: Binary floating-slash and extended range binary floating-slash representation entails a loss of storage efficiency limited to approximately one bit of the word size for any word size.

The results of addition, subtraction, multiplication and division of binary floating-slash numbers are all finite precision fractions whose maximum lengths influence the arithmetic register requirements.

Theorem 3: For any $N \geq 2$,

(i) the result of a multiplication or (nonzero) division of any two fractions of KL(N) is a fraction in KL(2N), and

(ii) the result of an addition or subtraction of any two fractions of KL(N) is a fraction of KL(3N-2).

Proof: If either fraction has value zero, the result is immediate, and the results for negative fractions follow from the results for positive fractions, so we may assume

$$(p,q),(r,s) \in KL(N), \quad p,q,r,s \geq 1.$$

Using the defining equation (4),

$$\lfloor \log_2 pr \rfloor + \lfloor \log_2 qs \rfloor$$
$$\le \lfloor \log_2 p \rfloor + \lfloor \log_2 r \rfloor + \lfloor \log_2 q \rfloor + \lfloor \log_2 s \rfloor + 2$$
$$\le N-2,$$

so $(pr,qs) \in K(2N)$, and similarly $(ps,qr) \in K(2N)$, so (i) follows. To prove (ii) it is sufficient to show that $(ps+qr,qs) \in KL(3N-2)$, and we consider two cases.

Case 1. Assume $1 \le q,s \le 2^{N-2}-1$, and we may then further assume $qr \ge ps$. Utilizing (4)

$$\lfloor \log_2 (ps+qr) \rfloor + \lfloor \log_2 qs \rfloor$$
$$\le +3\lfloor \log_2 r \rfloor + \lfloor \log_2 s \rfloor + 2\lfloor \log_2 q \rfloor$$
$$\le +3N-5.$$

Thus in this case $(ps+qr,qs) \in KL(3N-3)$.

Case 2. Assume $q \ge s$, and $2^{N-2} \le q \le 2^{N-1}-1$. In this case $p=1$, so $ps+qr \le q(r+1) \le 2^{N-1}r+2^{N-1}-r-1 \le 2^{N-1}r$

and

$$\lfloor \log_2 (ps+qr) \rfloor + \lfloor \log_2 qs \rfloor$$
$$\le N-1+\lfloor \log_2 r \rfloor + \lfloor \log_2 q \rfloor + \lfloor \log_2 s \rfloor + 1$$
$$\le 3N-4.$$

Thus in this case $(ps+qr,qs) \in KL(3N-2)$, proving (ii). |

As an application of Theorem 3 note that an arithmetic operation performed on two 41 bit binary floating-slash numbers as represented in the 6 byte (48 bit) word in Example III will always yield a result exactly representable as a 121 bit binary floating-slash number in the 16 byte (128 bit) word of Example I.

In designing a floating-slash arithmetic unit it is desirable to first generate the exact arithmetic result in intermediate registers and then perform the rounding. It is evident from Theorem 3 and the discussion of rounding in our companion paper [1] that the arithmetic unit requirements for N-bit floating-slash arithmetic require only N-bit integer multiply and 2N-bit integer addition and subtraction to effect both the four standard arithmetic operations (+, −, ×, ÷) and the rounding needed to map the result back to an approximate N-bit floating-slash number. Note that the arithmetic unit requirements for extended range binary floating-slash are considerably more involved and will not be discussed here.

III. Rounding and Adaptive Precision

If the irreducible fractions of a fixed-slash or floating-slash number system are arranged in increasing order, the size of the gaps between successive fractions is quite variable. Building on the classical theory of Farey fractions [7,Ch. 2] and continued fractions [7,Ch. 10], an extensive theory is developed [4,5] that provides considerable insight into limited precision fraction number systems specifically and finite precision arithmetic generally. Only a brief survey of this theory sufficient to indicate the rounding behavior in fixed- and floating-slash systems will be presented here.

The fractions $\frac{p}{q}, \frac{r}{s}$ are termed adjacent if $|qr-ps| = 1$, and the interval between the quotients of the adjacent fractions $\frac{p}{q}$ and $\frac{r}{s}$, is the gap between $\frac{p}{q}$ and $\frac{r}{s}$. Since $\frac{r}{s} - \frac{p}{q} = \frac{qr-ps}{qs}$, the adjacent fractions $\frac{p}{q}, \frac{r}{s}$ have a gapsize of $\frac{1}{qs}$ and are in a sense "optimally close" distinct fractions relative to their precision, that is, relative to the size of their denominators. The fraction $\frac{r}{s}$ is simpler than the fraction $\frac{p}{q}$ if $|r| \le |p|$, $s \le q$, where at least one of these inequalities is strict. The following results on adjacent fractions should be noted.

Lemma 4: If $\frac{p}{q}, \frac{r}{s}$ are adjacent fractions, then both $\frac{p}{q}$ and $\frac{r}{s}$ are irreducible.

Proof: The g.c.d.(p,q) and g.c.d.(r,s) both divide $|qr-ps| = 1$. |

Lemma 5: If $\frac{p}{q}, \frac{r}{s}$ are adjacent fractions other than the pair $\frac{0}{1}, \frac{1}{0}$, then one of them is simpler than the other.

Proof: If $p = r$ the result is immediate. For $p \ne r$ we may assume $r \ge p+1 \ge 1$. If also $q \ge s+1$, then $qr - ps \ge p + s + 1$, so $p = s = 0$, and $q = r = 1$. Otherwise $q \le s$, so then $\frac{p}{q}$ is simpler than $\frac{r}{s}$. |

Lemma 6: Assume the quotient $\frac{t}{u}$ is in the gap $(\frac{p}{q}, \frac{r}{s})$ between the adjacent fractions $\frac{p}{q}$ and $\frac{r}{s}$. Then $t \ge p + r$, $u \ge q + s$.

Proof: We may assume $\frac{p}{q} < \frac{t}{u} < \frac{r}{s}$. Then

$$\frac{qr-ps}{qs} = \frac{ur-ts}{us} + \frac{qt-up}{qu},$$

$$\frac{1}{qs} \ge \frac{1}{us} + \frac{1}{qu},$$

so $u \ge q + s$. Since by inverting the fractions, $\frac{s}{r} < \frac{u}{t} < \frac{q}{p}$, the above then yields $t \ge p + r$. |

Lemma 7: For $\frac{p}{q}$ adjacent to $\frac{r}{s}$ and $\frac{t}{u}$ adjacent to $\frac{v}{w}$, the gaps $(\frac{p}{q}, \frac{r}{s})$ and $(\frac{t}{u}, \frac{v}{w})$ are either nested or disjoint (except for end points).

Proof: If not then it may be assumed that $0 \le \frac{p}{q} < \frac{t}{u} < \frac{r}{s} < \frac{v}{w}$. Then by Lemma 6 $t \geq p + r$, $u \geq q + s$, and also $r \geq t + v$, $s \geq u + w$, so then $p = v = 0$, $q = w = 0$, a contradiction. |

These adjacency lemmas thus state that adjacent fractions are irreducible, comparable in the "simpler than" ordering, and simpler than any fraction falling in the gap between them, and furthermore, that two pairs of adjacent fractions have either nested or disjoint gaps . Lemma 6 states that no fraction simpler than $\frac{p + r}{q + s}$ lies in the gap between the adjacent fractions $\frac{p}{q}$ and $\frac{r}{s}$. In general the fraction $\frac{p + r}{q + s}$ is termed the medians of the fractions $\frac{p}{q}$ and $\frac{r}{s}$ and is defined for any pair of fractions. For the case of adjacent fractions, the mediant has many properties.

Theorem 8: The mediant $\frac{p + r}{q + s}$ of the adjacent fractions $\frac{p}{q} < \frac{r}{s}$ satisfies:

(i) Irreducibility: $\frac{p + r}{q + s}$ is irreducible,

(ii) Quotient ordering: $\frac{p}{q} < \frac{p + r}{q + s} < \frac{r}{s}$,

(iii) Precision ordering: $\frac{p}{q}$ and $\frac{r}{s}$ or both simpler than $\frac{p + r}{q + s}$,

(iv) Adjacency: $\frac{p + r}{q + s}$ is adjacent to $\frac{p}{q}$ and $\frac{r}{s}$,

(v) Simplicity: $\frac{p}{q} < \frac{t}{u} < \frac{r}{s}$ implies $\frac{p + r}{q + s}$ is simpler than or identical to $\frac{t}{u}$.

Proof: Property (iii) is immediate. Note then that

$$\frac{p + r}{q + s} - \frac{p}{q} = \frac{pq + rq - pq - ps}{(q + s)q} = \frac{1}{(q + s)q} > 0,$$

establishing both that $\frac{p}{q} < \frac{p + r}{q + s}$ and that $\frac{p}{q}$ and $\frac{p + r}{q + s}$ are adjacent. Also $\frac{r}{s} - \frac{p + r}{q + s} =$ $\frac{rq + rs - ps - rs}{s(q + s)} = \frac{1}{s(q + s)} > 0$, so that $\frac{p + r}{q + s} < \frac{r}{s}$ and they are likewise adjacent fractions. This proves (iv) and (ii), and property (v) then follows from Lemma 6. Property (i) follows from property (v), completing the theorem. |

The notion of mediant is seen to be fundamental in that every finite non-zero irreducible fraction, $\frac{p}{q}$, is the mediant of a unique pair of adjacent fractions, the parents of $\frac{p}{q}$. Furthermore, the proof of the following theorem indicates a canonical procedure for determining the parents and all "ancestors" of $\frac{p}{q}$.

Theorem 9: Every finite non-zero irrreducible fraction is the mediant of precisely one pair of adjacent fractions.

Proof: It is sufficient to consider $\frac{0}{1} < \frac{p}{q} < \frac{1}{0}$, where $\frac{p}{q}$ is irreducible. Form the sequence $F_i(\frac{p}{q})$ as follows for $i = 0,1,2,\ldots, N(p,q)$.

$$F_0(\frac{p}{q}) = \frac{0}{1}, \frac{1}{0} ,$$

$$F_1(\frac{p}{q}) = \frac{0}{1}, \frac{1}{1}, \frac{1}{0} ,$$

and $F_i(\frac{p}{q})$, $i \geq 2$, is formed recursively from $F_{i-1}(\frac{p}{q})$ by inserting the mediant in the gap of $F_{i-1}(\frac{p}{q})$ that contains the quotient of $\frac{p}{q}$. If the mediant inserted to form $F_i(\frac{p}{q})$ is $\frac{p}{q}$, the process terminates with $N(p,q) = i$, otherwise it continues. Note from Theorem 8(iv) that each F_i is a sequence of consecutive adjacent fractions, so before termination $\frac{p}{q}$ always falls in a unique gap between adjacent fractions. By Theorem 8(iii) this procedure must terminate after $N(p,q) \leq \max \{p,q\}$ insertions, determining a pair $\frac{r}{s}$, $\frac{t}{u}$ for which $\frac{p}{q}$ is the mediant. It then follows from Lemma 7 and Theorem 8 that this pair of adjacent fractions, $\frac{r}{s}$,$\frac{t}{u}$, is unique. |

These results allow us to prove an important alternative characterization of adjacent fractions.

Theorem 10: Let $\frac{p}{q} < \frac{r}{s}$ be irreducible fractions. Then if no fraction simpler than at least one of $\frac{p}{q}, \frac{r}{s}$ lies in the open interval $(\frac{p}{q}, \frac{r}{s})$, they are adjacent.

Proof: Assume $\frac{p}{q} < \frac{r}{s}$ irreducible with no fraction simpler than at least one of $\frac{p}{q}$, $\frac{r}{s}$ in the interval $(\frac{p}{q}, \frac{r}{s})$. Let $\frac{p''}{q''}$ be the larger parent, $\frac{p}{q} < \frac{p''}{q''}$, of $\frac{p}{q}$, and $\frac{r'}{s'}$ the smaller parent, $\frac{r'}{s'} < \frac{r}{s}$, of $\frac{r}{s}$. The parents are simpler fractions and cannot fall in the open interval $(\frac{p}{q}, \frac{r}{s})$ by assumption, so $\frac{r'}{s'} \leq \frac{p}{q} < \frac{r}{s} \leq \frac{p''}{q''}$. Now $\frac{r'}{s'}$, $\frac{r}{s}$ are adjacent and $\frac{p}{q}$, $\frac{p''}{q''}$ are adjacent by Theorem 8(iv), and both inequalities cannot be strict by Lemma 7. So one of

$\frac{p}{q}, \frac{r}{s}$ is a parent of the other, proving the theorem. |

A <u>Farey chain</u> $\frac{p_1}{q_1} < \frac{p_2}{q_2} < \ldots < \frac{p_n}{q_n}$ is an increasing sequence of consecutive adjacent fractions, i.e. $q_i p_{i+1} - q_{i+1} p_i = 1$ for $1 \le i \le n-1$. The <u>gaps</u> of the Farey chain are the gaps between consecutive fractions of the chain. The following are immediate from Theorem 10 and the definitions.

<u>Corollary 10.1</u>: The monotonically increasing sequence of irreducible fractions of KX(2N) from a Farey chain for every $N \ge 1$.

<u>Corollary 10.2</u>: The monotonically increasing sequence of irreducible fractions of KL(N) form a Farey chain for every $N \ge 2$.

For the Farey chain $\frac{p_1}{q_1} < \frac{p_2}{q_2} < \ldots < \frac{p_n}{q_n}$, it is evident that the maximum and minimum gapsizes are given by the maximum and minimum of

$\{\frac{1}{q_i q_{i+1}}, 1 \le i \le n-1\}$. If $\frac{0}{1} \le \frac{p}{q} < \frac{r}{s} \le \frac{1}{1}$ are consecutive adjacent fractions of the Farey chain for KX(2N), then $q+s \ge 2^N$ since $\frac{p+r}{q+s} \notin$ KX(2N). From these observations the following is immediate.

<u>Lemma 11</u>: For the interval [0,1], the size of the intervals between successive representable values of KX(2N) obtains a maximum of $1/(2^N-1)$ and a minimum of $1/[(2^N-1)(2^N-2)]$.

From Lemma 11 we conclude that the precision of the binary fixed-slash system KX(2N) varies approximately from what we normally term N-bits to 2N-bits over the interval [0,1], or loosely from single to double precision. To measure the "average precision" it is desirable to specify the natural rounding for limited precision rational systems.

Let K denote the Farey chain $\frac{p_1}{q_1} < \frac{p_2}{q_2} < \cdots < \frac{p_n}{p_n}$, and let $[\frac{p_1}{q_1}, \frac{p_n}{q_n}]$ denote the closed interval of real numbers between $\frac{p_1}{q_1}$ and $\frac{p_n}{q_n}$. Then <u>mediant rounding</u> is the mapping $\Phi: [\frac{p_1}{q_1}, \frac{p_n}{q_n}] \to K$ defined by

$$\Phi(\frac{p_i}{q_i}) = \frac{p_i}{q_i} \quad \text{for } 1 \le i \le n,$$

and for $1 \le i \le n-1$,

$$\Phi(x) = \begin{cases} \frac{p_i}{q_i} & \text{if } \frac{p_i}{q_i} < x < \frac{p_i+p_{i+1}}{q_i+q_{i+1}}, \\ \frac{p_{i+1}}{q_{i+1}} & \text{if } \frac{p_i+p_{i+1}}{q_i+q_{i+1}} < x < \frac{p_{i+1}}{q_{i+1}}, \\ \text{the simpler of } \frac{p_i}{q_i}, \frac{p_{i+1}}{q_{i+1}} & \text{for } x = \frac{p_i+p_{i+1}}{q_i+q_{i+1}}. \end{cases} \qquad (6)$$

Now we show that although the gapsizes in KX(2N) vary so that the accuracy essentially varies from N to 2N bits, the average rounding error on [0,1] is of the order $\ell n N$ bits, and thus is acceptably close to the full 2N bit precision level. First note:

<u>Lemma 12</u>: Let $\Phi:[0,1] \to$ KX(2N). For $0 \le x \le 1$, let $\Phi(x) = \frac{p}{q}$ where $\frac{p}{q}$ is irreducible. Then

$$|x - \Phi(x)| \le \frac{1}{q2^N}.$$

<u>Proof</u>: Note that the mediant of the adjacent fractions $\frac{p}{q}, \frac{r}{s}$ of the Farey chain for KX(2N) has $q+s \ge 2^N$, and the result follows. |

<u>Theorem 13</u>: Let $\Phi:[0,1] \to$ KX(2N) denote mediant rounding of the interval [0,1] into the fixed-slash system KX(2N). For the random variable X chosen uniformly on [0,1], the expectation of $|X-\Phi(X)|$ satisfies:

$$E(|X-\Phi(X)|) \le \frac{1}{2^{2N}} + \frac{N\ell n2}{2^{2N}}. \qquad (7)$$

<u>Proof</u>: For X chosen uniformly on [0,1], let $\Phi(X) = \frac{p}{q}$. Note from Lemma 12 that the interval rounding to $\frac{p}{q}$ by mediant rounding extends no more than $1/q2^N$ on either side of $\frac{p}{q}$, so

$$\text{Prob}\{\Phi(X) = \frac{p}{q}\} \le 2/q2^N,$$

and the average rounding error rounding to $\frac{p}{q}$ is no greater than $(1/2) \times (1/q2^N)$.

Therefore

$$E(|X-\Phi(X)|) \le \frac{1}{2^{2N}} + \frac{1}{2^{2N}} \sum_{\substack{1 \le p < q \le 2^{N-1} \\ \frac{p}{q} \text{ irreducible}}} \frac{1}{q^2}$$

$$\le \frac{1}{2^{2N}} + \frac{1}{2^{2N}} \sum_{1 \le q \le 2^{N-1}} \frac{q-1}{q}$$

$$\le \frac{1}{2^{2N}} + \frac{N\ell n2}{2^{2N}}. \;|$$

Similar results on the maximum, minimum and average <u>relative</u> rounding error in floating-slash systems using a log uniform distribution for average case analysis can be obtained [4]. The significance of Lemma 12 is that the greater rounding deviations $|x-\Phi(x)|$ are associated with roundings to simpler fractions $\frac{p}{q}$, that is, where q is a smaller integer. The resulting variable single to double precision feature is intriquing and is investigated in [4].

References

[1] Kornerup, P. and Matula, D. W., A Feasibility Analysis of Fixed-Slash Rational Arithmetic, This proceedings, 1978.

[2] Matula, D. W., "Number Theoretic Foundations of Finite Precision Arithmetic," in Applications of Number Theory to Numerical Analysis, W. Zarenba, ed., Academic Press, New York, 1972, 479-489.

[3] Matula, D. W., "Fixed-Slash and Floating-Slash Rational Arithmetic," Proceedings of the 3rd IEEE Symposium on Computer Arithmetic, Dallas, 1975, pp. 90-91.

[4] Matula, D. W. and Kornerup, P, "Finite Precision Rational Arithmetic, Part 1: The Number Systems," in preparation.

[5] Kornerup, P and Matula, D. W., "Finite Precision Rational Arithmetic, Part II: The Arithmetic," in preparation.

[6] Knuth, D. E., "The Art of Computer Programming," Vol. 2/Seminumerical Algorithms, Addison-Wesley 1969.

[7] Hardy, G. H. and Wright, E. M., "An Introduction to the Theory of Numbers," Clarendon Press, Oxford 1954.