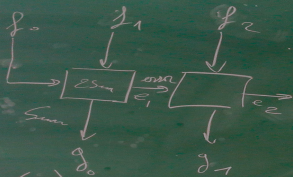


Multiword Arithmetic: Long-Standing Algorithms, Modern Challenges

Mioara Joldes

LAAS-CNRS, Toulouse, France

$$|g_1| \leq (1+2^{-p}) \left(\frac{1}{2} \text{ulp}(g_0) + 2^{-p+\epsilon} \text{ulp}(g_1) \right) = (1+2^{-p}) \text{ulp}(g_0)$$

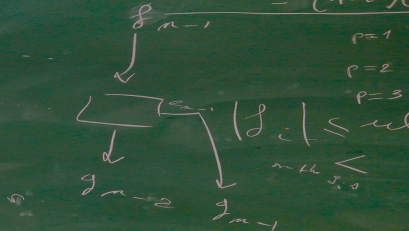


$$|e_1| \leq \frac{1}{2} \text{ulp}(g_0)$$

$$|e_1| \leq |L_1| \leq \text{ulp}(f_0)$$

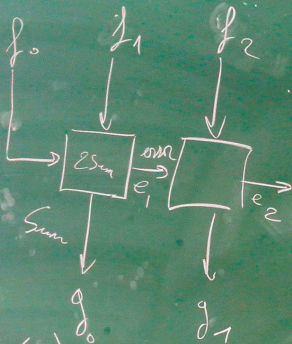
$$|f_2| \leq \text{ulp}(f_1) \leq 2^{-p+1} \text{ulp}(f_0)$$

$$\frac{1}{2} \text{ulp}(f_0) \leq \text{ulp}(f_{n-2}) \leq \text{ulp}(f_0) \leq 2^{-p+\epsilon} \text{ulp}(f_0)$$



$$|g_1| = |\text{RN}(e_1 + f_2)| \leq (1+2^{-p}) \text{ulp}(g_0) \leq (1+2^{-p}) \text{ulp}(g_0) \leq (1+2^{-p}) \text{ulp}(g_0)$$

$$|g_i| \leq (1+2^{-p}) \left(\frac{1}{2} \text{ulp}(g_0) + 2^{-p+1} \text{ulp}(g_1) \right) = (1+2^{-p})$$



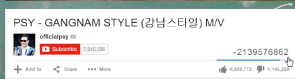
$$|e_1| \leq \frac{1}{2} \text{ulp}(g_0)$$

$$|e_1| \leq |f_1| \leq \text{ulp}(f_0)$$

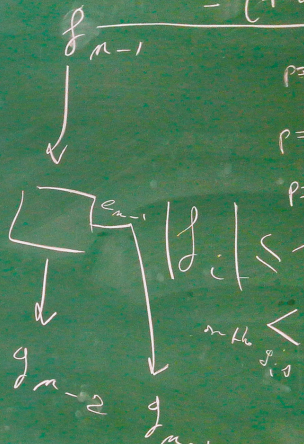
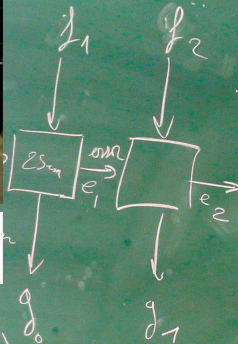
$$|f_2| \leq \text{ulp}(f_1) \leq 2^{-p+1}$$

$$\frac{1}{2} \text{ulp}(g_0) \leq \text{ulp}(f_0) \leq \text{ulp}(g_0) \leq 2^{-p+1} \text{ulp}(g_0)$$

This talk is based on
Valentina Popescu's PhD
co-supervised with Jean-Michel Muller
and our joint work with
Sylvie Boldo



$$|g_1| \leq (1+2^{-p}) \left| \frac{1}{2} \text{ulp}(g_0) + 2^{-p+1} \text{ulp}(g_1) \right| = (1+2^{-p})$$



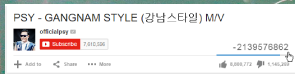
$$|e_1| \leq \frac{1}{2} \text{ulp}(g_0)$$

$$|e_1| \leq |L_1| \leq \text{ulp}(L_0)$$

$$|L_2| \leq \text{ulp}(L_1) \leq 2^{-p+1}$$

$$\frac{1}{2} \text{ulp}(g_0) \leq \text{ulp}(L_0) \leq \text{ulp}(g_0) \leq 2^{-p+1} \text{ulp}(g_0)$$

This talk is based on
Valentina Popescu's PhD
co-supervised with Jean-Michel Muller
and our joint work with
Sylvie Boldo

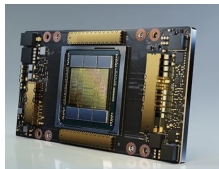


This talk is based on
Valentina Popescu's PhD
co-supervised with Jean-Michel Muller
and our joint work with
Sylvie Boldo

A Growing Demand for More (and Less) Precision

AI / Machine Learning Accelerators

- **Low-prec formats:** FP16, bfloat16, FP8
- Optimized tensor cores (NVIDIA, Intel AMX, Google TPU)
- Focus on throughput, energy efficiency
- Increasingly used for mixed-precision algorithms



(e.g. NVIDIA A100, Google TPU)

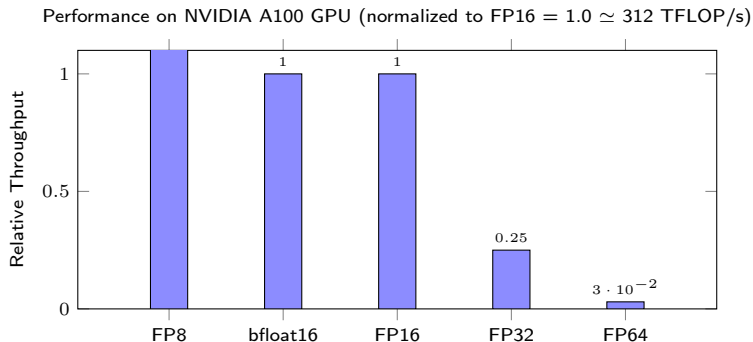
Scientific Computing / HPC

- **High-precision needs:** FP64 and beyond
- Uncertainty quantification, numerical stability
- Limited hardware support beyond double
- Software-based quad or arbitrary-prec libs



(e.g. x86 CPUs, HPC nodes)

Hardware landscape is changing



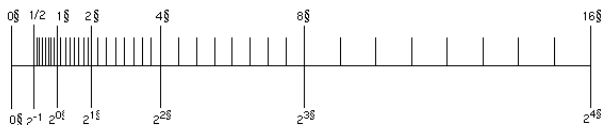
Hardware innovation has focused on **low precision for speed**,
not on **extended precision for accuracy**.

Multiword arithmetic: combine low-precision words to reach higher precision flexibly

Outline

- Floating-Point (FP) Fundamentals and Error-Free Transforms
- Double words
- Multiple words (FP Expansions)
 - `VecSumErrBranch` – A sketch of a proof
- Triple words
- Conclusion

Floating-point data



$$\mathcal{F} := \{0\} \cup \{\pm M \cdot 2^{e-p+1} : 2^{p-1} \leq M < 2^p, e_{\min} \leq e \leq e_{\max}\}$$

- Base 2
- Precision p
- Exponent range defined by e_{\min} and e_{\max}

Assumption:

- $e_{\min} = -\infty$ and $e_{\max} = +\infty$ (unbounded exponent range)

Floating-point data (continued)

For any $x \in \mathcal{F} \setminus \{0\}$:

$$|x| = m \cdot 2^e, \quad m = (*.*\dots*) \in [1, 2)$$

Useful units

$$\text{Unit in the first place:} \quad \text{ufp}(x) = 2^e$$

$$\text{Unit in the last place:} \quad \text{ulp}(x) = 2^{e-p+1}$$

$$\text{Unit in the last significant place:} \quad \text{uls}(x) = 2^k, \quad k = \max \{ i \in \mathbb{Z} \mid x \in 2^i \mathbb{Z} \}$$

$$\text{Unit roundoff:} \quad u = 2^{-p}$$

Alternative views (structure of \mathcal{F})

$$x \in \text{ulp}(x) \mathbb{Z}$$

$$|x| = (1 + 2ku) \text{ufp}(x), \quad k \in \mathbb{N}$$

$$\Rightarrow \mathcal{F} \cap [1, 2) = \{1, 1 + 2u, 1 + 4u, \dots\}$$

Rounding function

Define the **round-to-nearest** function:

$$\text{RN} : \mathbb{R} \rightarrow \mathcal{F} \quad \text{such that} \quad \forall t \in \mathbb{R}, \quad |\text{RN}(t) - t| = \min_{f \in \mathcal{F}} |f - t|$$

- If $t \in \mathcal{F}$, then $\text{RN}(t) = t$
- RN is non-decreasing
- For a reasonable tie-breaking rule:

$$\text{RN}(-t) = -\text{RN}(t)$$

$$\text{RN}(t2^e) = \text{RN}(t) 2^e, \quad e \in \mathbb{Z}$$

- Error bound:

$$|\text{RN}(t) - t| \leq \frac{1}{2} \text{ulp}(t) \leq \frac{1}{2} \text{ulp}(\text{RN}(t))$$

- Relative error bounds:

$$\frac{|\text{RN}(t) - t|}{|t|} \leq \frac{u}{1+u}, \quad \frac{|\text{RN}(t) - t|}{|\text{RN}(t)|} \leq u$$

Error-free transformations (EFT)

Floating-point algorithms for computing exact rounding errors:

- **Addition:**

$x + y - \text{RN}(x + y)$ can be computed in 6 additions (Møller'65, Knuth)

and not fewer (Kornerup, Lefèvre, Louvet, Muller'12 – *Most inelegant proof award*)

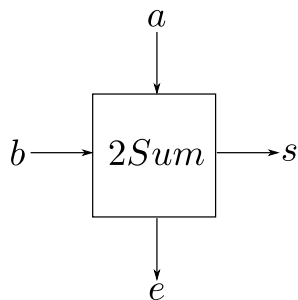
- **Multiplication:**

$$xy - \text{RN}(xy)$$

can be obtained

- in 17 operations (+ and \times) [Dekker'71, Boldo'06]
- or in only 2 ops if an FMA is available

$$\hat{z} := \text{RN}(xy) \quad \Rightarrow \quad xy - \hat{z} = \text{FMA}(x, y, -\hat{z})$$



Algorithm 1: 2Sum (a, b)

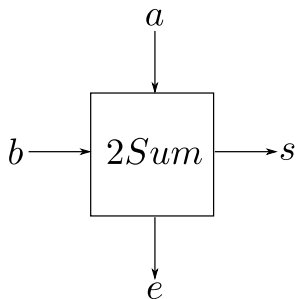
```

 $s \leftarrow \text{RN}(a + b)$ 
 $t \leftarrow \text{RN}(s - b)$ 
 $e \leftarrow \text{RN}(\text{RN}(a - t) + \text{RN}(b - \text{RN}(s - t)))$ 
return ( $s, e$ )
  
```

Thm. (2Sum)

Let $a, b \in \mathcal{F}$. Algorithm 2Sum computes $s, e \in \mathcal{F}$ s.t.

- $s + e = a + b$ exactly
- $s = \text{RN}(a + b)$.



Algorithm 2: Fast2Sum(a, b)

```

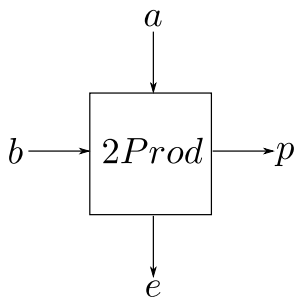
 $s \leftarrow \text{RN}(a + b)$ 
 $t \leftarrow \text{RN}(s - b)$ 
 $e \leftarrow \text{RN}(\text{RN}(a - t) + \text{RN}(b - \text{RN}(s - t)))$ 
return ( $s, e$ )
  
```

Thm. (2Sum)

Let $a, b \in \mathcal{F}$. Algorithm 2Sum computes $s, e \in \mathcal{F}$ s.t.

- $s + e = a + b$ exactly
- $s = \text{RN}(a + b)$.

If $|b| \geq |a|$ we can use Fast2Sum.



Algorithm 3: 2ProdFMA(a, b)

$p \leftarrow \text{RN}(a \times b)$
 $e \leftarrow \text{fma}(a, b, -p)$
return (p, e)

Thm. (2ProdFMA)

Let $a, b \in \mathcal{F}$. Algorithm 2ProdFMA computes $p, e \in \mathcal{F}$ s.t.

- $p + e = a \times b$ exactly
- $s = \text{RN}(a \times b)$.

When considering a bounded exponent range, condition on the exponents needed: $e_a + e_b \geq e_{min} + p - 1$.

Double-Word Arithmetic

- Fast2Sum, 2Sum, and 2Prod return unevaluated sum of two FP numbers.
- Perform more accurate calculations in critical parts of a numerical program.
- Leads to **double-word** or **double-double** arithmetic [Dekker, 1971], [Hida, Li, and Bailey], [Briggs],...
- One of most recent forms: **pair arithmetic** [Rump and Lange, 2020].

Double-word (DW)

A **double-word (DW)** number x is the unevaluated sum $x_h + x_\ell$ of two FP numbers x_h and x_ℓ such that

$$x_h = \text{RN}(x)$$

Algorithm DWPlusFP

$(s_h, s_\ell) \leftarrow 2\text{Sum}(x_h, y)$

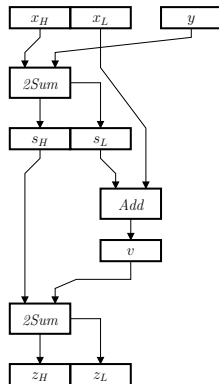
$v \leftarrow \text{RN}(x_\ell + s_\ell)$

$(z_h, z_\ell) \leftarrow \text{Fast2Sum}(s_h, v)$

return (z_h, z_ℓ)

The relative error satisfies:

$$\frac{(z_h + z_\ell) - (x + y)}{x + y} \leq 2u^2$$



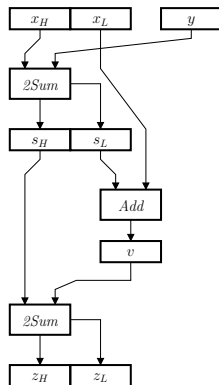
Algorithm DWPlusFP

```

 $(s_h, s_\ell) \leftarrow 2\text{Sum}(x_h, y)$ 
 $v \leftarrow \text{RN}(x_\ell + s_\ell)$ 
 $(z_h, z_\ell) \leftarrow \text{Fast2Sum}(s_h, v)$ 
return  $(z_h, z_\ell)$ 
    
```

The relative error satisfies:

$$\frac{(z_h + z_\ell) - (x + y)}{x + y} \leq 2u^2$$



The error is tight:

Let $x_h = 1$, $x_\ell = (2^p - 1) \cdot 2^{-2p}$, $y = -\frac{1}{2}(1 - 2^{-p})$. Output is $\frac{1}{2} + 3 \cdot 2^{-p-1}$ and exact sum is $\frac{1}{2} + 3 \cdot 2^{-p-1} - 2^{-2p}$, so a relative error of

$$\frac{2^{-2p+1}}{1 + 3 \cdot 2^{-p} - 2^{-2p+1}} \approx 2^{-2p+1} - 3 \cdot 2^{-3p+1}$$

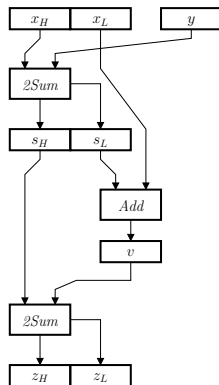
Algorithm DWPlusFP

```

 $(s_h, s_\ell) \leftarrow 2\text{Sum}(x_h, y)$ 
 $v \leftarrow \text{RN}(x_\ell + s_\ell)$ 
 $(z_h, z_\ell) \leftarrow \text{Fast2Sum}(s_h, v)$ 
return  $(z_h, z_\ell)$ 
    
```

The relative error satisfies:

$$\frac{(z_h + z_\ell) - (x + y)}{x + y} \leq 2u^2$$



The error is tight:

Let $x_h = 1$, $x_\ell = (2^p - 1) \cdot 2^{-2p}$, $y = -\frac{1}{2}(1 - 2^{-p})$. Output is $\frac{1}{2} + 3 \cdot 2^{-p-1}$ and exact sum is $\frac{1}{2} + 3 \cdot 2^{-p-1} - 2^{-2p}$, so a relative error of

$$\frac{2^{-2p+1}}{1 + 3 \cdot 2^{-p} - 2^{-2p+1}} \approx 2^{-2p+1} - 3 \cdot 2^{-3p+1}$$

When the signs of x_h and y agree, the relative error is bounded by 2^{-2p} [Lefèvre, Louvet, Muller, Picot, Rideau, ACM TOMS 2023]

Algorithm DWPlusDW

$(s_h, s_\ell) \leftarrow 2\text{Sum}(x_h, y_h)$

$(t_h, t_\ell) \leftarrow 2\text{Sum}(x_\ell, y_\ell)$

$c \leftarrow \text{RN}(s_\ell + t_h)$

$(v_h, v_\ell) \leftarrow \text{Fast2Sum}(s_h, c)$

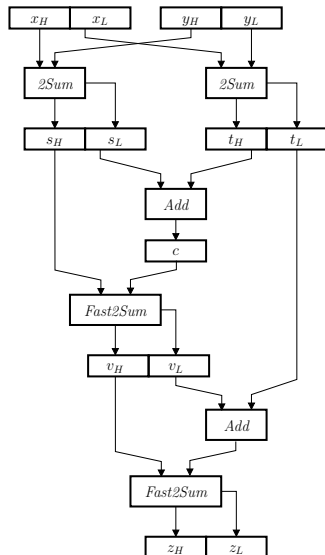
$w \leftarrow \text{RN}(t_\ell + v_\ell)$

$(z_h, z_\ell) \leftarrow \text{Fast2Sum}(v_h, w)$

return (z_h, z_ℓ)

If $p \geq 3$, the relative error is bounded by:

$$\frac{3u^2}{1 - 4u} = 3u^2 + 12u^3 + 48u^4 + \dots$$



Algorithm DWPlusDW

$(s_h, s_\ell) \leftarrow 2\text{Sum}(x_h, y_h)$

$(t_h, t_\ell) \leftarrow 2\text{Sum}(x_\ell, y_\ell)$

$c \leftarrow \text{RN}(s_\ell + t_h)$

$(v_h, v_\ell) \leftarrow \text{Fast2Sum}(s_h, c)$

$w \leftarrow \text{RN}(t_\ell + v_\ell)$

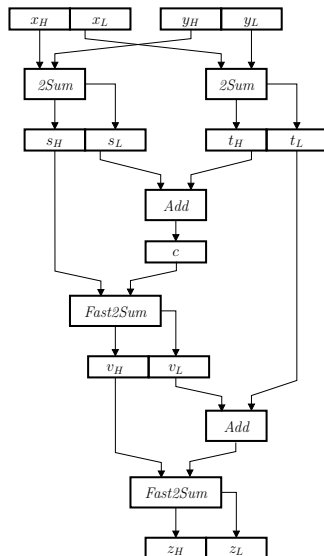
$(z_h, z_\ell) \leftarrow \text{Fast2Sum}(v_h, w)$

return (z_h, z_ℓ)

If $p \geq 3$, the relative error is bounded by:

$$\frac{3u^2}{1 - 4u} = 3u^2 + 12u^3 + 48u^4 + \dots$$

- Claim of an error bound of $2u^2$ (in binary64)
- An example was found with error $\approx 2.25u^2$
- When formalizing [MullerRideau2022], minor error found in the pen and paper proof



Outline

- Floating-Point (FP) Fundamentals and Error-Free Transforms
- Double words
- Multiple words (FP Expansions)
 - VecSumErrBranch – A sketch of a proof
- Triple words
- Conclusion

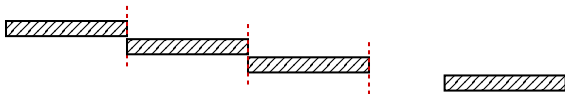
Definition

A **multiword number** (or *floating-point expansion*) represents a real number as an **unevaluated sum** of standard floating-point words:

$$u = u_0 + u_1 + \cdots + u_{n-1},$$

with components ordered by magnitude and **non-overlapping**

$$|u_{i+1}| \leq \text{Func}(u_i).$$



- Each term u_i is a standard / hardware float (e.g., double)
- The higher terms capture rounding residuals from lower ones
- Typical cases: **double-double (2 words)**, **triple-double (3 words)**

[Dekker '71, Knuth '97, Priest '92]

Wobbling Precision: Multi-word vs Multi-digit Representations

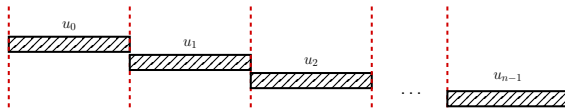
- **Multi-digit representation (fixed precision):**

- long mantissa + a single shared exponent
- arithmetic is **correctly rounded** and **IEEE 754 compatible**
- implemented in arbitrary-precision libraries such as **GNU MPFR**



- **Multiple-word representation (wobbling precision):**

- unevaluated sum of several FP words
- overall precision can **wobble slightly** depending on rounding and overlap
- used in libraries such as **QD**, **CAMPARY**



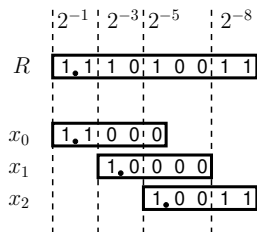
Redundancy in Multiword Representations

Example: $p = 5$ (radix 2)

The real number $R = 1.11010011_2 \times 2^{-1}$ can be represented in several ways:

$$R = x_0 + x_1 + x_2:$$

$$\begin{cases} x_0 = 1.1000 \times 2^{-1}, \\ x_1 = 1.0010 \times 2^{-3}, \\ x_2 = 1.0110 \times 2^{-6}. \end{cases}$$



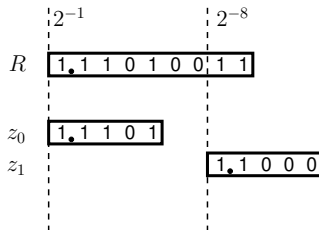
Redundancy in Multiword Representations

Example: $p = 5$ (radix 2)

The real number $R = 1.11010011_2 \times 2^{-1}$ can be represented in several ways:

Most compact $R = z_0 + z_1$:

$$\begin{cases} z_0 = 1.1101 \times 2^{-1}, \\ z_1 = 1.1000 \times 2^{-8}. \end{cases}$$



Redundancy in Multiword Representations

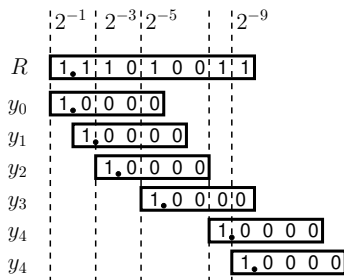
Example: $p = 5$ (radix 2)

The real number $R = 1.11010011_2 \times 2^{-1}$ can be represented in several ways:

Least compact

$$R = y_0 + y_1 + y_2 + y_3 + y_4 + y_5:$$

$$\left\{ \begin{array}{l} y_0 = 1.0000 \times 2^{-1}, \\ y_1 = 1.0000 \times 2^{-2}, \\ y_2 = 1.0000 \times 2^{-3}, \\ y_3 = 1.0000 \times 2^{-5}, \\ y_4 = 1.0000 \times 2^{-8}, \\ y_5 = 1.0000 \times 2^{-9}. \end{array} \right.$$



Redundancy in Multiword Representations

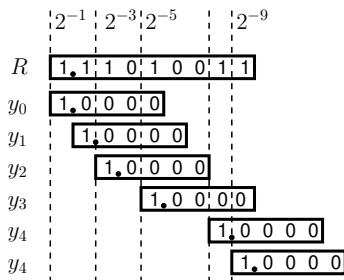
Example: $p = 5$ (radix 2)

The real number $R = 1.11010011_2 \times 2^{-1}$ can be represented in several ways:

Least compact

$$R = y_0 + y_1 + y_2 + y_3 + y_4 + y_5:$$

$$\left\{ \begin{array}{l} y_0 = 1.0000 \times 2^{-1}, \\ y_1 = 1.0000 \times 2^{-2}, \\ y_2 = 1.0000 \times 2^{-3}, \\ y_3 = 1.0000 \times 2^{-5}, \\ y_4 = 1.0000 \times 2^{-8}, \\ y_5 = 1.0000 \times 2^{-9}. \end{array} \right.$$



Redundancy in Multiword Representations

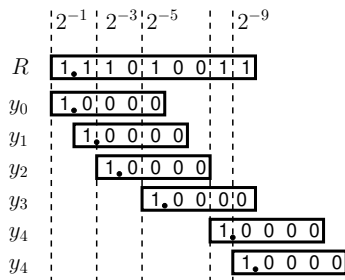
Example: $p = 5$ (radix 2)

The real number $R = 1.11010011_2 \times 2^{-1}$ can be represented in several ways:

Least compact

$$R = y_0 + y_1 + y_2 + y_3 + y_4 + y_5:$$

$$\begin{cases} y_0 = 1.0000 \times 2^{-1}, \\ y_1 = 1.0000 \times 2^{-2}, \\ y_2 = 1.0000 \times 2^{-3}, \\ y_3 = 1.0000 \times 2^{-5}, \\ y_4 = 1.0000 \times 2^{-8}, \\ y_5 = 1.0000 \times 2^{-9}. \end{cases}$$



Key point

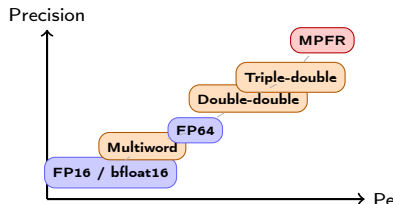
To convey more information \Rightarrow *non-overlapping* representation

\Rightarrow requires (re)normalization algorithms

Why It Matters

Scientific and Engineering Context

- **Precision gap:** applications which require more than 53 bits, but MPFR may be too slow
- **Examples:**
 - Long-time simulations (weather, long time iterations in dynamical systems)
 - Ill-conditioned linear algebra
- Leveraging hardware-compatible, tunable precision.



Multiword arithmetic bridges the gap between standard hardware and full arbitrary precision

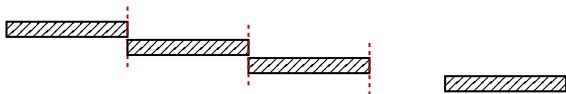
Specification issues

- Each variant (Dekker, Knuth, Priest, etc.) has subtle non-overlapping rules
- Correctness proofs and tight error bounds are **non-trivial but essential**

Nonoverlapping representations

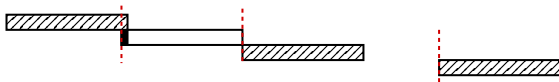
PNonoverlap (Priest's definition)

For an expansion u_0, u_1, \dots, u_{n-1} , we have $|u_i| < \text{ulp}(u_{i-1})$ for all $0 < i < n$.



UlpNonoverlap (Relaxed Priest's definition)

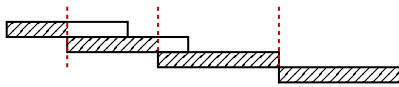
For an expansion u_0, u_1, \dots, u_{n-1} , we have $|u_i| \leq \text{ulp}(u_{i-1})$ for all $0 < i < n$.



Nonoverlapping representations

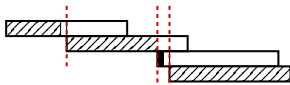
Nonzero-overlapping: SNonoverlap (Shewchuk's definition)

For an expansion u_0, u_1, \dots, u_{n-1} , we have $|u_i| < \text{uls}(u_{i-1})$ for all $0 < i < n$.



Nonzero-overlapping: FNonoverlap (Fabiano's definition)

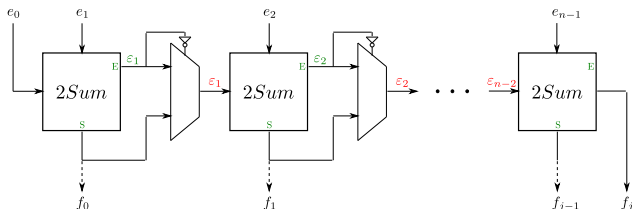
For an expansion u_0, u_1, \dots, u_{n-1} , we have $|u_i| \leq \frac{1}{2} \text{uls}(u_{i-1})$ for all $0 < i < n$.



Outline

- Floating-Point (FP) Fundamentals and Error-Free Transforms
- Double words
- Multiple words (FP Expansions)
 - `VecSumErrBranch` – A sketch of a proof
- Triple words
- Conclusion

VecSumErrBranch



Algorithm 4: VecSumErrBranch

Input: (e_0, \dots, e_{n-1})

Output: (f_0, \dots, f_{j-1})

$j = 0$

$\varepsilon_0 = e_0$

for $i \leftarrow 1$ **to** $n - 1$ **do**

$(s_i, \varepsilon_i) \leftarrow 2Sum(\varepsilon_{i-1}, e_i)$

if $\varepsilon_i \neq 0$ **then**

$f_j = s_i$

$j = j + 1$

else

$\varepsilon_i = s_i$

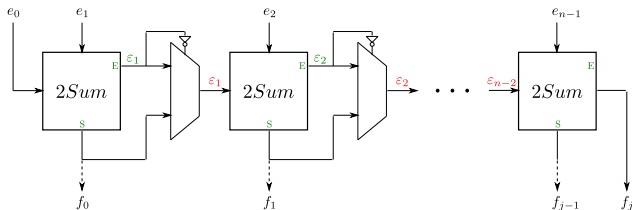
return (f_0, \dots, f_{j-1})

- Error free summation:

$$\sum_{i=0}^{n-1} e_i = \sum_{k=0}^{j-1} f_k$$

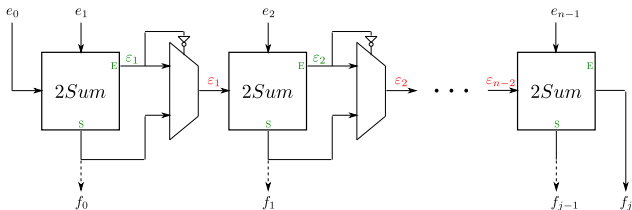
- carries are propagated to the right

Input vs. Output Non-Overlapping in VecSumErrBranch



	Input hypothesis	Output guarantee
Shewchuk	$ e_{i+1} < \text{uls}(e_i)$	$ f_{j+1} \leq \text{ulp}(f_j)$
Fabiano	$ e_{i+1} \leq \frac{1}{2} \text{uls}(e_i)$ and $n \leq p - 1$	$ f_{j+1} < \text{ulp}(f_j)$

Input vs. Output Non-Overlapping in VecSumErrBranch



	Input hypothesis	Output guarantee	Formally Proved!
Shewchuk	$ e_{i+1} < \text{uls}(e_i)$	$ f_{j+1} \leq \text{ulp}(f_j)$	
Fabiano	$ e_{i+1} \leq \frac{1}{2} \text{uls}(e_i)$ and $n \leq p - 1$	$ f_{j+1} < \text{ulp}(f_j)$	

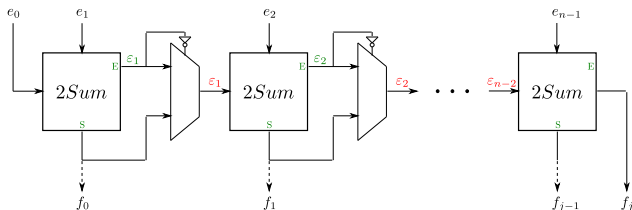
Boldo, Joldes, Muller & Popescu. Formal verification of a floating-point expansion renormalization algorithm, International Conference on Interactive Theorem Proving, 2017.

Fabiano, Muller & Picot. Algorithms for triple-word arithmetic, IEEE Transactions on Computers, 2019.



Photo taken by Jean-Michel at RAIM 2016 in Banyuls.

VecSumErrBranch



Thm. (Shewchuk-Nonoverlap):

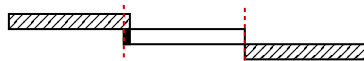
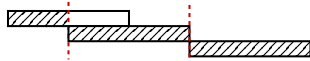
Let $e = (e_0, \dots, e_{n-1})$ with $e_i \in \mathcal{F}$, and $e_i \neq 0$, for $0 \leq i \leq n-1$.

$$\text{SNonoverlap } e \quad \Rightarrow \quad \text{UlpNonoverlap } \underbrace{(\text{VecSumErrBranch } e)}_{(f_0, \dots, f_j)}$$

i.e.,

$$|e_{i+1}| < \text{uls}(e_i) \quad \forall i \quad \Rightarrow$$

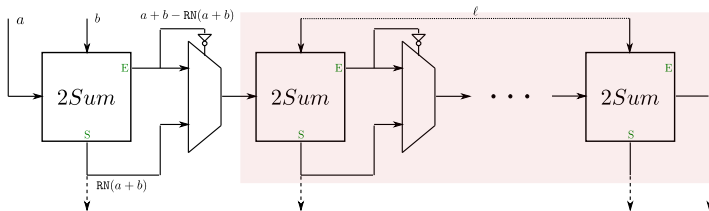
$$|f_{j+1}| \leq \text{ulp}(f_j) \quad \forall j$$



Observations:

- Trivial for at most 2 inputs.
- Interleaving zeros can be safely ignored.

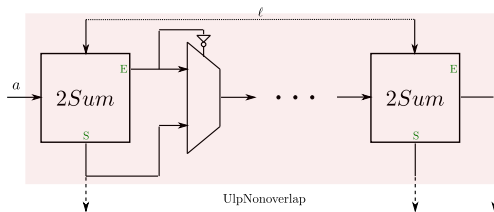
Proof of Thm. Shewchuk-Nonoverlap [ITP2017]



- Proved by **structural induction on tail of input list** $(a :: b :: \ell)$.
- Two elements, a and b are processed at a time, then recursion on a smaller list:

$$\begin{cases} \text{VecSumErrBranch } ((a + b) :: \ell), & \text{if } a + b - RN(a + b) = 0 \\ RN(a + b) :: \text{VecSumErrBranch } (a + b - RN(a + b) :: \ell), & \text{otherwise} \end{cases}$$

Proof of Thm. Shewchuk-Nonoverlap [ITP2017]



- Induction Hypothesis:

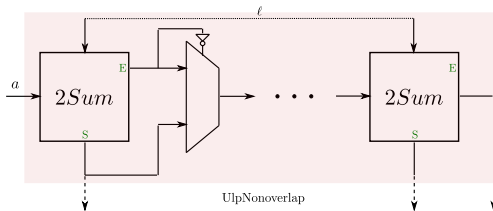
$\forall a,$

Forall nonzero $(a :: \ell),$

Forall FP $(a :: \ell),$

$SNonoverlap(a :: \ell) \rightarrow UlpNonoverlap (VecSumErrBranch (a :: \ell))$

Proof of Thm. Shewchuk-Nonoverlap [ITP2017]



- Induction Hypothesis:

$\forall a,$

Forall nonzero $(a :: \ell),$

Forall FP $(a :: \ell),$

$SNonoverlap(a :: \ell) \rightarrow UlpNonoverlap (VecSumErrBranch (a :: \ell))$

- Additional Lemmas:

$\forall a, b,$

Forall nonzero $(a :: b :: \ell),$

Forall FP $(a :: b :: \ell),$

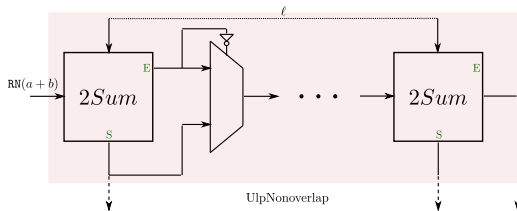
$SNonoverlap(a :: b :: \ell)$

\rightarrow

$RN(a + b) \neq 0,$

$SNonoverlap (RN(a + b) :: \ell),$

Proof of Thm. Shewchuk-Nonoverlap [ITP2017]



- Induction Hypothesis:

$\forall a,$

Forall nonzero $(a :: \ell),$

Forall FP $(a :: \ell),$

$SNonoverlap(a :: \ell) \rightarrow UlpNonoverlap (VecSumErrBranch (a :: \ell))$

- Additional Lemmas:

$\forall a, b,$

Forall nonzero $(a :: b :: \ell),$

Forall FP $(a :: b :: \ell),$

$SNonoverlap(a :: b :: \ell)$

\rightarrow

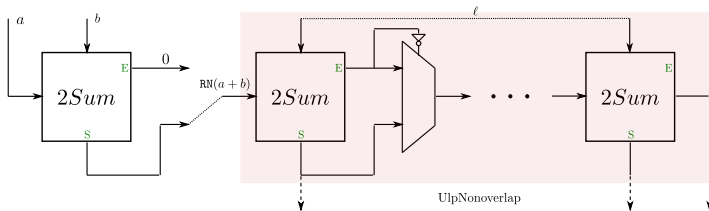
$RN(a+b) \neq 0,$

$SNonoverlap (RN(a+b) :: \ell),$

- Combined with IH:

$\rightarrow UlpNonoverlap (VecSumErrBranch (RN(a+b) :: \ell))$

Proof of Thm. Shewchuk-Nonoverlap [ITP2017]



- Induction Hypothesis:

$\forall a,$

Forall nonzero $(a :: \ell),$

Forall FP $(a :: \ell),$

$\text{SNonoverlap}(a :: \ell) \rightarrow \text{UlpNonoverlap}(\text{VecSumErrBranch } (a :: \ell))$

- Additional Lemmas:

$\forall a, b,$

Forall nonzero $(a :: b :: \ell),$

Forall FP $(a :: b :: \ell),$

$\text{SNonoverlap}(a :: b :: \ell)$

\rightarrow

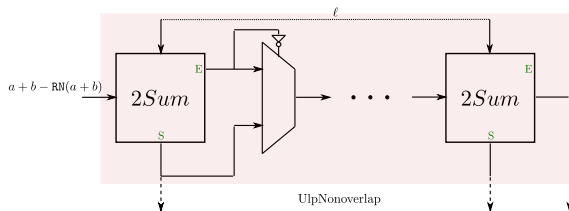
$\text{RN}(a + b) \neq 0,$

$\text{SNonoverlap}(\text{RN}(a + b) :: \ell),$

- Combined with IH:

$\rightarrow \text{UlpNonoverlap}(\text{VecSumErrBranch}(\text{RN}(a + b) :: \ell)) \quad \checkmark (\text{Exact addition case})$

Proof of Thm. Shewchuk-Nonoverlap [ITP2017]



- Induction Hypothesis:

$\forall a,$

Forall nonzero $(a :: \ell),$

Forall FP $(a :: \ell),$

$SNonoverlap(a :: \ell) \rightarrow UlpNonoverlap (VecSumErrBranch (a :: \ell))$

- Additional Lemmas:

$\forall a, b,$

Forall nonzero $(a :: b :: \ell),$

Forall FP $(a :: b :: \ell),$

$SNonoverlap(a :: b :: \ell)$

\rightarrow

$RN(a + b) \neq 0,$

$SNonoverlap (RN(a + b) :: \ell),$

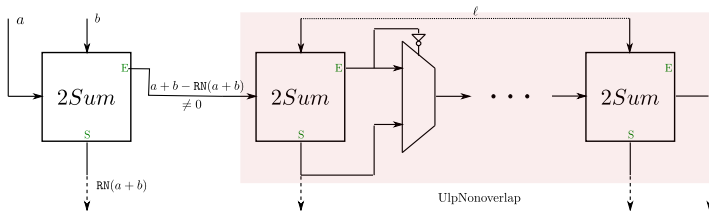
$SNonoverlap(a + b - RN(a + b) :: \ell)$

- Combined with IH:

$\rightarrow UlpNonoverlap (VecSumErrBranch (RN(a + b) :: \ell))$ ✓(Exact addition case)

$\rightarrow UlpNonoverlap (VecSumErrBranch (a + b - RN(a + b) :: \ell))$

Proof of Thm. Shewchuk-Nonoverlap [ITP2017]



- Induction Hypothesis:

$\forall a,$

Forall nonzero $(a :: \ell),$

Forall FP $(a :: \ell),$

$SNonoverlap(a :: \ell) \rightarrow UlpNonoverlap (VecSumErrBranch (a :: \ell))$

- Additional Lemmas:

$\forall a, b,$

Forall nonzero $(a :: b :: \ell),$

Forall FP $(a :: b :: \ell),$

$SNonoverlap(a :: b :: \ell)$

\rightarrow

$RN(a+b) \neq 0,$

$SNonoverlap (RN(a+b) :: \ell),$

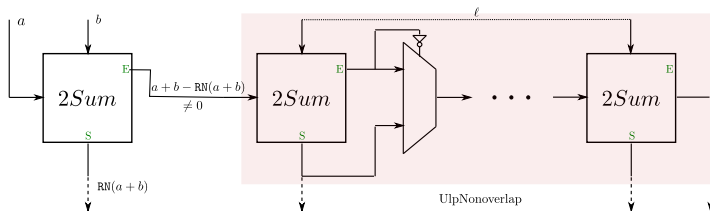
$SNonoverlap(a+b - RN(a+b) :: \ell)$

- Combined with IH:

$\rightarrow UlpNonoverlap (VecSumErrBranch (RN(a+b) :: \ell))$ ✓(Exact addition case)

$\rightarrow UlpNonoverlap (VecSumErrBranch (a+b - RN(a+b) :: \ell))$

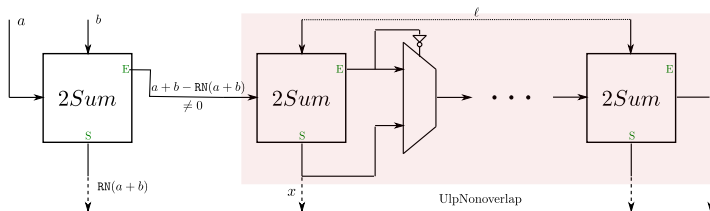
Proof of Thm. Shewchuk-Nonoverlap [ITP2017]



- ✓ Exact addition case follows directly from IH.
- Otherwise, reconstruction of output:

$$RN(a+b) :: \underbrace{\text{VecSumErrBranch}(a+b - RN(a+b) :: \ell)}_{UlpNonoverlap}$$

Proof of Thm. Shewchuk-Nonoverlap [ITP2017]

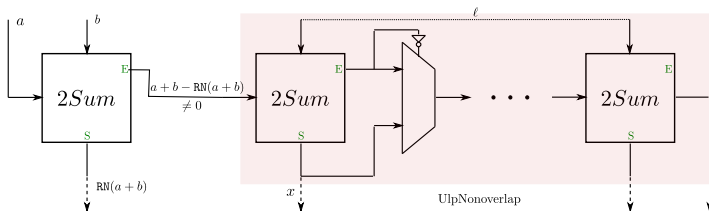


- ✓ Exact addition case follows directly from IH.
- Otherwise, reconstruction of output:

$$\text{RN}(a+b) :: \underbrace{\text{VecSumErrBranch}(a+b - \text{RN}(a+b) :: \ell)}_{(x \quad \dots)}$$

$$x = \text{RN}(a+b - \text{RN}(a+b)) + \sum_{i \leq k} \ell_i \neq 0$$

Proof of Thm. Shewchuk-Nonoverlap [ITP2017]



- ✓ Exact addition case follows directly from IH.
- Otherwise, reconstruction of output:

$$RN(a+b) :: \underbrace{VecSumErrBranch(a+b - RN(a+b) :: \ell)}_{(x \quad \dots)}$$

Prove that:

$$|x| = |RN(a+b - RN(a+b) + \sum_{i \leq k} \ell_i)| \leq ulp(RN(a+b))$$

Proof of Thm. Shewchuk-Nonoverlap [ITP2017]

Need to prove that:

For all nonzero $a, b, \ell_1, \dots, \ell_k \in \mathcal{F}$, with $\text{SNonoverlap}(a, b :: \ell)$ and $a + b - \text{RN}(a + b) \neq 0$, we have

$$|\text{RN}(a + b - \text{RN}(a + b) + \sum_{i \leq k} \ell_i)| \leq \text{ulp}(\text{RN}(a + b)).$$

Recall that:

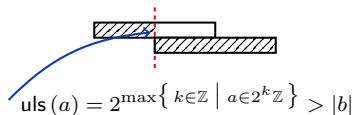
✓ $|a + b - \text{RN}(a + b)| \leq \frac{1}{2} \text{ulp}(a + b) \leq \frac{1}{2} \text{ulp}(\text{RN}(a + b))$

✓ If $y \in \mathcal{F}$ and $x \leq y$, then $\text{RN}(x) \leq y$

It remains to prove that $|\sum_{i \leq k} \ell_i| \leq |a + b - \text{RN}(a + b)|$ under the conditions above.

SNonoverlap property

For all $a, b \in \mathcal{F}$, $a \neq 0$, $\text{uls}(a) > |b|$ iff there exist $e, n \in \mathbb{Z}$ s.t. $a = n 2^e$ and $|b| < 2^e$.



SNonoverlap property

For all nonzero $a, b, c \in \mathcal{F}$,

$\text{SNonoverlap}(a :: b :: c) \implies \text{SNonoverlap}(\text{RN}(a + b) :: c)$

and

$\text{SNonoverlap}(a + b - \text{RN}(a + b) :: c).$

Proof.

$|b| < \text{uls}(a) \iff \text{there exist } e_a \in \mathbb{Z} \text{ and } n_a \in \mathbb{Z}, \text{ s.t. } a = n_a 2^{e_a} \text{ and } |b| < 2^{e_a}$

$|c| < \text{uls}(b) \iff \text{there exist } e_b \in \mathbb{Z} \text{ and } n_b \in \mathbb{Z}, \text{ s.t. } b = n_b 2^{e_b} \text{ and } |c| < 2^{e_b}$

So $|b| < |a|$ and $e_b < e_a$, which implies $\text{RN}(a + b) \in 2^{e_b} \mathbb{Z}$ and $\text{RN}(a + b) \neq 0$.

Hence $|\text{RN}(a + b)| \geq 2^{e_b} > c$.

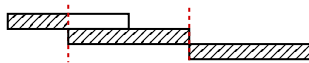
A similar argument gives $a + b - \text{RN}(a + b) \in 2^{e_b} \mathbb{Z}$.

Proof of Thm. Shewchuk-Nonoverlap [ITP2017]

SNonoverlap property

For all nonzero $a, b, c \in \mathcal{F}$,

$$\text{SNonoverlap}(a :: b :: c) \implies |b + c| < \text{uls}(a).$$



Proof of Thm. Shewchuk-Nonoverlap [ITP2017]

SNonoverlap property

For all nonzero $a, b, c \in \mathcal{F}$,

$$\text{SNonoverlap } (a :: b :: c) \implies |b + c| < \text{uls}(a).$$

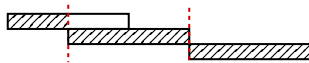
Proof.

$$|b| < \text{uls}(a) \iff \text{there exist } e_a \in \mathbb{Z} \text{ and } n_a \in \mathbb{Z}, \text{ s.t. } a = n_a 2^{e_a} \text{ and } |b| < 2^{e_a}$$

$$|c| < \text{uls}(b) \iff \text{there exist } e_b \in \mathbb{Z} \text{ and } n_b \in \mathbb{Z}, \text{ s.t. } b = n_b 2^{e_b} \text{ and } |c| < 2^{e_b}$$

First note that $e_b < e_a$ and $|n_b| \leq 2^{e_a - e_b} - 1$.

So $|b + c| < |n_b| 2^{e_b} + 2^{e_b} \leq (|n_b| + 1) 2^{e_b} \leq 2^{e_a}$.



Proof of Thm. Shewchuk-Nonoverlap [ITP2017]

SNonoverlap property

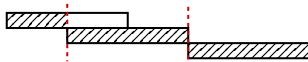
For all nonzero $a, b, c \in \mathcal{F}$,

$$\text{SNonoverlap}(a :: b :: c) \implies |b + c| < \text{uls}(a).$$

SNonoverlap property

For all nonzero $a, \ell_1, \dots, \ell_k \in \mathcal{F}$,

$$\text{SNonoverlap}(a :: \ell_1 :: \dots :: \ell_k) \implies \left| \sum_{i \leq k} \ell_i \right| < \text{uls}(a).$$



Proof of Thm. Shewchuk-Nonoverlap [ITP2017]

SNonoverlap property

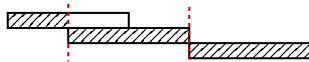
For all nonzero $a, b, c \in \mathcal{F}$,

$$\text{SNonoverlap}(a :: b :: c) \implies |b + c| < \text{uls}(a).$$

SNonoverlap property

For all nonzero $a, \ell_1, \dots, \ell_k \in \mathcal{F}$,

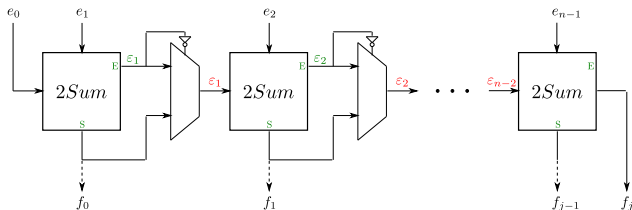
$$\text{SNonoverlap}(a :: \ell_1 :: \dots :: \ell_k) \implies \left| \sum_{i \leq k} \ell_i \right| < \text{uls}(a).$$



$$\text{So, } \text{SNonoverlap}(a + b - \text{RN}(a + b) :: \ell_1 :: \dots :: \ell_k) \implies \left| \sum_{i \leq k} \ell_i \right| < \text{uls}(a + b - \text{RN}(a + b)).$$

$$\text{Finally, } \left| \sum_{i \leq k} \ell_i \right| < |a + b - \text{RN}(a + b)|.$$

Input vs. Output Non-Overlapping in VecSumErrBranch

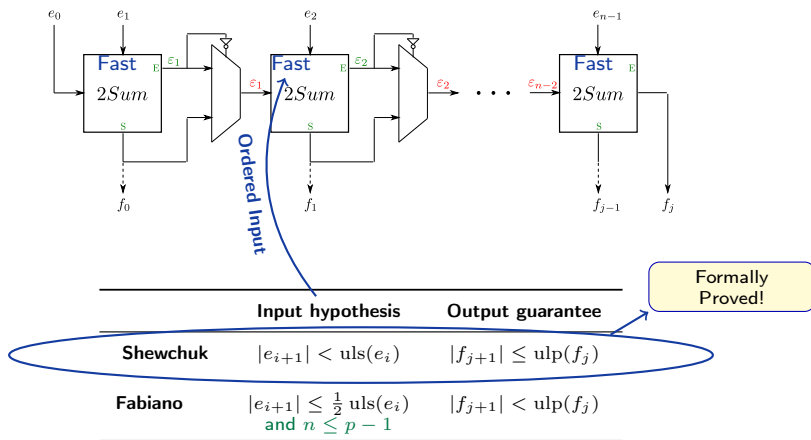


	Input hypothesis	Output guarantee	Formally Proved!
Shewchuk	$ e_{i+1} < \text{uls}(e_i)$	$ f_{j+1} \leq \text{ulp}(f_j)$	Formally Proved!
Fabiano	$ e_{i+1} \leq \frac{1}{2} \text{uls}(e_i)$ and $n \leq p - 1$	$ f_{j+1} < \text{ulp}(f_j)$	

Boldo, Joldes, Muller & Popescu. Formal verification of a floating-point expansion renormalization algorithm, International Conference on Interactive Theorem Proving, 2017.

Fabiano, Muller & Picot. Algorithms for triple-word arithmetic, IEEE Transactions on Computers, 2019.

Input vs. Output Non-Overlapping in VecSumErrBranch



Boldo, Joldes, Muller & Popescu. Formal verification of a floating-point expansion renormalization algorithm, International Conference on Interactive Theorem Proving, 2017.

Fabiano, Muller & Picot. Algorithms for triple-word arithmetic, IEEE Transactions on Computers, 2019.



Yuchen Jin  

@Yuchenj_UW

I saw a guy coding today.

Tab 1 ChatGPT.

Tab 2 Gemini.

Tab 3 Claude.

Tab 4 Grok.

Tab 5 DeepSeek.

He asked every AI the same exact question.

Patiently waited, then pasted each response into
5 different Python files.

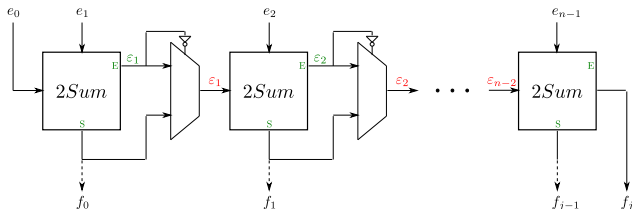
Hit run on all five.

Pick the best one.

Like a psychopath.

It's me.

VecSumErrBranch



Thm. (Shewchuk-Nonoverlap):

Let $e = (e_0, \dots, e_{n-1})$ with $e_i \in \mathcal{F}$, and $e_i \neq 0$, for $0 \leq i \leq n-1$.

$\text{SNonoverlap } e$

\Rightarrow

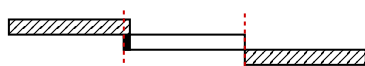
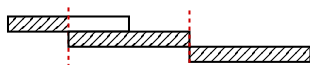
$\text{UlpNonoverlap}(\underbrace{\text{VecSumErrBranch } e}_{(f_0, \dots, f_j)})$

i.e.,

$$|e_{i+1}| < \text{uls}(e_i) \forall i$$

\Rightarrow

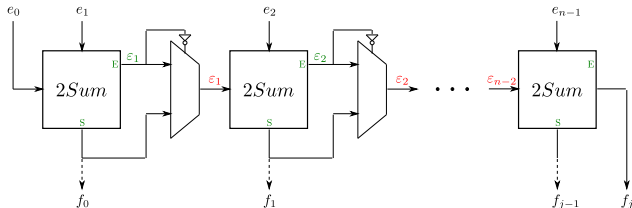
$$|f_{j+1}| \leq \text{ulp}(f_j) \forall j$$



Observations:

- Trivial for at most 2 inputs.
- Interleaving zeros can be safely ignored.

VecSumErrBranch



Thm. (Fabiano-Nonoverlap):

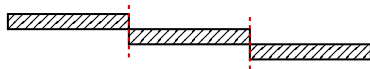
Let $e = (e_0, \dots, e_{n-1})$ with $e_i \in \mathcal{F}$, and $e_i \neq 0$, for $0 \leq i \leq n-1$, $n \leq p-1$.

$$\text{FNonoverlap } e \Rightarrow \text{PNonoverlap}(\underbrace{\text{VecSumErrBranch } e}_{(f_0, \dots, f_j)}),$$

i.e.,

$$|e_{i+1}| \leq \frac{1}{2} \text{uls}(e_i) \forall i \Rightarrow$$

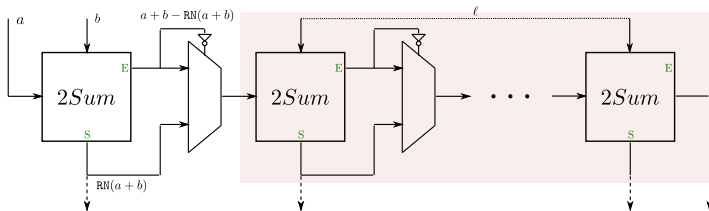
$$|f_{j+1}| < \text{ulp}(f_j) \forall j$$



Observations:

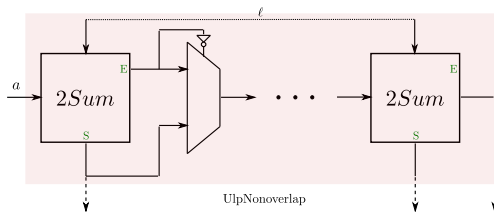
- Trivial for at most 2 inputs.
- Interleaving zeros can be safely ignored.

Proof of Thm. Fabiano-Nonoverlap [Sylvie's code ROCq & Flocq + ChatGPT]



- Proved by **structural induction on tail of input list** $(a :: b :: \ell)$.
- Two elements, a and b are processed at a time, then recursion on a smaller list:

$$\begin{cases} \text{VecSumErrBranch } ((a + b) :: \ell), & \text{if } a + b - RN(a + b) = 0 \\ RN(a + b) :: \text{VecSumErrBranch } (a + b - RN(a + b) :: \ell), & \text{otherwise} \end{cases}$$



• Induction Hypothesis:

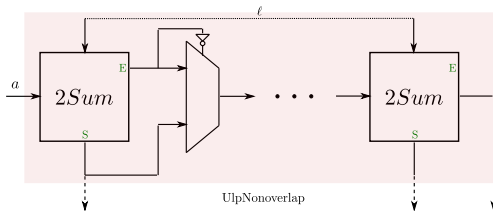
$\forall a,$

Forall nonzero $(a :: \ell),$

Forall FP $(a :: \ell),$

$\text{FNonoverlap } (a :: \ell) \rightarrow \text{PNonoverlap } (\text{VecSumErrBranch } (a :: \ell))$

Proof of Thm. Fabiano-Nonoverlap [Sylvie's code ROCQ & Flocq + ChatGPT]



- Induction Hypothesis:

$\forall a,$

Forall nonzero $(a :: \ell),$

Forall FP $(a :: \ell),$

$\text{FNonoverlap } (a :: \ell) \rightarrow \text{PNonoverlap } (\text{VecSumErrBranch } (a :: \ell))$

- Additional Lemmas:

$\forall a, b,$

Forall nonzero $(a :: b :: \ell),$

Forall FP $(a :: b :: \ell),$

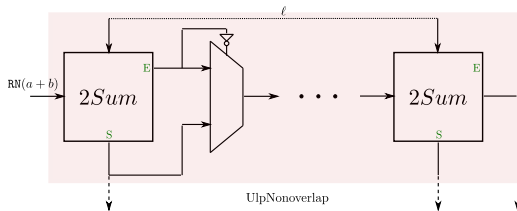
$\text{FNonoverlap}(a :: b :: \ell)$

\rightarrow

$\text{RN } (a + b) \neq 0,$

$\text{FNonoverlap}(\text{RN } (a + b) :: \ell),$

Proof of Thm. Fabiano-Nonoverlap [Sylvie's code ROCQ & Flocq + ChatGPT]



- Induction Hypothesis:

$\forall a,$

Forall nonzero $(a :: \ell),$

Forall FP $(a :: \ell),$

$\text{FNonoverlap } (a :: \ell) \rightarrow \text{PNonoverlap } (\text{VecSumErrBranch } (a :: \ell))$

- Additional Lemmas:

$\forall a, b,$

Forall nonzero $(a :: b :: \ell),$

Forall FP $(a :: b :: \ell),$

$\text{FNonoverlap}(a :: b :: \ell)$

\rightarrow

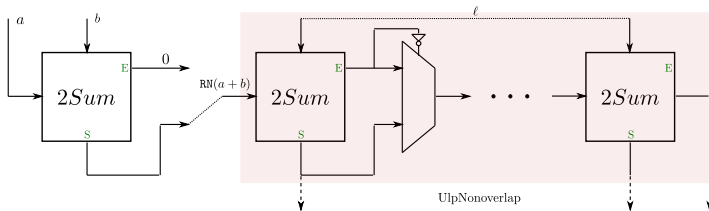
$RN(a+b) \neq 0,$

$\text{FNonoverlap}(RN(a+b) :: \ell),$

- Combined with IH:

$\rightarrow \text{PNonoverlap } (\text{VecSumErrBranch } (RN(a+b) :: \ell))$

Proof of Thm. Fabiano-Nonoverlap [Sylvie's code ROCQ & Flocq + ChatGPT]



- Induction Hypothesis:

$\forall a,$

Forall nonzero $(a :: \ell),$

Forall FP $(a :: \ell),$

$\text{FNonoverlap } (a :: \ell) \rightarrow \text{PNonoverlap } (\text{VecSumErrBranch } (a :: \ell))$

- Additional Lemmas:

$\forall a, b,$

Forall nonzero $(a :: b :: \ell),$

Forall FP $(a :: b :: \ell),$

$\text{FNonoverlap}(a :: b :: \ell)$

\rightarrow

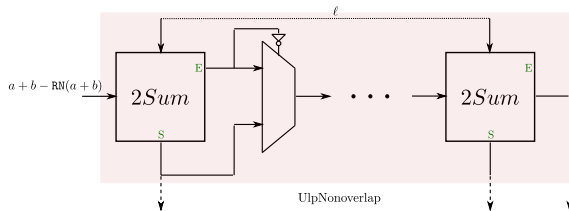
$\text{RN } (a + b) \neq 0,$

$\text{FNonoverlap}(\text{RN } (a + b) :: \ell),$

- Combined with IH:

$\rightarrow \text{PNonoverlap } (\text{VecSumErrBranch } (\text{RN } (a + b) :: \ell)) \quad \checkmark (\text{Exact addition case})$

Proof of Thm. Fabiano-Nonoverlap [Sylvie's code ROCQ & Flocq + ChatGPT]



- Induction Hypothesis:

$\forall a,$

Forall nonzero $(a :: \ell),$

Forall FP $(a :: \ell),$

$\text{FNonoverlap } (a :: \ell) \rightarrow \text{PNonoverlap } (\text{VecSumErrBranch } (a :: \ell))$

- Additional Lemmas:

$\forall a, b,$

Forall nonzero $(a :: b :: \ell),$

Forall FP $(a :: b :: \ell),$

$\text{FNonoverlap}(a :: b :: \ell)$

\rightarrow

$\text{RN } (a + b) \neq 0,$

$\text{FNonoverlap}(\text{RN } (a + b) :: \ell),$

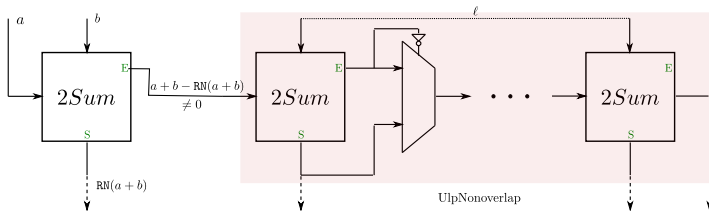
$\text{FNonoverlap}(a + b - \text{RN } (a + b) :: \ell)$

- Combined with IH:

$\rightarrow \text{PNonoverlap } (\text{VecSumErrBranch } (\text{RN } (a + b) :: \ell)) \quad \checkmark (\text{Exact addition case})$

$\rightarrow \text{PNonoverlap } (\text{VecSumErrBranch } (a + b - \text{RN } (a + b) :: \ell))$

Proof of Thm. Fabiano-Nonoverlap [Sylvie's code ROCQ & Flocq + ChatGPT]



- Induction Hypothesis:

$\forall a,$

Forall nonzero $(a :: \ell),$

Forall FP $(a :: \ell),$

$\text{FNonoverlap } (a :: \ell) \rightarrow \text{PNonoverlap } (\text{VecSumErrBranch } (a :: \ell))$

- Additional Lemmas:

$\forall a, b,$

Forall nonzero $(a :: b :: \ell),$

Forall FP $(a :: b :: \ell),$

$\text{FNonoverlap}(a :: b :: \ell)$

\rightarrow

$RN(a+b) \neq 0,$

$\text{FNonoverlap}(RN(a+b) :: \ell),$

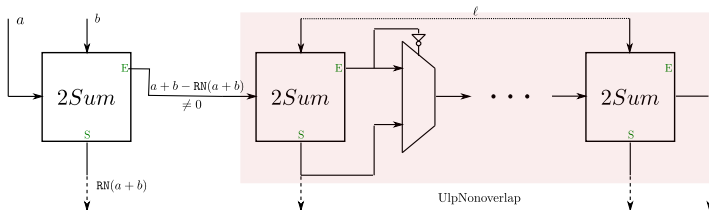
$\text{FNonoverlap}(a+b - RN(a+b) :: \ell)$

- Combined with IH:

$\rightarrow \text{PNonoverlap } (\text{VecSumErrBranch } (RN(a+b) :: \ell))$ ✓(Exact addition case)

$\rightarrow \text{PNonoverlap } (\text{VecSumErrBranch } (a+b - RN(a+b) :: \ell))$

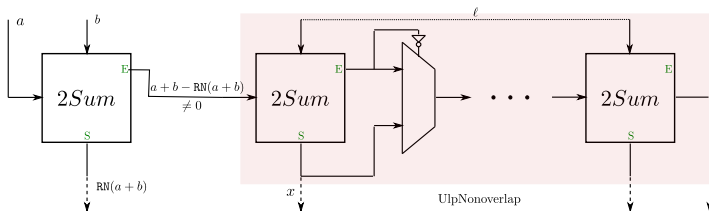
Proof of Thm. Fabiano-Nonoverlap [Sylvie's code ROCQ & Flocq + ChatGPT]



- ✓ Exact addition case follows directly from IH.
- Otherwise, reconstruction of output:

$$RN(a+b) :: \underbrace{\text{VecSumErrBranch}(a+b - RN(a+b) :: \ell)}_{\text{PNonoverlap}}$$

Proof of Thm. Fabiano-Nonoverlap [Sylvie's code ROCQ & Flocq + ChatGPT]

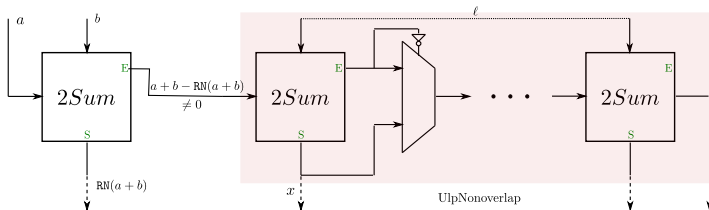


- ✓ Exact addition case follows directly from IH.
- Otherwise, reconstruction of output:

$$RN(a+b) :: \underbrace{\text{VecSumErrBranch}(a+b - RN(a+b) :: \ell)}_{(x \quad \dots)}$$

$$x = RN(a+b - RN(a+b)) + \sum_{i \leq k} \ell_i \neq 0$$

Proof of Thm. Fabiano-Nonoverlap [Sylvie's code ROCQ & Flocq + ChatGPT]



- ✓ Exact addition case follows directly from IH.
- Otherwise, reconstruction of output:

$$\text{RN}(a+b) :: \underbrace{\text{VecSumErrBranch}(a+b - \text{RN}(a+b) :: \ell)}_{(x \quad \dots)}$$

Prove that:

$$|x| = |\text{RN}(a+b - \text{RN}(a+b) + \sum_{i \leq k} \ell_i)| < \text{ulp}(\text{RN}(a+b))$$

Need to prove that:

For all nonzero $a, b, \ell_1, \dots, \ell_k \in \mathcal{F}$, with $\text{FNonoverlap}(a, b :: \ell)$ and $a + b - \text{RN}(a + b) \neq 0$, we have

$$|\text{RN}(a + b - \text{RN}(a + b) + \sum_{i \leq k} \ell_i)| < \text{ulp}(\text{RN}(a + b)).$$

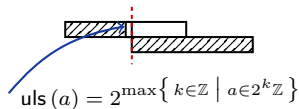
Recall that:

$$\checkmark |a + b - \text{RN}(a + b)| \leq \frac{1}{2} \text{ulp}(a + b) \leq \frac{1}{2} \text{ulp}(\text{RN}(a + b))$$

\checkmark If $y \in \mathcal{F}$ and $x \leq y$, then $\text{RN}(x) \leq y$... not very useful directly...

FNonoverlap property

For all $a, b \in \mathcal{F}$, $a \neq 0$, $\frac{1}{2} \text{uls}(a) \geq |b|$ iff there exist $e, n \in \mathbb{Z}$ s.t. $a = n 2^e$ and $|b| \leq 2^{e-1}$.



$$\text{uls}(a) = 2^{\max\{k \in \mathbb{Z} \mid a \in 2^k \mathbb{Z}\}}$$

FNonoverlap property

For all nonzero $a, b, c \in \mathcal{F}$,

$$\begin{aligned} \text{FNonoverlap}(a :: b :: c) \implies & \text{FNonoverlap}(\text{RN}(a + b) :: c) \\ & \text{and} \\ & \text{FNonoverlap}(a + b - \text{RN}(a + b) :: c). \end{aligned}$$

Proof.

$$|b| \leq \frac{1}{2} \text{uls}(a) \iff \text{there exist } e_a \in \mathbb{Z} \text{ and } n_a \in \mathbb{Z}, \text{ s.t. } a = n_a 2^{e_a} \text{ and } |b| \leq 2^{e_a-1}$$

$$|c| \leq \frac{1}{2} \text{uls}(b) \iff \text{there exist } e_b \in \mathbb{Z} \text{ and } n_b \in \mathbb{Z}, \text{ s.t. } b = n_b 2^{e_b} \text{ and } |c| \leq 2^{e_b-1}$$

So $|b| \leq \frac{1}{2}|a|$ and $e_b \leq e_a - 1$, which implies $\text{RN}(a + b) \in 2^{e_b} \mathbb{Z}$ and $\text{RN}(a + b) \neq 0$.

A similar argument gives $a + b - \text{RN}(a + b) \in 2^{e_b} \mathbb{Z}$.

SNonoverlap property

For all nonzero $a, b, c \in \mathcal{F}$,

$$\text{SNonoverlap } (a :: b :: c) \implies |b + c| < \text{uls}(a).$$

FNonoverlap property

For all nonzero $a, b, c \in \mathcal{F}$,

$$\text{FNonoverlap } (a :: b :: c) \implies |b + c| \not\leq \frac{1}{2} \text{uls}(a).$$

Example: $b = \frac{1}{2} \text{uls}(a)$ and $c > 0$.



Realizing this little change implies modifying some Rocq code...
(that I barely understand)

Proof of Thm. Fabiano-Nonoverlap [Sylvie's code ROCQ & Flocq + ChatGPT]

FNonoverlap property

For all nonzero $a, \ell_1, \dots, \ell_k \in \mathcal{F}$,

$$\text{FNonoverlap } (a :: \ell_1 :: \dots :: \ell_k) \implies \left| a + \sum_{i \leq k} \ell_i \right| \leq |a| (2 - 2^{-k}).$$

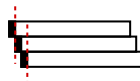
Proof.

$$|\ell_1| \leq \frac{1}{2} |a|$$

$$|\ell_2| \leq \frac{1}{4} |a|$$

...

$$|\ell_k| \leq \frac{1}{2^k} |a|$$



Proof of Thm. Fabiano-Nonoverlap [Sylvie's code ROCQ & Flocq + ChatGPT]

FNonoverlap property

For all nonzero $a, \ell_1, \dots, \ell_k \in \mathcal{F}$,

$$\text{FNonoverlap } (a :: \ell_1 :: \dots :: \ell_k) \implies \left| a + \sum_{i \leq k} \ell_i \right| \leq |a| (2 - 2^{-k}).$$

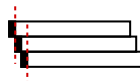
Proof.

$$|\ell_1| \leq \frac{1}{2} |a|$$

$$|\ell_2| \leq \frac{1}{4} |a|$$

...

$$|\ell_k| \leq \frac{1}{2^k} |a|$$



So, $\text{FNonoverlap } (a + b - \text{RN } (a + b) :: \ell_1 :: \dots :: \ell_k) \implies$

$$\left| a + b - \text{RN } (a + b) + \sum_{i \leq k} \ell_i \right| \leq \underbrace{\text{ulp } (\text{RN } (a + b))}_{\text{ulp } (\text{RN } (a + b))} (1 - 2^{-k-1}).$$

Proof of Thm. Fabiano-Nonoverlap [Sylvie's code ROCQ & Flocq + ChatGPT]

FNonoverlap property

For all nonzero $a, \ell_1, \dots, \ell_k \in \mathcal{F}$,

$$\text{FNonoverlap}(a :: \ell_1 :: \dots :: \ell_k) \implies \left| a + \sum_{i \leq k} \ell_i \right| \leq |a| (2 - 2^{-k}).$$

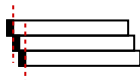
Proof.

$$|\ell_1| \leq \frac{1}{2} |a|$$

$$|\ell_2| \leq \frac{1}{4} |a|$$

...

$$|\ell_k| \leq \frac{1}{2^k} |a|$$



So, $\text{FNonoverlap}(a + b - \text{RN}(a + b) :: \ell_1 :: \dots :: \ell_k) \implies$

$$\left| a + b - \text{RN}(a + b) + \sum_{i \leq k} \ell_i \right| \leq \underbrace{\text{ulp}(\text{RN}(a + b)) (1 - 2^{-k-1})}_{\in \mathcal{F}, \text{ when } k+1 \leq p}.$$

✓ If $y \in \mathcal{F}$ and $x \leq y$, then $\text{RN}(x) \leq y$.

Proof of Thm. Fabiano-Nonoverlap [Sylvie's code ROCQ & Flocq + ChatGPT]

FNonoverlap property

For all nonzero $a, \ell_1, \dots, \ell_k \in \mathcal{F}$,

$$\text{FNonoverlap } (a :: \ell_1 :: \dots :: \ell_k) \implies \left| a + \sum_{i \leq k} \ell_i \right| \leq |a| (2 - 2^{-k}).$$

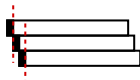
Proof.

$$|\ell_1| \leq \frac{1}{2} |a|$$

$$|\ell_2| \leq \frac{1}{4} |a|$$

...

$$|\ell_k| \leq \frac{1}{2^k} |a|$$



So, $\text{FNonoverlap } (a + b - \text{RN}(a + b) :: \ell_1 :: \dots :: \ell_k) \implies$

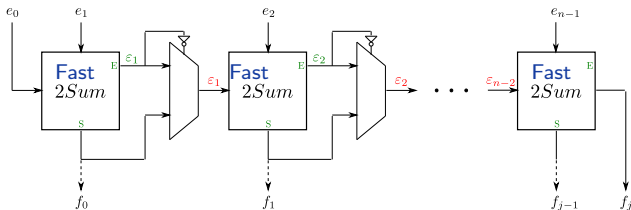
$$\left| a + b - \text{RN}(a + b) + \sum_{i \leq k} \ell_i \right| \leq \underbrace{\text{ulp}(\text{RN}(a + b)) \left(1 - 2^{-k-1} \right)}_{\in \mathcal{F}, \text{ when } k+1 \leq p}.$$

✓ If $y \in \mathcal{F}$ and $x \leq y$, then $\text{RN}(x) \leq y$.

Finally,

$$\left| \text{RN}(a + b - \text{RN}(a + b) + \sum_{i \leq k} \ell_i) \right| < \text{ulp}(\text{RN}(a + b)).$$

Input vs. Output Non-Overlapping in VecSumErrBranch



Formally
Proved!

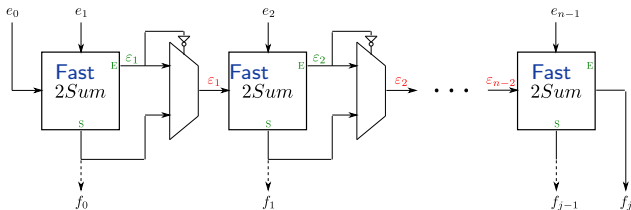
	Input hypothesis	Output guarantee
Shewchuk	$ e_{i+1} < \text{uls}(e_i)$	$ f_{j+1} \leq \text{ulp}(f_j)$
Fabiano	$ e_{i+1} \leq \frac{1}{2} \text{uls}(e_i)$ and $n \leq p - 1$	$ f_{j+1} < \text{ulp}(f_j)$

Note: The proof also handles subnormals without any issues.

Boldo, Joldes, Muller & Popescu. Formal verification of a floating-point expansion renormalization algorithm, International Conference on Interactive Theorem Proving, 2017.

Fabiano, Muller & Picot. Algorithms for triple-word arithmetic, IEEE Transactions on Computers, 2019.

Input vs. Output Non-Overlapping in VecSumErrBranch



Formally
Proved!

	Input hypothesis	Output guarantee
Shewchuk	$ e_{i+1} < \text{uls}(e_i)$	$ f_{j+1} \leq \text{ulp}(f_j)$
Fabiano	$ e_{i+1} \leq \frac{1}{2} \text{uls}(e_i)$ and $n \leq p - 1$	$ f_{j+1} < \text{ulp}(f_j)$

Note: The proof also handles subnormals without any issues.

Boldo, Joldes, Muller & Popescu. Formal verification of a floating-point expansion renormalization algorithm, International Conference on Interactive Theorem Proving, 2017.

Fabiano, Muller & Picot. Algorithms for triple-word arithmetic, IEEE Transactions on Computers, 2019.

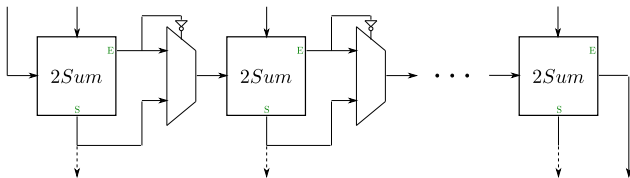
ChatGPT
+
Existing Rocq Code
+
me



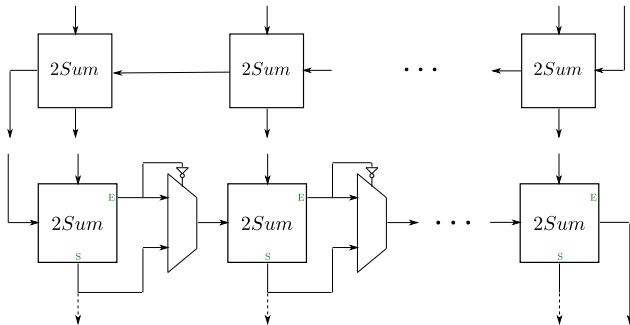
Jean-Michel
+
pen and paper proofs

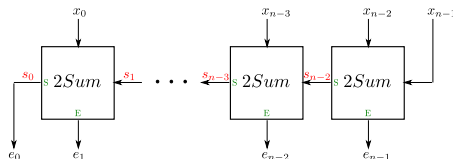


Renormalisation Algorithms



Renormalisation Algorithms





Algorithm 5: VecSum

Input: x_0, \dots, x_{n-1}

Output: e_0, \dots, e_{n-1}

for $i \leftarrow n - 1$ **down to** 1 **do**

$(s_{i-1}, e_i) \leftarrow 2Sum(x_i, x_{i-1})$

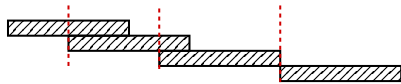
$e_0 \leftarrow s_0$

return e_0, \dots, e_{n-1}

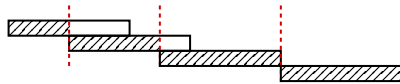
Thm.

If $x_0, \dots, x_{n-1} \in \mathcal{F}$ overlap by at most $d \leq p - 2$ digits
 $\rightarrow \text{SNonoverlap } (e_0, \dots, e_{n-1})$.

x



e



Proof of the *VecSum* algorithm property Here is some excerpt of the proof in [10]:

$$\begin{aligned} & |x_{j+1}| + |x_{j+2}| + \dots \leq \\ & \leq [2^d + 2^{2d-p} + 2^{3d-2p} + 2^{4d-3p} + \dots] \text{ulp}(x_j) \\ & \leq 2^d \cdot 2^p / (2^p - 1) \cdot \text{ulp}(x_j). \end{aligned}$$

This is partly wrong! In fact the geometric series should be bounded by:

$$2^d + 2^{2d-p} + 2^{3d-2p} + 2^{4d-3p} + \dots \leq 2^d / (1 - 2^{d-p}).$$

The proof can be fixed as the two inequalities were coarse enough, so there is no problem at this point.

[10] is Joldes, Marty, Muller, Popescu. Arithmetic algorithms for extended precision using floating-point expansions, IEEE TC, 2015

Boldo, Joldes, Muller & Popescu. Formal verification of a floating-point expansion renormalization algorithm, International Conference on Interactive Theorem Proving, 2017.

2.1 VecSum

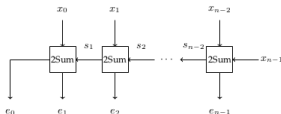
The VecSum algorithm (Algorithm 4) first appears as a part of Priest's normalization algorithm [26]. The name "VecSum" was coined by Ogita et al [24]. The aim of this algorithm is to turn a sequence that is "slightly" nonoverlapping into one that is "more" nonoverlapping, with no error. It is illustrated Fig. 1.

Algorithm 4 – VecSum(x_0, \dots, x_{n-1}). ($6n - 6$ operations)

Ensure: $e_0 + \dots + e_{n-1} = x_0 + \dots + x_{n-1}$

```

 $s_{n-1} \leftarrow x_{n-1}$ 
for  $i = n - 2$  to  $0$  do
     $s_i, e_{i+1} \leftarrow \text{2Sum}(x_i, s_{i+1})$ 
end for
 $e_0 \leftarrow s_0$ 
return ( $e_0, e_1, \dots, e_{n-1}$ )
    
```



... and no formal proof yet!

or $\mathcal{L}u$, that must be the case for one of the $x_j, j \leq i - \mathcal{L}$. In particular, we have $2^{k_j} \leq \frac{1}{2}$, hence $2^{k_{i-2}} \leq \frac{1}{2}$, so $2^{k_{i-1}} \leq \frac{1}{4}$, which contradicts $|e_i| \leq 2u2^{k_{i-1}}$. \square

The conditions on the input of Theorem 1 are complex, so we will use the following corollary:

Corollary 1. Assume that we have $I \subset [1, n - 2]$ with no 2 consecutive indices such that

$$\forall i \in [0, n - 2], i \notin I, \text{ufp}(x_{i+1}) \leq \frac{1}{2} \text{ufp}(x_i),$$

and

$$\forall i \in I, \text{ufp}(x_{i+1}) \leq 2^{p-2} \text{uls}(x_i) \text{ and } \text{ufp}(x_{i+1}) \leq \frac{1}{4} \text{ufp}(x_{i-1})$$

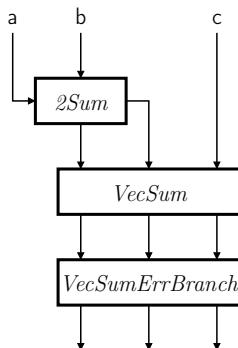
Then $\text{VecSum}(x_0, \dots, x_{n-1})$ is F -nonoverlapping with the same sum. In this case, Fast2Sum can be used instead of 2Sum , so that Algorithm 4 only costs $3n - 3$ operations.

Proof: For $i \notin I$, we take $k_i = e_{x_i}$ the canonical exponent, and for $i \in I$, we take $k_i = \max(k_{i+1} + 1, e_{x_i})$. This is possible because: $2^{k_{i+1} - p + 2} |x_i|$ and $2^{e_{x_i} - p + 1} |x_i|$, which imply $2^{k_i - p + 1} |x_i|$, and $|x_i| \leq 2 \cdot 2^{e_{x_i}}$, which imply $|x_i| \leq 2 \cdot 2^{k_i}$.

For $i, i + 1 \notin I$, we have $k_{i+1} \leq k_i - 1$. For $i \in I$, we have on one hand $k_{i+1} \leq k_i - 1$, and on the other hand $e_{x_i} \leq k_{i-1} - 1$ and $k_{i+1} \leq k_{i-1} - 2$ so $k_i \leq k_{i-1} - 1$. \square

Triple-word (TW) algorithms

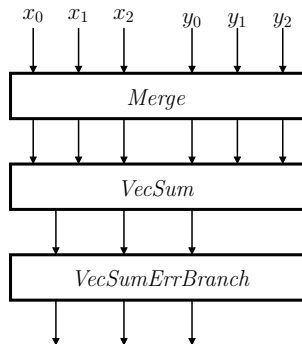
To-TW



If $a, b, c \in \mathcal{F}$, then $\text{To-TW}(a, b, c)$ is a TW, provided that $p \geq 4$.

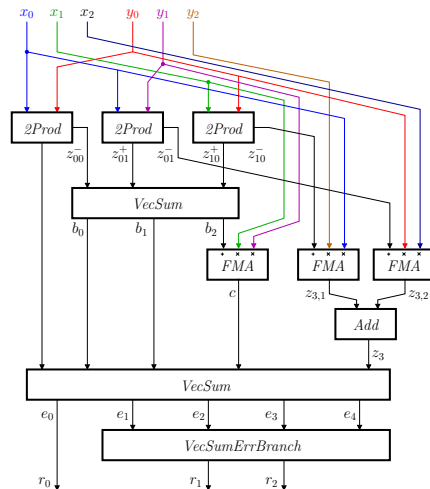
Triple-word (TW) algorithms

TWSum



Triple-double algorithms

3ProdAcc



Outline

- Floating-Point (FP) Fundamentals and Error-Free Transforms
- Double words
- Multiple words (FP Expansions)
 - `VecSumErrBranch` – A sketch of a proof
- Triple words
- Conclusion

- An insight into the fundamentals of EFT-based algorithms for Multiword arithmetic

Conclusion

- An insight into the fundamentals of EFT-based algorithms for Multiword arithmetic
- Many (interesting) variants were left unexplored in this talk e.g., use of FD2, FD2A, ADD3 cf. [Hubrecht, Jeannerod, Muller, 2024]

Conclusion

- An insight into the fundamentals of EFT-based algorithms for Multiword arithmetic
- Many (interesting) variants were left unexplored in this talk e.g., use of FD2, FD2A, ADD3 cf. [Hubrecht, Jeannerod, Muller, 2024]
- Subtle non-overlapping rules
- Correctness proofs and tight error bounds are **non-trivial but essential** in a scientific computing context

Conclusion

- An insight into the fundamentals of EFT-based algorithms for Multiword arithmetic
- Many (interesting) variants were left unexplored in this talk e.g., use of FD2, FD2A, ADD3 cf. [Hubrecht, Jeannerod, Muller, 2024]
- Subtle non-overlapping rules
- Correctness proofs and tight error bounds are **non-trivial but essential** in a scientific computing context
- Complementing pen and paper with formal proofs seems indispensable – yet we are not at a stage to fully automate this

Conclusion

- An insight into the fundamentals of EFT-based algorithms for Multiword arithmetic
- Many (interesting) variants were left unexplored in this talk e.g., use of FD2, FD2A, ADD3 cf. [Hubrecht, Jeannerod, Muller, 2024]
- Subtle non-overlapping rules
- Correctness proofs and tight error bounds are **non-trivial but essential** in a scientific computing context
- Complementing pen and paper with formal proofs seems indispensable – yet we are not at a stage to fully automate this
- <https://homepages.laas.fr/mmjoldes/stageM2Arith.html>

This meme is trivial and has been left
as an exercise to the audience.