Useful applications of correctly-rounded operators of the form ab + cd + e

Tom Hubrecht, Claude-Pierre Jeannerod, Jean-Michel Muller

Useful applications of correctly-rounded operators of the form $ab + cd + e$

Tom Hubrecht*, Claude-Pierre Jeannerod†, Jean-Michel Muller‡

* Département d'informatique de l'ENS, École Normale Supérieure, CNRS, PSL University, F-75005 Paris, France
† Inria, Univ. Lyon, Laboratoire LIP (UMR 5668, CNRS, ENS de Lyon, Inria, UCBL), F-69342 Lyon, France
‡ CNRS, Univ. Lyon, Laboratoire LIP (UMR 5668, CNRS, ENS de Lyon, Inria, UCBL), F-69342 Lyon, France

***Abstract*—We show that the availability of fused arithmetic operators that evaluate expressions of the form $ab + cd$ (FD2 instruction) or $ab + cd + e$ (FD2A instruction) in floating-point arithmetic with one final rounding only would significantly facilitate many calculations that are hard to perform with high accuracy at small cost using only the traditional operations $+$, $-$, $\times$, $\div$, $\sqrt{}$, and fused multiply-add (FMA).**

***Index Terms*—Floating-point arithmetic, correct rounding, fused operators, dot-product instructions, complex arithmetic, double-word arithmetic, rounding error analysis.**

I. Introduction

A. Fused dot-product operators

The recent years have seen the emergence of new arithmetic instructions, that allow to perform *fused* "exact" dot-product (or dot-product-add) calculations for low-dimensional vectors: the arithmetic operations involved are fused, with one final rounding only. Most frequently, these instructions correspond to some types of dot products, such as $ab + cd$ or $ab + cd + e$ or longer [1], [2], [3], [4]. Typical applications of such instructions are the implementation of ML algorithms, but the FFT is also a target [5]. The predecessor of these instructions is the fused multiply-add (FMA), which evaluates expressions of the form $ab + c$ with only one final rounding. The first widely available processor with an FMA was the IBM POWER [6]. In addition to speeding up (and generally making more accurate) the computation of dot products and the evaluation of polynomials, the availability of an FMA instruction allows efficient software implementation of correctly rounded division and square root [7], [8], and makes it easy to compute the error of a floating-point multiplication, paving the way for efficient double-word multiplication and better complex arithmetic [9]. The goal of this paper is to show that these new fused instructions can bring to the field of numerical computing a progress comparable to the progress brought by the FMA 30 years ago.

So far, the fused exact dot-product instructions presented in the literature are mainly targeted at low-precision arithmetic. Furthermore, they frequently use different formats at the same time: typically, the products are accumulated in a format larger than the format of the input operands. Brunie [2] considers 16-bit inputs and an 80-bit accumulator, Bertaccini et al. [3] present such a sum-of-dot-product operator for NN training and inference, which takes four inputs a, b, c, d in a w-bit format (with $w = 8$ or 16) and one input e in a $2w$-bit format, and outputs $ab + cd + e$ rounded to the $2w$-bit format. Desrentes et al. [4] investigate the general case of $2n + 1$ input operands with $n \geq 2$. Kalray's MPPA architectures offer an FDMDA[1] (fused dual-multiply dual-add) instruction, that computes $ab+cd+e$ correctly rounded in binary32 arithmetic.

Lauter [10] describes software implementations of the fused, correctly-rounded operations $a + b + c$ and $ab + cd$, and lists a few applications: accurate complex multiplication, faster compensated algorithms, simplification of the last rounding step in correctly rounded elementary functions, and accurate computation of discriminants of quadratic polynomials.

Several other fused sum or dot-product operators have been introduced in the literature, especially for ML applications; see for example [11], [12], [13]. They are of little use for our study, either because they do not provide correct rounding or because the way they round the results is not fully specified.

B. Our contributions

In this paper, we consider only homogeneous operators (that is, operators whose input and output operands are all represented in the same format) that evaluate expressions of the form $ab+cd$ or $ab+cd+e$ with correct rounding to nearest. We show that these correctly-rounded fused operators allow for simple and accurate implementation of *complex arithmetic* (section II); that they simplify the evaluation of some rounding errors, either exactly (*error-free transforms*) or with correct rounding, paving the way for efficient compensated algorithms (section III); that they allow for more accurate computations of *products of three or four numbers*, with an application to the accurate computation of integer powers (section IV); that they simplify the evaluation of the sign of the discriminant of some *quadratic and cubic equations* (section V) as well as the implementation of *correctly-rounded multiplication by constants* such as π (section VI); that they allow for implementing *sums and inner products* with fewer instructions and smaller error bounds than the classical algorithms (section VII); and that they allow for *evaluating polynomials* as accurately as the classical and compensated FMA-based Horner schemes but with significantly fewer instructions (section VIII). Finally, we show that such operators allow for simple and efficient implementations of *double-word arithmetic* (section IX).

For each of these eight application domains, the possible gains are quantified precisely and systematically via detailed

[1] See https://patents.google.com/patent/US20220222073A1/en

operation counts and/or the derivation of rigorous, worst-case accuracy bounds (several of them being shown to be tight).

C. Notation and background

We assume radix-2, precision-p, floating-point (FP) arithmetic, with an unbounded exponent range (which means that the algorithms and properties presented below apply to an arithmetic conforming to the IEEE 754 standard [14] as long as underflows and overflows do not occur).

We write $\mathbb{F}$ to denote the set of FP numbers. We also assume that the rounding function, denoted RN, is round to nearest with ties to even, which is the default in IEEE 754 arithmetic. The associated unit roundoff is then $u = 2^{-p}$.

Given $a, b, c, d, e \in \mathbb{F}$, we assume that the following two instructions are available for performing *fused dot-products of order two* (FD2), possibly *followed by an addition* (FD2A):

- **FD2**(a, b, c, d) returns $\mathrm{RN}(ab + cd)$;
- **FD2A**(a, b, c, d, e) returns $\mathrm{RN}(ab + cd + e)$.

The necessary background in FP arithmetic and error analysis can be found for example in [15]. In particular, recall that for $x \in \mathbb{R}$, $\mathrm{ufp}(x) = 2^{\lfloor \log_2 |x| \rfloor}$ if $x \neq 0$ (and zero otherwise), that $\mathrm{ulp}(x) = 2u \cdot \mathrm{ufp}(x)$, and that the absolute and relative errors due to rounding to nearest are bounded, respectively, as $|\mathrm{RN}(x) - x| \leq \frac{1}{2}\mathrm{ulp}(x)$ and as $\mathrm{RN}(x) = x(1 + \delta)$, $|\delta| \leq u/(1 + u)$. Recall also that if $x \in \mathbb{F}$, then it is an integral multiple of its ulp: $x \in \mathrm{ulp}(x)\mathbb{Z}$.

II. Complex arithmetic

For complex FP numbers $a + ib$ and $c + id$, addition and subtraction can be implemented perfectly well with basic FP arithmetic, as $\mathrm{RN}(a \pm c) + i\,\mathrm{RN}(b \pm d)$. However, for complex multiplication and division, even FMA-based implementations can suffer heavy cancellation in the real or imaginary part of the result, and so an FD2 instruction will be most useful here to get highly-accurate results.

Specifically, for complex products $(a + ib)(c + id)$ with $a, b, c, d \in \mathbb{F}$, two FD2 calls yield the correctly-rounded real and imaginary parts $\mathrm{RN}(ac - bd)$ and $\mathrm{RN}(ad + bc)$. When $c+id = a-ib$ (complex multiplication by complex conjugate), this ensures that the imaginary part of the result is zero; with an FMA, this algebraic property can be lost, since $\mathrm{RN}(-ab + \mathrm{RN}(ab)) \neq 0$ unless $ab \in \mathbb{F}$.

For complex division, we have $(a + ib)/(c + id) = R + iI$ with $R = \frac{ac+bd}{c^2+d^2}$ and $I = \frac{bc-ad}{c^2+d^2}$, so that three FD2 calls and two FP divisions suffice to get approximations $\widehat{R}$ and $\widehat{I}$ having high relative accuracy. More precisely, $\widehat{R} = \mathrm{RN}(\mathrm{RN}(ac + bd)/\mathrm{RN}(c^2 + d^2))$, where the relative error of FD2 is at most $u/(1+u)$ and the one of FP division is at most $u - 2u^2$ [16]; the same holds for $\widehat{I}$, and it is then easily checked that this implies

$$\widehat{R} = R(1 + \delta_R), \quad \widehat{I} = I(1 + \delta_I), \quad |\delta_R|, |\delta_I| \leq 3u.$$

Another immediate consequence of the availability of an FD2 instruction is a more accurate complex absolute value $\sqrt{a^2 + b^2}$ (i.e., hypotenuse): with or without an FMA, the relative error is known to be tightly bounded by $2u$ [16], [17]; with an FD2, we now compute $\widehat{h} := \mathrm{RN}\big(\sqrt{\mathrm{RN}(a^2 + b^2)}\big)$. By applying twice the relative error bound $u/(1+u)$, we deduce that

$$\widehat{h} = \sqrt{a^2 + b^2}\,(1 + \delta_h), \qquad |\delta_h| < \tfrac{3}{2}u.$$

Furthermore, this new bound is tight: taking $a = 1$ and $b = \mathrm{RN}(\sqrt{3u}(1 - u))$ gives $|\delta_h| > 1 - 1/\sqrt{1 + 3u - 12u^2}$ for $p \geq 3$, which is $3u/2 - O(u^2)$ as $u \to 0$.

Note also that having an FD2A (and not just an FD2) gives directly a *complex fused multiply-add* operation, which is for example key for the FFT [4]: since $(a+ib)(c+id)+e+if = ac - bd + e + i(ad + bc + f)$, two FD2A calls suffice to get the correctly-rounded values of the real and imaginary parts of this complex number.

Finally, another advantage of these FD2(A)-based solutions is to allow to avoid spurious underflows and overflows, either completely (for multiplication and complex FMA) or partially (for division and absolute value).

III. Error-free transforms

It is well known that under mild conditions the error of an FP multiplication and the error of a (rounded to nearest) FP addition are FP numbers, and that these errors can be computed using the 2Sum, Fast2Sum, and 2MultFMA algorithms; see for example [15, §4.2]. These algorithms are called error-free transformations (EFT) in [18].

The 2Sum algorithm requires 6 FP additions and, assuming some kind of ordering on the input, the Fast2Sum algorithm needs only 3 FP additions. Without an FMA, computing the exact error of an FP product can be done using 17 FP operations [19], but with an FMA this becomes much easier and cheaper: if one computes $P_h = \mathrm{RN}(ab)$ and $P_\ell = \mathrm{RN}(ab - P_h)$, then $P_h + P_\ell = ab$ exactly.[2] In other words, the FMA makes it possible to implement the EFT of multiplication with only two operations.

Similarly, as noted in [18], the availability of an ADD3 operation, which for $a, b, c \in \mathbb{F}$ returns $\mathrm{RN}(a + b + c)$, would greatly simplify the computation of the error of an FP addition: if $S_h = \mathrm{RN}(a+b)$ and $S_\ell = \mathrm{RN}(a+b-S_h)$, then $S_h + S_\ell = a + b$ exactly.

Since the FMA and ADD3 are special cases of the FD2 and FD2A operations, respectively, having an FD2 makes it easy to obtain the error of an FP multiplication, and having an FD2A makes it easy to obtain the error of an FP addition: in each case, two operations now suffice to implement the EFT.

What about the error of the FMA $ab + c - \mathrm{RN}(ab + c)$? As shown in [20], [21], this error is in general not an FP number but can always be expressed as the exact sum $e_1 + e_2$ of two FP numbers such that $e_1 = \mathrm{RN}(e_1 + e_2)$; furthermore, the pair (e_1, e_2) can be produced using only basic FP operations, but the cost is high: Algorithm ErrFmac from [20] uses 2 FMA, 1 multiplication, and 17 additions, for a total of 20 basic FP operations. Interestingly, with an ADD3 (and thus a

[2] We recall that we have assumed an unbounded exponent range. If this is not the case, then there is a condition on the exponents.

fortiori with an FD2A), this operation count drops to 10: by inspecting Algorithm ErrFmac and its proof in [20], one sees that 2 FMA, 1 multiplication, 3 additions, and 4 ADD3 are sufficient.

If one needs only the correctly-rounded value of the error of the FMA, that is,

$$e_1 = \mathrm{RN}(ab + c - \mathrm{RN}(ab + c)), \tag{1}$$

then Algorithm ErrFmaNearest from [21] can be used: it performs 18 basic FP operations and, again, it is easily seen that with an ADD3 this operation count is halved too, dropping to 9. However, in this case, having an FD2A makes an even bigger difference, since e_1 in (1) can now be obtained using only 2 operations: one FMA followed by one FD2A. We will see in section VIII how this can be used to design a new, FMA/FD2A-based, compensated Horner algorithm, which avoids using the expensive EFT of the FMA without sacrificing for accuracy.

Similarly, only one call to the FD2A makes it possible to obtain the correct rounding of the error of an FD2 (and thus the correct rounding of the errors in a complex product):

$$\begin{array}{rcl} P_h & = & \mathrm{RN}(ab + cd), \\ P_\ell & = & \mathrm{RN}(ab + cd - P_h). \end{array} \tag{2}$$

As for the FMA, we will have $P_h + P_\ell \neq ab + cd$ in general. However, the theorem below shows that the scheme in (2) does define an EFT for $ab + cd$ provided some conditions on the ratio ab/cd are satisfied.

Theorem 1. *Given $a, b, c, d \in \mathbb{F}$, let P_h and P_ℓ be as in (2). If $ab/2 \leq -cd \leq 2ab$, then*

$$ab + cd = P_h + P_\ell.$$

Proof. We will use two basic properties: for $x, y, z, t \in \mathbb{R}$,

(i) if $|x| \leq 2^k$ then $|\mathrm{RN}(x) - x| \leq 2^{-p-1} \cdot 2^k$;
(ii) if $|xy| \leq |zt|$ then $\mathrm{ulp}(x)\mathrm{ulp}(y) \leq 2\mathrm{ulp}(z)\mathrm{ulp}(t)$.

Since ab and cd play a symmetric role, we can assume that $-cd = |cd| \leq ab$, so $0 \leq ab + cd \leq |cd| \leq 2^{2p}\mathrm{ulp}(c)\mathrm{ulp}(d)$. Using ($i$) we deduce that $|ab + cd - P_h| \leq 2^{p-1}\mathrm{ulp}(c)\mathrm{ulp}(d)$. On the other hand, $ab + cd \in \mu\mathbb{Z}$ with $\mu := \min\{\mathrm{ulp}(a)\mathrm{ulp}(b), \mathrm{ulp}(c)\mathrm{ulp}(d)\}$. Since $|cd| \leq ab$, it follows from (ii) that $\mu \geq \frac{1}{2}\mathrm{ulp}(c)\mathrm{ulp}(d) =: 2^e$, so that $ab+cd$ and P_h are in $2^e\mathbb{Z}$. Hence $|ab + cd - P_h|$ is an integral multiple of 2^e that is less than or equal to 2^{p+e}. This implies $ab+cd-P_h \in \mathbb{F}$ and thus $P_\ell = \mathrm{RN}(ab + cd - P_h) = ab + cd - P_h$. □

We conclude this section by noting that even when the exact sum $P_h + P_\ell$ differs from the exact value $ab + cd$, its relative error is extremely small, as we always have the following bound, which is asymptotically optimal:

$$P_h + P_\ell = (ab + cd)(1 + \delta), \qquad |\delta| \leq \tfrac{1}{2}u^2.$$

This bound and its asymptotic optimality follow as a special case of the more general property below (which will also be useful in order to establish Property 2 in section IV, and when designing double-word arithmetic algorithms in section IX).

Property 1. *Let $x \in \mathbb{R}$, $\widehat{x} = \mathrm{RN}(x)$, and $\widehat{e} = \mathrm{RN}(x - \widehat{x})$. Then the relative error of the exact sum $\widehat{x} + \widehat{e}$ is at most $\frac{1}{2}u^2$. Furthermore, this bound is asymptotically optimal.*

Proof. The bound is clear for $x = 0$, so let $x \neq 0$. Writing $e = x - \widehat{x}$, we have $|e| \leq u\,\mathrm{ufp}(x) =: B$. This bound B is an integral power of 2, so $|\widehat{x} + \widehat{e} - x| = |\mathrm{RN}(e) - e| \leq \frac{u}{2}B = \frac{1}{2}u^2\mathrm{ufp}(x)$. Using $|x| \geq \mathrm{ufp}(x)$ then leads to the relative error bound. Asymtpotic optimality can be shown by considering $x = 1 + u + \frac{1}{2}u^2$, for which $\widehat{x} = 1 + 2u$, $\widehat{e} \in \{-u, -u + u^2\}$, and $|(\widehat{x} + \widehat{e})/x - 1| = \frac{u^2/2}{1+u+\frac{1}{2}u^2} \sim \frac{1}{2}u^2$ as $u \to 0$. □

IV. Products of three or four FP numbers

Note first that given $a, b, c, d \in \mathbb{F}$, one can compute the correctly-rounded value of $abc + d$ very easily with an FD2A: compute $p_h := \mathrm{RN}(ab)$, $p_\ell := \mathrm{RN}(ab - p_h)$, and $q_h := \mathrm{RN}(p_h c + p_\ell c + d)$; since $p_h + p_\ell = ab$, we see immediately that q_h satisfies

$$q_h = \mathrm{RN}(abc + d). \tag{3}$$

This has several interesting properties. A first one (detailed in section V) is to ease the correct computation of the sign of the discriminant of some cubic equations. Evaluating expressions of the form $\mathrm{RN}(abc + d)$ can also be useful for elementary function libraries. For example, the cosine of a small argument x is typically approximated by $1 + x^2P(x)$ with P a low-degree polynomial.

Of course, (3) leads in particular to correctly-rounded products of three FP numbers. But what if we want to multiply four FP numbers together? Interestingly, the FD2A can again be used easily in order to get an approximation to $abcd$ with relative error at most u, as follows: by applying the method underlying (3), compute $q_h := \mathrm{RN}(abc)$ and $q_\ell := \mathrm{RN}(abc - q_h)$, and then deduce

$$\widehat{r} := \mathrm{RN}(q_h d + q_\ell d). \tag{4}$$

Note that since $abc - q_h$ is not always in $\mathbb{F}$, the exact sum $q_h + q_\ell$ can differ from abc, and so $\widehat{r}$ may not be the correctly-rounded value of $abcd$. Such a situation occurs for example when $(a, b, c, d) = (1 + 2u, 1 + 2u, 1 - u, 1)$: in this case, one can check that for $u \leq 2^{-3}$, $q_h = 1 + 2u$ and $\widehat{r} = 1 + 4u$. Despite this, the relative accuracy of $\widehat{r}$ in (4) is essentially best possible, as the next property shows.

Property 2. *For $\widehat{r}$ as in (4), $|\widehat{r} - abcd| \leq u|abcd|$.*

Proof. By Property 1, $q_h + q_\ell = abc(1+\delta)$ with $|\delta| \leq \frac{1}{2}u^2$; on the other hand, $\widehat{r} = (q_h d + q_\ell d)(1 + \delta')$ with $|\delta'| \leq u/(1+u)$. Hence $\widehat{r} = abcd(1 + \delta'')$ with $\delta'' = (1 + \delta)(1 + \delta') - 1$, which satisfies $|\delta''| \leq u$ for $u \leq 1/4$. □

For comparison, when evaluating $abcd$ in the most usual way with three FP multiplications, the associated relative error bound is about $3u$. Longer products, of the form $x_1x_2 \cdots x_n$ with $n \geq 5$, will obviously benefit from the above lemma in about the same way, with the usual bound $\approx (n - 1)u$ (see for example [22]) being replaced by a smaller one $\approx \frac{n}{4}u + (\frac{n}{4} - 1)u = nu/2$.

V. Quadratic and cubic equations

When solving quadratic or cubic equations in FP arithmetic, a first key step is to get the sign of their discriminant right, as this tells about the nature of the solutions and in particular the number of real solutions.

For quadratic equations $ax^2+bx+c=0$ with $a,b,c\in\mathbb{F}$, the discriminant is $\Delta=b^2-4ac$ and, as already noted in [10], one FD2 instruction suffices to produce $\mathrm{RN}(\Delta)$ and thus $\mathrm{sign}(\Delta)$ as well. (This holds at least for binary arithmetic, since $4a\in\mathbb{F}$ is obtained exactly as $\mathrm{RN}(4a)$.)

By contrast, other ways to evaluate Δ, such as computing $\Delta_{+\times}:=\mathrm{RN}(\mathrm{RN}(b^2)-\mathrm{RN}(4ac))$, $\Delta_{\text{FMA\_1}}:=\mathrm{RN}(\mathrm{RN}(b^2)-4ac)$, and $\Delta_{\text{FMA\_2}}:=\mathrm{RN}(b^2-\mathrm{RN}(4ac))$ can be so wrong that even the sign is not obtained correctly [23]. For example, one has $\Delta_{+\times}=0$ and $\Delta\neq 0$ when $(a,b,c)=(1/4-u/2,1,1+2u)$. One can also have $\Delta_{\text{FMA\_1}}<0$ or $\Delta_{\text{FMA\_2}}<0$, while $\Delta=0$, which leads to conclude wrongly that there is no real solution; an example for $\Delta_{\text{FMA\_1}}$ is given by $(a,b,c)=(1/4-u/4,1-u,1-u)$.

An FD2 instruction can also be immediately useful when solving so-called depressed cubic equations $x^3+bx+c=0$ with $b,c\in\mathbb{F}$. Here the discriminant is $\Delta=-4b^3-27c^2$ and its sign can be computed wrongly by direct implementations, with or without FMA. For example, in binary32 arithmetic with $b=-14628329$ and $c=21534767104$, we have $\Delta_{\text{FMA}}<0<\Delta$ with $\Delta_{\text{FMA}}:=\mathrm{RN}(-\mathrm{RN}(4b\cdot\mathrm{RN}(b^2))-c\cdot\mathrm{RN}(27c))$.

With an FD2 instruction, however, one can use the solution presented in section IV to easily obtain the correctly-rounded products $\delta_1=\mathrm{RN}(4b^3)$ and $\delta_2=\mathrm{RN}(27c^2)$, and then compute $\Delta_{\text{FD2}}=\mathrm{RN}(-\delta_1-\delta_2)$. By monotonicity of rounding, this approach ensures that $\Delta_{\text{FD2}}>0$ implies $\Delta>0$ (and that there are three distinct real roots), and that $\Delta_{\text{FD2}}<0$ implies $\Delta<0$ (and that there is one real root). Hence only the case $\Delta_{\text{FD2}}=0$ will require more work in order to compute $\mathrm{sign}(\Delta)$ correctly.

VI. Correctly-rounded multiplication by real constants

In [24], the authors consider the problem of performing the correctly rounded (to nearest) multiplication by a constant $C\notin\mathbb{F}$. Typical useful values of C are π, $1/\pi$, $\ln(2)$, $\sqrt{2}$, etc. Assuming that $C_h=\mathrm{RN}(C)$ and $C_\ell=\mathrm{RN}(C-C_h)$ are precomputed, for an input $x\in\mathbb{F}$ they assume that $y^*=C\cdot x$ is approximated by

$$y_1=\mathrm{RN}(C_h\cdot x+\mathrm{RN}(C_\ell\cdot x)),\tag{5}$$

whose computation requires an FP multiplication followed by an FMA. There are values of C and x for which $y\neq\mathrm{RN}(y^*)$. However, it is shown in [24] that for a given C, a simple algorithm can check if $y_1=\mathrm{RN}(y^*)$ for all $x\in\mathbb{F}$, and it is observed that this is indeed the case for many useful values of C. The underlying reasoning is as follows:

- First, there are ≤ 4 values of the significand of x for which y_1 and y^* may belong to different binades. One easily checks if $y_1=\mathrm{RN}(y^*)$ for these values.
- Then, for all other numbers x and with $\epsilon:=|C-(C_h+C_\ell)|$, one can bound $|y_1-y^*|$ by

$$\tfrac{1}{2}\mathrm{ulp}(y^*)+\eta_1,\quad \eta_1:=\tfrac{1}{2}\mathrm{ulp}(C_\ell x)+\epsilon|x|.\tag{6}$$

- Finally, (6) implies that if $y_1\neq\mathrm{RN}(y^*)$, then Cx is within distance $\leq\eta_1$ from the middle of two consecutive FP numbers. This in turn implies that C is very close to a rational number of the form $(2A+1)/(2X)$ with A and X two integers between 2^{p-1} and 2^p, which can be checked using the continued-fraction expansion of C.

Method 3 in [24] implements this strategy, and either tells that there is no x such that $y_1\neq\mathrm{RN}(y^*)$, or gives all values x (up to a multiplication by a power of 2) for which $y_1\neq\mathrm{RN}(y^*)$.

With an FD2 instruction we can change the way the approximation to y^* is computed and, instead of (5), use

$$y_2=\mathrm{RN}(C_h\cdot x+C_\ell\cdot x).$$

We easily obtain that there are at most 2 values of the significand of x for which y and y^* may belong to different binades (as previously, one checks these values separately), and that for all other values we have

$$|y_2-y^*|\leq\tfrac{1}{2}\mathrm{ulp}(y^*)+\eta_2,\quad \eta_2:=\epsilon|x|.$$

Hence, if $y_2\neq\mathrm{RN}(y^*)$, then Cx is within distance $\leq\eta_2$ from the middle of two consecutive FP numbers. As $\eta_2\leq\eta_1$, if Method 3 in [24] tells us there is no x such that $y_1\neq\mathrm{RN}(y^*)$, then there is no x such that $y_2\neq\mathrm{RN}(y^*)$ either. If Method 3 yields x such that $y_1\neq\mathrm{RN}(y^*)$, then we must check if $y_2=\mathrm{RN}(y^*)$. Using this strategy we can show that with π, $1/\pi$, $\ln(2)$, $1/\ln(2)$, $\ln(10)$, $1/\ln(10)$, in the binary32, binary64, and binary128 formats we always have $y_2=\mathrm{RN}(y^*)$.

VII. Inner products and sums

Another natural application of the FD2A is the computation of inner products with both fewer instructions and fewer rounding errors than when just using an FMA. Given two floating-point vectors $x,y\in\mathbb{F}^n$, recall from [25] that a sequence of n FMAs suffices to produce an approximation $\widehat{r}_{\text{FMA}}$ to the inner product x^Ty such that[3]

$$|\widehat{r}_{\text{FMA}}-x^Ty|\leq nu|x|^T|y|.\tag{7}$$

This bound is asymptotically optimal: for fixed n, there exist $x,y\in\mathbb{F}^n$ such that both sides of (7) are asymptotically equivalent as $u\to 0$. For example, with $x^T=[1+2u,1,\ldots,1]$ and $y^T=[1-u,u-u^2,\ldots,u-u^2]$, we have $\widehat{r}_{\text{FMA}}=1$ and $x^Ty=1+nu-(n+1)u^2$.

With an FD2A this scheme is easily adapted by considering the following recursion: let $\widehat{r}_1:=\mathrm{RN}(x_1y_1+x_2y_2)$ and, for $i=2,\ldots,\lceil n/2\rceil$, let

$$\widehat{r}_i:=\mathrm{RN}(\widehat{r}_{i-1}+x_{2i-1}y_{2i-1}+x_{2i}y_{2i}),$$

[3]Note that instead of nu in (7), a term of the form $(1+u)^n-1=nu+O(u^2)$ can be obtained classically, following [26, Ch. 3]; however, the fact that the term $O(u^2)$ can be removed is a consequence of [25, Lemma 2.1].

with $x_{n+1} = y_{n+1} = 0$ if n is odd. As expected, the output $\widehat{r}_{\text{FD2A}} := \widehat{r}_{\lceil n/2 \rceil}$ has an error bound about twice smaller than the one in (7). The next theorem makes this claim precise.

Theorem 2. *Given $x, y \in \mathbb{F}^n$, a sequence of $\lceil n/2 \rceil$ FD2A instructions suffices to produce an approximation $\widehat{r}_{FD2A} \in \mathbb{F}$ to the inner product $x^T y$ such that*

$$|\widehat{r}_{FD2A} - x^T y| \leq mu|x|^T|y|, \quad m := \lceil n/2 \rceil.$$

Furthermore, this bound is asymptotically optimal.

Proof. Let the vectors $z, \varepsilon, \delta, s \in \mathbb{R}^m$ be defined by $z_i = x_{2i-1}y_{2i-1} + x_{2i}y_{2i}$, $s_i = \widehat{r}_i$, $s_1 = z_1 + \delta_1 = z_1(1+\varepsilon_1)$ and, for $i \geq 2$, $s_i = z_i + s_{i-1} + \delta_i = (z_i + s_{i-1})(1+\varepsilon_i)$. For $i \geq 2$, we thus have $s_i = \text{RN}(s_{i-1} + z_i)$ and $|\delta_i| = |\text{RN}(s_{i-1}+z_i) - (s_{i-1}+z_i)| = \min_{f \in \mathbb{F}} |f - (s_{i-1}+z_i)|$; since $s_{i-1} \in \mathbb{F}$, taking in particular $f = s_{i-1}$ implies that $|\delta_i| \leq |z_i|$. Hence the assumption needed to apply [25, Lemma 2.1] to $(z, \varepsilon, \delta, s)$ is satisfied and since $z_1 + \cdots + z_m = x^T y$ and $|z_1| + \cdots + |z_m| \leq |x|^T|y|$, we deduce from this lemma that $|\widehat{r}_m - x^T y| \leq mu|x|^T|y|$. The asymptotic optimality of this bound can be seen by considering $x^T = [1+2u, 0, 1, 0, 1, \ldots]$, and $y^T = [1-u, 0, u-u^2, 0, u-u^2, \ldots]$, for which we have $\widehat{r}_m = 1$ and $x^T y = 1 + mu - (m+1)u^2$. □

Alternatively, one could approximate $x^T y$ by first computing the m values $\widehat{z}_i := \text{RN}(x_{2i-1}y_{2i-1} + x_{2i}y_{2i})$ and then, if $m \geq 2$, adding them together by means of a ternary summation tree having $\lfloor m/2 \rfloor$ inner nodes. This would use m calls to FD2 followed by $\lfloor m/2 \rfloor$ calls to ADD3, and thus $m + \lfloor m/2 \rfloor \approx 3n/4$ operations. Although this operation count is larger than m by about $n/4$, the error bound associated with this second approach will be smaller than the one in Theorem 2 by at least about $n/4$, and thus at least about twice smaller. Specifically, classical analysis as in [26, Ch. 4] would show that the constant mu obtained so far can be replaced by $(h+1)u + O(u^2)$, with h denoting the height of the underlying ternary tree. Since this tree has $\lfloor m/2 \rfloor$ inner nodes, its height h ranges from $\lceil \log_3 \lfloor m/2 \rfloor \rceil$ to $\lfloor m/2 \rfloor$. Hence $(h+1)u \lesssim m/2 \cdot u \approx n/4 \cdot u$.

Another way to reduce even further the error bound (7) is via *compensation*: besides the FMA-based recursion

$$\widehat{r}_i := \text{RN}(\widehat{r}_{i-1} + x_i y_i), \quad i = 1, \ldots, n,$$

which yields $\widehat{r}_{\text{FMA}} := \widehat{r}_n$, one computes correctly-rounded approximations to the errors $e_i := \widehat{r}_{i-1} + x_i y_i - \widehat{r}_i$ by means of n calls to FD2A:

$$\widehat{e}_i := \text{RN}(\widehat{r}_{i-1} + x_i y_i - \widehat{r}_i), \quad i = 1, \ldots, n.$$

(Of course, we set $\widehat{r}_0 := 0$ and so an FMA suffices to produce $e_1 = x_1 y_1 - \widehat{r}_1 \in \mathbb{F}$.) These n values $\widehat{e}_i$ are then added together by means of $\lfloor n/2 \rfloor$ calls to ADD3 into a single value $\widehat{e}$. Finally, this value, which approximates the exact overall error $e_1 + \cdots + e_n = x^T y - \widehat{r}_{\text{FMA}}$, is used to produce the refined approximation $\widehat{r}_{\text{COMP}} := \text{RN}(\widehat{r}_{\text{FMA}} + \widehat{e})$.

Theorem 3. *Under the assumptions of the previous theorem, a sequence of n FMA instructions and $\lfloor 3n/2 \rfloor$ FD2A instructions suffices to produce $\widehat{r}_{COMP} \in \mathbb{F}$ such that*

$$|\widehat{r}_{COMP} - x^T y| \leq u|x^T y| + \lambda_{COMP}|x|^T|y|$$

with $\lambda_{COMP} = \frac{1}{2}(n^2+3n)u^2 + \frac{1}{4}(n^3+n^2)u^3$.

Proof. For readability define here $r = x^T y$, $\widetilde{r} = |x|^T|y|$, $\widehat{r} = \widehat{r}_{\text{FMA}}$, $e = \sum_{i=1}^n e_i$, and $v = u/(1+u)$. Since $\widehat{r}_{\text{COMP}} = \text{RN}(\widehat{r}+\widehat{e})$ and $\widehat{r} + e = r$, we have $|\widehat{r}_{\text{COMP}} - r| = |(\widehat{r}+\widehat{e})(1+\delta) - r| = |r\delta + (\widehat{e} - e)(1+\delta)|$, $|\delta| \leq v$. Hence

$$|\widehat{r}_{\text{COMP}} - r| \leq v|r| + (1+v)|\widehat{e} - e| \tag{8}$$

and it remains to bound $|\widehat{e} - e|$. First, $|\widehat{e} - e| \leq |\widehat{e} - \sum_{i=1}^n \widehat{e}_i| + \sum_{i=2}^n |\widehat{e}_i - e_i|$ since $\widehat{e}_1 = e_1$. On the one hand, $|\widehat{e}_i - e_i| \leq v|e_i|$ for all i. On the other hand, adding the $\widehat{e}_i$'s recursively by means of $\lfloor n/2 \rfloor$ ADD3, we deduce from [25, Lemma 2.1] that $|\widehat{e} - \sum_{i=1}^n \widehat{e}_i| \leq \lfloor n/2 \rfloor v \sum_{i=1}^n |\widehat{e}_i|$. Hence

$$|\widehat{e} - e| \leq \left\lfloor \tfrac{n}{2} \right\rfloor v|e_1| + \Big(\left\lfloor \tfrac{n}{2} \right\rfloor (v + v^2) + v \Big) \sum_{i=2}^n |e_i|. \tag{9}$$

Second, to bound $|e_i|$, let us define $r_i = \sum_{j \leq i} x_i y_i$ and $\widetilde{r}_i = \sum_{j \leq i} |x_i y_i|$. Since the $\widehat{r}_i$'s are produced recursively by the FMA, we deduce from [25, Lemma 2.1] that $|\widehat{r}_i - r_i| \leq iv\widetilde{r}_i$. Consequently, for $i \geq 1$ and with $\widehat{r}_0 = 0$, $|e_i| \leq v|\widehat{r}_{i-1} + x_i y_i| \leq v\big(\widetilde{r}_i + (i-1)v\widetilde{r}_{i-1}\big)$. Using $\widetilde{r}_{i-1} \leq \widetilde{r}_i \leq \widetilde{r}_n = \widetilde{r}$ for $i \leq n$, we obtain $|e_i| \leq (v + (i-1)v^2)\widetilde{r}$ and, therefore,

$$|e_1| \leq v\widetilde{r} \quad \text{and} \quad \sum_{i=2}^n |e_i| \leq nv\big(1 + \tfrac{n-1}{2}v\big)\widetilde{r}. \tag{10}$$

The claimed error bound then follows from the inequalities (8), (9), (10). Furthermore, this was achieved using $n+1$ FMA, $n-1$ FD2A, and $\lfloor n/2 \rfloor$ ADD3. □

What is interesting with Theorem 3 is that $\lambda_{\text{COMP}} \sim \frac{1}{2}n^2u^2$, which is about half of the classical one from [18], and that this smaller bound is obtained using only about $3n/2$ FD2A besides the initial FMA-based recursion.

Note also that λ_{COMP} can be reduced even further by using a more balanced ternary tree when adding up the n approximate values $\widehat{e}_i$'s (and thus without increasing the operation count): a ternary tree of height h leads to $\lambda_{\text{COMP}} \sim (h+1)nu^2$, so that with the minimal height $h = \lceil \log_3 \lfloor n/2 \rfloor \rceil$ we arrive at $\lambda_{\text{COMP}} \sim 0.91n \ln n \, u^2$.

Note finally that similar gains can of course be obtained for the special case of summation, where $y^T = [1, 1, \ldots, 1]$, using exclusively the ADD3 instruction.

VIII. Polynomial evaluation

To be able to rapidly evaluate a low-degree polynomial on a specific architecture, taking into account the number of available operators, the depth of the arithmetic pipelines, etc., one can conduct an exhaustive or heuristic-based search among the various possible evaluation schemes [27], [28]. Having an FD2A instruction increases the number of possible schemes, potentially leading to a better optimum. We illustrate this by focusing on the most classical way (Horner's scheme with an

FMA) and detailing how it can be adapted to exploit the FD2A when computing $a(x) = \sum_{i=0}^{n} a_i x^i$ given $x, a_i \in \mathbb{F}$.

Recall first that with an FMA, Horner's recursion is given by $\widehat{r}_0 := a_n$ and

$$\widehat{r}_i := \mathrm{RN}(\widehat{r}_{i-1}x + a_{n-i}), \quad i = 1, \ldots, n, \tag{11}$$

and that this sequence of n FMA instructions yields an approximation $\widehat{r}_{\text{FMA}} := \widehat{r}_n \in \mathbb{F}$ such that

$$|\widehat{r}_{\text{FMA}} - a(x)| \leq \gamma_n \widetilde{a}(|x|), \tag{12}$$

where $\gamma_n := nu/(1-nu)$ if $nu < 1$ and $\widetilde{a}(x) := \sum_i |a_i| x^i$; see [29, §6.1]. The bound (12) is essentially best possible for fixed n and as $u \to 0$, since taking in particular $x = a_n = 1$ and $a_{n-i} = u - u^2$ for $i \leq n$ gives $\widehat{r}_{\text{FMA}} = 1$ and $a(x) = \widetilde{a}(|x|) = 1 + nu - nu^2$.

With an FD2A, the recursion in (11) can be adapted immediately by computing first $\widehat{y} := \mathrm{RN}(x^2)$ and then

$$\widehat{r}_i := \mathrm{RN}(\widehat{r}_{i-1}\widehat{y} + a_{n-2i+1}x + a_{n-2i}), \quad i = 1 : \lceil n/2 \rceil.$$

(Here, $\widehat{r}_0 = a_n$ and, if n is odd, $a_{-1} = 0$.) This scheme costs about $n/2$ FD2A instructions. In terms of accuracy, if x^2 is computed exactly, then the factor $\gamma_n \approx nu$ in (12) can be decreased down to $\gamma_{\lceil n/2 \rceil} \approx \frac{1}{2}nu$; unfortunately, in the general case where $\widehat{y} \neq x^2$, the rounding error due to this initial squaring operation must be taken into account, bringing the factor γ_n back and thus leading to the same a priori bound (12) as for Horner with FMA. This is in contrast with what we have seen for inner products in section VII and Theorem 2.

There is, however, a more interesting way of exploiting the FD2A instruction in the context of Horner evaluation: the ability of the FD2A to recover the correctly-rounded value of the error of an FMA makes it possible to design a very simple *compensated* version of (11), as follows. Set $\widehat{r}_0 := a_n$ and $\widehat{e} := 0$; then compute for i from n

$$\begin{aligned} \widehat{r}_i &:= \mathrm{RN}(\widehat{r}_{i-1}x + a_{n-i}); && \text{// FMA} \\ \widehat{e}_i &:= \mathrm{RN}(\widehat{r}_{i-1}x + a_{n-i} - \widehat{r}_i); && \text{// FD2A} \\ \widehat{e} &:= \mathrm{RN}(\widehat{e}x + \widehat{e}_i); && \text{// FMA} \end{aligned}$$

and finally return $\widehat{r}_{\text{COMP}} := \mathrm{RN}(\widehat{r}_n + \widehat{e})$.

Since $\widehat{e}$ is initialized to zero, its final value is obtained using $n-1$ FMA and is equal to the approximation to $\sum_{i=1}^{n} \widehat{e}_i x^{n-i}$ produced by Horner with FMA. The result $\widehat{r}_{\text{COMP}}$ is thus obtained by means of $2n-1$ FMA, n FD2A, and 1 addition, that is, $3n$ instructions.

An accuracy bound is given by the following theorem. Interestingly, this bound matches the best one from [29, §6.3.2], where it needed $19n$ FP instructions: n calls to an EFT for the FMA, $n+1$ additions, and $n-1$ FMA. (Remark that the EFT used in [29] is ErrFmac [20] without the last Fast2Sum, so its cost is of 17 FP instructions instead of 20.)

Theorem 4. *Given $a_0, a_1, \ldots, a_n, x \in \mathbb{F}$, the compensated Horner algorithm described above yields an approximation $\widehat{r}_{COMP} \in \mathbb{F}$ to $a(x) = \sum_{i=0}^{n} a_i x^i$ such that*

$$|\widehat{r}_{COMP} - a(x)| \leq u|a(x)| + \gamma_n^2 \widetilde{a}(|x|).$$

Proof. Let $r = a(x)$, $\widetilde{r} = \widetilde{a}(|x|)$, and $v = u/(1+u)$. For $1 \leq i \leq n$, define also $r_i = \sum_{j=0}^{i} a_{n-j}x^{i-j}$ and let e_i be such that $\widehat{r}_{i-1}x + a_{n-i} = \widehat{r}_i + e_i$. Multiplying both sides of this equality by x^{n-i} and then summing from $i = 1$ to n yields after simplification $r = \widehat{r}_n + e$, where $e = \sum_{i=1}^{n} e_i x^{n-i}$. Since $\widehat{r}_{\text{COMP}} = (\widehat{r}_n + \widehat{e})(1+\delta)$ with $|\delta| \leq v$, it follows that

$$|\widehat{r}_{\text{COMP}} - r| \leq v|r| + (1+v)|\widehat{e} - e|. \tag{13}$$

Let us now bound $|\widehat{e} - e|$. Since $\widehat{e}$ is obtained by evaluating $|\sum_{i=1}^{n} \widehat{e}_i x^{n-i}|$ by Horner with FMA, it follows from (12) and $|\widehat{e}_i - e_i| \leq v|e_i|$ that

$$|\widehat{e} - e| \leq ((1+v)\gamma_{n-1} + v)\widetilde{e}, \tag{14}$$

where $\widetilde{e} = \sum_{i=1}^{n} |e_i||x|^{n-i}$. Now, $|e_i| \leq u|\widehat{r}_i|$ with $\widehat{r}_i$ obtained by evaluating r_i by Horner with FMA. Hence (12) implies $|\widehat{r}_i| \leq |r_i| + \gamma_i \sum_{j=0}^{i} |a_{n-j}||x|^{i-j}$, so that $|e_i||x|^{n-i} \leq u(1+\gamma_n)\widetilde{r}$. Therefore, $\widetilde{e} \leq nu(1+\gamma_n)\widetilde{r}$ and the conclusion follows from combining this bound with the ones in (13) and (14). □

IX. Double-word arithmetic

Double-word arithmetic (often referred to as *double-double* in the literature) deals with real numbers x represented as the unevaluated sums of two FP numbers x_h and x_ℓ such that $x_h = \mathrm{RN}(x_h + x_\ell) = \mathrm{RN}(x)$. Several algorithms for manipulating double-word (DW) numbers have been introduced since the pioneering work of Dekker [19], in particular in [30], [31], [32], [33], [34].

The availability of an FD2A instruction makes it possible to design new algorithms for DW arithmetic, that are cheaper in terms of numbers of instructions, often more accurate, and simpler to state and analyze than the traditional algorithms mentioned above. A key building-block will be the following implementation of the EFT for addition, which appears in [18].

Algorithm 1 : TwoSum_Add3(a, b).

1: $s \leftarrow \mathrm{RN}(a + b)$
2: $e \leftarrow \mathrm{RN}(a + b - s)$ // $a + b = e + s$ (EFT)

This algorithm performs only two operations, instead of six for 2Sum. It does not require a condition such as $|a| \geq |b|$ as Fast2Sum does. These features will, of course, benefit to all the DW algorithms described below.

A. Sum of a DW number and an FP number

Algorithm 2 : DWPlusFP_FD2A(a_h, a_ℓ, b). Computes a DW approximation c to $a + b$.

1: $(s, f) \leftarrow$ **TwoSum_Add3**(a_h, b)
2: $e \leftarrow \mathrm{RN}(f + a_\ell)$
3: $(c_h, c_\ell) \leftarrow$ **TwoSum_Add3**(s, e)

This algorithm is deduced from Algorithm DWPlusFP in [32], using TwoSum_Add3 instead of 2Sum and Fast2 Sum. It returns the same output in 5 FP instructions instead of 10, and one can proceed as in [32, §2] to check that an

asymptotically-optimal bound on its relative error is $2u$. The analysis is simpler as it avoids having to check the usual FastTwoSum condition $|s| \geq |e|$ used in [32].

B. Sum of two DW numbers

The authors of [32] analyze algorithm AccurateDWPlusDW of Hida, Li and Bailey's library [31]. They show that it has an asymptotically-optimal relative error bound of $3u^2/(1-4u)$. If an FD2A operation is available, one can use that algorithm with each 2Sum and Fast2Sum replaced by a TwoSum_Add3, which saves 10 operations without changing the error. However, the availability of an FD2A operation allows one to design an algorithm that is both cheaper and more accurate:

Algorithm 3 : AccDWPlusDW_FD2A$(a_h, a_\ell, b_h, b_\ell)$. Computes a DW approximation z to $a + b$.

1: $(s_h, s_\ell) \leftarrow$ **TwoSum_Add3**(a_h, b_h)
2: $(t_h, t_\ell) \leftarrow$ **TwoSum_Add3**(a_ℓ, b_ℓ)
3: $(c_h, c_\ell) \leftarrow$ **TwoSum_Add3**(s_ℓ, t_h)
4: $(v_h, v_\ell) \leftarrow$ **TwoSum_Add3**(s_h, c_h)
5: $w \leftarrow \text{RN}(t_\ell + v_\ell + c_\ell)$
6: $(z_h, z_\ell) \leftarrow$ **TwoSum_Add3**(v_h, w)

This algorithm uses 5 FP additions and 6 FD2A operations (instead of 20 additions for AccurateDWPlusDW), and its relative error is bounded by $2u^2$ (the proof is similar to the proof of AccurateDWPlusDW in [32]).

C. Product of a DW number by an FP number

Without an FD2A, multiplying a DW number by an FP number is a rather complex operation (for example, the most accurate of the three algorithms analyzed in [32] takes 10 FP operations and has a tight error bound of $\frac{3}{2}u^2 + 4u^3$). With an FD2A, this becomes much simpler and more accurate:

Algorithm 4 : DWTimesFP_FD2A(a_h, a_ℓ, b). Computes a DW approximation z to ab.

1: $s \leftarrow \text{RN}(a_h \cdot b + a_\ell \cdot b)$
2: $e \leftarrow \text{RN}(a_h \cdot b + a_\ell \cdot b - s)$
3: $(z_h, z_\ell) \leftarrow$ **TwoSum_Add3**(s, e)

It follows from Property 1 that the relative error of this algorithm is at most $\frac{1}{2}u^2$. This bound is asymptotically optimal: with the values $a_h = 1 + 2u$, $a_\ell = \frac{3}{2}u^2$, and $b = 1 - u$, we obtain $s = 1$ and $e = u - u^2$, which leads to a relative error equal to $\frac{\frac{1}{2}u^2 - \frac{3}{2}u^3}{1+u-\frac{1}{2}u^2-\frac{3}{2}u^3} \sim \frac{1}{2}u^2$ as $u \to 0$.

As $e = \text{RN}(ab - s)$, in most applications the last line of the algorithm can be omitted. That line is however necessary if we wish to always output a DW, as we may have $s \neq \text{RN}(s + e)$ (an example is with $a_h = 1 + 2u$, $a_\ell = \frac{5}{2}u^2$, $b = 1 - u$).

D. Product of two DW numbers

Three algorithms for computing the product of two DW numbers are analyzed in [35]. The most accurate of them has an error bound of $4u^2$ and requires 9 FP operations. The availability of an FD2A operation would improve the situation, both in terms of accuracy and number of operations. Consider the following algorithm, which uses 5 FP operations:

Algorithm 5 : DWTimesDW_FD2A$(a_h, a_\ell, b_h, b_\ell)$. Computes a DW approximation z to ab.

1: $(c_h, e) \leftarrow$ **TwoProd**(a_h, b_h)
2: $c_\ell \leftarrow \text{RN}(a_\ell \cdot b_h + a_h \cdot b_\ell + e)$
3: $(z_h,\ z_\ell) \leftarrow$ **TwoSum_Add3**(c_h, c_ℓ)

This algorithm has relative error $\leq 3u^2$. This can be shown as follows. The absolute error is $|\epsilon_1 + \epsilon_2|$ with $\epsilon_1 = \text{RN}(a_\ell b_h + a_h b_\ell + e) - (a_\ell b_h + a_h b_\ell + e)$ the rounding error occurring when computing c_ℓ, and ϵ_2 the error due to $a_\ell b_\ell$ being ignored. Without loss of generality we assume $1 \leq a_h, b_h \leq 2 - 2u$. This implies that $|a_\ell|, |b_\ell| \leq u$, so that $|\epsilon_2| \leq u^2$.

- If $a_h b_h \leq 2$, then $|e| \leq u$ and, from [35, Lemma 2.11], $a_h + b_h \leq 3 - 2u$. Thus $|a_\ell b_h + a_h b_\ell + e| \leq u(a_h + b_h) + u < 4u$, so that $|\epsilon_1| \leq 2u^2$. Therefore, $|(c_h + c_\ell) - ab| \leq 3u^2$. As $ab \geq 1$, the relative error is $\leq B_1 = 3u^2$.
- If $a_h b_h > 2$, then $|e| \leq 2u$, so that $|a_\ell b_h + a_h b_\ell + e| \leq 2u(2 - 2u) + 2u < 6u$ and $|\epsilon_1| \leq 4u^2$. Hence $|\epsilon_1 + \epsilon_2| \leq 5u^2$. Since $ab \geq a_h(1-u) \cdot b_h(1-u)$, the relative error is $\leq B_2 = 5u^2/(2(1-u)^2)$.

If $u \leq \frac{1}{16}$ (which always holds in practice), $B_1 \geq B_2$, giving the final bound $3u^2$. Furthermore, this bound is asymptotically optimal. To see this, let $a_h = b_h = 1 + 2u$ and $a_\ell = b_\ell = u - u^2$. One can check that $c_h = 1 + 4u$, $e = 4u^2$, and $c_\ell = 2u + 4u^2$, so that the relative error of $c_h + c_\ell$ (and thus of $z_h + z_\ell$ as well) is $\frac{3u^2 - 6u^3 + u^4}{1 + 6u + 7u^2 - 6u^3 + u^4} \sim 3u^2$ as $u \to 0$.

A more accurate calculation is possible at a higher cost (8 FP operations), using the following algorithm.

Algorithm 6 : DWTimesDW2_FD2A$(a_h, a_\ell, b_h, b_\ell)$. Computes a DW approximation z to ab.

1: $(c, e) \leftarrow$ **TwoProd**(a_h, b_h)
2: $v_h \leftarrow \text{RN}(a_\ell \cdot b_h + a_h \cdot b_\ell)$
3: $v_\ell \leftarrow \text{RN}(a_\ell \cdot b_h + a_h \cdot b_\ell - v_h)$
4: $w \leftarrow \text{RN}(v_\ell + a_\ell b_\ell)$
5: $s \leftarrow \text{RN}(e + v_h + w)$
6: $(z_h,\ z_\ell) \leftarrow$ **TwoSum_Add3**(c, s)

The relative error of this algorithm is bounded by $\frac{4u^2 + 3u^3}{2(1-u)^2} \sim 2u^2 + \frac{11}{2}u^3$. The proof is similar to the one of the previous algorithm.

E. Division of a DW number by a DW number

This algorithm has the same general structure as DWDivDW3 in [32], whose relative error bound is $9.8u^2$. Its relative error is $\leq 7.8u^2$ when $p \geq 13$. The proof follows the steps of the proof of [32, Theorem 7.2] (using the error bounds on DWTimesFP_FD2A, DWPlusFP_FD2A, and DWTimesDW_FD2A given in this paper).

Algorithm 7 : DWDivDW_FD2A$(a_h, a_\ell, b_h, b_\ell)$. Computes a DW approximation z to a/b.

$t_h \leftarrow \text{RN}(1/b_h)$
$r_h \leftarrow \text{RN}(1 - t_h \cdot b_h)$
$r_l \leftarrow -\text{RN}(t_h \cdot b_\ell)$
$(e_h, e_\ell) \leftarrow$ **TwoSum_Add3**(r_h, r_ℓ)
$(\delta_h, \delta_\ell) \leftarrow$ **DWTimesFP_FD2A**(e_h, e_ℓ, t_h)
$(m_h, m_\ell) \leftarrow$ **DWPlusFP_FD2A**$(\delta_h, \delta_\ell, t_h)$
$(z_h, z_\ell) \leftarrow$ **DWTimesDW_FD2A**$(a_h, a_\ell, m_h, m_\ell)$

F. Square root of a DW number

The authors of [34] analyze and formally prove a DW square root algorithm (that already appears in [36], without the final renormalization). The availability of an FD2A operator makes it possible to fuse two steps of this algorithm, and to replace its final renormalization by a call to TwoSum_Add3. This gives

Algorithm 8 : DWSQRT_FD2A(a_h, a_ℓ). Computes a DW approximation z to $\sqrt{a}$.

if $a_h = 0$ **then** $(z_h, z_\ell) \leftarrow (0, 0)$
else
$s_h \leftarrow \text{RN}(\sqrt{a_h})$
$\rho \leftarrow \text{RN}(a_h - s_h^2 + a_\ell)$
$s_\ell \leftarrow \text{RN}(\rho/(2 \cdot s_h))$
$(z_h, z_\ell) \leftarrow$ **TwoSum_Add3**(s_h, s_ℓ)

The relative error bound is the same as for the original algorithm, namely, $(25/8)u^2 = 3.125u^2$. The proof is the same as the one given in [34, Theorem 3.6].

X. Concluding remarks

We have shown that many basic building blocks of numerical computing (complex arithmetic, error-free transforms, inner products, polynomial evaluation, double-word arithmetic, etc.) could greatly benefit from the availability of FD2 and FD2A operations. This would allow for more accurate calculations, via algorithms that use fewer FP instructions and are simpler to state and analyze. An additional benefit would be a somehow simpler management of spurious overflow issues.

References

[1] Y. Tao, G. Deyuan, F. Xiaoya, and J. Nurmi, "Correctly rounded architectures for floating-point multi-operand addition and dot-product computation," in *ASAP Proc.*, 2013.

[2] N. Brunie, "Modified fused multiply and add for exact low precision product accumulation," in *ARITH Proc.*, 2017.

[3] L. Bertaccini, G. Paulin, T. Fischer, S. Mach, and L. Benini, "MiniFloat-NN and ExSdotp: An ISA extension and a modular open hardware unit for low-precision training on RISC-V cores," in *ARITH Proc.*, 2022.

[4] O. Desrentes, B. Dupont de Dinechin, and F. de Dinechin, "Exact fused dot product add operators," in *ARITH Proc.*, 2023.

[5] B. Dupont de Dinechin, J. Hascoet, and O. Desrentes, "In-Place Multi-core SIMD Fast Fourier Transforms," in *HPEC Proc.*, 2023.

[6] R. K. Montoye, E. Hokonek, and S. L. Runyan, "Design of the IBM RISC System/6000 floating-point execution unit," *IBM J. Res. Dev.*, 1990.

[7] P. Markstein, "Computation of elementary functions on the IBM RISC System/6000 processor," *IBM J. Res. Dev.*, 1990.

[8] 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 *ARITH Proc.*, 1999.

[9] C.-P. Jeannerod, N. Louvet, and J.-M. Muller, "Further analysis of Kahan's algorithm for the accurate computation of 2×2 determinants," *Math. Comp.*, 2013.

[10] C. Lauter, "An efficient software implementation of correctly rounded operations extending FMA: $a + b + c$ and $a \times b + c \times d$," in *ACSSC Proc.*, 2017.

[11] J. Sohn and E. E. Swartzlander, "A fused floating-point four-term dot product unit," *IEEE Trans. Circuits and Systems*, 2016.

[12] B. Hickmann and D. Bradford, "Experimental analysis of matrix multiplication functional units," in *ARITH Proc.*, 2019.

[13] B. Hickmann *et al.*, "Intel Nervana Neural Network Processor-T (NNP-T) fused floating point many-term dot product," in *ARITH Proc.*, 2020.

[14] IEEE, *IEEE Standard for Floating-Point Arithmetic (IEEE Std 754-2019)*, 2019.

[15] S. Boldo, C.-P. Jeannerod, G. Melquiond, and J.-M. Muller, "Floating-point arithmetic," *Acta Numerica*, 2023.

[16] C.-P. Jeannerod and S. M. Rump, "On relative errors of floating-point operations: optimal bounds and applications," *Math. Comp.*, 2018.

[17] 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 *ARITH Proc.*, 2017.

[18] T. Ogita, S. M. Rump, and S. Oishi, "Accurate sum and dot product," *SIAM J. Sci. Comput.*, 2005.

[19] T. J. Dekker, "A floating-point technique for extending the available precision," *Numer. Math.*, 1971.

[20] S. Boldo and J.-M. Muller, "Some functions computable with a Fused-Mac," in *ARITH Proc.*, 2005.

[21] ——, "Exact and approximated error of the FMA," *IEEE Trans. Comput.*, 2011.

[22] S. M. Rump, F. Bünger, and C.-P. Jeannerod, "Improved error bounds for floating-point products and Horner's scheme," *BIT Numer. Math.*, 2016.

[23] W. Kahan, "Lecture notes on the status of IEEE-754," 1997, available at http://www.cs.berkeley.edu/\~{}wkahan/ieee754status/IEEE754.PDF.

[24] N. Brisebarre and J.-M. Muller, "Correctly rounded multiplication by arbitrary precision constants," *IEEE Trans. Comp.*, 2008.

[25] M. Lange and S. M. Rump, "Error estimates for the summation of real numbers with application to floating-point summation," *BIT Numer. Math.*, 2017.

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

[27] G. Revy, "Implementation of binary floating-point arithmetic on embedded integer processors : polynomial evaluation-based algorithms and certified code generation," Ph.D. dissertation, U. Lyon - ÉNS de Lyon, France, 2009.

[28] M. Cornea, J. Harrison, and P. T. P. Tang, *Scientific Computing on Itanium-Based Systems*. Intel Press, 2002.

[29] N. Louvet, "Algorithmes compensés en arithmétique flottante : Précision, validation, performances," Ph.D. dissertation, U. Perpignan, France, 2007.

[30] X. Li *et al.*, "Design, implementation and testing of extended and mixed precision BLAS," *ACM Trans. Math. Soft.*, 2002.

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

[32] M. M. Joldes, J.-M. Muller, and V. Popescu, "Tight and rigorous error bounds for basic building blocks of double-word arithmetic," *ACM Trans. Math. Soft.*, 2017.

[33] V. Popescu, "Towards fast and certified multiple-precision librairies," Ph.D. dissertation, U. Lyon, France, 2017.

[34] V. Lefèvre, N. Louvet, J.-M. Muller, J. Picot, and L. Rideau, "Accurate calculation of Euclidean norms using double-word arithmetic," *ACM Trans. Math. Soft.*, 2023.

[35] J.-M. Muller and L. Rideau, "Formalization of double-word arithmetic, and comments on "Tight and rigorous error bounds for basic building blocks of double-word arithmetic"," *ACM Trans. Math. Soft.*, 2022.

[36] S. M. Rump and M. Lange, "Faithfully rounded floating-point computation," *ACM Trans. Math. Soft.*, 2020.