Appendix A

Number Theory Tools for Floating-Point Arithmetic

A.1 Continued Fractions

Continued fractions make it possible to build very good (indeed, the best possible, in a sense that will be made explicit by Theorems A.1 and A.2) rational approximations to real numbers. As such, they naturally appear in many problems of number theory, discrete mathematics, and computer science. Since floating-point numbers are rational approximations to real numbers, it is not surprising that continued fractions play a role in some areas of floating-point arithmetic. A typical example is Table 10.1: that table gives, among other results, the floating-point numbers that are nearest to an integer multiple of $\pi/2$, which is a typical continued-fraction problem. It allows one to design accurate range-reduction algorithms for evaluating trigonometric functions. Another example is the continued-fraction-based proof of the algorithm for performing correctly rounded multiplication by an arbitrary-precision constant, presented in Section 4.11.

Excellent surveys can be found in [237, 565, 334]. Here, we will just present some general definitions, as well as the few results that are needed in this book, especially in Chapters 4 and 10.

Let α be a real number. From α, consider the two sequences (a_i) and (r_i) defined by

$$\begin{cases} r_0 &= \alpha, \\ a_i &= \lfloor r_i \rfloor, \\ r_{i+1} &= \dfrac{1}{r_i - a_i}, \end{cases} \tag{A.1}$$

J.-M. Muller et al., *Handbook of Floating-Point Arithmetic*,
https://doi.org/10.1007/978-3-319-76526-6

where "$\lfloor . \rfloor$" is the usual floor function. Note that the a_i's are integers and that the r_i's are real numbers.

If α is an irrational number, then these sequences are defined for any $i \geq 0$ (i.e., r_i is never equal to a_i), and the rational number

$$\frac{P_i}{Q_i} = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cfrac{1}{\ddots + \cfrac{1}{a_i}}}}}$$

is called the i-th *convergent* of α. The a_i's constitute the *continued fraction expansion* of α. We write

$$\alpha = a_0 + \cfrac{1}{a_1 + \cfrac{1}{a_2 + \cfrac{1}{a_3 + \cdots}}}$$

or, to save space,

$$\alpha = [a_0; a_1, a_2, a_3, \ldots].$$

If α is rational, then these sequences terminate for some k, and $P_k/Q_k = \alpha$ exactly. The P_i's and the Q_i's can be deduced from the a_i's using the following recurrences:

$$\begin{cases} P_0 &= a_0, \\ Q_0 &= 1, \\ P_1 &= a_1 a_0 + 1, \\ Q_1 &= a_1, \\ P_k &= P_{k-1} a_k + P_{k-2} \quad \text{for } k \geq 2, \\ Q_k &= Q_{k-1} a_k + Q_{k-2} \quad \text{for } k \geq 2. \end{cases}$$

Note that these recurrences give irreducible fractions P_i/Q_i: the values P_i and Q_i that are deduced from them satisfy $\gcd(P_i, Q_i) = 1$.

The major interest in the continued fractions lies in the fact that P_i/Q_i is the best rational approximation to α among all rational numbers of denominator less than or equal to Q_i. More precisely, we have the following two results [237].

Theorem A.1. *Let $(P_j/Q_j)_{j\geq 0}$ be the convergents of α. If a rational number P/Q is a better approximation to α than P_k/Q_k (namely, if $|P/Q - \alpha| < |P_k/Q_k - \alpha|$), then $Q > Q_k$.*

Theorem A.2. *Let $(P_j/Q_j)_{j\geq 0}$ be the convergents of α. If Q_{k+1} exists, then for any $(P, Q) \in \mathbb{Z} \times \mathbb{N}^*$, with $Q < Q_{k+1}$, we have*

$$|P - \alpha Q| \geq |P_k - \alpha Q_k|.$$

If Q_{k+1} does not exist (which implies that α is rational), then the previous inequality holds for any $(P, Q) \in \mathbb{Z} \times \mathbb{N}^*$.

Interestingly enough, a kind of converse result exists: if a rational approximation to some number α is extremely good, then it must be a convergent of its continued fraction expansion.

Theorem A.3 ([237]). *Let P, Q be integers, $Q \neq 0$. If*

$$\left|\frac{P}{Q} - \alpha\right| < \frac{1}{2Q^2},$$

then P/Q is a convergent of α.

An example of continued fraction expansion of an irrational number is

$$e = \exp(1) = 2 + \cfrac{1}{1 + \cfrac{1}{2 + \cfrac{1}{1 + \cfrac{1}{1 + \cfrac{1}{4 + \cfrac{1}{\ddots}}}}}} = [2; 1, 2, 1, 1, 4, \ldots],$$

which gives the following rational approximations to e:

$$\frac{P_0}{Q_0} = 2, \quad \frac{P_1}{Q_1} = 3, \quad \frac{P_2}{Q_2} = \frac{8}{3}, \quad \frac{P_3}{Q_3} = \frac{11}{4}, \quad \frac{P_4}{Q_4} = \frac{19}{7}, \quad \frac{P_5}{Q_5} = \frac{87}{32}.$$

Other examples are

$$\pi = 3 + \cfrac{1}{7 + \cfrac{1}{15 + \cfrac{1}{1 + \cfrac{1}{292 + \cfrac{1}{1 + \cfrac{1}{\ddots}}}}}} = [3; 7, 15, 1, 292, 1, \ldots]$$

and

$$\sqrt{2} = 1 + \cfrac{1}{2 + \cfrac{1}{2 + \cfrac{1}{2 + \cfrac{1}{2 + \cfrac{1}{\ddots}}}}} = [1; 2, 2, 2, 2, \ldots] = [1; \overline{2}].$$

A.2 Euclidean Lattices

A Euclidean lattice is a set of points that are regularly spaced in the Euclidean space $\mathbb{R}^n$ (see Definition A.1). It is a discrete algebraic object that is encountered in several domains of various sciences, including mathematics, computer science, electrical engineering, and chemistry. It is a rich and powerful modeling tool, thanks to the deep and numerous theoretical results, algorithms, and implementations available (see [90, 112, 226, 397, 557, 599] for example).

Applications of Euclidean lattices to floating-point arithmetic include the calculation of polynomial approximations with coefficients that are floating-point numbers (or double-word numbers) [68, 97] and the search for hardest-to-round points of elementary functions (see Chapter 10 and [571]).

Let $x = (x_1, \ldots, x_n) \in \mathbb{R}^n$. We set

$$\|x\|_2 = (x|x)^{1/2} = \left(x_1^2 + \cdots + x_n^2\right)^{1/2} \text{ and } \|x\|_\infty = \max_{1 \le i \le n} |x_i|.$$

Definition A.1. *Let L be a nonempty subset of $\mathbb{R}^n$. The set L is a (Euclidean) lattice if there exists a set of $\mathbb{R}$-linearly independent vectors $b_1, \ldots, b_d$ such that*

$$L = \mathbb{Z} \cdot b_1 \oplus \cdots \oplus \mathbb{Z} \cdot b_d = \left\{ \sum_{i \le d} x_i \cdot b_i, x_i \in \mathbb{Z} \right\}.$$

The family $(b_1, \ldots, b_d)$ is a basis of the lattice L, and d is called the rank of the lattice L.

For example, the set $\mathbb{Z}^n$ and all of its additive subgroups are lattices. These lattices play a central role in computer science since they can be represented exactly. We say that a lattice L is integer (resp. rational) when $L \subseteq \mathbb{Z}^n$ (resp. $\mathbb{Q}^n$). An integer lattice of rank 2 is given in Figure A.1, as well as one of its bases.

A lattice is often given by one of its bases (in practice, a matrix whose rows or columns are the basis vectors). Unfortunately, as soon as the rank of the lattice is greater than 1, there are infinitely many such representations for any given lattice. In Figure A.2, we give another basis of the lattice of Figure A.1.

Proposition A.1. *If $(e_1, \ldots, e_k)$ and $(f_1, \ldots, f_j)$ are two families of $\mathbb{R}$-linearly independent (column) vectors that generate the same lattice, then $k = j$ (this is the rank of the lattice), and there exists a $(k \times k)$ matrix M with integer coefficients and determinant equal to ± 1 such that $(e_i) = (f_i) \cdot M$.*

Among the infinitely many bases of a specified lattice (if $k \ge 2$), some are more interesting than others. One can define various notions of what a

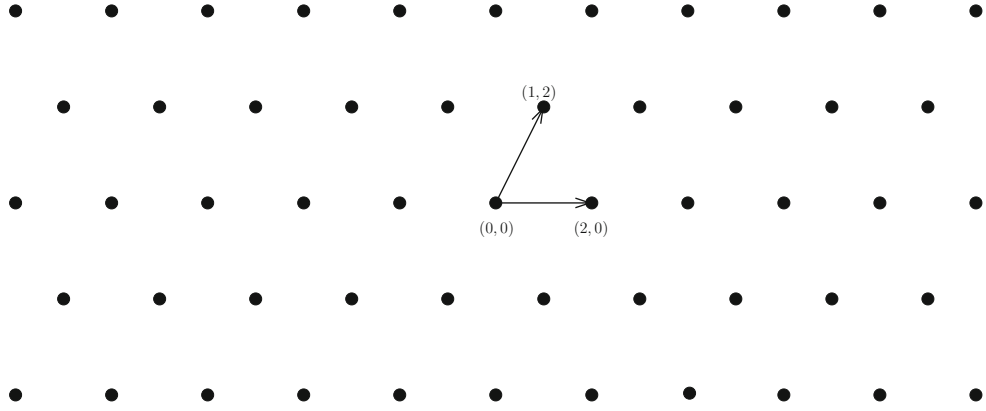

Figure A.1: The lattice $\mathbb{Z}(2,0) \oplus \mathbb{Z}(1,2)$.

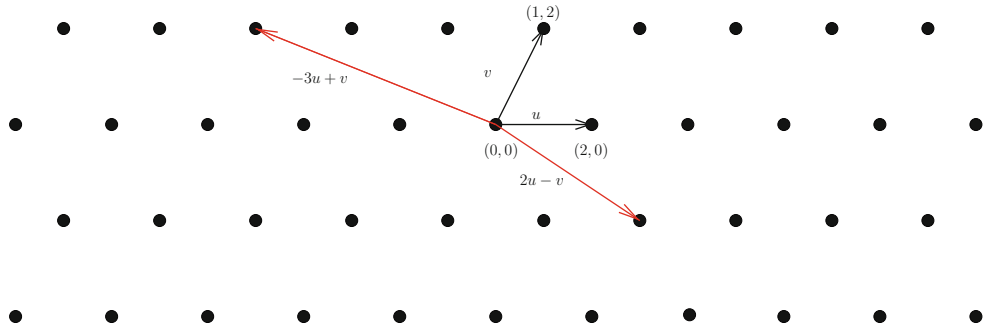

Figure A.2: Two bases of the lattice $\mathbb{Z}(2,0) \oplus \mathbb{Z}(1,2)$.

"good" basis is, but most of the time, it is required to consist of somewhat short vectors.

The two most famous computational problems related to lattices are the shortest and closest vector problems (SVP and CVP). Since a lattice is discrete, it contains a vector of smallest nonzero norm. That norm is denoted by λ and is called the *minimum* of the lattice. Note that the minimum is reached at least twice (a vector and its opposite), and may be reached more times. The discreteness also implies that, given an arbitrary vector of the space, there always exists a lattice vector closest to it (note that there can be several such vectors). We now state the search versions of SVP and CVP.

Problem A.1. Shortest vector problem (SVP). *Given a basis of a lattice* $L \subseteq \mathbb{Q}^n$*, find a shortest nonzero vector of* L*, i.e., a vector of norm* $\lambda(L)$*.*

SVP naturally leads to the following approximation problem, which we call γ*-SVP, where* γ *is a function of the rank only: given a basis of a lattice* $L \subseteq \mathbb{Q}^n$*, find* $b \in L$ *such that*

$$0 < \|b\| \leq \gamma \cdot \lambda(L).$$

Problem A.2. Closest vector problem (CVP). *Given a basis of a lattice $L \subseteq \mathbb{Q}^n$ and a target vector $t \in \mathbb{Q}^n$, find $b \in L$ such that $\|b - t\| = \mathrm{dist}(t, L)$.*

CVP naturally leads to the following approximation problem, which we call γ-CVP, where γ is a function of the rank only: given a basis of a lattice $L \subseteq \mathbb{Q}^n$ and a target vector $t \in \mathbb{Q}^n$, find $b \in L$ such that

$$\|b - t\| \leq \gamma \cdot \mathrm{dist}(t, L).$$

Note that SVP and CVP can be defined with respect to any norm of $\mathbb{R}^n$, and we will write SVP_2 and CVP_2 to explicitly refer to the Euclidean norm. These two computational problems have been studied extensively. We very briefly describe some of the results, and refer to [420] for more details.

Ajtai [4] showed in 1998 that the decisional version of SVP_2 (i.e., given a lattice L and a scalar x, compare $\lambda(L)$ and x) is NP-hard under randomized polynomial reductions. The NP-hardness had been conjectured in the early 1980s by van Emde Boas [612], who proved the result for the infinity norm instead of the Euclidean norm. Khot [335] showed that Ajtai's result still holds for the decisional version of the relaxed problem γ-SVP_2, where γ is an arbitrary constant. Goldreich and Goldwasser [216] proved that, under very plausible complexity theory assumptions, approximating SVP_2 within a factor $\gamma = \sqrt{d/\ln d}$ is not NP-hard, where d is the rank of the lattice. No polynomial-time algorithm is known for approximating SVP_2 within a factor $f(d)$ with f a polynomial in d. On the constructive side, Kannan [325] described an algorithm that solves SVP_2 in time

$$d^{\frac{d(1+o(1))}{2e}} \approx d^{0.184 \cdot d},$$

and in polynomial space (the complexity bound is proved in [233]). Ajtai, Kumar, and Sivakumar [5] gave an algorithm of complexity $2^{\mathcal{O}(d)}$ both in time and space that solves SVP with high probability. Becker et al. [34] described a heuristic variant of that algorithm that runs is time $\approx 2^{0.292 \cdot d}$. Similar results hold for the infinity norm instead of the Euclidean norm.

In 1981, van Emde Boas [612] proved that the decisional version of CVP_2 is NP-hard (see also [419]). On the other hand, Goldreich and Goldwasser [216] showed, under very plausible assumptions, that approximating CVP_2 within a factor $\sqrt{d/\ln d}$ is not NP-hard. Similar results also hold for the infinity norm (see [488]). Unfortunately, no polynomial-time algorithm is known for approximating CVP to a polynomial factor. On the constructive side, Kannan [325] described an algorithm that solves CVP in time

$$d^{\frac{d(1+o(1))}{2}}$$

for the Euclidean norm and

$$d^{d(1+o(1))}$$

for the infinity norm (see [233] for the proofs of the complexity bounds).

If we sufficiently relax the parameter γ, the situation becomes far better. In 1982, Lenstra, Lenstra, and Lovász [382] gave an algorithm that allows one to get relatively short vectors in polynomial time. Their algorithm is now commonly referred to by the acronym LLL.

Theorem A.4 (LLL [382]). *Given an arbitrary basis $(a_1, \ldots, a_d)$ of a lattice $L \subseteq \mathbb{Z}^n$, the LLL algorithm provides a basis $(b_1, \ldots, b_d)$ of L that is made of relatively short vectors. Among others, we have $\|b_1\| \leq 2^{(d-1)/2} \cdot \lambda(L)$. Furthermore, LLL terminates within $\mathcal{O}(d^5 n \ln^3 A)$ bit operations with $A = \max_i \|a_i\|$.*

More precisely, the LLL algorithm computes what is called a δ-LLL-reduced basis, where δ is a fixed parameter which belongs to $(1/4, 1)$ (if δ is omitted, then its value is $3/4$, which is the historical choice). To define what a LLL-reduced basis is, we need to recall the Gram–Schmidt orthogonalization of a basis. Consider a basis $(b_1, \ldots, b_d)$. Its Gram–Schmidt orthogonalization $(b_1^*, \ldots, b_d^*)$ is defined recursively as follows:

- the vector b_1^* is b_1;
- for $i > 1$, we set $b_i^* = b_i - \sum_{j<i} \mu_{i,j} b_j^*$, where $\mu_{i,j} = \frac{\langle b_i, b_j^* \rangle}{\|b_j^*\|^2}$.

Geometrically, the vector b_i^* is the projection of the vector b_i orthogonally to the span of the previous basis vectors $b_1, \ldots, b_{i-1}$. We say that the basis $(b_1, \ldots, b_d)$ is δ-LLL-reduced if the following two conditions are satisfied:

- for any $i > j$, the quantity $\mu_{i,j}$ has magnitude less than or equal to $1/2$. This condition is called the *size-reduction condition*;
- for any i, we have $\delta \|b_i^*\|^2 \leq \|b_{i+1}^*\|^2 + \mu_{i+1,i}^2 \|b_i^*\|^2$. This condition is called Lovász's condition. It means that orthogonally to the first $i-1$ vectors, the $(i+1)$-th vector cannot be arbitrarily small compared to the i-th vector.

LLL-reduced bases have many interesting properties. The most important one is probably that the first basis vector cannot be more than $2^{(d-1)/2}$ times longer than the lattice minimum. The LLL algorithm computes an LLL-reduced basis $(b_1, \ldots, b_d)$ of the lattice spanned by $(a_1, \ldots, a_d)$ by incrementally trying to make the LLL conditions satisfied. It uses an index k, which starts at 2 and eventually reaches $d+1$. At any moment, the first $k-1$ vectors satisfy the LLL conditions, and we are trying to make the first k vectors satisfy the conditions. To make the size-reduction condition satisfied for the k-th vector, one subtracts from it an adequate integer linear combination of the vectors $b_1, \ldots, b_{k-1}$. This is essentially the same process as Babai's nearest plane algorithm, described below. After that, one tests Lovász's condition. If it is satisfied, then the index k can be incremented. If not, the vectors b_k and b_{k-1} are swapped, and the index k is decremented. The correctness of

the LLL algorithm is relatively simple to prove, but the complexity analysis is significantly more involved. We refer the interested reader to [382].

The LLL algorithm has been extensively studied since its invention [324, 544, 576, 463, 465, 457]. Very often in practice, the returned basis is of better quality than the worst-case bound given above and is obtained faster than expected. We refer to [463] for more details about the practical behavior of the LLL algorithm.

Among many important applications of the LLL algorithm, Babai [23] extracted from it a polynomial-time algorithm for solving CVP with an exponentially bounded approximation factor. We present it in Algorithm A.1.

Theorem A.5 (Babai [23])**.** *Given an arbitrary basis $(b_1, \ldots, b_d)$ of a lattice $L \subseteq \mathbb{Z}^n$, and a target vector $t \in \mathbb{Z}^n$, Babai's nearest plane algorithm (Algorithm A.1) finds a vector $b \in L$ such that*

$$\|b - t\|_2 \leq 2^d \cdot \mathrm{dist}_2(t, L).$$

Moreover, it finishes in polynomial time in d, n, $\ln A$, and $\ln \|t\|$, where $A = \max_i \|a_i\|$.

Algorithm A.1 Babai's nearest plane algorithm. The inputs are an LLL-reduced basis $(b_i)_{1 \leq i \leq d}$, its Gram–Schmidt orthogonalization $(b_i^*)_{1 \leq i \leq d}$, and a target vector t. The output is a vector in the lattice spanned by the b_i's that is close to t.

$v \leftarrow t$
for ($j = d$; $j \geq 1$; j--) **do**
  $v \leftarrow v - \left\lfloor \frac{\langle v, b_j^* \rangle}{\langle b_j^*, b_j^* \rangle} \right\rceil b_j$
end for
return $t - v$

Babai's algorithm may also be described with the LLL algorithm directly. This may be seen as a particular case of Kannan's embedding technique [326]. This may be simpler to implement, in particular, if one has access to an implementation of LLL. We give that variant in Algorithm A.2.

Algorithm A.2 Babai's nearest plane algorithm, using LLL. The inputs are an LLL-reduced basis $(b_i)_{1 \leq i \leq d}$ and a target vector t. The output is a vector in the lattice spanned by the b_i's that is close to t.

for ($j = 1$; $j \leq d$; j++) **do**
  $c_j \leftarrow (b_j, 0)$
end for
$B \leftarrow \max_i \|b_i\|$; $c_{d+1} \leftarrow (t, B)$
$(c'_1, \ldots, c'_{d+1}) \leftarrow \mathrm{LLL}(c_1, \ldots, c_{d+1})$
return $c_{d+1} - c'_{d+1}$

Appendix B

Previous Floating-Point Standards

B.1 The IEEE 754-1985 Standard

The binary part of the IEEE 754-2008 standard is an evolution of IEEE 754-1985. Hence a large part of what we have presented in Chapter 3 was already present in IEEE 754-1985. In the following, we therefore mainly focus on the differences.

B.1.1 Formats specified by IEEE 754-1985

Of course, since IEEE 754-1985 focused on binary floating-point arithmetic, there were no decimal formats. The two basic formats were

- single precision: this is the format now called binary32 in IEEE 754-2008 (same parameters and bit encoding);
- double precision: this is the format now called binary64 (same parameters and bit encoding).

There was no 16-bit or 128-bit format.

To each basic format, the IEEE 754-1985 standard associated an *extended format*. The standard recommended an extended format for the widest basic format supported only. Hence, to the best of our knowledge, the single-extended precision has never been implemented: when double precision was available, it fulfilled all the purposes of a single-extended format. Table B.1 gives the main parameters of the formats specified by IEEE 754-1985.

J.-M. Muller et al., *Handbook of Floating-Point Arithmetic*,
https://doi.org/10.1007/978-3-319-76526-6

Format	Hidden bit	Precision p	$e_{\min}$	$e_{\max}$
Single precision	yes	24	-126	127
Double precision	yes	53	-1022	1023
Single-extended	optional	≥ 32	≤ -1022	≥ 1023
Double-extended	optional	≥ 64	≤ -16382	≥ 16383
Double-ext. (IA32)	no	64	-16382	16383

Table B.1: Main parameters of the four formats specified by the IEEE 754-1985 standard [12] (©IEEE, 1985, with permission). A specific single-extended format has not been implemented in practice. The last line describes the double-extended format introduced by Intel in the 387 FPU, and available in subsequent IA32 compatible processors by Intel, Cyrix, AMD, and others.

B.1.2 Rounding modes specified by IEEE 754-1985

The IEEE 754-1985 standard defined four rounding modes: round toward $-\infty$ (rounding function RD), round toward $+\infty$ (rounding function RU), round toward zero (rounding function RZ), and round to nearest *ties to even* (RN), also called *round to nearest even*. There was no round to nearest *ties to away* (it was brought up by IEEE 754-2008).

B.1.3 Operations specified by IEEE 754-1985

B.1.3.1 Arithmetic operations and square root

The IEEE 754-1985 standard required that addition, subtraction, multiplication, and division of operands of the same format be provided, for all supported formats, with correct rounding (with the four rounding modes presented above). The standard also required a correctly rounded square root in all supported formats. The FMA (fused multiply-add) operator was not specified.

It was also permitted (but not required) that these operations be provided (still with correct rounding) for operands of different formats (in such a case, the destination format had to be at least as wide as the wider operand's format).

B.1.3.2 Conversions to and from decimal strings

At the time the IEEE 754-1985 standard was released, some of the algorithms presented in Section 4.9 were not known. This explains why the requirements of the standard were clearly below what one could now expect. The 2008 version of the standard has much stronger requirements.

	Decimal to binary		Binary to decimal	
	$M_{10,\max}^{(1)}$	$E_{10,\max}^{(1)}$	$M_{10,\max}^{(2)}$	$E_{10,\max}^{(2)}$
Single precision	$10^9 - 1$	99	$10^9 - 1$	53
Double precision	$10^{17} - 1$	999	$10^{17} - 1$	340

Table B.2: The thresholds for conversion from and to a decimal string, as specified by the IEEE 754-1985 standard [12] (©IEEE, 1985, with permission).

The requirements of the IEEE 754-1985 standard were:

- conversions must be provided between decimal strings in at least one format and binary floating-point numbers in all basic floating-point formats, for numbers of the form

$$\pm M_{10} \times 10^{\pm E_{10}}$$

  with $M_{10} \geq 0$ and $E_{10} \geq 0$;

- conversions must be correctly rounded for operands in the ranges specified in Table B.3;

- when the operands are not in the ranges specified in Table B.3:
  - in round-to-nearest mode, the conversion error cannot exceed $0.97\,\text{ulp}$ of the target format;
  - in the directed rounding modes, the "direction" of the rounding must be honored (e.g., for round-toward $-\infty$ mode, the delivered result must be less than or equal to the initial value), and the rounding error cannot exceed $1.47\,\text{ulp}$ of the target format;

- conversions must be *monotonic* (if $x \leq y$ before conversion, then $x \leq y$ after conversion);

- when rounding to nearest, as long as the decimal strings have at least 9 digits for single precision and 17 digits for double precision, conversion from binary to decimal and back to binary must produce the initial value exactly (Table B.2). This allows one to store intermediate results in files, and to read them later on, without losing any information, as explained in Chapter 2, Section 4.9.

B.1.4 Exceptions specified by IEEE 754-1985

The five exceptions listed in Section 2.5 (invalid, division by zero, overflow, underflow, inexact) had to be signaled when detected, either by taking a *trap* or by setting a *status flag*. The default was not to use traps.

	Decimal to binary		Binary to decimal	
	$M^{(1)}_{10,\max}$	$E^{(1)}_{10,\text{corr}}$	$M^{(2)}_{10,\max}$	$E^{(2)}_{10,\text{corr}}$
Single precision	$10^9 - 1$	13	$10^9 - 1$	13
Double precision	$10^{17} - 1$	27	$10^{17} - 1$	27

Table B.3: Correctly rounded decimal conversion range, as specified by the IEEE 754-1985 standard [12] (©IEEE, 1985, with permission).

B.2 The IEEE 854-1987 Standard

The IEEE 854-1987 standard [13] covered "radix-independent" floating-point arithmetic. This does not mean that all possible radices were considered: actually, that standard only focused on radices 2 and 10. We will just present it briefly (it is now superseded by IEEE 754-2008 [267]).

Unlike IEEE 754-1985, the IEEE 854-1987 standard did not fully specify formats or internal encodings. It merely expressed constraints between the parameters β, $e_{\min}$, $e_{\max}$, and p of the various precisions provided by an implementation. It also said that for each available precision, we must have two infinities, at least one signaling NaN, and at least one quiet NaN (as in the IEEE 754-1985 standard). In the remainder of this section, β is equal to 2 or 10. The same radix must be used for all available precisions: an arithmetic system compliant to IEEE 854-1987 is either binary or decimal, but it cannot mix up the two kinds of representations.

B.2.1 Constraints internal to a format

A balance must be found between the precision p and the value of the extremal exponents $e_{\min}$ and $e_{\max}$. If p is too large compared to $|e_{\min}|$ and $e_{\max}$, then underflows or overflows may occur too often. Also, there must be some balance between $e_{\min}$ and $e_{\max}$: to avoid underflows or overflows when computing reciprocals of normalized floating-point numbers as much as possible, one might want $e_{\min} \approx -e_{\max}$. Since underflow (more precisely, *gradual underflow*, with subnormal numbers available) is less harmful than overflow, it is preferable to have $e_{\min}$ very slightly above[1] $-e_{\max}$. Here are the constraints specified by the IEEE 854-1987 standard.

- We *must* have
$$\frac{e_{\max} - e_{\min}}{p} > 5,$$
and it is *recommended* that
$$\frac{e_{\max} - e_{\min}}{p} > 10.$$

[1] This is why the IEEE 754-2008 standard now requires $e_{\min} = 1 - e_{\max}$ for all formats.

- We must have $\beta^{p-1} \geq 10^5$.
- $\beta^{e_{\max}+e_{\min}+1}$ should be the smallest power of β greater than or equal to 4 (which was a very complicated way of saying that $e_{\min}$ should be $1 - e_{\max}$ in radix 2 and $-e_{\max}$ in radix 10).

For instance, the binary32 format of IEEE 754-2008 satisfies these requirements: with $\beta = 2$, $p = 24$, $e_{\min} = -126$, and $e_{\max} = 127$, we have

$$\begin{cases} \dfrac{e_{\max} - e_{\min}}{p} & = & 10.54\ldots & > & 10; \\ \beta^{p-1} & = & 2^{23} & \geq & 10^5; \\ \beta^{e_{\max}+e_{\min}+1} & = & 2^2 & = & 4. \end{cases}$$

B.2.2 Various formats and the constraints between them

The narrowest supported format was called *single precision*. When a second, wider basic format is supported, it was called *double precision*. The required constraints between their respective parameters $e_{\min s}$, $e_{\max s}$, p_s and $e_{\min d}$, $e_{\max d}$, p_d were:

- $\beta^{p_d} \geq 10\beta^{2p_s}$;
- $e_{\max d} \geq 8e_{\max s} + 7$;
- $e_{\min d} \leq 8e_{\min s}$.

Extended precisions were also possible. For obvious reasons, the only extended precision that was recommended was the one associated with the widest supported basic precision. If $e_{\min}$, $e_{\max}$, and p are the extremal exponents and precision of that widest basic precision, the parameters $e_{\min e}$, $e_{\max e}$, and p_e of the corresponding extended precision had to satisfy:

- $e_{\max e} \geq 8e_{\max} + 7$;
- $e_{\min e} \leq 8e_{\min}$;
- if $\beta = 2$,
$$p_e \geq p + \lceil \log_2 (e_{\max} - e_{\min}) \rceil ; \tag{B.1}$$
- for all β, $p_e \geq 1.2p$.

It was also recommended that

$$p_e > 1 + p + \frac{\log\left(3\log(\beta)\left(e_{\max}+1\right)\right)}{\log(\beta)}. \tag{B.2}$$

The purpose of constraint (B.1) was to facilitate the support of conversion to and from decimal strings for the basic formats, using algorithms that were available at that time. The purpose of (B.2) was to make accurate implementation, in the basic formats, of the power function x^y simpler.

B.2.3 Rounding

The IEEE 854-1987 standard required that the arithmetic operations and the square root be correctly rounded. Exactly as for IEEE 754-1985, four rounding modes were specified: rounding toward $-\infty$, toward $+\infty$, toward 0, and to nearest ties to even.

B.2.4 Operations

The arithmetic operations, the remainder operation, and the square root (including the $\sqrt{-0} = -0$ requirement) were defined very much as in IEEE 754-1985.

B.2.5 Comparisons

The comparisons were defined very much as in IEEE 754-1985. Especially, every NaN compares "unordered" with everything including itself: the test "$x \neq x$" must return **true** if and only if x is a NaN.

B.2.6 Exceptions

The IEEE 754-1985 way of handling exceptions was also chosen for IEEE 854-1987.

B.3 The Need for a Revision

The IEEE 754-1985 standard was a huge improvement. It soon became implemented on most platforms of commercial significance. And yet, 15 years after its release, there was a clear need for a revision.

- Some features that had become common practice needed to be standardized: e.g., the "quadruple-precision" (i.e., 128-bit wide, binary) format, the fused multiply-add operator.
- Since 1985, new algorithms were published that allowed one to easily perform computations that were previously thought too complex. Typical examples are the radix conversion algorithms presented in Section 4.9: now, for an internal binary format, it is possible to have much stronger requirements on the accuracy of the conversions that must be done when reading or printing decimal strings. Another example is the availability of reasonably fast libraries for some correctly rounded elementary functions: the revised standard can now deal with transcendental functions and recommend that some should be correctly rounded.

- There were some ambiguities in IEEE 754-1985. For instance, when evaluating expressions, when a larger internal format is available in hardware, it was unclear in which format the implicit intermediate variables should be represented.

B.4 The IEEE 754-2008 Revision

The IEEE 754-1985 standard has been revised from 2000 to 2006, and the revised standard was adopted in June 2008. Some of the various goals of the working group were as follows (see http://grouper.ieee.org/groups/754/revision.html):

- merging the 854-1987 standard into the 754-1985 standard;
- reducing the implementation choices;
- resolving some ambiguities in the 754-1985 standard (especially concerning expression evaluation and exception handling). The revised standard allows languages and users to focus on portability and reproducibility, or on performance;
- standardizing the fused multiply-add (FMA) operation, and
- including quadruple precision.

Also, the working group had to cope with a very strong constraint: the revised standard would rather not invalidate hardware that conformed to the old IEEE 754-1985 standard.

Bibliography

[1] E. Abu-Shama and M. Bayoumi. A new cell for low power adders. In *International Symposium on Circuits and Systems (ISCAS)*, pages 1014–1017, 1996.

[2] Advanced Micro Devices. 128-bit SSE5 instruction set, 2007. Available at http://pdinda.org/icsclass/doc/AMD_ARCH_MANUALS/AMD64_128_Bit_SSE5_Instrs.pdf.

[3] R. C. Agarwal, F. G. Gustavson, and M. S. Schmookler. Series approximation methods for divide and square root in the Power3™ processor. In *14th IEEE Symposium on Computer Arithmetic (ARITH-14)*, pages 116–123, April 1999.

[4] M. Ajtai. The shortest vector problem in L_2 is NP-hard for randomized reductions. Extended abstract. In *30th Annual ACM Symposium on Theory of Computing (STOC)*, pages 10–19, 1998.

[5] M. Ajtai, R. Kumar, and D. Sivakumar. A sieve algorithm for the shortest lattice vector problem. In *33rd Annual ACM Symposium on Theory of Computing (STOC)*, pages 601–610, 2001.

[6] B. Akbarpour, A. T. Abdel-Hamid, S. Tahar, and J. Harrison. Verifying a synthesized implementation of IEEE-754 floating-point exponential function using HOL. *The Computer Journal*, 53(4):465, 2010.

[7] L. Aksoy, E. Costa, P. Flores, and J. Monteiro. Optimization of area in digital FIR filters using gate-level metrics. In *Design Automation Conference*, pages 420–423, 2007.

[8] J. Alexandre dit Sandretto and A. Chapoutot. Validated Explicit and Implicit Runge-Kutta Methods. *Reliable Computing*, 22:78–103, 2016.

J.-M. Muller et al., *Handbook of Floating-Point Arithmetic*,
https://doi.org/10.1007/978-3-319-76526-6

[9] E. Allen, D. Chase, V. Luchangco, J.-W. Maessen, and G. L. Steele, Jr. Object-oriented units of measurement. In *19th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA)*, pages 384–403, 2004.

[10] Altera Corporation. *FFT/IFFT Block Floating Point Scaling*, 2005. Application note 404-1.0.

[11] B. Amedro, V. Bodnartchouck, D. Caromel, C. Delbé, F. Huet, and G. L. Taboada. Current state of Java for HP. Preprint, Technical Report 0353, INRIA, 2008. Available at http://hal.inria.fr/inria-00312039/en.

[12] American National Standards Institute and Institute of Electrical and Electronic Engineers. *IEEE Standard for Binary Floating-Point Arithmetic*. ANSI/IEEE Standard 754–1985, 1985.

[13] American National Standards Institute and Institute of Electrical and Electronic Engineers. *IEEE Standard for Radix Independent Floating-Point Arithmetic*. ANSI/IEEE Standard 854–1987, 1987.

[14] C. Anderson, N. Astafiev, and S. Story. Accurate math functions on the Intel IA-32 architecture: A performance-driven design. In *Real Numbers and Computers*, pages 93–105, July 2006.

[15] S. F. Anderson, J. G. Earle, R. E. Goldschmidt, and D. M. Powers. The IBM 360/370 model 91: floating-point execution unit. *IBM Journal of Research and Development*, 1967. Reprinted in [583].

[16] M. Andrysco, R. Jhala, and S. Lerner. Printing floating-point numbers: A faster, always correct method. In *43rd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)*, pages 555–567, 2016.

[17] ARM. *ARM Developer Suite: Compilers and Libraries Guide*. ARM Limited, November 2001. Available at http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.dui0067d/index.html or in PDF at http://infocenter.arm.com/help/topic/com.arm.doc.dui0067d/DUI0067.pdf.

[18] ARM. *ARM Developer Suite: Developer Guide*. ARM Limited, 1.2 edition, November 2001. Available at http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.dui0056d/index.html or in PDF at http://infocenter.arm.com/help/topic/com.arm.doc.dui0056d/DUI0056.pdf.

[19] ARM. *ARM Compiler: armasm User Guide*. ARM Limited, 6.7 edition, 2017. Available at http://infocenter.arm.com/help/topic/com.arm.doc.100069_0607_00_en/ or in PDF at http://infocenter.arm.com/help/topic/com.arm.doc.100069_0607_00_en/armasm_user_guide_100069_0607_00_en.pdf.

[20] W. Aspray, A. G. Bromley, M. Campbell-Kelly, P. E. Ceruzzi, and M. R. Williams. *Computing Before Computers*. Iowa State University Press, Ames, Iowa, 1990. Available at http://ed-thelen.org/comp-hist/CBC.html.

[21] A. Avizienis. Signed-digit number representations for fast parallel arithmetic. *IRE Transactions on Electronic Computers*, 10:389–400, 1961. Reprinted in [584].

[22] A. Azmi and F. Lombardi. On a tapered floating point system. In *9th IEEE Symposium on Computer Arithmetic (ARITH-9)*, pages 2–9, September 1989.

[23] L. Babai. On Lovász' lattice reduction and the nearest lattice point problem. *Combinatorica*, 6(1):1–13, 1986.

[24] I. Babuška. Numerical stability in mathematical analysis. In *Proceedings of the 1968 IFIP Congress*, volume 1, pages 11–23, 1969.

[25] D. H. Bailey. Some background on Kanada's recent pi calculation. Technical report, Lawrence Berkeley National Laboratory, 2003. Available at http://crd.lbl.gov/~dhbailey/dhbpapers/dhb-kanada.pdf.

[26] D. H. Bailey, R. Barrio, and J. M. Borwein. High precision computation: Mathematical physics and dynamics. *Applied Mathematics and Computation*, 218:10106–10121, 2012.

[27] D. H. Bailey and J. M. Borwein. Experimental mathematics: examples, methods and implications. *Notices of the AMS*, 52(5):502–514, 2005.

[28] D. H. Bailey, J. M. Borwein, P. B. Borwein, and S. Plouffe. The quest for pi. *Mathematical Intelligencer*, 19(1):50–57, 1997.

[29] D. H. Bailey, Y. Hida, X. S. Li, and B. Thompson. ARPREC: an arbitrary precision computation package. Technical report, Lawrence Berkeley National Laboratory, 2002. Available at http://crd.lbl.gov/~dhbailey/dhbpapers/arprec.pdf.

[30] S. Banescu, F. de Dinechin, B. Pasca, and R. Tudoran. Multipliers for floating-point double precision and beyond on FPGAs. *ACM SIGARCH Computer Architecture News*, 38:73–79, 2010.

[31] G. Barrett. Formal methods applied to a floating-point system. *IEEE Transactions on Software Engineering*, 15(5):611–621, 1989.

[32] M. Baudin. Error bounds of complex arithmetic. Technical report, 2011. Available at http://forge.scilab.org/upload/compdiv/files/complexerrorbounds_v0.2.pdf.

[33] P.-D. Beck and M. Nehmeier. Parallel interval Newton method on CUDA. In *International Workshop on Applied Parallel Computing*, pages 454–464, 2012.

[34] A. Becker, L. Ducas, N. Gama, and T. Laarhoven. New directions in nearest neighbor searching with applications to lattice sieving. In *27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 10–24, 2016.

[35] F. Benford. The law of anomalous numbers. *Proceedings of the American Philosophical Society*, 78(4):551–572, 1938.

[36] M. Bennani and M. C. Brunet. PRECISE: simulation of round-off error propagation model. In *12th World IMACS Congress*, July 1988.

[37] C. Berg. *Formal Verification of an IEEE Floating-Point Adder*. Master's thesis, Universität des Saarlandes, Germany, 2001.

[38] D. J. Bernstein. Multidigit multiplication for mathematicians. Available at https://cr.yp.to/papers/m3.pdf, 2001.

[39] F. Blomquist, W. Hofschuster, and W. Krämer. Real and complex staggered (interval) arithmetics with wide exponent range. Technical Report 2008/1, Universität Wuppertal, Germany, 2008. In German.

[40] A. Blot, J.-M. Muller, and L. Théry. Formal correctness of comparison algorithms between binary64 and decimal64 floating-point numbers. In *10th International Workshop on Numerical Software Verification (NSV)*, pages 25–37, 2017.

[41] G. Bohlender, W. Walter, P. Kornerup, and D. W. Matula. Semantics for exact floating point operations. In *10th IEEE Symposium on Computer Arithmetic (ARITH-10)*, pages 22–26, June 1991.

[42] S. Boldo. Pitfalls of a full floating-point proof: example on the formal proof of the Veltkamp/Dekker algorithms. In *3rd International Joint Conference on Automated Reasoning (IJCAR)*, volume 4130 of *Lecture Notes in Computer Science*, pages 52–66, Seattle, WA, USA, 2006.

[43] S. Boldo. Kahan's algorithm for a correct discriminant computation at last formally proven. *IEEE Transactions on Computers*, 58(2):220–225, 2009.

[44] S. Boldo. How to compute the area of a triangle: a formal revisit. In *21th IEEE Symposium on Computer Arithmetic (ARITH-21)*, pages 91–98, Austin, TX, USA, April 2013.

[45] S. Boldo. Formal verification of programs computing the floating-point average. In *17th International Conference on Formal Engineering Methods (ICFEM)*, volume 9407 of *Lecture Notes in Computer Science*, pages 17–32, Paris, France, November 2015.

[46] S. Boldo and M. Daumas. Representable correcting terms for possibly underflowing floating point operations. In *16th IEEE Symposium on Computer Arithmetic (ARITH-16)*, pages 79–86, Santiago de Compostela, Spain, 2003.

[47] S. Boldo, M. Daumas, and R.-C. Li. Formally verified argument reduction with a fused multiply-add. *IEEE Transactions on Computers*, 58(8):1139–1145, 2009.

[48] S. Boldo, M. Daumas, C. Moreau-Finot, and L. Théry. Computer validated proofs of a toolset for adaptable arithmetic. Technical report, École Normale Supérieure de Lyon, 2001. Available at http://arxiv.org/pdf/cs.MS/0107025.

[49] S. Boldo, S. Graillat, and J.-M. Muller. On the robustness of the 2Sum and Fast2Sum algorithms. *ACM Transactions on Mathematical Software*, 44(1):4:1–4:14, 2017.

[50] S. Boldo, M. Joldeş, J.-M. Muller, and V. Popescu. Formal verification of a floating-point expansion renormalization algorithm. In *8th International Conference on Interactive Theorem Proving (ITP)*, Brasilia, Brazil, 2017.

[51] S. Boldo, J.-H. Jourdan, X. Leroy, and G. Melquiond. Verified compilation of floating-point computations. *Journal of Automated Reasoning*, 54(2):135–163, 2015.

[52] S. Boldo and G. Melquiond. Emulation of FMA and correctly rounded sums: proved algorithms using rounding to odd. *IEEE Transactions on Computers*, 57(4):462–471, 2008.

[53] S. Boldo and G. Melquiond. Flocq: A unified library for proving floating-point algorithms in Coq. In *20th IEEE Symposium on Computer Arithmetic (ARITH-20)*, pages 243–252, Tübingen, Germany, 2011.

[54] S. Boldo and G. Melquiond. *Computer Arithmetic and Formal Proofs*. ISTE Press – Elsevier, 2017.

[55] S. Boldo and J.-M. Muller. Exact and approximated error of the FMA. *IEEE Transactions on Computers*, 60(2):157–164, 2011.

[56] S. Boldo and C. Muñoz. Provably faithful evaluation of polynomials. In *ACM Symposium on Applied Computing*, pages 1328–1332, Dijon, France, 2006.

[57] A. D. Booth. A signed binary multiplication technique. *Quarterly Journal of Mechanics and Applied Mathematics*, 4(2):236–240, 1951. Reprinted in [583].

[58] J. Borwein and D. H. Bailey. *Mathematics by Experiment: Plausible Reasoning in the 21st Century*. A. K. Peters, Natick, MA, 2004.

[59] P. Borwein and T. Erdélyi. *Polynomials and Polynomial Inequalities*. Graduate Texts in Mathematics, 161. Springer-Verlag, New York, 1995.

[60] W. Bosma, J. Cannon, and C. Playoust. The Magma algebra system. I. The user language. *Journal of Symbolic Computation*, 24(3–4):235–265, 1997.

[61] N. Boullis and A. Tisserand. Some optimizations of hardware multiplication by constant matrices. *IEEE Transactions on Computers*, 54(10):1271–1282, 2005.

[62] R. T. Boute. The Euclidean definition of the functions div and mod. *ACM Transactions on Programming Languages and Systems*, 14(2):127–144, 1992.

[63] R. P. Brent. On the precision attainable with various floating-point number systems. *IEEE Transactions on Computers*, C-22(6):601–607, 1973.

[64] R. P. Brent. A FORTRAN multiple-precision arithmetic package. *ACM Transactions on Mathematical Software*, 4(1):57–70, 1978.

[65] R. P. Brent, C. Percival, and P. Zimmermann. Error bounds on complex floating-point multiplication. *Mathematics of Computation*, 76:1469–1481, 2007.

[66] R. P. Brent and P. Zimmermann. *Modern Computer Arithmetic*. Cambridge University Press, 2011.

[67] K. Briggs. The `doubledouble` library, 1998. Available at http://www.boutell.com/fracster-src/doubledouble/doubledouble.html.

[68] N. Brisebarre and S. Chevillard. Efficient polynomial L^∞ approximations. In *18th IEEE Symposium on Computer Arithmetic (ARITH-18)*, pages 169–176, Montpellier, France, 2007.

[69] N. Brisebarre, F. de Dinechin, and J.-M. Muller. Integer and floating-point constant multipliers for FPGAs. In *Application-specific Systems, Architectures and Processors*, pages 239–244, 2008.

[70] N. Brisebarre and G. Hanrot. Floating-point L^2-approximations to functions. In *18th IEEE Symposium on Computer Arithmetic (ARITH-18)*, pages 177–186, Montpellier, France, 2007.

[71] N. Brisebarre, G. Hanrot, and O. Robert. Exponential sums and correctly-rounded functions. *IEEE Transactions on Computers*, 66(12):2044–2057, 2017.

[72] N. Brisebarre, M. Joldeş, É. Martin-Dorel, M. Mayero, J.-M. Muller, I. Paşca, L. Rideau, and L. Théry. Rigorous polynomial approximation using Taylor models in Coq. In *4th International Symposium on NASA Formal Methods (NFM)*, volume 7226 of *Lecture Notes in Computer Science*, pages 85–99, Norfolk, VA, USA, 2012.

[73] N. Brisebarre, C. Lauter, M. Mezzarobba, and J.-M. Muller. Comparison between binary and decimal floating-point numbers. *IEEE Transactions on Computers*, 65(7):2032–2044, 2016.

[74] N. Brisebarre and J.-M. Muller. Correctly rounded multiplication by arbitrary precision constants. *IEEE Transactions on Computers*, 57(2):165–174, 2008.

[75] N. Brisebarre, J.-M. Muller, and S.-K. Raina. Accelerating correctly rounded floating-point division when the divisor is known in advance. *IEEE Transactions on Computers*, 53(8):1069–1072, 2004.

[76] W. S. Brown. A simple but realistic model of floating-point computation. *ACM Transactions on Mathematical Software*, 7(4), 1981.

[77] W. S. Brown and P. L. Richman. The choice of base. *Communications of the ACM*, 12(10):560–561, 1969.

[78] C. Bruel. If-conversion SSA framework for partially predicated VLIW architectures. In *4th Workshop on Optimizations for DSP and Embedded Systems (ODES)*, New York, NY, USA, March 2006.

[79] J. D. Bruguera and T. Lang. Leading-one prediction with concurrent position correction. *IEEE Transactions on Computers*, 48(10):1083–1097, 1999.

[80] J. D. Bruguera and T. Lang. Floating-point fused multiply-add: Reduced latency for floating-point addition. In *17th IEEE Symposium on Computer Arithmetic (ARITH-17)*, Cape Cod, MA, USA, June 2005.

[81] M. C. Brunet and F. Chatelin. A probabilistic round-off error propagation model, application to the eigenvalue problem. In *Reliable Numerical Software*, 1987. Available at http://www.boutell.com/fracster-src/doubledouble/doubledouble.html.

[82] N. Brunie. Modified FMA for exact low precision product accumulation. In *24th IEEE Symposium on Computer Arithmetic (ARITH-24)*, pages 106–113, July 2017.

[83] N. Brunie, F. de Dinechin, and B. Dupont de Dinechin. A mixed-precision fused multiply and add. In *45th Asilomar Conference on Signals, Systems, and Computers*, November 2011.

[84] N. Brunie, F. de Dinechin, and B. Dupont de Dinechin. Mixed-precision merged multiplication and addition operator. Patent WO/2012/175828, December 2012.

[85] H. T. Bui, Y. Wang, and Y. Jiang. Design and analysis of low-power 10-transistor full adders using novel XORXNOR gates. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 49(1), 2003.

[86] R. G. Burger and R. K. Dybvig. Printing floating-point numbers quickly and accurately. In *SIGPLAN'96 Conference on Programming Languages Design and Implementation (PLDI)*, pages 108–116, June 1996.

[87] F. Y. Busaba, C. A. Krygowski, W. H. Li, E. M. Schwarz, and S. R. Carlough. The IBM z900 decimal arithmetic unit. In *35th Asilomar Conference on Signals, Systems, and Computers*, volume 2, pages 1335–1339, November 2001.

[88] P. R. Cappello and K. Steiglitz. A VLSI layout for a pipelined Dadda multiplier. *ACM Transactions on Computer Systems*, 1(2):157–174, 1983. Reprinted in [584].

[89] S. Carlough, A. Collura, S. Mueller, and M. Kroener. The IBM zEnterprise-196 decimal floating-point accelerator. In *20th IEEE Symposium on Computer Arithmetic (ARITH-20)*, pages 139–146, July 2011.

[90] J. W. S. Cassels. *An Introduction to the Geometry of Numbers*. Classics in Mathematics. Springer-Verlag, Berlin, 1997. Corrected reprint of the 1971 edition.

[91] A. Cauchy. Sur les moyens d'éviter les erreurs dans les calculs numériques. *Comptes Rendus de l'Académie des Sciences, Paris*, 11:789–798, 1840. Republished in: Augustin Cauchy, œuvres complètes, 1ère série, Tome V, pages 431–442.

[92] P. E. Ceruzzi. The early computers of Konrad Zuse, 1935 to 1945. *Annals of the History of Computing*, 3(3):241–262, 1981.

[93] P. E. Ceruzzi. *A History of Modern Computing*. MIT Press, 2nd edition, 2003.

[94] K. D. Chapman. Fast integer multipliers fit in FPGAs (EDN 1993 design idea winner). *EDN Magazine*, 1994.

[95] T. C. Chen and I. T. Ho. Storage-efficient representation of decimal data. *Communications of the ACM*, 18(1):49–52, 1975.

[96] S. Chevillard, J. Harrison, M. Joldeş, and C. Lauter. Efficient and accurate computation of upper bounds of approximation errors. *Theoretical Computer Science*, 412(16):1523–1543, 2011.

[97] S. Chevillard, M. Joldeş, and C. Q. Lauter. Sollya: An environment for the development of numerical codes. In *3rd International Congress on Mathematical Software (ICMS)*, volume 6327 of *Lecture Notes in Computer Science*, pages 28–31, September 2010.

[98] S. Chevillard and C. Q. Lauter. A certified infinite norm for the implementation of elementary functions. In *7th International Conference on Quality Software (QSIC)*, pages 153–160, Portland, OR, USA, 2007.

[99] C. W. Chou, D. B. Hume, T. Rosenband, and D. J. Wineland. Optical clocks and relativity. *Science*, 329(5999):1630–1633, 2010.

[100] C. W. Clenshaw and F. W. J. Olver. Beyond floating point. *Journal of the ACM*, 31:319–328, 1985.

[101] W. D. Clinger. How to read floating-point numbers accurately. *ACM SIGPLAN Notices*, 25(6):92–101, 1990.

[102] W. D. Clinger. Retrospective: how to read floating-point numbers accurately. *ACM SIGPLAN Notices*, 39(4):360–371, 2004.

[103] D. Cochran. Algorithms and accuracy in the HP 35. *Hewlett Packard Journal*, 23:10–11, 1972.

[104] W. J. Cody. Static and dynamic numerical characteristics of floating-point arithmetic. *IEEE Transactions on Computers*, C-22(6):598–601, 1973.

[105] W. J. Cody. Implementation and testing of function software. In *Problems and Methodologies in Mathematical Software Production*, volume 142 of *Lecture Notes in Computer Science*, 1982.

[106] W. J. Cody. MACHAR: a subroutine to dynamically determine machine parameters. *ACM Transactions on Mathematical Software*, 14(4):301–311, 1988.

[107] W. J. Cody and W. Waite. *Software Manual for the Elementary Functions*. Prentice-Hall, Englewood Cliffs, NJ, 1980.

[108] T. Coe and P. T. P. Tang. It takes six ones to reach a flaw. In *12th IEEE Symposium on Computer Arithmetic (ARITH-12)*, pages 140–146, July 1995.

[109] S. Collange, M. Daumas, and D. Defour. État de l'intégration de la virgule flottante dans les processeurs graphiques. *Technique et Science Informatiques*, 27(6):719–733, 2008. In French.

[110] S. Collange, M. Daumas, and D. Defour. *GPU Computing Gems Jade Edition*, chapter Interval arithmetic in CUDA, pages 99–107. Morgan Kaufmann, 2011.

[111] J. L. D. Comba and J. Stolfi. Affine arithmetic and its applications to computer graphics. In *VI Simpósio Brasileiro de Computação Gráfica e Processamento de Imagens (SIBGRAPI'93)*, pages 9–18, 1993.

[112] J. H. Conway and N. J. A. Sloane. *Sphere Packings, Lattices and Groups*. Springer-Verlag, New York, 1988.

[113] D. Coppersmith. Finding a small root of a univariate modular equation. In *Advances in Cryptology – EUROCRYPT*, volume 1070 of *Lecture Notes in Computer Science*, pages 155–165, Saragossa, Spain, May 1996.

[114] D. Coppersmith. Finding small solutions to small degree polynomials. In *International Conference on Cryptography and Lattices (CaLC)*, volume 2146 of *Lecture Notes in Computer Science*, pages 20–31, Providence, RI, USA, March 2001.

[115] R. M. Corless and N. Fillion. *A Graduate Introduction to Numerical Methods, From the Viewpoint of Backward Error Analysis*. Springer, 2013.

[116] M. Cornea, C. Anderson, J. Harrison, P. T. P. Tang, E. Schneider, and C. Tsen. A software implementation of the IEEE 754R decimal floating-point arithmetic using the binary encoding format. In *18th IEEE Symposium on Computer Arithmetic (ARITH-18)*, pages 29–37, June 2007.

[117] M. Cornea, J. Harrison, C. Anderson, P. T. P. Tang, E. Schneider, and E. Gvozdev. A software implementation of the IEEE 754R decimal floating-point arithmetic using the binary encoding format. *IEEE Transactions on Computers*, 58(2):148–162, 2009.

[118] M. Cornea, J. Harrison, and P. T. P. Tang. *Scientific Computing on Itanium®-based Systems*. Intel Press, Hillsboro, OR, 2002.

[119] M. Cornea, C. Iordache, J. Harrison, and P. Markstein. Integer divide and remainder operations in the IA-64 architecture. In *4th Conference on Real Numbers and Computers (RNC-4)*, 2000.

[120] M. A. Cornea-Hasegan, R. A. Golliver, and P. Markstein. Correctness proofs outline for Newton–Raphson based floating-point divide and square root algorithms. In *14th IEEE Symposium on Computer Arithmetic (ARITH-14)*, pages 96–105, April 1999.

[121] M. F. Cowlishaw. Decimal floating-point: algorism for computers. In *16th IEEE Symposium on Computer Arithmetic (ARITH-16)*, pages 104–111, June 2003.

[122] M. F. Cowlishaw, E. M. Schwarz, R. M. Smith, and C. F. Webb. A decimal floating-point specification. In *15th IEEE Symposium on Computer Arithmetic (ARITH-15)*, pages 147–154, June 2001.

[123] O. Creţ, F. de Dinechin, I. Trestian, R. Tudoran, L. Creţ, and L. Văcariu. FPGA-based acceleration of the computations involved in transcranial magnetic stimulation. In *Southern Programmable Logic Conference*, pages 43–48, 2008.

[124] X. Cui, W. Liu, D. Wenwen, and F. Lombardi. A parallel decimal multiplier using hybrid binary coded decimal (BCD) codes. In *23rd IEEE Symposium on Computer Arithmetic (ARITH-23)*, pages 150–155, July 2016.

[125] A. Cuyt, B. Verdonk, S. Becuwe, and P. Kuterna. A remarkable example of catastrophic cancellation unraveled. *Computing*, 66:309–320, 2001.

[126] L. Dadda. Some schemes for parallel multipliers. *Alta Frequenza*, 34:349–356, 1965. Reprinted in [583].

[127] L. Dadda. On parallel digital multipliers. *Alta Frequenza*, 45:574–580, 1976. Reprinted in [583].

[128] A. Dahan-Dalmedico and J. Pfeiffer. *Histoire des Mathématiques*. Editions du Seuil, Paris, 1986. In French.

[129] C. Daramy-Loirat, D. Defour, F. de Dinechin, M. Gallet, N. Gast, C. Q. Lauter, and J.-M. Muller. CR-LIBM, a library of correctly-rounded elementary functions in double-precision. Technical report, LIP Laboratory, Arenaire team, December 2006. Available at https://hal-ens-lyon.archives-ouvertes.fr/ensl-01529804.

[130] D. Das Sarma and D. W. Matula. Measuring the accuracy of ROM reciprocal tables. *IEEE Transactions on Computers*, 43(8):932–940, 1994.

[131] D. Das Sarma and D. W. Matula. Faithful bipartite ROM reciprocal tables. In *12th IEEE Symposium on Computer Arithmetic (ARITH-12)*, pages 17–28, June 1995.

[132] D. Das Sarma and D. W. Matula. Faithful interpolation in reciprocal tables. In *13th IEEE Symposium on Computer Arithmetic (ARITH-13)*, pages 82–91, July 1997.

[133] M. Daumas and C. Finot. Division of floating point expansions with an application to the computation of a determinant. *Journal of Universal Computer Science*, 5(6):323–338, 1999.

[134] M. Daumas and G. Melquiond. Certification of bounds on expressions involving rounded operators. *Transactions on Mathematical Software*, 37(1):1–20, 2010.

[135] M. Daumas, G. Melquiond, and C. Muñoz. Guaranteed proofs using interval arithmetic. In *17th IEEE Symposium on Computer Arithmetic (ARITH-17)*, pages 188–195, Cape Cod, MA, USA, 2005.

[136] M. Daumas, L. Rideau, and L. Théry. A generic library of floating-point numbers and its application to exact computing. In *14th International Conference on Theorem Proving in Higher Order Logics (TPHOLs)*, pages 169–184, Edinburgh, Scotland, 2001.

[137] B. de Dinechin. From machine scheduling to VLIW instruction scheduling. *ST Journal of Research*, 1(2), 2004.

[138] F. de Dinechin. The price of routing in FPGAs. *Journal of Universal Computer Science*, 6(2):227–239, 2000.

[139] F. de Dinechin. Multiplication by rational constants. *IEEE Transactions on Circuits and Systems, II*, 52(2):98–102, 2012.

[140] F. de Dinechin and L.-S. Didier. Table-based division by small integer constants. In *Applied Reconfigurable Computing*, pages 53–63, March 2012.

[141] F. de Dinechin, P. Echeverría, M. López-Vallejo, and B. Pasca. Floating-point exponentiation units for reconfigurable computing. *ACM Transactions on Reconfigurable Technology and Systems*, 6(1), 2013.

[142] F. de Dinechin, A. V. Ershov, and N. Gast. Towards the post-ultimate libm. In *17th IEEE Symposium on Computer Arithmetic (ARITH-17)*, pages 288–295, 2005.

[143] F. de Dinechin and M. Istoan. Hardware implementations of fixed-point Atan2. In *22nd IEEE Symposium of Computer Arithmetic (ARITH-22)*, pages 34–41, June 2015.

[144] F. de Dinechin, M. Istoan, and G. Sergent. Fixed-point trigonometric functions on FPGAs. *SIGARCH Computer Architecture News*, 41(5):83–88, 2013.

[145] F. de Dinechin, M. Joldeş, and B. Pasca. Automatic generation of polynomial-based hardware architectures for function evaluation. In *Application-specific Systems, Architectures and Processors (ASAP)*, 2010.

[146] F. de Dinechin, M. Joldeş, B. Pasca, and G. Revy. Multiplicative square root algorithms for FPGAs. In *Field-Programmable Logic and Applications*, pages 574–577, 2010.

[147] F. de Dinechin, C. Q. Lauter, and G. Melquiond. Certifying the floating-point implementation of an elementary function using Gappa. *IEEE Transactions on Computers*, 60(2):242–253, 2011.

[148] F. de Dinechin, C. Q. Lauter, and J.-M. Muller. Fast and correctly rounded logarithms in double-precision. *Theoretical Informatics and Applications*, 41:85–102, 2007.

[149] F. de Dinechin, C. Q. Lauter, J.-M. Muller, and S. Torres. On Ziv's rounding test. *ACM Transactions on Mathematical Software*, 39(4), 2013.

[150] F. de Dinechin and V. Lefèvre. Constant multipliers for FPGAs. In *Parallel and Distributed Processing Techniques and Applications*, pages 167–173, 2000.

[151] F. de Dinechin, C. Loirat, and J.-M. Muller. A proven correctly rounded logarithm in double-precision. In *6th Conference on Real Numbers and Computers (RNC-6)*, 2004.

[152] F. de Dinechin, E. McIntosh, and F. Schmidt. Massive tracking on heterogeneous platforms. In *9th International Computational Accelerator Physics Conference (ICAP)*, October 2006.

[153] F. de Dinechin and B. Pasca. Large multipliers with fewer DSP blocks. In *Field Programmable Logic and Applications*, pages 250–255, August 2009.

[154] F. de Dinechin and B. Pasca. Floating-point exponential functions for DSP-enabled FPGAs. In *Field Programmable Technologies*, pages 110–117, December 2010. Best paper candidate.

[155] F. de Dinechin, B. Pasca, O. Creţ, and R. Tudoran. An FPGA-specific approach to floating-point accumulation and sum-of-products. In *Field-Programmable Technologies*, 2008.

[156] F. de Dinechin and A. Tisserand. Multipartite table methods. *IEEE Transactions on Computers*, 54(3):319–330, 2005.

[157] A. DeHon and N. Kapre. Optimistic parallelization of floating-point accumulation. In *18th Symposium on Computer Arithmetic (ARITH-18)*, pages 205–213, June 2007.

[158] T. J. Dekker. A floating-point technique for extending the available precision. *Numerische Mathematik*, 18(3):224–242, 1971.

[159] M. deLorimier and A. DeHon. Floating-point sparse matrix-vector multiply for FPGAs. In *Field-Programmable Gate Arrays*, pages 75–85, 2005.

[160] J. Demmel. Underflow and the reliability of numerical software. *SIAM Journal on Scientific and Statistical Computing*, 5(4):887–919, 1984.

[161] J. Demmel and H. D. Nguyen. Fast reproducible floating-point summation. In *21th IEEE Symposium on Computer Arithmetic (ARITH-21)*, pages 163–172, April 2013.

[162] J. Demmel and H. D. Nguyen. Parallel reproducible summation. *IEEE Transactions on Computers*, 64(7):2060–2070, 2015.

[163] J. W. Demmel and X. Li. Faster numerical algorithms via exception handling. In *11th IEEE Symposium on Computer Arithmetic*, pages 234–241, June 1993.

[164] J. W. Demmel and X. Li. Faster numerical algorithms via exception handling. *IEEE Transactions on Computers*, 43(8):983–992, 1994.

[165] J. Demmel, P. Ahrens, and H. D. Nguyen. Efficient reproducible floating point summation and BLAS. Technical Report UCB/EECS-2016-121, EECS Department, University of California, Berkeley, June 2016.

[166] J. Demmel and Y. Hida. Accurate and efficient floating point summation. *SIAM Journal of Scientific Computing*, 25(4):1214–1248, 2003.

[167] J. Demmel and Y. Hida. Fast and accurate floating point summation with application to computational geometry. *Numerical Algorithms*, 37(1):101–112, 2004.

[168] A. G. Dempster and M. D. Macleod. Constant integer multiplication using minimum adders. *Circuits, Devices and Systems, IEE Proceedings*, 141(5):407–413, 1994.

[169] R. Descartes. *La Géométrie*. Paris, 1637.

[170] J. Detrey and F. de Dinechin. Table-based polynomials for fast hardware function evaluation. In *Application-Specific Systems, Architectures and Processors*, pages 328–333, 2005.

[171] J. Detrey and F. de Dinechin. Floating-point trigonometric functions for FPGAs. In *Field-Programmable Logic and Applications*, pages 29–34, August 2007.

[172] J. Detrey and F. de Dinechin. Parameterized floating-point logarithm and exponential functions for FPGAs. *Microprocessors and Microsystems, Special Issue on FPGA-based Reconfigurable Computing*, 31(8):537–545, 2007.

[173] J. Detrey and F. de Dinechin. A tool for unbiased comparison between logarithmic and floating-point arithmetic. *Journal of VLSI Signal Processing*, 49(1):161–175, 2007.

[174] J. Detrey, F. de Dinechin, and X. Pujol. Return of the hardware floating-point elementary function. In *18th Symposium on Computer Arithmetic (ARITH-18)*, pages 161–168, June 2007.

[175] W. R. Dieter, A. Kaveti, and H. G. Dietz. Low-cost microarchitectural support for improved floating-point accuracy. *IEEE Computer Architecture Letters*, 6(1):13–16, 2007.

[176] V. Dimitrov, L. Imbert, and A. Zakaluzny. Multiplication by a constant is sublinear. In *18th IEEE Symposium on Computer Arithmetic (ARITH-18)*, pages 261–268, June 2007.

[177] W. S. Dorn. Generalizations of Horner's rule for polynomial evaluation. *IBM Journal of Research and Development*, 6(2):239–245, 1962.

[178] C. Doss and R. L. Riley, Jr. FPGA-based implementation of a robust IEEE-754 exponential unit. In *Field-Programmable Custom Computing Machines*, pages 229–238, 2004.

[179] Y. Dou, S. Vassiliadis, G. K. Kuzmanov, and G. N. Gaydadjiev. 64-bit floating-point FPGA matrix multiplication. In *Field-Programmable Gate Arrays*, pages 86–95, 2005.

[180] T. Drane, W.-C. Cheung, and G. Constantinides. Correctly rounded constant integer division via multiply-add. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1243–1246, Seoul, South Korea, May 2012.

[181] P. Echeverría and M. López-Vallejo. An FPGA implementation of the powering function with single precision floating-point arithmetic. In *8th Conference on Real Numbers and Computers (RNC-8)*, pages 17–26, 2008.

[182] J.-P. Eckmann, A. Malaspinas, and S. O. Kamphorst. *A Software Tool for Analysis in Function Spaces*, pages 147–167. Springer, 1991.

[183] A. Edelman. The mathematics of the Pentium division bug. *SIAM Review*, 39(1):54–67, 1997.

[184] L. Eisen, J. W. Ward, H. W. Tast, N. Mäding, J. Leenstra, S. M. Mueller, C. Jacobi, J. Preiss, E. M. Schwarz, and S. R. Carlough. IBM POWER6 accelerators: VMX and DFU. *IBM Journal of Research and Development*, 51(6):1–21, 2007.

[185] M. Ercegovac. Radix-16 evaluation of certain elementary functions. *IEEE Transactions on Computers*, C-22(6):561–566, 1973.

[186] M. D. Ercegovac and T. Lang. *Division and Square Root: Digit-Recurrence Algorithms and Implementations*. Kluwer Academic Publishers, Boston, MA, 1994.

[187] M. D. Ercegovac and T. Lang. *Digital Arithmetic*. Morgan Kaufmann Publishers, San Francisco, CA, 2004.

[188] M. D. Ercegovac and J.-M. Muller. Complex division with prescaling of the operands. In *14th IEEE Conference on Application-Specific Systems, Architectures and Processors (ASAP'2003)*, pages 304–314, June 2003.

[189] M. Ercegovac, J.-M. Muller, and A. Tisserand. Simple seed architectures for reciprocal and square root reciprocal. In *39th Asilomar Conference on Signals, Systems, and Computers*, November 2005.

[190] M. A. Erle and M. J. Schulte. Decimal multiplication via carry-save addition. In *Application-specific Systems, Architectures and Processors*, pages 348–355, 2003.

[191] M. A. Erle, M. J. Schulte, and B. J. Hickmann. Decimal floating-point multiplication via carry-save addition. In *18th IEEE Symposium on Computer Arithmetic (ARITH-18)*, pages 46–55, June 2007.

[192] M. A. Erle, M. J. Schulte, and B. J. Hickmann. Decimal floating-point multiplication. *IEEE Transactions on Computers*, 58(7):902–916, 2009.

[193] M. A. Erle, M. J. Schulte, and J. M. Linebarger. Potential speedup using decimal floating-point hardware. In *36th Asilomar Conference on Signals, Systems, and Computers*, volume 2, pages 1073–1077, November 2002.

[194] M. A. Erle, E. M. Schwarz, and M. J. Schulte. Decimal multiplication with efficient partial product generation. In *17th IEEE Symposium on Computer Arithmetic (ARITH-17)*, 2005.

[195] G. Even and W. J. Paul. On the design of IEEE compliant floating-point units. *IEEE Transactions on Computers*, 49(5):398–413, 2000.

[196] G. Even and P.-M. Seidel. A comparison of three rounding algorithms for IEEE floating-point multiplication. *IEEE Transactions on Computers*, 49(7):638–650, 2000.

[197] G. Even, P.-M. Seidel, and W. E. Ferguson. A parametric error analysis of Goldschmidt's division algorithm. *Journal of Computer and System Sciences*, 70(1):118–139, 2005.

[198] H. A. H. Fahmy, A. A. Liddicoat, and M. J. Flynn. Improving the effectiveness of floating point arithmetic. In *35th Asilomar Conference on Signals, Systems, and Computers*, volume 1, pages 875–879, November 2001.

[199] W. E. Ferguson, Jr. Exact computation of a sum or difference with applications to argument reduction. In *12th IEEE Symposium on Computer Arithmetic (ARITH-12)*, pages 216–221, Bath, UK, July 1995.

[200] S. A. Figueroa. When is double rounding innocuous? *ACM SIGNUM Newsletter*, 30(3), 1995.

[201] B. P. Flannery, W. H. Press, S. A. Teukolsky, and W. T. Vetterling. *Numerical Recipes in C*, 2nd edition. Cambridge University Press, New York, NY, 1992.

[202] G. E. Forsythe and C. B. Moler. *Computer Solution of Linear Algebraic Systems*. Prentice-Hall, Englewood Cliffs, NJ, 1967.

[203] P. Fortin, M. Gouicem, and S. Graillat. Towards solving the Table Maker's Dilemma on GPU. In *20th Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, pages 407–415, February 2012.

[204] L. Fousse, G. Hanrot, V. Lefèvre, P. Pélissier, and P. Zimmermann. MPFR: A multiple-precision binary floating-point library with correct rounding. *ACM Transactions on Mathematical Software*, 33(2), 2007. 15 pages. Available at http://www.mpfr.org/.

[205] D. Fowler and E. Robson. Square root approximations in old Babylonian mathematics: YBC 7289 in context. *Historia Mathematica*, 25:366–378, 1998.

[206] W. Fraser. A survey of methods of computing minimax and near-minimax polynomial approximations for functions of a single independent variable. *Journal of the ACM*, 12(3):295–314, 1965.

[207] P. Friedland. Algorithm 312: Absolute value and square root of a complex number. *Communications of the ACM*, 10(10):665, 1967.

[208] Fujitsu. *SPARC64^{TM} VI Extensions*. Fujitsu Limited, 1.3 edition, March 2007.

[209] M. Fürer. Faster integer multiplication. In *39th Annual ACM Symposium on Theory of Computing (STOC)*, pages 57–66, June 2007.

[210] P. Gaudry, A. Kruppa, and P. Zimmermann. A GMP-based implementation of Schönhage-Strassen's large integer multiplication algorithm. In *International Symposium on Symbolic and Algebraic Computation (ISSAC)*, pages 167–174, Waterloo, ON, Canada, 2007.

[211] D. M. Gay. Correctly-rounded binary-decimal and decimal-binary conversions. Technical Report Numerical Analysis Manuscript 90–10, ATT & Bell Laboratories (Murray Hill, NJ), November 1990.

[212] W. M. Gentleman and S. B. Marovitch. More on algorithms that reveal properties of floating-point arithmetic units. *Communications of the ACM*, 17(5):276–277, 1974.

[213] G. Gerwig, H. Wetter, E. M. Schwarz, J. Haess, C. A. Krygowski, B. M. Fleischer, and M. Kroener. The IBM eServer z990 floating-point unit. *IBM Journal of Research and Development*, 48(3.4):311–322, 2004.

[214] D. Goldberg. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys*, 23(1):5–48, 1991. An edited reprint is available at https://docs.oracle.com/cd/E19059-01/fortec6u2/806-7996/806-7996.pdf from Sun's Numerical Computation Guide; it contains an addendum *Differences Among IEEE 754 Implementations*, also available at http://www.validlab.com/goldberg/addendum.html.

[215] I. B. Goldberg. 27 bits are not enough for 8-digit accuracy. *Commun. ACM*, 10(2):105–106, 1967.

[216] O. Goldreich and S. Goldwasser. On the limits of non-approximability of lattice problems. In *30th Annual ACM Symposium on Theory of Computing (STOC)*, pages 1–9, May 1998.

[217] R. E. Goldschmidt. Applications of division by convergence. Master's thesis, Dept. of Electrical Engineering, Massachusetts Institute of Technology, Cambridge, MA, June 1964.

[218] A. Goldsztejn. Modal intervals revisited, part 1: A generalized interval natural extension. *Reliable Computing*, 16:130–183, 2012.

[219] A. Goldsztejn. Modal intervals revisited, part 2: A generalized interval mean value extension. *Reliable Computing*, 16:184–209, 2012.

[220] J. Gosling, B. Joy, G. Steele, G. Bracha, and A. Buckley. *The Java Language Specification, Java SE 8 edition*. Oracle, 2015.

[221] F. Goualard. Fast and correct SIMD algorithms for interval arithmetic. In *Workshop on State-of-the-Art in Scientific and Parallel Computing (PARA)*, May 2008.

[222] S. Graillat, P. Langlois, and N. Louvet. Algorithms for accurate, validated and fast computations with polynomials. *Japan Journal of Industrial and Applied Mathematics*, 26(2):215–231, 2009.

[223] S. Graillat, V. Lefèvre, and J.-M. Muller. On the maximum relative error when computing integer powers by iterated multiplications in floating-point arithmetic. *Numerical Algorithms*, 70:653–667, 2015.

[224] T. Granlund. The GNU multiple precision arithmetic library, release 6.1.2. Accessible electronically at https://gmplib.org/, September 2016.

[225] B. Greer, J. Harrison, G. Henry, W. Li, and P. Tang. Scientific computing on the Itanium™ processor. In *ACM/IEEE Conference on Supercomputing (Supercomputing '01)*, pages 41–41, 2001.

[226] P. M. Gruber and C. G. Lekkerkerker. *Geometry of Numbers*, volume 37 of *North-Holland Mathematical Library*. North-Holland Publishing Co., Amsterdam, second edition, 1987.

[227] A. Guntoro and M. Glesner. High-performance FPGA-based floating-point adder with three inputs. In *Field Programmable Logic and Applications*, pages 627–630, 2008.

[228] J. L. Gustafson. *The End of Error: Unum Computing*. Chapman & Hall/CRC Computational Science. Taylor & Francis, 2015.

[229] J. L. Gustafson and I. Yonemoto. Beating floating point at its own game: Posit arithmetic. In *Supercomputing Frontiers and Innovations*, pages 71–86, July 2017.

[230] O. Gustafsson, A. G. Dempster, K. Johansson, and M. D. Macleod. Simplified design of constant coefficient multipliers. *Circuits, Systems, and Signal Processing*, 25(2):225–251, 2006.

[231] R. W. Hamming. On the distribution of numbers. *The Bell System Technical Journal*, 49:1609–1625, 1970. Reprinted in [583].

[232] G. Hanrot, V. Lefèvre, P. Pélissier, P. Théveny, and P. Zimmermann. The MPFR library, version 3.1.5, 2016. Available at http://www.mpfr.org/mpfr-3.1.5/.

[233] G. Hanrot and D. Stehlé. Improved analysis of Kannan's shortest lattice vector algorithm. In *CRYPTO*, volume 4622 of *Lecture Notes in Computer Science*, pages 170–186, 2007.

[234] G. Hanrot and P. Zimmermann. A long note on Mulders' short product. *Journal of Symbolic Computation*, 37(3):391–401, 2004.

[235] E. R. Hansen and R. I. Greenberg. An interval Newton method. *Journal of Applied Mathematics and Computing*, 12:89–98, 1983.

[236] E. R. Hansen and W. Walster. *Global optimization using interval analysis*. MIT Press, Cambridge, MA, 2004.

[237] G. H. Hardy and E. M. Wright. *An Introduction to the Theory of Numbers*. Oxford University Press, London, 1979.

[238] J. Harrison. Floating-point verification in HOL light: The exponential function. Technical Report 428, University of Cambridge Computer Laboratory, 1997.

[239] J. Harrison. Verifying the accuracy of polynomial approximations in HOL. In *10th International Conference on Theorem Proving in Higher Order Logics (TPHOLs)*, volume 1275 of *Lecture Notes in Computer Science*, pages 137–152, Murray Hill, NJ, USA, 1997.

[240] J. Harrison. A machine-checked theory of floating point arithmetic. In *12th International Conference in Theorem Proving in Higher Order Logics (TPHOLs)*, volume 1690 of *Lecture Notes in Computer Science*, pages 113–130, Nice, France, September 1999.

[241] J. Harrison. Formal verification of floating-point trigonometric functions. In *3rd International Conference on Formal Methods in Computer-Aided Design (FMCAD)*, number 1954 in Lecture Notes in Computer Science, pages 217–233, Austin, TX, USA, 2000.

[242] J. Harrison. Formal verification of IA-64 division algorithms. In *13th International Conference on Theorem Proving in Higher Order Logics (TPHOLs)*, volume 1869 of *Lecture Notes in Computer Science*, pages 233–251, 2000.

[243] J. Harrison. Floating-point verification using theorem proving. In *Formal Methods for Hardware Verification, 6th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM 2006*, volume 3965 of *Lecture Notes in Computer Science*, pages 211–242, Bertinoro, Italy, 2006.

[244] J. Harrison. Verifying nonlinear real formulas via sums of squares. In *20th International Conference on Theorem Proving in Higher Order Logics (TPHOLs)*, volume 4732 of *Lecture Notes in Computer Science*, pages 102–118, Kaiserslautern, Germany, 2007.

[245] J. Harrison, T. Kubaska, S. Story, and P. T. P. Tang. The computation of transcendental functions on the IA-64 architecture. *Intel Technology Journal*, Q4, 1999. Available at http://developer.intel.com/technology/itj/archive/1999.htm.

[246] J. F. Hart, E. W. Cheney, C. L. Lawson, H. J. Maehly, C. K. Mesztenyi, J. R. Rice, H. G. Thacher, and C. Witzgall. *Computer Approximations*. John Wiley & Sons, New York, 1968.

[247] D. Harvey, J. van der Hoeven, and G. Lecerf. Even faster integer multiplication. *Journal of Complexity*, 36:1–30, 2016.

[248] J. R. Hauser. The SoftFloat and TestFloat Packages. Available at http://www.jhauser.us/arithmetic/.

[249] J. R. Hauser. Handling floating-point exceptions in numeric programs. *ACM Transactions on Programming Languages and Systems*, 18(2):139–174, 1996.

[250] B. Hayes. Third base. *American Scientist*, 89(6):490–494, 2001.

[251] C. He, G. Qin, M. Lu, and W. Zhao. Group-alignment based accurate floating-point summation on FPGAs. In *Engineering of Reconfigurable Systems and Algorithms*, pages 136–142, 2006.

[252] O. Heimlich. Interval arithmetic in GNU Octave. In *Summer Workshop on Interval Methods (SWIM)*, 2016.

[253] T. J. Hickey, Q. Ju, and M. H. van Emden. Interval arithmetic: From principles to implementation. *Journal of the ACM*, 48(5):1038–1068, 2001.

[254] Y. Hida, X. S. Li, and D. H. Bailey. Algorithms for quad-double precision floating-point arithmetic. In *15th IEEE Symposium on Computer Arithmetic (ARITH-15)*, pages 155–162, June 2001.

[255] Y. Hida, X. S. Li, and D. H. Bailey. C++/fortran-90 double-double and quad-double package, release 2.3.17, March 2012. Accessible electronically at http://crd-legacy.lbl.gov/~dhbailey/mpdist/.

[256] D. J. Higham and N. J. Higham. *MATLAB Guide*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, third edition, 2017.

[257] N. J. Higham. The accuracy of floating point summation. *SIAM Journal on Scientific Computing*, 14(4):783–799, 1993.

[258] N. J. Higham. *Accuracy and Stability of Numerical Algorithms*. SIAM, Philadelphia, PA, 2nd edition, 2002.

[259] A. Hirshfeld. *Eureka Man, The life and legacy of Archimedes*. Walker & Company, 2009.

[260] E. Hokenek, R. K. Montoye, and P. W. Cook. Second-generation RISC floating point with multiply-add fused. *IEEE Journal of Solid-State Circuits*, 25(5):1207–1213, 1990.

[261] J. E. Holm. *Floating-Point Arithmetic and Program Correctness Proofs*. Ph.D. thesis, Cornell University, 1980.

[262] M. S. Hrishikesh, D. Burger, N. P. Jouppi, S. W. Keckler, K. I. Farkas, and P. Shivakumar. The optimal logic depth per pipeline stage is 6 to 8 FO4 inverter delays. In *29th Annual International Symposium on Computer Architecture (ISCA)*, pages 14–24, 2002.

[263] S.-F. Hsiao, P.-H. Wu, C.-S. Wen, and P. K. Meher. Table size reduction methods for faithfully rounded lookup-table-based multiplierless function evaluation. *Transactions on Circuits and Systems II*, 62(5):466–470, 2015.

[264] T. E. Hull, T. F. Fairgrieve, and P. T. P. Tang. Implementing complex elementary functions using exception handling. *ACM Transactions on Mathematical Software*, 20(2):215–244, 1994.

[265] T. E. Hull, T. F. Fairgrieve, and P. T. P. Tang. Implementing the complex arcsine and arccosine functions using exception handling. *ACM Transactions on Mathematical Software*, 23(3):299–335, 1997.

[266] T. E. Hull and J. R. Swenson. Test of probabilistic models for propagation of round-off errors. *Communications of the ACM*, 9:108–113, 1966.

[267] IEEE Computer Society. *IEEE Standard for Floating-Point Arithmetic*. IEEE Standard 754-2008, August 2008. Available at http://ieeexplore.ieee.org/servlet/opac?punumber=4610933.

[268] IEEE Computer Society. *IEEE Standard for Interval Arithmetic*. IEEE Standard 1788-2015, June 2015.

[269] IEEE Computer Society. *IEEE Standard for Floating-Point Arithmetic (revision of IEEE Std 754-2008)*. 2017.

[270] G. Inoue. Leading one anticipator and floating point addition/subtraction apparatus, August 30 1994. US Patent 5,343,413.

[271] Intel Corporation. *Intrinsics Guide*. Available at https://software.intel.com/sites/landingpage/IntrinsicsGuide/. Accessed in June 2017.

[272] Intel Corporation. *Intel 64 and IA-32 Architectures Software Developer's Manual*, March 2017. Order Number: 325462-062US.

[273] International Organization for Standardization. *Programming Languages – C*. ISO/IEC Standard 9899:1999, Geneva, Switzerland, December 1999.

[274] International Organization for Standardization. *Information technology — Language independent arithmetic — Part 2: Elementary numerical functions*. ISO/IEC standard 10967-2, 2001.

[275] *Rationale for International Standard—Programming Languages—C*, 2003. Revision 5.10. Available at http://www.open-std.org/jtc1/sc22/wg14/www/C99RationaleV5.10.pdf.

[276] International Organization for Standardization. *Programming languages – Fortran – Part 1: Base language*. International Standard ISO/IEC 1539-1:2004, 2004.

[277] International Organization for Standardization. *ISO/IEC/IEEE Standard 60559:2011: Information technology – Microprocessor Systems – Floating-Point arithmetic*. International Electrotechnical Commission, 1st edition, 2011.

[278] International Organization for Standardization. *Programming Languages – C*. ISO/IEC Standard 9899:2011, Geneva, Switzerland, November 2011.

[279] International Organization for Standardization. *Floating-point extensions for C – Part 1: Binary floating-point arithmetic*. ISO/IEC Technical Specification 18661-1:2014, Geneva, Switzerland, July 2014.

[280] International Organization for Standardization. *Floating-point extensions for C – Part 2: Decimal floating-point arithmetic*. ISO/IEC Technical Specification 18661-2:2015, Geneva, Switzerland, May 2015.

[281] International Organization for Standardization. *Floating-point extensions for C – Part 3: Interchange and extended types*. ISO/IEC Technical Specification 18661-3:2015, Geneva, Switzerland, October 2015.

[282] International Organization for Standardization. *Floating-point extensions for C – Part 4: Supplementary functions*. ISO/IEC Technical Specification 18661-4:2015, Geneva, Switzerland, October 2015.

[283] International Organization for Standardization. *Floating-point extensions for C – Part 5: Supplementary attributes*. ISO/IEC Technical Specification 18661-5:2016, Geneva, Switzerland, August 2016.

[284] C. Iordache and P. T. P. Tang. An overview of floating-point support and math library on the Intel XScale architecture. In *16th IEEE Symposium on Computer Arithmetic (ARITH-16)*, pages 122–128, June 2003.

[285] C. Jacobi and C. Berg. Formal verification of the VAMP floating point unit. *Formal Methods in System Design*, 26(3):227–266, 2005.

[286] C. Jacobsen, A. Solovyev, and G. Gopalakrishnan. A parameterized floating-point formalization in HOL Light. In *7th and 8th International Workshops on Numerical Software Verification (NSV)*, volume 317 of *Electronic Notes in Theoretical Computer Science*, pages 101–107, 2015.

[287] C.-P. Jeannerod. A radix-independent error analysis of the Cornea-Harrison-Tang method. *ACM Transactions on Mathematical Software*, 42(3):19:1–19:20, 2016.

[288] C.-P. Jeannerod and J. Jourdan-Lu. Simultaneous floating-point sine and cosine for VLIW integer processors. In *23rd International Conference on Application-Specific Systems, Architectures and Processors (ASAP'12)*, pages 69–76, 2012.

[289] C.-P. Jeannerod, J. Jourdan-Lu, and C. Monat. Non-generic floating-point software support for embedded media processing. In *IEEE Symposium on Industrial Embedded Systems (SIES'12)*, pages 283–286, 2012.

[290] C.-P. Jeannerod, J. Jourdan-Lu, C. Monat, and G. Revy. How to square floats accurately and efficiently on the ST231 integer processor. In *20th IEEE Symposium on Computer Arithmetic (ARITH-20)*, pages 77–81, Tübingen, Germany, July 2011.

[291] C.-P. Jeannerod, H. Knochel, C. Monat, and G. Revy. Faster floating-point square root for integer processors. In *IEEE Symposium on Industrial Embedded Systems (SIES'07)*, pages 324–327, 2007.

[292] C.-P. Jeannerod, H. Knochel, C. Monat, and G. Revy. Computing floating-point square roots via bivariate polynomial evaluation. *IEEE Transactions on Computers*, 60(2):214–227, 2011.

[293] C.-P. Jeannerod, H. Knochel, C. Monat, G. Revy, and G. Villard. A new binary floating-point division algorithm and its software implementation on the ST231 processor. In *19th IEEE Symposium on Computer Arithmetic (ARITH-19)*, June 2009.

[294] C.-P. Jeannerod, P. Kornerup, N. Louvet, and J.-M. Muller. Error bounds on complex floating-point multiplication with an FMA. *Mathematics of Computation*, 86(304):881–898, 2017.

[295] C.-P. Jeannerod, N. Louvet, and J.-M. Muller. Further analysis of Kahan's algorithm for the accurate computation of 2×2 determinants. *Mathematics of Computation*, 82(284):2245–2264, 2013.

[296] C.-P. Jeannerod, N. Louvet, and J.-M. Muller. On the componentwise accuracy of complex floating-point division with an FMA. In *21st IEEE Symposium on Computer Arithmetic (ARITH-21)*, pages 83–90, April 2013.

[297] C.-P. Jeannerod, N. Louvet, J.-M. Muller, and A. Panhaleux. Midpoints and exact points of some algebraic functions in floating-point arithmetic. *IEEE Transactions on Computers*, 60(2), 2011.

[298] C.-P. Jeannerod, N. Louvet, J.-M. Muller, and A. Plet. Sharp error bounds for complex floating-point inversion. *Numerical Algorithms*, 73(3):735–760, 2016.

[299] C.-P. Jeannerod, J.-M. Muller, and A. Plet. The classical relative error bounds for computing $\sqrt{a^2+b^2}$ and $c/\sqrt{a^2+b^2}$ in binary floating-point arithmetic are asymptotically optimal. In *24th IEEE Symposium on Computer Arithmetic (ARITH-24)*, July 2017.

[300] C.-P. Jeannerod and G. Revy. FLIP 1.0: a fast floating-point library for integer processors. http://flip.gforge.inria.fr/, February 2009.

[301] C.-P. Jeannerod and G. Revy. Optimizing correctly-rounded reciprocal square roots for embedded VLIW cores. In *43rd Asilomar Conference on Signals, Systems, and Computers*, pages 731–735, November 2009.

[302] C.-P. Jeannerod and S. M. Rump. Improved error bounds for inner products in floating-point arithmetic. *SIAM Journal on Matrix Analysis and Applications*, 34(2):338–344, 2013.

[303] C.-P. Jeannerod and S. M. Rump. On relative errors of floating-point operations: optimal bounds and applications. *Mathematics of Computation*, 2016. To appear.

[304] F. Johansson. Arb: a C library for ball arithmetic. *ACM Communications in Computer Algebra*, 47(4):166–169, 2013.

[305] K. Johansson, O. Gustafsson, and L. Wanhammar. A detailed complexity model for multiple constant multiplication and an algorithm to minimize the complexity. In *Circuit Theory and Design*, pages 465–468, 2005.

[306] P. Johnstone and F. E. Petry. Rational number approximation in higher radix floating-point systems. *Computers & Mathematics with Applications*, 25(6):103–108, 1993.

[307] M. Joldeş. *Rigorous polynomial approximations and applications*. Ph.D. thesis, École Normale Supérieure de Lyon, Lyon, France, 2011.

[308] M. Joldeş, J.-M. Muller, and V. Popescu. Tight and rigourous error bounds for basic building blocks of double-word arithmetic. *ACM Transactions on Mathematical Software*, 44(2), 2017.

[309] M. Joldeş, J.-M. Muller, V. Popescu, and W. Tucker. CAMPARY: Cuda multiple precision arithmetic library and applications. In *5th International Congress on Mathematical Software (ICMS)*, July 2016.

[310] M. Joldeş, O. Marty, J.-M. Muller, and V. Popescu. Arithmetic algorithms for extended precision using floating-point expansions. *IEEE Transactions on Computers*, 65(4):1197–1210, 2016.

[311] J. Jourdan-Lu. *Custom floating-point arithmetic for integer processors: algorithms, implementation, and selection*. Ph.D. thesis, Université de Lyon - ÉNS de Lyon, France, November 2012.

[312] E. Kadric, P. Gurniak, and A. DeHon. Accurate parallel floating-point accumulation. In *21th IEEE Symposium on Computer Arithmetic (ARITH-21)*, pages 153–162, April 2013.

[313] W. Kahan. Pracniques: further remarks on reducing truncation errors. *Communications of the ACM*, 8(1):40, 1965.

[314] W. Kahan. A more complete interval arithmetic. Lecture notes for the University of Michigan, 1968.

[315] W. Kahan. Why do we need a floating-point standard? Technical report, Computer Science, UC Berkeley, 1981. Available at http://www.cs.berkeley.edu/~wkahan/ieee754status/why-ieee.pdf.

[316] W. Kahan. Minimizing q*m-n. Text accessible electronically at http://http.cs.berkeley.edu/~wkahan/. At the beginning of the file "nearpi.c", 1983.

[317] W. Kahan. Branch cuts for complex elementary functions. In *The State of the Art in Numerical Analysis*, pages 165–211, 1987.

[318] W. Kahan. Lecture notes on the status of IEEE-754. Available at http://www.cs.berkeley.edu/~wkahan/ieee754status/IEEE754.PDF, 1997.

[319] W. Kahan. Matlab's loss is nobody's gain. Technical report, Computer Science, UC Berkeley, 1998. Available at https://people.eecs.berkeley.edu/~wkahan/MxMulEps.pdf.

[320] W. Kahan. How futile are mindless assessments of roundoff in floating-point computation? Available at http://http.cs.berkeley.edu/~wkahan/Mindless.pdf, 2004.

[321] W. Kahan. A logarithm too clever by half. Available at http://http.cs.berkeley.edu/~wkahan/LOG10HAF.TXT, 2004.

[322] W. Kahan and J. Darcy. How Java's floating-point hurts everyone everywhere. Available at http://www.cs.berkeley.edu/~wkahan/JAVAhurt.pdf, 1998.

[323] R. Kaivola and K. Kohatsu. Proof engineering in the large: formal verification of Pentium®4 floating-point divider. *International Journal on Software Tools for Technology Transfer*, 4(3):323–334, 2003.

[324] E. Kaltofen. On the complexity of finding short vectors in integer lattices. In *European Computer Algebra Conference (EUROCAL)*, volume 162 of *Lecture Notes in Computer Science*, pages 236–244, 1983.

[325] R. Kannan. Improved algorithms for integer programming and related lattice problems. In *Annual ACM Symposium on Theory of Computing (STOC)*, pages 193–206, 1983.

[326] R. Kannan. Minkowski's convex body theorem and integer programming. *Mathematics of Operations Research*, 12(3):415–440, 1987.

[327] L. V. Kantorovich. On some new approaches to computational methods and to processing of observations. *Siberian Mathematical Journal*, 3(5):701–709, 1962. In Russian.

[328] A. Karatsuba and Y. Ofman. Multiplication of many-digital numbers by automatic computers. *Doklady Akad. Nauk SSSR*, 145:293–294, 1962. Translation in Physics-Doklady 7, 595–596, 1963.

[329] R. Karpinsky. PARANOIA: a floating-point benchmark. *BYTE*, 10(2), 1985.

[330] E. Kaucher. Interval analysis in the extended interval space IR. In *Fundamentals of Numerical Computation (Computer-Oriented Numerical Analysis)*, pages 33–49. Springer, 1980.

[331] R. B. Kearfott. *Rigorous global search: continuous problems*. Kluwer, Dordrecht, 1996.

[332] R. B. Kearfott, M. T. Nakao., A. Neumaier, S. M. Rump, S. P. Shary, and P. V. Hentenryck. Standardized notation in interval analysis. In *XIII Baikal International School-seminar "Optimization methods and their applications"*, volume 4, pages 106–113, 2005.

[333] B. W. Kernighan and D. M. Ritchie. *The C Programming Language*. Prentice-Hall, Englewood Cliffs, NJ, 1978.

[334] A. Y. Khinchin. *Continued Fractions*. Dover, New York, 1997.

[335] S. Khot. Hardness of approximating the shortest vector problem in lattices. In *FOCS*, pages 126–135, 2004.

[336] Khronos OpenCL Working Group. *The OpenCL SPIR-V Environment Specification, version 2.2*, May 2017. Available at https://www.khronos.org/registry/OpenCL/specs/opencl-2.2-environment.pdf.

[337] N. G. Kingsbury and P. J. W. Rayner. Digital filtering using logarithmic arithmetic. *Electronic Letters*, 7:56–58, 1971. Reprinted in [583].

[338] P. Kirchberger. *Ueber Tchebychefsche Annaeherungsmethoden*. Ph.D. thesis, Gottingen, 1902.

[339] A. Klein. A generalized Kahan-Babuška-summation-algorithm. *Computing*, 76:279–293, 2006.

[340] A. Knöfel. Fast hardware units for the computation of accurate dot products. In *10th IEEE Symposium on Computer Arithmetic (ARITH-10)*, pages 70–74, June 1991.

[341] S. Knowles. A family of adders. In *14th IEEE Symposium on Computer Arithmetic (ARITH-14)*, pages 30–34, April 1999.

[342] D. E. Knuth. *The Art of Computer Programming*, volume 2. Addison-Wesley, Reading, MA, 3rd edition, 1998.

[343] J. Koenig, D. Biancolin, J. Bachrach, and K. Asanovic. A hardware accelerator for computing an exact dot product. In *24th IEEE Symposium on Computer Arithmetic (ARITH-24)*, July 2017.

[344] P. M. Kogge and H. S. Stone. A parallel algorithm for the efficient solution of a general class of recurrence equations. *IEEE Transactions on Computers*, 100(8):786–793, 1973.

[345] I. Koren. *Computer Arithmetic Algorithms*. Prentice-Hall, Englewood Cliffs, NJ, 1993.

[346] P. Kornerup, C. Lauter, V. Lefèvre, N. Louvet, and J.-M. Muller. Computing correctly rounded integer powers in floating-point arithmetic. *ACM Transactions on Mathematical Software*, 37(1):4:1–4:23, 2010.

[347] P. Kornerup, V. Lefèvre, N. Louvet, and J.-M. Muller. On the computation of correctly rounded sums. *IEEE Transactions on Computers*, 61(3):289–298, 2012.

[348] P. Kornerup and D. W. Matula. Finite-precision rational arithmetic: an arithmetic unit. *IEEE Transactions on Computers*, C-32:378–388, 1983.

[349] P. Kornerup and D. W. Matula. Finite precision lexicographic continued fraction number systems. In *7th IEEE Symposium on Computer Arithmetic (ARITH-7)*, 1985. Reprinted in [584].

[350] P. Kornerup and J.-M. Muller. Choosing starting values for certain Newton–Raphson iterations. *Theoretical Computer Science*, 351(1):101–110, 2006.

[351] W. Krandick and J. R. Johnson. Efficient multiprecision floating point multiplication with optimal directional rounding. In *11th IEEE Symposium on Computer Arithmetic (ARITH-11)*, pages 228–233, June 1993.

[352] H. Kuki and W. J. Cody. A statistical study of the accuracy of floating point number systems. *Communications of the ACM*, 16(4):223–230, 1973.

[353] U. W. Kulisch. Circuitry for generating scalar products and sums of floating-point numbers with maximum accuracy. United States Patent 4622650, 1986.

[354] U. W. Kulisch. *Advanced Arithmetic for the Digital Computer: Design of Arithmetic Units*. Springer-Verlag, Berlin, 2002.

[355] U. W. Kulisch. *Computer Arithmetic and Validity: Theory, Implementation, and Applications*. de Gruyter, Berlin, 2008.

[356] M. Kumm, O. Gustafsson, M. Garrido, and P. Zipf. Optimal single constant multiplication using ternary adders. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2016.

[357] M. Kumm and P. Zipf. Pipelined compressor tree optimization using integer linear programming. In *Field Programmable Logic and Applications*, 2014.

[358] B. Lambov. Interval arithmetic using SSE-2. In *Reliable Implementation of Real Number Algorithms: Theory and Practice*, volume 5045 of *Lecture Notes in Computer Science*, pages 102–113, August 2008.

[359] T. Lang and J. D. Bruguera. Floating-point multiply-add-fused with reduced latency. *IEEE Transactions on Computers*, 53(8):988–1003, 2004.

[360] T. Lang and A. Nannarelli. A radix-10 combinational multiplier. In *40th Asilomar Conference on Signals, Systems, and Computers*, pages 313–317, October/November 2006.

[361] M. Lange and S. M. Rump. Error estimates for the summation of real numbers with application to floating-point summation. *BIT Numerical Mathematics*, 57(3):927–941, 2017.

[362] M. Lange and S. M. Rump. Faithfully rounded floating-point computations. Manuscript available at http://www.ti3.tu-harburg.de/rump/, 2017.

[363] M. Lange and S. M. Rump. Sharp estimates for perturbation errors in summations. Manuscript available at http://www.ti3.tu-harburg.de/rump/, 2017.

[364] M. Langhammer and B. Pasca. Faithful single-precision floating-point tangent for FPGAs. In *ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pages 39–42, 2013.

[365] M. Langhammer and B. Pasca. Floating-point DSP block architecture for FPGAs. In *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 117–125, 2015.

[366] M. Langhammer and B. Pasca. Single precision logarithm and exponential architectures for hard floating-point enabled FPGAs. *IEEE Transactions on Computers*, 66(12):2031–2043, 2017.

[367] P. Langlois. Automatic linear correction of rounding errors. *BIT Numerical Algorithms*, 41(3):515–539, 2001.

[368] P. Langlois and N. Louvet. How to ensure a faithful polynomial evaluation with the compensated Horner algorithm. In *18th IEEE Symposium on Computer Arithmetic (ARITH-18)*, pages 141–149, June 2007.

[369] J. Laskar, P. Robutel, F. Joutel, M. Gastineau, A. C. M. Correia, and B. Levrard. A long term numerical solution for the insolation quantities of the Earth. *Astronomy & Astrophysics*, 428:261–285, 2004.

[370] C. Q. Lauter. Basic building blocks for a triple-double intermediate format. Technical Report 2005-38, LIP, École Normale Supérieure de Lyon, September 2005.

[371] C. Q. Lauter. *Arrondi Correct de Fonctions Mathématiques*. Ph.D. thesis, École Normale Supérieure de Lyon, Lyon, France, October 2008. In French, available at http://www.ens-lyon.fr/LIP/Pub/Rapports/PhD/PhD2008/PhD2008-07.pdf.

[372] J. Le Maire, N. Brunie, F. de Dinechin, and J.-M. Muller. Computing floating-point logarithms with fixed-point operations. In *23rd IEEE Symposium of Computer Arithmetic (ARITH-23)*, July 2016.

[373] G. Lecerf and J. van der Hoeven. Evaluating Straight-Line Programs over Balls. In *23rd IEEE Symposium on Computer Arithmetic*, pages 142–149, 2016.

[374] B. Lee and N. Burgess. Parameterisable floating-point operations on FPGA. In *36th Asilomar Conference on Signals, Systems, and Computers*, volume 2, pages 1064–1068, November 2002.

[375] V. Lefèvre. Multiplication by an integer constant. Technical Report RR1999-06, Laboratoire de l'Informatique du Parallélisme, Lyon, France, 1999.

[376] V. Lefèvre. *Moyens Arithmétiques Pour un Calcul Fiable*. Ph.D. thesis, École Normale Supérieure de Lyon, Lyon, France, 2000.

[377] V. Lefèvre. The Euclidean division implemented with a floating-point division and a floor. Research report RR-5604, INRIA, June 2005.

[378] V. Lefèvre. New results on the distance between a segment and $\mathbb{Z}^2$. Application to the exact rounding. In *17th IEEE Symposium on Computer Arithmetic (ARITH-17)*, pages 68–75, June 2005.

[379] V. Lefèvre. Correctly rounded arbitrary-precision floating-point summation. *IEEE Transactions on Computers*, 66(12):2111–2124, 2017.

[380] V. Lefèvre and J.-M. Muller. Worst cases for correct rounding of the elementary functions in double precision. In *15th IEEE Symposium on Computer Arithmetic (ARITH-15)*, June 2001.

[381] V. Lefèvre and P. Zimmermann. Optimized binary64 and binary128 arithmetic with GNU MPFR. In *24th IEEE Symposium on Computer Arithmetic (ARITH-24)*, July 2017.

[382] A. K. Lenstra, H. W. Lenstra, Jr., and L. Lovász. Factoring polynomials with rational coefficients. *Mathematische Annalen*, 261:515–534, 1982.

[383] X. Leroy. Formal verification of a realistic compiler. *Communications of the ACM*, 52(7):107–115, 2009.

[384] X. Li, J. Demmel, D. H. Bailey, G. Henry, Y. Hida, J. Iskandar, W. Kahan, A. Kapur, M. Martin, T. Tung, and D. J. Yoo. Design, implementation and testing of extended and mixed precision BLAS. Technical Report 45991, Lawrence Berkeley National Laboratory, 2000. https://publications.lbl.gov/islandora/object/ir%3A115848.

[385] X. Li, J. Demmel, D. H. Bailey, G. Henry, Y. Hida, J. Iskandar, W. Kahan, A. Kapur, M. Martin, T. Tung, and D. J. Yoo. Design, implementation and testing of extended and mixed precision BLAS. *ACM Transactions on Mathematical Software*, 28(2):152–205, 2002.

[386] Y. Li and W. Chu. Implementation of single precision floating-point square root on FPGAs. In *FPGAs for Custom Computing Machines*, pages 56–65, 1997.

[387] C. Lichtenau, S. Carlough, and S. M. Mueller. Quad precision floating point on the IBM z13^{TM}. *23rd IEEE Symposium on Computer Arithmetic (ARITH-23)*, pages 87–94, 2016.

[388] G. Lienhart, A. Kugel, and R. Männer. Using floating-point arithmetic on FPGAs to accelerate scientific N-body simulations. In *FPGAs for Custom Computing Machines*, 2002.

[389] W. B. Ligon, S. McMillan, G. Monn, K. Schoonover, F. Stivers, and K. D. Underwood. A re-evaluation of the practicality of floating-point operations on FPGAs. In *FPGAs for Custom Computing Machines*, 1998.

[390] T. Lindholm, F. Yellin, G. Bracha, and A. Buckley. *The Java^{TM} Virtual Machine Specification, Java SE 8 edition*. Oracle, 2015.

[391] S. Linnainmaa. Software for doubled-precision floating-point computations. *ACM Transactions on Mathematical Software*, 7(3):272–283, 1981.

[392] J. Liu, M. Chang, and C.-K. Cheng. An iterative division algorithm for FPGAs. In *Field-Programmable Gate Arrays*, pages 83–89, 2006.

[393] E. Loh and G. W. Walster. Rump's example revisited. *Reliable Computing*, 8(3):245–248, 2002.

[394] F. Loitsch. Printing floating-point numbers quickly and accurately with integers. In *31st ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI '10)*, pages 233–243, 2010.

[395] A. R. Lopes and G. A. Constantinides. A fused hybrid floating-point and fixed-point dot-product for FPGAs. In *6th International Symposium on Reconfigurable Computing: Architectures, Tools and Applications (ARC)*, volume 5992 of *Lecture Notes in Computer Science*, pages 157–168, Bangkok, Thailand, March 2010.

[396] N. Louvet. *Algorithmes Compensés en Arithmétique Flottante: Précision, Validation, Performances*. Ph.D. thesis, Université de Perpignan, Perpignan, France, November 2007. In French.

[397] L. Lovász. *An Algorithmic Theory of Numbers, Graphs and Convexity*, volume 50 of *CBMS-NSF Regional Conference Series in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), 1986.

[398] Z. Luo and M. Martonosi. Accelerated pipelined integer and floating-point accumulations in configurable hardware with delayed addition techniques. *IEEE Transactions on Computers*, 49(3):208–218, 2000.

[399] D. Lutz. Fused multiply-add microarchitecture comprising separate early-normalizing multiply and add pipelines. In *20th IEEE Symposium on Computer Arithmetic (ARITH-20)*, pages 123–128, 2011.

[400] D. Lutz and N. Burgess. Overcoming double-rounding errors under IEEE 754-2008 using software. In *44th Asilomar Conference on Signals, Systems, and Computers*, pages 1399–1401, November 2010.

[401] N. Macon and A. Spitzbart. Inverses of Vandermonde matrices. *American Mathematical Monthly*, 65(2):95–100, 1958.

[402] K. Makino and M. Berz. Taylor models and other validated functional inclusion methods. *International Journal of Pure and Applied Mathematics*, 6(3):239–312, 2003.

[403] M. A. Malcolm. Algorithms to reveal properties of floating-point arithmetic. *Communications of the ACM*, 15(11):949–951, 1972.

[404] M. V. Manoukian and G. A. Constantinides. Accurate floating point arithmetic through hardware error-free transformations. In *Reconfigurable Computing: Architectures, Tools and Applications (ARC)*, pages 94–101, 2011.

[405] P. Markstein. Computation of elementary functions on the IBM RISC System/6000 processor. *IBM Journal of Research and Development*, 34(1):111–119, 1990.

[406] P. Markstein. *IA-64 and Elementary Functions: Speed and Precision*. Hewlett-Packard Professional Books. Prentice-Hall, Englewood Cliffs, NJ, 2000.

[407] M. Martel. Propagation of Roundoff Errors in Finite Precision Computations: A Semantics Approach. In *ESOP*, pages 194–208, 2002.

[408] É. Martin-Dorel and G. Melquiond. Proving tight bounds on univariate expressions with elementary functions in Coq. *Journal of Automated Reasoning*, 57(3):187–217, 2016.

[409] É. Martin-Dorel, G. Melquiond, and J.-M. Muller. Some issues related to double rounding. *BIT Numerical Mathematics*, 53(4):897–924, 2013.

[410] W. F. Mascarenhas. Floating point numbers are real numbers. Manuscript available at https://arxiv.org/abs/1605.09202, 2016.

[411] D. W. Matula. In-and-out conversions. *Communications of the ACM*, 11(1):47–50, 1968.

[412] D. W. Matula and P. Kornerup. Finite precision rational arithmetic: Slash number systems. *IEEE Transactions on Computers*, 34(1):3–18, 1985.

[413] W. M. McKeeman. Representation error for real numbers in binary computer arithmetic. *IEEE Transactions on Electronic Computers*, EC-16(5):682–683, 1967.

[414] P. Meher, J. Valls, T.-B. Juang, K. Sridharan, and K. Maharatna. 50 years of CORDIC: Algorithms, architectures, and applications. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 56(9):1893–1907, 2009.

[415] G. Melquiond. *De l'arithmétique d'intervalles à la certification de programmes*. Ph.D. thesis, École Normale Supérieure de Lyon, November 2006. In French, available at http://www.ens-lyon.fr/LIP/Pub/PhD2006.php.

[416] G. Melquiond. Proving bounds on real-valued functions with computations. In *4th International Joint Conference on Automated Reasoning (IJCAR)*, volume 5195 of *Lecture Notes in Artificial Intelligence*, pages 2–17, 2008.

[417] G. Melquiond. Floating-point arithmetic in the Coq system. *Information and Computation*, 216:14–23, 2012.

[418] V. Ménissier. *Arithmétique Exacte*. Ph.D. thesis, Université Pierre et Marie Curie, Paris, December 1994. In French.

[419] D. Micciancio. The hardness of the closest vector problem with preprocessing. *IEEE Transactions on Information Theory*, 47(3):1212–1215, 2001.

[420] D. Micciancio and S. Goldwasser. *Complexity of Lattice Problems: a Cryptographic Perspective*, volume 671 of *The Kluwer International Series in Engineering and Computer Science*. Kluwer Academic Publishers, Boston, MA, 2002.

[421] J. H. Min and E. E. Swartzlander. Fused floating-point two-term sum-of-squares unit. In *Application-Specific Systems, Architectures and Processors (ASAP)*, 2013.

[422] O. Møller. Quasi double-precision in floating-point addition. *BIT*, 5:37–50, 1965.

[423] D. Monniaux. The pitfalls of verifying floating-point computations. *ACM TOPLAS*, 30(3):1–41, 2008. A preliminary version is available at http://hal.archives-ouvertes.fr/hal-00128124.

[424] R. K. Montoye, E. Hokonek, and S. L. Runyan. Design of the IBM RISC System/6000 floating-point execution unit. *IBM Journal of Research and Development*, 34(1):59–70, 1990.

[425] P. Montuschi and P. M. Mezzalama. Survey of square rooting algorithms. *Computers and Digital Techniques, IEE Proceedings E.*, 137(1):31–40, 1990.

[426] J. S. Moore, T. Lynch, and M. Kaufmann. A mechanically checked proof of the correctness of the kernel of the AMD5K86 floating point division algorithm. *IEEE Transactions on Computers*, 47(9):913–926, 1998.

[427] R. E. Moore. *Interval arithmetic and automatic error analysis in digital computing*. Ph.D. thesis, Applied Math Statistics Lab., Report 25, Stanford, 1962.

[428] R. E. Moore. *Interval analysis*. Prentice Hall, 1966.

[429] R. E. Moore. *Methods and applications of interval analysis*. SIAM Studies in Applied Mathematics, Philadelphia, PA, 1979.

[430] R. E. Moore, R. B. Kearfott, and M. J. Cloud. *Introduction to Interval Analysis*. SIAM, Philadelphia, PA, 2009.

[431] S. K. Moore. Intel makes a big jump in computer math. *IEEE Spectrum*, 2008.

[432] R. Morris. Tapered floating point: A new floating-point representation. *IEEE Transactions on Computers*, 20(12):1578–1579, 1971.

[433] C. Mouilleron and G. Revy. Automatic generation of fast and certified code for polynomial evaluation. In *20th IEEE Symposium on Computer Arithmetic*, pages 233–242, 2011.

[434] S. Muchnik. *Advanced Compiler Design and Implementation*. Morgan Kaufmann, 1997.

[435] T. Mulders. On short multiplications and divisions. *Applicable Algebra in Engineering, Communication and Computing*, 11(1):69–88, 2000.

[436] J.-M. Muller. *Arithmétique des Ordinateurs*. Masson, Paris, 1989. In French.

[437] J.-M. Muller. Algorithmes de division pour microprocesseurs: illustration à l'aide du "bug" du pentium. *Technique et Science Informatiques*, 14(8), 1995.

[438] J.-M. Muller. A few results on table-based methods. *Reliable Computing*, 5(3):279–288, 1999.

[439] J.-M. Muller. On the definition of ulp(x). Technical Report 2005-09, LIP Laboratory, ENS Lyon, 2005.

[440] J.-M. Muller. Avoiding double roundings in scaled Newton-Raphson division. In *47th Asilomar Conference on Signals, Systems, and Computers*, pages 396–399, November 2013.

[441] J.-M. Muller. On the error of computing $ab + cd$ using Cornea, Harrison and Tang's method. *ACM Transactions on Mathematical Software*, 41(2):7:1–7:8, 2015.

[442] J.-M. Muller. *Elementary Functions, Algorithms and Implementation*. Birkhäuser Boston, MA, 3rd edition, 2016.

[443] J.-M. Muller, V. Popescu, and P. T. P. Tang. A new multiplication algorithm for extended precision using floating-point expansions. In *23nd IEEE Symposium on Computer Arithmetic (ARITH-23)*, pages 39–46, July 2016.

[444] J.-M. Muller, A. Scherbyna, and A. Tisserand. Semi-logarithmic number systems. *IEEE Transactions on Computers*, 47(2):145–151, 1998.

[445] M. Müller, C. Rüb, and W. Rülling. Exact accumulation of floating-point numbers. In *10th IEEE Symposium on Computer Arithmetic (ARITH-10)*, pages 64–69, June 1991.

[446] A. Munk-Nielsen and J.-M. Muller. Borrow-save adders for real and complex number systems. In *Real Numbers and Computers 2*, April 1996.

[447] C. Muñoz and A. Narkawicz. Formalization of Bernstein polynomials and applications to global optimization. *Journal of Automated Reasoning*, 51(2):151–196, 2013.

[448] A. Naini, A. Dhablania, W. James, and D. Das Sarma. 1-GHz HAL SPARC64 dual floating-point unit with RAS features. In *15th IEEE Symposium on Computer Arithmetic (ARITH-15)*, pages 174–183, June 2001.

[449] P. Nataraj and K. Kotecha. Higher Order Convergence for Multidimensional Functions with a New Taylor-Bernstein Form as Inclusion Function. *Reliable Computing*, 9:185–203, 2003.

[450] R. Nathan, B. Anthonio, S.-L. Lu, H. Naeimi, D. J. Sorin, and X. Sun. Recycled error bits: Energy-efficient architectural support for floating point accuracy. In *International Conference for High Performance Computing, Networking, Storage and Analysis (SC '14)*, pages 117–127, 2014.

[451] R. Nave. Implementation of transcendental functions on a numerics processor. *Microprocessing and Microprogramming*, 11:221–225, 1983.

[452] M. Nehmeier. libieeep1788: A C++ implementation of the IEEE interval standard P1788. In *Norbert Wiener in the 21st Century (21CW), 2014 IEEE Conference on*, pages 1–6, 2014.

[453] H. Neto and M. Véstias. Decimal multiplier on FPGA using embedded binary multipliers. In *Field Programmable Logic and Applications*, pages 197–202, 2008.

[454] A. Neumaier. Rundungsfehleranalyse einiger Verfahren zur Summation endlicher Summen. *ZAMM*, 54:39–51, 1974. In German.

[455] A. Neumaier. *Interval methods for systems of equations*. Cambridge University Press, Cambridge, UK, 1990.

[456] A. Neumaier. *Introduction to Numerical Analysis*. Cambridge University Press, 2001.

[457] A. Neumaier and D. Stehlé. Faster LLL-type reduction of lattice bases. In *ACM International Symposium on Symbolic and Algebraic Computation (ISSAC)*, pages 373–380, 2016.

[458] I. Newton. *Methodus Fluxionum et Serierum Infinitarum*. 1664–1671.

[459] K. C. Ng. Argument reduction for huge arguments: Good to the last bit. Technical report, SunPro, 1992.

[460] K. Ng. Method and apparatus for exact leading zero prediction for a floating-point adder, April 20 1993. US Patent 5,204,825.

[461] H. D. Nguyen. *Efficient algorithms for verified scientific computing : Numerical linear algebra using interval arithmetic*. Phd thesis, École Normale Supérieure de Lyon, January 2011.

[462] H. D. Nguyen, B. Pasca, and T. Preusser. FPGA-specific arithmetic optimizations of short-latency adders. In *Field Programmable Logic and Applications*, pages 232–237, 2011.

[463] P. Nguyen and D. Stehlé. An LLL algorithm with quadratic complexity. *SIAM Journal on Computing*, 39(3):874–903, 2009.

[464] K. R. Nichols, M. A. Moussa, and S. M. Areibi. Feasibility of floating-point arithmetic in FPGA based artificial neural networks. In *Computer Applications in Industry and Engineering (CAINE)*, pages 8–13, 2002.

[465] A. Novocin, D. Stehlé, and G. Villard. An LLL-reduction algorithm with quasi-linear time complexity: extended abstract. In *43rd ACM Symposium on Theory of Computing (STOC)*, pages 403–412, 2011.

[466] J. Oberg. Why the Mars probe went off course. *IEEE Spectrum*, 36(12):34–39, 1999.

[467] S. F. Oberman. Floating point division and square root algorithms and implementation in the AMD-K7 microprocessor. In *14th IEEE Symposium on Computer Arithmetic (ARITH-14)*, pages 106–115, April 1999.

[468] S. F. Oberman, H. Al-Twaijry, and M. J. Flynn. The SNAP project: design of floating-point arithmetic units. In *13th Symposium on Computer Arithmetic (ARITH-13)*, 1997.

[469] S. F. Oberman and M. J. Flynn. Division algorithms and implementations. *IEEE Transactions on Computers*, 46(8):833–854, 1997.

[470] S. F. Oberman and M. Y. Siu. A high-performance area-efficient multifunction interpolator. In *17th IEEE Symposium on Computer Arithmetic (ARITH-17)*, June 2005.

[471] T. Ogita, S. M. Rump, and S. Oishi. Accurate sum and dot product. *SIAM Journal on Scientific Computing*, 26(6):1955–1988, 2005.

[472] T. Ogita, S. M. Rump, and S. Oishi. Verified solutions of linear systems without directed rounding. Technical Report 2005-04, Advanced Research Institute for Science and Engineering, Waseda University, Tokyo, Japan, 2005.

[473] V. G. Oklobdzija. An algorithmic and novel design of a leading zero detector circuit: Comparison with logic synthesis. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 2, 1994.

[474] J. O'Leary, X. Zhao, R. Gerth, and C.-J. H. Seger. Formally verifying IEEE compliance of floating point hardware. *Intel Technology Journal*, 3(1), 1999.

[475] F. W. J. Olver and P. R. Turner. Implementation of level-index arithmetic using partial table look-up. In *8th IEEE Symposium on Computer Arithmetic (ARITH-8)*, May 1987.

[476] F. Ortiz, J. Humphrey, J. Durbano, and D. Prather. A study on the design of floating-point functions in FPGAs. In *Field Programmable Logic and Applications*, volume 2778 of *Lecture Notes in Computer Science*, pages 1131–1135, Lisbon, Portugal, September 2003.

[477] M. L. Overton. *Numerical Computing with IEEE Floating Point Arithmetic*. SIAM, Philadelphia, PA, 2001.

[478] K. Ozaki, F. Bünger, T. Ogita, S. Oishi, and S. M. Rump. Simple floating-point filters for the two-dimensional orientation problem. *BIT Numerical Mathematics*, 56(2):729–749, 2016.

[479] K. Ozaki, T. Ogita, F. Bünger, and S. Oishi. Accelerating interval matrix multiplication by mixed precision arithmetic. *Nonlinear Theory and its Applications, IEICE*, 6(3):364–376, 2015.

[480] A. Paidimarri, A. Cevrero, P. Brisk, and P. Ienne. Fpga implementation of a single-precision floating-point multiply-accumulator with single-cycle accumulation. In *17th IEEE Symposium on Field Programmable Custom Computing Machines*, 2009.

[481] A. Panhaleux. Génération d'itérations de type Newton-Raphson pour la division de deux flottants à l'aide d'un FMA. Master's thesis, École Normale Supérieure de Lyon, Lyon, France, 2008. In French.

[482] B. Parhami. On the complexity of table lookup for iterative division. *IEEE Transactions on Computers*, C-36(10):1233–1236, 1987.

[483] B. Parhami. *Computer Arithmetic: Algorithms and Hardware Designs*. Oxford University Press, 2000.

[484] M. Parks. Number-theoretic test generation for directed roundings. *IEEE Transactions on Computers*, 49(7):651–658, 2000.

[485] B. Pasca. Correctly rounded floating-point division for DSP-enabled FPGAs. In *Field Programmable Logic and Applications*, 2012.

[486] D. Patil, O. Azizi, M. Horowitz, R. Ho, and R. Ananthraman. Robust energy-efficient adder topologies. In *18th IEEE Symposium on Computer Arithmetic (ARITH-18)*, pages 29–37, June 2007.

[487] M. Payne and R. Hanek. Radian reduction for trigonometric functions. *SIGNUM Newsletter*, 18:19–24, 1983.

[488] C. Peikert. Limits on the hardness of lattice problems in l_p norms. *Computational Complexity*, 17(2):300–351, 2008.

[489] P. Pélissier and P. Zimmermann. The DPE library. Available at http://www.loria.fr/~zimmerma/free/dpe-1.4.tar.gz.

[490] M. Pichat. Correction d'une somme en arithmétique à virgule flottante. *Numerische Mathematik*, 19:400–406, 1972. In French.

[491] R. V. K. Pillai, D. Al-Khalili, and A. J. Al-Khalili. A low power approach to floating point adder design. In *International Conference on Computer Design*, 1997.

[492] J. A. Pineiro and J. D. Bruguera. High-speed double-precision computation of reciprocal, division, square root, and inverse square root. *IEEE Transactions on Computers*, 51(12):1377–1388, 2002.

[493] S. Pion. *De la Géométrie Algorithmique au Calcul Géométrique*. Ph.D. thesis, Université de Nice Sophia-Antipolis, France, November 1999. In French.

[494] V. Popescu. *Towards fast and certified multiple-precision libraries*. Ph.D. thesis, Université de Lyon, 2017. Available at https://hal.archives-ouvertes.fr/tel-01534090.

[495] D. M. Priest. Algorithms for arbitrary precision floating point arithmetic. In *10th IEEE Symposium on Computer Arithmetic (ARITH-10)*, pages 132–143, June 1991.

[496] D. M. Priest. *On Properties of Floating-Point Arithmetics: Numerical Stability and the Cost of Accurate Computations*. Ph.D. thesis, University of California at Berkeley, 1992.

[497] D. M. Priest. Efficient scaling for complex division. *ACM Transactions on Mathematical Software*, 30(4), 2004.

[498] C. Proust. Masters' writings and students' writings: School material in Mesopotamia. In Gueudet, Pepin, and Trouche, editors, *Mathematics curriculum material and teacher documentation: from textbooks to shared living resources*, pages 161–180. Springer, 2011.

[499] J. D. Pryce and G. F. Corliss. Interval arithmetic with containment sets. *Computing*, 78:251–276, 2006.

[500] S. Putot. *Static Analysis of Numerical Programs and Systems*. Habilitation, Université Paris-Sud, 2012.

[501] E. Quinnell, E. E. Swartzlander, and C. Lemonds. Floating-point fused multiply-add architectures. In *41st Asilomar Conference on Signals, Systems, and Computers*, pages 331–337, November 2007.

[502] S.-K. Raina. *FLIP: a Floating-point Library for Integer Processors*. Ph.D. thesis, École Normale Supérieure de Lyon, September 2006. Available at http://www.ens-lyon.fr/LIP/Pub/PhD2006.php.

[503] J. Ramos and A. Bohorquez. Two operand binary adders with threshold logic. *IEEE Transactions on Computers*, 48(12):1324–1337, 1999.

[504] B. Randell. From analytical engine to electronic digital computer: the contributions of Ludgate, Torres, and Bush. *IEEE Annals of the History of Computing*, 4(4):327–341, 1982.

[505] E. Remez. Sur un procédé convergent d'approximations successives pour déterminer les polynômes d'approximation. *C.R. Académie des Sciences, Paris*, 198:2063–2065, 1934. In French.

[506] N. Revol, K. Makino, and M. Berz. Taylor models and floating-point arithmetic: proof that arithmetic operations are validated in COSY. *Journal of Logic and Algebraic Programming*, 64:135–154, 2005.

[507] N. Revol and F. Rouillier. Motivations for an arbitrary precision interval arithmetic and the MPFI library. *Reliable Computing*, 11:1–16, 2005.

[508] N. Revol and P. Théveny. Numerical reproducibility and parallel computations: Issues for interval algorithms. *IEEE Transactions on Computers*, 63(8):1915–1924, 2014.

[509] G. Revy. *Implementation of binary floating-point arithmetic on embedded integer processors: polynomial evaluation-based algorithms and certified code generation*. Ph.D. thesis, Université de Lyon - ÉNS de Lyon, France, December 2009.

[510] T. J. Rivlin. *Chebyshev polynomials. From approximation theory to algebra*. John Wiley & Sons, New York, 2nd edition, 1990.

[511] J. E. Robertson. A new class of digital division methods. *IRE Transactions on Electronic Computers*, EC-7:218–222, 1958. Reprinted in [583].

[512] E. Roesler and B. Nelson. Novel optimizations for hardware floating-point units in a modern FPGA architecture. In *Field Programmable Logic and Applications*, volume 2438 of *Lecture Notes in Computer Science*, pages 637–646, 2002.

[513] R. Rojas. Konrad Zuse's legacy: the architecture of the Z1 and Z3. *IEEE Annals of the History of Computing*, 19(2):5–16, 1997.

[514] R. Rojas. The Z1: Architecture and algorithms of Konrad Zuse's first computer. Technical report, Freie Universität Berlin, June 2014. Available at https://arxiv.org/abs/1406.1886.

[515] R. Rojas, F. Darius, C. Göktekin, and G. Heyne. The reconstruction of Konrad Zuse's Z3. *IEEE Annals of the History of Computing*, 27(3):23–32, 2005.

[516] P. Roux. Innocuous double rounding of basic arithmetic operations. *Journal of Formalized Reasoning*, 7(1):131–142, 2014.

[517] S. M. Rump. Solving algebraic problems with high accuracy (Habilitationsschrift). In *A New Approach to Scientific Computation*, pages 51–120, 1983.

[518] S. M. Rump. Algorithms for verified inclusions: theory and practice. In *Reliability in Computing, Perspectives in Computing*, pages 109–126, 1988.

[519] S. M. Rump. Fast and parallel interval arithmetic. *BIT*, 39(3):534–554, 1999.

[520] S. M. Rump. INTLAB - INTerval LABoratory. In T. Csendes, editor, *Developments in Reliable Computing*, pages 77–104. Kluwer Academic Publishers, Dordrecht, 1999. http://www.ti3.tuhh.de/rump/.

[521] S. M. Rump. Ultimately fast accurate summation. *SIAM Journal on Scientific Computing*, 31(5):3466–3502, 2009.

[522] S. M. Rump. Error estimation of floating-point summation and dot product. *BIT Numerical Mathematics*, 52(1):201–220, 2012.

[523] S. M. Rump. Fast interval matrix multiplication. *Numerical Algorithms*, 1(61):1–34, 2012.

[524] S. M. Rump. Interval arithmetic over finitely many endpoints. *BIT Numerical Mathematics*, 52(4):1059–1075, 2012.

[525] S. M. Rump. Computable backward error bounds for basic algorithms in linear algebra. *Nonlinear Theory and its Applications, IEICE*, 6(3):360–363, 2015.

[526] S. M. Rump and H. Böhm. Least significant bit evaluation of arithmetic expressions in single-precision. *Computing*, 30:189–199, 1983.

[527] S. M. Rump, F. Bünger, and C.-P. Jeannerod. Improved error bounds for floating-point products and Horner's scheme. *BIT Numerical Mathematics*, 56(1):293–307, 2016.

[528] S. M. Rump and C.-P. Jeannerod. Improved backward error bounds for LU and Cholesky factorizations. *SIAM Journal on Matrix Analysis and Applications*, 35(2):684–698, 2014.

[529] S. M. Rump and M. Kashiwagi. Implementation and improvements of affine arithmetic. *Nonlinear Theory and its Applications, IEICE*, 6(3):341–359, 2015.

[530] S. M. Rump and M. Lange. On the definition of unit roundoff. *BIT Numerical Mathematics*, 56(1):309–317, 2016.

[531] S. M. Rump, T. Ogita, and S. Oishi. Accurate floating-point summation part I: Faithful rounding. *SIAM Journal on Scientific Computing*, 31(1):189–224, 2008.

[532] S. M. Rump, T. Ogita, and S. Oishi. Accurate floating-point summation part II: Sign, K-fold faithful and rounding to nearest. *SIAM Journal on Scientific Computing*, 31(2):1269–1302, 2008.

[533] S. M. Rump, P. Zimmermann, S. Boldo, and G. Melquiond. Computing predecessor and successor in rounding to nearest. *BIT Numerical Mathematics*, 49(2):419–431, 2009.

[534] D. M. Russinoff. A mechanically checked proof of IEEE compliance of a register-transfer-level specification of the AMD-K7 floating-point multiplication, division, and square root instructions. *LMS Journal of Computation and Mathematics*, 1:148–200, 1998.

[535] D. M. Russinoff. A mechanically checked proof of correctness of the AMD K5 floating point square root microcode. *Formal Methods in System Design*, 14(1):75–125, 1999.

[536] D. M. Russinoff. A case study in formal verification of register-transfer logic with ACL2: The floating point adder of the AMD Athlon processor. *Lecture Notes in Computer Science*, 1954:3–36, 2000.

[537] E. Salamin. Computation of π using arithmetic-geometric mean. *Mathematics of Computation*, 30:565–570, 1976.

[538] H. H. Saleh and E. E. Swartzlander. A floating-point fused dot-product unit. In *International Conference on Computer Design (ICCD)*, pages 426–431, 2008.

[539] J.-L. Sanchez, A. Jimeno, H. Mora, J. Mora, and F. Pujol. A CORDIC-based architecture for high performance decimal calculation. In *IEEE International Symposium on Industrial Electronics*, pages 1951–1956, June 2007.

[540] J. Sawada and R. Gamboa. Mechanical verification of a square root algorithm using Taylor's theorem. In *4th International Conference on Formal Methods in Computer-Aided Design (FMCAD)*, volume 2517 of *Lecture Notes in Computer Science*, pages 274–291, 2002.

[541] H. Schichl and A. Neumaier. Interval analysis on directed acyclic graphs for global optimization. *Journal of Global Optimization*, 33:541–562, 2005.

[542] H. Schmid and A. Bogacki. Use decimal CORDIC for generation of many transcendental functions. *EDN*, pages 64–73, 1973.

[543] M. M. Schmookler and K. J. Nowka. Leading zero anticipation and detection – a comparison of methods. In *15th IEEE Symposium on Computer Arithmetic (ARITH-15)*, pages 7–12, June 2001.

[544] C. P. Schnorr. A more efficient algorithm for lattice basis reduction. *J. Algorithms*, 9(1):47–62, 1988.

[545] A. Schönhage. Schnelle Berechnung von Kettenbruchentwicklungen. *Acta Informatica*, 1:139–144, 1971. In German.

[546] A. Schönhage, A. F. W. Grotefeld, and E. Vetter. *Fast algorithms: a Multitape Turing Machine Implementation*. Bibliographisches Institut, Mannheim, 1994.

[547] A. Schönhage and V. Strassen. Schnelle Multiplikation grosser Zahlen. *Computing*, 7:281–292, 1971. In German.

[548] E. M. Schwarz, J. S. Kapernick, and M. F. Cowlishaw. Decimal floating-point support on the IBM system Z10 processor. *IBM Journal of Research and Development*, 53(1):36–45, 2009.

[549] E. M. Schwarz, M. Schmookler, and S. D. Trong. FPU implementations with denormalized numbers. *IEEE Transactions on Computers*, 54(7):825–836, 2005.

[550] P.-M. Seidel. Multiple path IEEE floating-point fused multiply-add. In *46th International Midwest Symposium on Circuits and Systems*, pages 1359–1362, 2003.

[551] P.-M. Seidel and G. Even. How many logic levels does floating-point addition require. In *International Conference on Computer Design*, pages 142–149, 1998.

[552] P.-M. Seidel and G. Even. On the design of fast IEEE floating-point adders. In *15th IEEE Symposium on Computer Arithmetic (ARITH-15)*, pages 184–194, June 2001.

[553] C. Severance. IEEE 754: An interview with William Kahan. *Computer*, 31(3):114–115, 1998.

[554] A. M. Shams and M. A. Bayoumi. A novel high-performance CMOS 1-bit full-adder cell. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 47(5), 2000.

[555] J. R. Shewchuk. Adaptive precision floating-point arithmetic and fast robust geometric predicates. *Discrete Computational Geometry*, 18:305–363, 1997.

[556] N. Shirazi, A. Walters, and P. Athanas. Quantitative analysis of floating point arithmetic on FPGA based custom computing machine. In *FPGAs for Custom Computing Machines*, pages 155–162, 1995.

[557] V. Shoup. NTL, a library for doing number theory, version 10.5.0. http://shoup.net/ntl/, 2017.

[558] T. Simpson. *Essays on several curious and useful subjects in speculative and mix'd mathematicks, illustrated by a variety of examples*. London, 1740.

[559] D. P. Singh, B. Pasca, and T. S. Czajkowski. High-level design tools for floating point FPGAs. In *ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, Monterey, CA, USA, February 22–24, 2015*, pages 9–12, 2015.

[560] R. A. Smith. A continued-fraction analysis of trigonometric argument reduction. *IEEE Transactions on Computers*, 44(11):1348–1351, 1995.

[561] R. L. Smith. Algorithm 116: Complex division. *Communications of the ACM*, 5(8):435, 1962.

[562] A. Solovyev, C. Jacobsen, Z. Rakamarić, and G. Gopalakrishnan. *Rigorous Estimation of Floating-Point Round-off Errors with Symbolic Taylor Expansions*, pages 532–550. Springer International Publishing, 2015.

[563] A. Solovyev, C. Jacobsen, Z. Rakamarić, and G. Gopalakrishnan. Rigorous estimation of floating-point round-off errors with symbolic Taylor expansions. In *20th International Symposium on Formal Methods (FM)*, June 2015.

[564] E. Sprangle and D. Carmean. Increasing processor performance by implementing deeper pipelines. In *29th Annual International Symposium on Computer Architecture (ISCA)*, pages 25–34, 2002.

[565] H. M. Stark. *An Introduction to Number Theory*. MIT Press, Cambridge, MA, 1981.

[566] G. L. Steele, Jr. and J. L. White. How to print floating-point numbers accurately. *ACM SIGPLAN Notices*, 25(6):112–126, 1990.

[567] G. L. Steele, Jr. and J. L. White. Retrospective: how to print floating-point numbers accurately. *ACM SIGPLAN Notices*, 39(4):372–389, 2004.

[568] D. Stehlé. *Algorithmique de la Réduction de Réseaux et Application à la Recherche de Pires Cas pour l'Arrondi de Fonctions Mathématiques*. Ph.D. thesis, Université Henri Poincaré – Nancy 1, France, December 2005.

[569] D. Stehlé. On the randomness of bits generated by sufficiently smooth functions. In *7th Algorithmic Number Theory Symposium (ANTS)*, volume 4078 of *Lecture Notes in Computer Science*, pages 257–274, 2006.

[570] D. Stehlé, V. Lefèvre, and P. Zimmermann. Worst cases and lattice reduction. In *16th IEEE Symposium on Computer Arithmetic (ARITH-16)*, pages 142–147, June 2003.

[571] D. Stehlé, V. Lefèvre, and P. Zimmermann. Searching worst cases of a one-variable function. *IEEE Transactions on Computers*, 54(3):340–346, 2005.

[572] P. H. Sterbenz. *Floating-Point Computation*. Prentice-Hall, Englewood Cliffs, NJ, 1974.

[573] G. W. Stewart. A note on complex division. *ACM Transactions on Mathematical Software*, 11(3):238–241, 1985.

[574] J. E. Stine and M. J. Schulte. The symmetric table addition method for accurate function approximation. *Journal of VLSI Signal Processing*, 21:167–177, 1999.

[575] J. Stolfi and L. de Figueiredo. *Self-Validated Numerical Methods and Applications*. Monograph for 21st Brazilian Mathematics Colloquium, Rio de Janeiro, Brazil, 1997.

[576] A. Storjohann. Faster algorithms for integer lattice basis reduction. Technical report, ETH Zürich, 1996.

[577] V. Strassen. Gaussian elimination is not optimal. *Numerische Mathematik*, 13:354–356, 1969.

[578] Sun. *Numerical Computation Guide – Sun^{TM} Studio 11*, 2005. Available at http://docs.sun.com/source/819-3693/.

[579] Sun Microsystems. Interval arithmetic in high performance technical computing. Technical report, 2002.

[580] T. Sunaga. Geometry of Numerals. Master's thesis, U. Tokyo, Japan, 1956.

[581] D. A. Sunderland, R. A. Strauch, S. W. Wharfield, H. T. Peterson, and C. R. Cole. CMOS/SOS frequency synthesizer LSI circuit for spread spectrum communications. *IEEE Journal of Solid State Circuits*, SC-19(4):497–506, 1984.

[582] A. Svoboda. Adder with distributed control. *IEEE Transactions on Computers*, C-19(8), 1970. Reprinted in [583].

[583] E. E. Swartzlander. *Computer Arithmetic*, volume 1. World Scientific Publishing Co., Singapore, 2015.

[584] E. E. Swartzlander. *Computer Arithmetic*, volume 2. World Scientific Publishing Co., Singapore, 2015.

[585] E. E. Swartzlander and A. G. Alexpoulos. The sign-logarithm number system. *IEEE Transactions on Computers*, 1975. Reprinted in [583].

[586] N. Takagi. A hardware algorithm for computing the reciprocal square root. In *15th IEEE Symposium on Computer Arithmetic (ARITH-15)*, pages 94–100, June 2001.

[587] N. Takagi and S. Kuwahara. A VLSI algorithm for computing the Euclidean norm of a 3D vector. *IEEE Transactions on Computers*, 49(10):1074–1082, 2000.

[588] P. T. P. Tang. Table-driven implementation of the exponential function in IEEE floating-point arithmetic. *ACM Transactions on Mathematical Software*, 15(2):144–157, 1989.

[589] P. T. P. Tang. Table-driven implementation of the logarithm function in IEEE floating-point arithmetic. *ACM Transactions on Mathematical Software*, 16(4):378–400, 1990.

[590] P. T. P. Tang. Table lookup algorithms for elementary functions and their error analysis. In *10th IEEE Symposium on Computer Arithmetic (ARITH-10)*, pages 232–236, June 1991.

[591] P. T. P. Tang. Table-driven implementation of the expm1 function in IEEE floating-point arithmetic. *ACM Transactions on Mathematical Software*, 18(2):211–222, 1992.

[592] Y. Tao, G. Deyuan, R. Xianglong, H. Limin, F. Xiaoya, and Y. Lei. A novel floating-point function unit combining MAF and 3-input adder. In *Signal Processing, Communication and Computing (ICSPCC)*, 2012.

[593] Y. Tao, G. Deyuan, and F. Xiaoya. A multi-path fused add-subtract unit for digital signal processing. In *Computer Science and Automation Engineering (CSAE)*, 2012.

[594] Y. Tao, G. Deyuan, F. Xiaoya, and J. Nurmi. Correctly rounded architectures for floating-point multi-operand addition and dot-product computation. In *Application-Specific Systems, Architectures and Processors (ASAP)*, 2013.

[595] Y. Tao, G. Deyuan, F. Xiaoya, and R. Xianglong. Three-operand floating-point adder. In *12th International Conference on Computer and Information Technology*, pages 192–196, 2012.

[596] M. B. Taylor. Is dark silicon useful? Harnessing the four horsemen of the coming dark silicon apocalypse. In *49th Design Automation Conference*, pages 1131–1136, 2012.

[597] A. F. Tenca. Multi-operand floating-point addition. In *19th IEEE Symposium on Computer Arithmetic (ARITH-19)*, pages 161–168, 2009.

[598] T. Teufel and M. Baesler. FPGA implementation of a decimal floating-point accurate scalar product unit with a parallel fixed-point multiplier. In *Reconfigurable Computing and FPGAs*, pages 6–11, 2009.

[599] The FPLLL development team. fplll, a lattice reduction library. Available at https://github.com/fplll/fplll, 2016.

[600] P. Théveny. *Numerical Quality and High Performance in Interval Linear Algebra on Multi-Core Processors*. Ph.D. thesis, Université de Lyon, École Normale Supérieure de Lyon, 2014.

[601] D. B. Thomas. A general-purpose method for faithfully rounded floating-point function approximation in FPGAs. In *22nd IEEE Symposium on Computer Arithmetic (ARITH-22)*, pages 42–49, 2015.

[602] K. D. Tocher. Techniques of multiplication and division for automatic binary computers. *Quarterly Journal of Mechanics and Applied Mathematics*, 11(3):364–384, 1958.

[603] A. L. Toom. The complexity of a scheme of functional elements realizing the multiplication of integers. *Soviet Mathematics Doklady*, 3:714–716, 1963.

[604] S. Torres. *Tools for the Design of Reliable and Efficient Function Evaluation Libraries*. Ph.D. thesis, Université de Lyon, 2016. Available at https://tel.archives-ouvertes.fr/tel-01396907.

[605] W. J. Townsend, E. E. Swartzlander, Jr., and J. A. Abraham. A comparison of Dadda and Wallace multiplier delays. In *SPIE's 48th Annual Meeting on Optical Science and Technology*, pages 552–560, 2003.

[606] L. Trefethen. *Approximation Theory and Approximation Practice*. SIAM, 2013.

[607] S. D. Trong, M. Schmookler, E. M. Schwarz, and M. Kroener. P6 binary floating-point unit. In *18th IEEE Symposium on Computer Arithmetic (ARITH-18)*, pages 77–86, 2007.

[608] W. Tucker. The Lorenz attractor exists. *Comptes Rendus de l'Académie des Sciences-Series I-Mathematics*, 328(12):1197–1202, 1999.

[609] W. Tucker. *Validated Numerics – A Short Introduction to Rigorous Computations*. Princeton University Press, 2011.

[610] A. Tyagi. A reduced-area scheme for carry-select adders. *IEEE Transactions on Computers*, 42(10):1163–1170, 1993.

[611] H. F. Ugurdag, A. Bayram, V. E. Levent, and S. Gören. Efficient combinational circuits for division by small integer constants. In *23rd IEEE Symposium on Computer Arithmetic (ARITH-23)*, pages 1–7, July 2016.

[612] P. van Emde Boas. Another NP-complete problem and the complexity of computing short vectors in a lattice. Technical Report 81-04, Mathematisch Instituut, University of Amsterdam, 1981.

[613] A. Vázquez. *High-Performance Decimal Floating-Point Units*. Ph.D. thesis, Universidade de Santiago de Compostela, 2009.

[614] A. Vázquez, E. Antelo, and P. Montuschi. A new family of high performance parallel decimal multipliers. In *18th IEEE Symposium on Computer Arithmetic (ARITH-18)*, pages 195–204, 2007.

[615] L. Veidinger. On the numerical determination of the best approximations in the Chebyshev sense. *Numerische Mathematik*, 2:99–105, 1960.

[616] G. W. Veltkamp. ALGOL procedures voor het berekenen van een inwendig product in dubbele precisie. Technical Report 22, RC-Informatie, Technishe Hogeschool Eindhoven, 1968.

[617] G. W. Veltkamp. ALGOL procedures voor het rekenen in dubbele lengte. Technical Report 21, RC-Informatie, Technishe Hogeschool Eindhoven, 1969.

[618] D. Villeger and V. G. Oklobdzija. Evaluation of Booth encoding techniques for parallel multiplier implementation. *Electronics Letters*, 29(23):2016–2017, 1993.

[619] J. E. Volder. The CORDIC trigonometric computing technique. *IRE Transactions on Electronic Computers*, EC-8(3):330–334, 1959.

[620] J. E. Volder. The birth of CORDIC. *Journal of VLSI Signal Processing Systems*, 25(2):101–105, 2000.

[621] Y. Voronenko and M. Püschel. Multiplierless multiple constant multiplication. *ACM Trans. Algorithms*, 3(2), 2007.

[622] J. E. Vuillemin. Exact real computer arithmetic with continued fractions. *IEEE Transactions on Computers*, 39(8), 1990.

[623] J. E. Vuillemin. On circuits and numbers. *IEEE Transactions on Computers*, 43(8):868–879, 1994.

[624] C. S. Wallace. A suggestion for a fast multiplier. *IEEE Transactions on Electronic Computers*, pages 14–17, 1964. Reprinted in [583].

[625] J. S. Walther. A unified algorithm for elementary functions. In *Joint Computer Conference Proceedings*, 1971. Reprinted in [583].

[626] J. S. Walther. The story of unified CORDIC. *Journal of VLSI Signal Processing Systems*, 25(2):107–112, 2000.

[627] L.-K. Wang and M. J. Schulte. Decimal floating-point division using Newton–Raphson iteration. In *Application-Specific Systems, Architectures and Processors*, pages 84–95, 2004.

[628] L.-K. Wang, M. J. Schulte, J. D. Thompson, and N. Jairam. Hardware designs for decimal floating-point addition and related operations. *IEEE Transactions on Computers*, 58(2):322–335, 2009.

[629] X. Wang, S. Braganza, and M. Leeser. Advanced components in the variable precision floating-point library. In *Field-Programmable Custom Computing Machines*, pages 249–258, 2006.

[630] W. H. Ware, editor. Soviet computer technology—1959. *Communications of the ACM*, 3(3):131–166, 1960.

[631] N. Whitehead and A. Fit-Florea. Precision & performance: Floating point and IEEE 754 compliance for NVIDIA GPUs. Technical report, NVIDIA, 2011. Available at http://developer.download.nvidia.com/devzone/devcenter/cuda/files/NVIDIA-CUDA-Floating-Point.pdf.

[632] Wikipedia. Slide rule — Wikipedia, The Free Encyclopedia, 2017. [Online; accessed 20-November-2017].

[633] Wikipedia. Square root of 2 — Wikipedia, The Free Encyclopedia, 2017. [Online; accessed 20-November-2017].

[634] J. H. Wilkinson. *Rounding Errors in Algebraic Processes*. Prentice-Hall, Englewood Cliffs, NJ, 1963.

[635] J. H. Wilkinson. Some Comments from a Numerical Analyst. *Journal of the ACM*, 18(2):137–147, 1971.

[636] M. J. Wirthlin. Constant coefficient multiplication using look-up tables. *VLSI Signal Processing*, 36(1):7–15, 2004.

[637] W. F. Wong and E. Goto. Fast hardware-based algorithms for elementary function computations using rectangular multipliers. *IEEE Transactions on Computers*, 43(3):278–294, 1994.

[638] S. Xing and W. Yu. FPGA adders: Performance evaluation and optimal design. *IEEE Design & Test of Computers*, 15:24–29, 1998.

[639] T. J. Ypma. Historical development of the Newton-Raphson method. *SIAM Rev.*, 37(4):531–551, 1995.

[640] X. Y. Yu, Y.-H. Chan, B. Curran, E. Schwarz, M. Kelly, and B. Fleischer. A 5GHz+ 128-bit binary floating-point adder for the POWER6 processor. In *European Solid-State Circuits Conference*, pages 166–169, 2006.

[641] N. Zhuang and H. Wu. A new design of the CMOS full adder. *IEEE Journal on Solid-State Circuits*, 27:840–844, 1992.

[642] L. Zhuo and V. K. Prasanna. Scalable and modular algorithms for floating-point matrix multiplication on FPGAs. In *18th International Parallel and Distributed Processing Symposium (IPDPS)*, April 2004.

[643] L. Zhuo and V. K. Prasanna. High performance linear algebra operations on reconfigurable systems. In *Supercomputing*, 2005.

[644] G. Zielke and V. Drygalla. Genaue Loesung Linearer Gleichungssysteme. *GAMM Mitteilungen der Gesellschaft für Angewandte Mathematik und Mechanik*, 26:7–107, 2003.

[645] R. Zimmermann. *Binary Adder Architectures for Cell-Based VLSI and Their Synthesis*. Ph.D. thesis, Swiss Federal Institute of Technology, Zurich, 1997.

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

[647] R. Zumkeller. Formal global optimisation with Taylor models. In *3th International Joint Conference on Automated Reasoning (IJCAR)*, volume 4130 of *Lecture Notes in Computer Science*, pages 408–422, August 2006.

[648] D. Zuras. More on squaring and multiplying large integers. *IEEE Transactions on Computers*, 43(8):899–908, 1994.

[649] V. Zyuban, D. Brooks, V. Srinivasan, M. Gschwind, P. Bose, P. N. Strenski, and P. G. Emma. Integrated analysis of power and performance for pipelined microprocessors. *IEEE Transactions on Computers*, 53(8):1004–1016, 2004.

Index

J.-M. Muller et al., *Handbook of Floating-Point Arithmetic*,
https://doi.org/10.1007/978-3-319-76526-6