Approximating Floating-Point Addition Using The Geometric Mean

Theodor Lindberg and Oscar Gustafsson

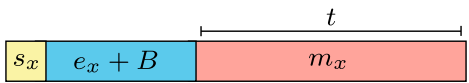
LINKÖPING UNIVERSITY

Floating-point system

A normalized FP number can be expressed as

$$x = (-1)^{s_x} 2^{e_x} (1 + m_x),$$

and stored in hardware as

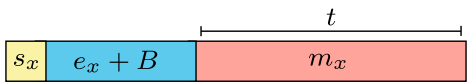


Floating-point system

A normalized FP number can be expressed as

$$x = (-1)^{s_x} 2^{e_x} (1 + m_x),$$

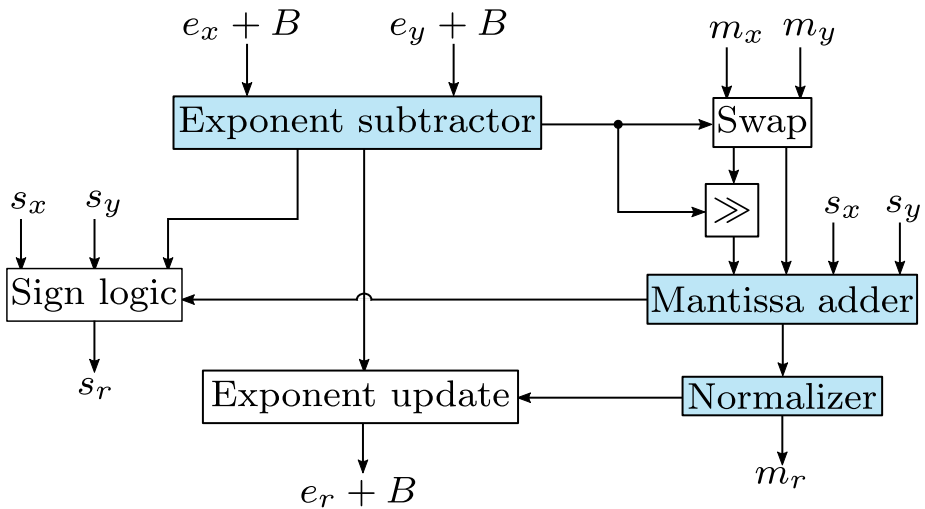
and stored in hardware as



Floating-point addition:

$$\begin{aligned} x \pm |y| &= (-1)^{s_x} 2^{e_x} (1 + m_x) \pm 2^{e_y} (1 + m_y) \\ &= (-1)^{s_x} 2^{e_x} \left(1 + m_x \pm 2^{e_y - e_x} (1 + m_y)\right) \end{aligned}$$

Previous work–Approximate parts in exact designs



This work

- Fundamentally different approach
  - Based on the geometric mean
  - Integer-only operations

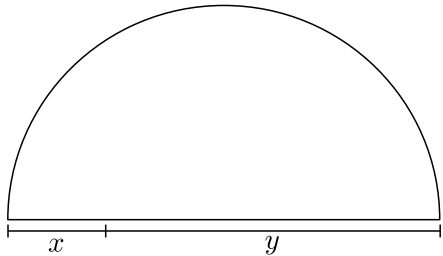
This work

- Fundamentally different approach
  - Based on the geometric mean
  - Integer-only operations
- Suitable for both hardware and software
  - Unique in being applicable for software
  - Usable alongside existing soft FP libraries, e.g. FLIP

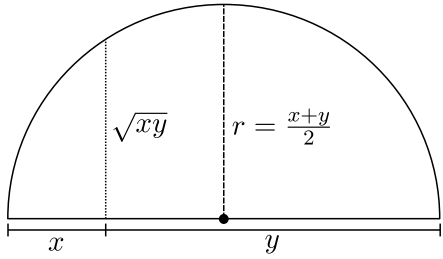
This work

- Fundamentally different approach
  - Based on the geometric mean
  - Integer-only operations
- Suitable for both hardware and software
  - Unique in being applicable for software
  - Usable alongside existing soft FP libraries, e.g. FLIP
- Restricted to positive inputs, still many applications
  - Squares, e.g. vectors norms: $\sqrt{\sum_{j=1} {x_j}^2}$
  - Exponentials, e.g. softmax: $\frac{e^{x_i}}{\sum_{j=1} e^{x_j}}$
  - Certain data, e.g. pixel values

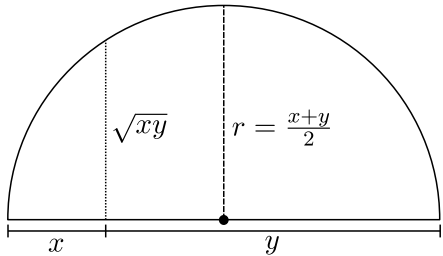
The arithmetic and geometric mean



The arithmetic and geometric mean

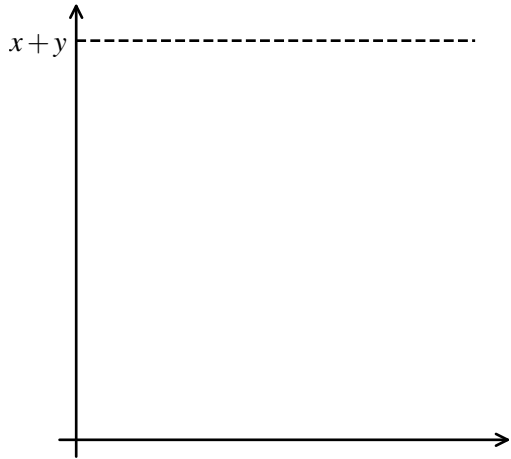


The arithmetic and geometric mean

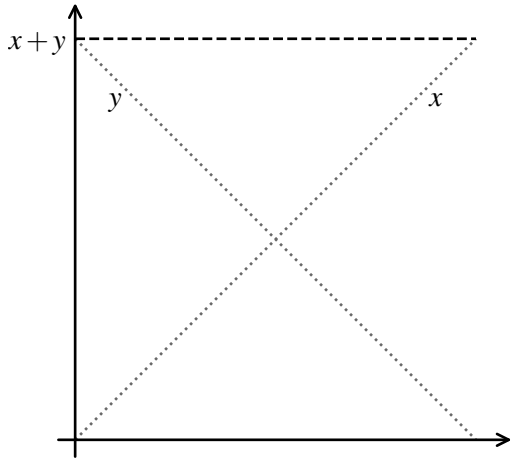


$$x + y = 2r \approx 2\sqrt{xy}$$

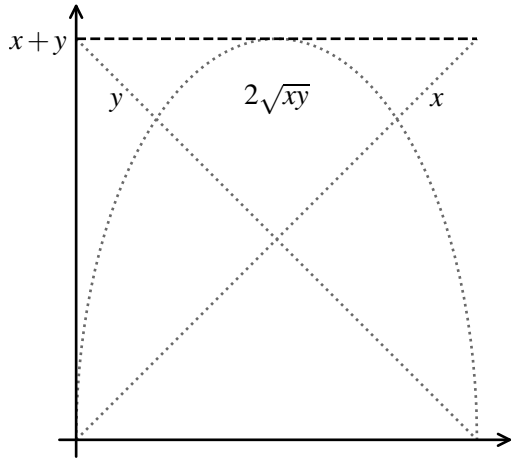
Proposed approach



Proposed approach

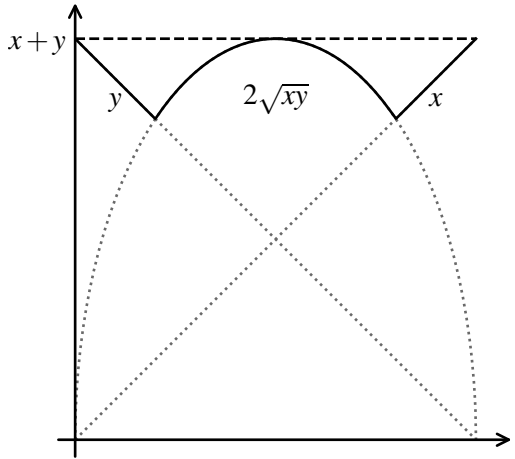


Proposed approach



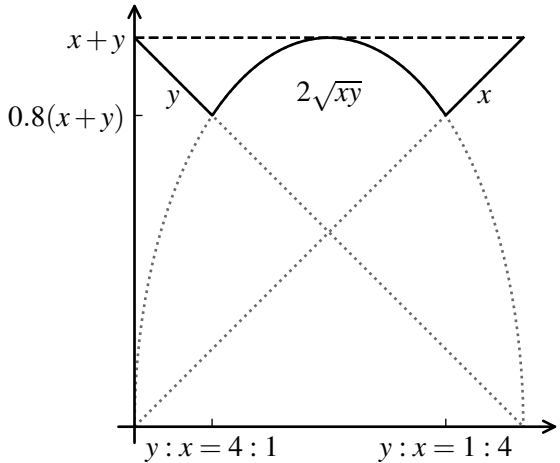
Proposed approach

$$x + y \approx \max\left(x,\, y,\, 2\sqrt{xy}\right)$$



Proposed approach

$$x + y \approx \max\left(x,\, y,\, 2\sqrt{xy}\right)$$



Implementation of $\max\left(x,\, y,\, 2\sqrt{xy}\right)$

1. Approximate to logarithmic fixed-point numbers
2. Perform computations
   - Multiplication $\Rightarrow$ addition, square-root $\Rightarrow$ right shift
3. Approximate back to floating-point

Mitchell's logarithm approximation

Based on

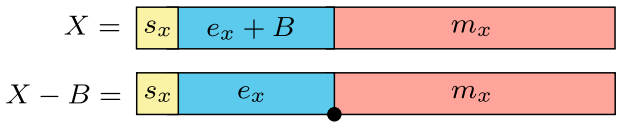
$$\log_2\left(2^e(1+m)\right) = e + \log_2(1+m) \approx e + m,\ 0 \leqslant m < 1$$

Mitchell's logarithm approximation

Based on

$$\log_2\left(2^e(1+m)\right) = e + \log_2(1+m) \approx e + m,\ 0 \leqslant m < 1$$

In practice:



$$(-1)^{s_x} \log_2 |x| \approx X - B$$

Implementation in the logarithmic domain

Operation	Mathematical notation	Base expression[1]
Multiplication	$x \times y$	$X + Y - B$
Multiplication by 2	$2x$	$X + (1 \ll t)$
Square root	$\sqrt{x}$	$(X + B) \gg 1$
Maximum	$\max(x,\, y)$	$\max(X,\, Y)$

[1] O. Gustafsson and N. Hellman, "Approximate floating-point operations with integer units by processing in the logarithmic domain", in *Proc. IEEE Symp. Comput. Arithmetic*, 2021

Implementation in the logarithmic domain

Operation	Mathematical notation	Base expression[1]
Multiplication	$x \times y$	$X + Y - B$
Multiplication by 2	$2x$	$X + (1 \ll t)$
Square root	$\sqrt{x}$	$(X + B) \gg 1$
Maximum	$\max(x,\, y)$	$\max(X,\, Y)$

This gives

$$\max\left(x,\, y,\, 2\sqrt{xy}\right) \Rightarrow \max\left(X,\, Y,\, (X + Y + C) \gg 1\right)$$

where $C = 2 \ll t$

[1] O. Gustafsson and N. Hellman, "Approximate floating-point operations with integer units by processing in the logarithmic domain", in *Proc. IEEE Symp. Comput. Arithmetic*, 2021

Derived numerical properties

$$\max\left(X,\, Y,\, (X + Y + C) \gg 1\right)$$

Derived numerical properties

$$\max\left(X,\, Y,\, (X + Y + C) \gg 1\right)$$

- Maximum relative error in closed-form:

$$-\frac{1+u}{5+u},$$

where $u = 2^{-t}$. Tends to -0.2 as t increases.

Derived numerical properties

$$\max\left(X,\, Y,\, (X + Y + C) \gg 1\right)$$

- Maximum relative error in closed-form:

$$-\frac{1+u}{5+u},$$

  where $u = 2^{-t}$. Tends to -0.2 as t increases.
- $C = (2 \ll t) + 1 \Rightarrow$ maximum relative error is exactly -0.2

Derived numerical properties

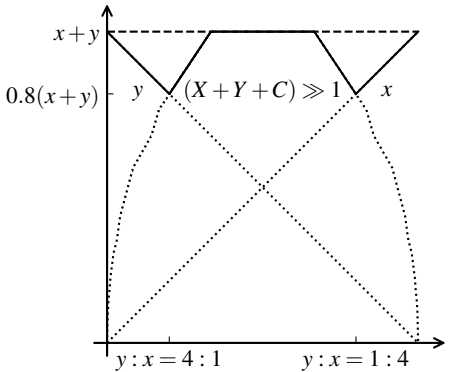
$$\max\left(X,\, Y,\, (X + Y + C) \gg 1\right)$$

- Maximum relative error in closed-form:

$$-\frac{1+u}{5+u},$$

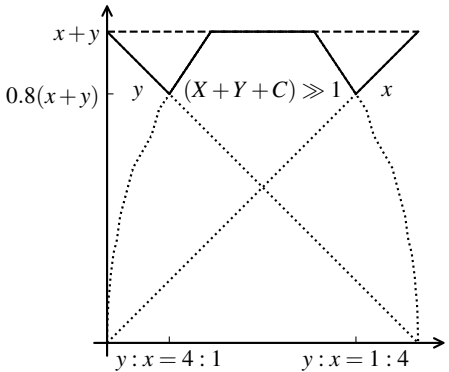
  where $u = 2^{-t}$. Tends to -0.2 as t increases.
- $C = (2 \ll t) + 1 \Rightarrow$ maximum relative error is exactly -0.2
- Correct rounding when
  - x and y share the same exponent
  - The difference in exponents is greater than t

Analysis of a constant sum

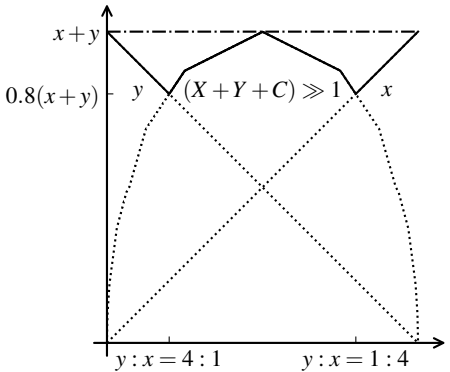


(a) Sum is exactly between two binades

Analysis of a constant sum

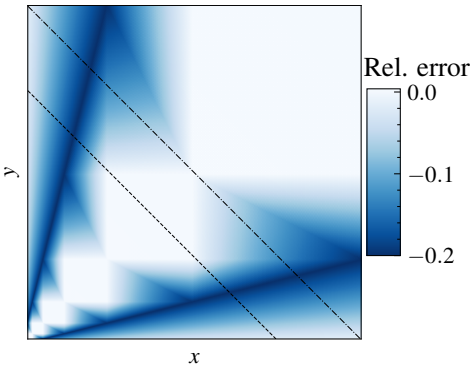


(a) Sum is exactly between two binades



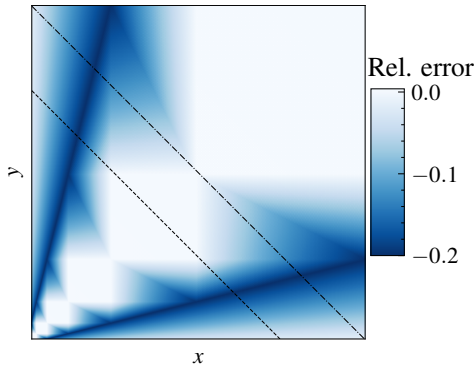
(b) Sum is an exact power of two

Sweep of six binades

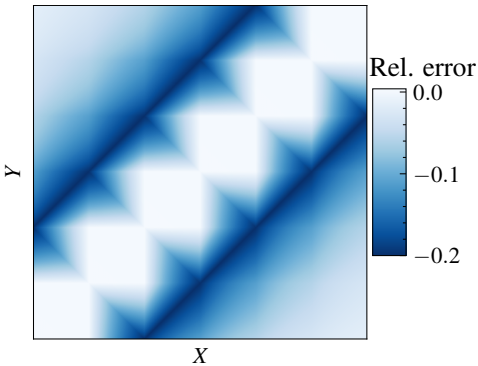


(a) Float values

Sweep of six binades

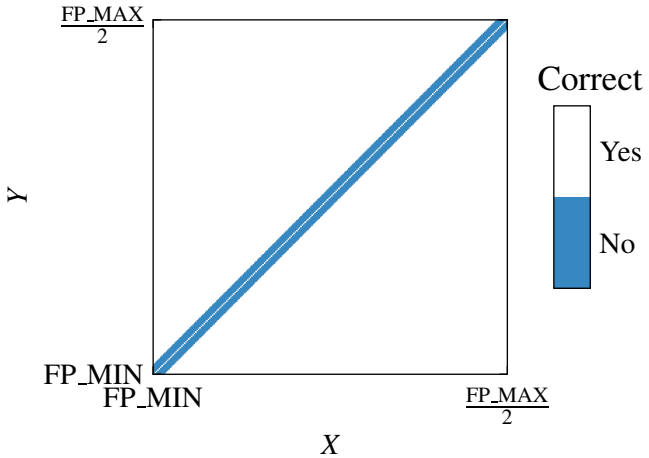


(a) Float values



(b) Bit patterns

Correct rounding is often obtained – BF16



Special numbers

- Infinity and NaN work when $X + Y + C$ doesn't overflow

Special numbers

- Infinity and NaN work when $X + Y + C$ doesn't overflow
- Overflow can be handled as

$$\max\left(X,\, Y,\, \min\left((X + Y + C) \gg 1,\, \texttt{FP\_INF}\right)\right)$$

Special numbers

- Infinity and NaN work when $X + Y + C$ doesn't overflow
- Overflow can be handled as

$$\max\left(X,\, Y,\, \min\left((X + Y + C) \gg 1,\, \texttt{FP\_INF}\right)\right)$$

- Subnormal numbers can be
  - Flushed to zero
  - Treated as normalized numbers

Evaluation

- Compared designs
  - Approximate: RLOA I and II[2], ACFPA[3]
  - Exact: FloPoCo

[2] W. Liu Chen, C. Wang, M. O'Neill, and F. Lombardi, "Design and Analysis of Inexact Floating-Point Adders", in *IEEE Trans. Comput., 2016*

[3] L. Tegazzini, G. Di Meo, A. Torino, F. Del Prete, D. de Caro, C. Parrella, and A. G. M. Strollo, "Novel Approximate Floating-Point Adder With Runtime Tunable Precision", in *IEEE Access, 2025*

Evaluation

- Compared designs
  - Approximate: RLOA I and II[2], ACFPA[3]
  - Exact: FloPoCo
- All designs have the same capabilities
  - Positive-only input
  - No subnormals, infinity, or NaN

[2] W. Liu Chen, C. Wang, M. O'Neill, and F. Lombardi, "Design and Analysis of Inexact Floating-Point Adders", in *IEEE Trans. Comput., 2016*

[3] L. Tegazzini, G. Di Meo, A. Torino, F. Del Prete, D. de Caro, C. Parrella, and A. G. M. Strollo, "Novel Approximate Floating-Point Adder With Runtime Tunable Precision", in *IEEE Access, 2025*

Evaluation

- Compared designs
  - Approximate: RLOA I and II[2], ACFPA[3]
  - Exact: FloPoCo
- All designs have the same capabilities
  - Positive-only input
  - No subnormals, infinity, or NaN
- Combinatorial circuits with registers on input/output

[2] W. Liu Chen, C. Wang, M. O'Neill, and F. Lombardi, "Design and Analysis of Inexact Floating-Point Adders", in *IEEE Trans. Comput., 2016*

[3] L. Tegazzini, G. Di Meo, A. Torino, F. Del Prete, D. de Caro, C. Parrella, and A. G. M. Strollo, "Novel Approximate Floating-Point Adder With Runtime Tunable Precision", in *IEEE Access, 2025*

Evaluation

- Compared designs
  - Approximate: RLOA I and II[2], ACFPA[3]
  - Exact: FloPoCo
- All designs have the same capabilities
  - Positive-only input
  - No subnormals, infinity, or NaN
- Combinatorial circuits with registers on input/output
- Synthesis results for ASIC 28 nm, FPGA Artix-7

[2] W. Liu Chen, C. Wang, M. O'Neill, and F. Lombardi, "Design and Analysis of Inexact Floating-Point Adders", in *IEEE Trans. Comput., 2016*

[3] L. Tegazzini, G. Di Meo, A. Torino, F. Del Prete, D. de Caro, C. Parrella, and A. G. M. Strollo, "Novel Approximate Floating-Point Adder With Runtime Tunable Precision", in *IEEE Access, 2025*

Evaluation

- Compared designs
  - Approximate: RLOA I and II[2], ACFPA[3]
  - Exact: FloPoCo
- All designs have the same capabilities
  - Positive-only input
  - No subnormals, infinity, or NaN
- Combinatorial circuits with registers on input/output
- Synthesis results for ASIC 28 nm, FPGA Artix-7
- Five FP format: **E5M2**, E4M3, BF16, FP16, and **FP32**

[2] W. Liu Chen, C. Wang, M. O'Neill, and F. Lombardi, "Design and Analysis of Inexact Floating-Point Adders", in *IEEE Trans. Comput., 2016*

[3] L. Tegazzini, G. Di Meo, A. Torino, F. Del Prete, D. de Caro, C. Parrella, and A. G. M. Strollo, "Novel Approximate Floating-Point Adder With Runtime Tunable Precision", in *IEEE Access, 2025*

Error metrics

Measure	Definition
Relative error distance (RED)	$\left\lvert \frac{\widetilde{x} - x}{x} \right\rvert$
Maximum RED	$\max(\mathrm{RED})$
Mean RED (MRED)	$\mathrm{mean}(\mathrm{RED})$
Normalized MED (NMED)	$\frac{\mathrm{mean}(\lvert\widetilde{x} - x\rvert)}{\texttt{FP\_MAX}}$
Error rate (ER)	Percentage of times not `roundTiesToEven`

Results for E5M2

Design	Max. RED	MRED	NMED	ER
Proposed	$\mathbf{2.00 \times 10^{-1}}$	$\mathbf{2.85 \times 10^{-2}}$	$\mathbf{1.05 \times 10^{-3}}$	**13.2%**
RLOA I	5.00×10^{-1}	4.81×10^{-2}	2.08×10^{-3}	18.7%
RLOA II	3.85×10^{-1}	3.14×10^{-2}	1.33×10^{-3}	**13.2%**

Results for E5M2

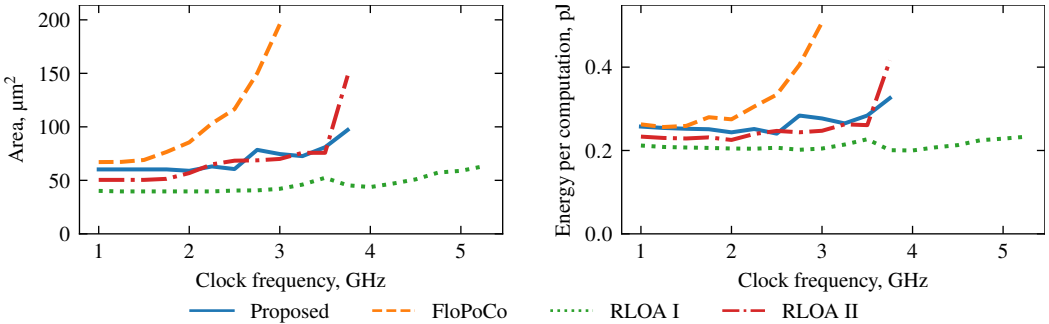
Design	Max. RED	MRED	NMED	ER
Proposed	$\mathbf{2.00 \times 10^{-1}}$	$\mathbf{2.85 \times 10^{-2}}$	$\mathbf{1.05 \times 10^{-3}}$	**13.2%**
RLOA I	5.00×10^{-1}	4.81×10^{-2}	2.08×10^{-3}	18.7%
RLOA II	3.85×10^{-1}	3.14×10^{-2}	1.33×10^{-3}	**13.2%**

Design	Delay, ns	LUTs
Proposed	3.881	27
RLOA I	**3.006**	**16**
RLOA II	4.684	24
FloPoCo	5.405	32

Results for E5M2

Design	Max. RED	MRED	NMED	ER
Proposed	$\mathbf{2.00 \times 10^{-1}}$	$\mathbf{2.85 \times 10^{-2}}$	$\mathbf{1.05 \times 10^{-3}}$	**13.2%**
RLOA I	5.00×10^{-1}	4.81×10^{-2}	2.08×10^{-3}	18.7%
RLOA II	3.85×10^{-1}	3.14×10^{-2}	1.33×10^{-3}	**13.2%**

Design	Delay, ns	LUTs
Proposed	3.881	27
RLOA I	**3.006**	**16**
RLOA II	4.684	24
FloPoCo	5.405	32



Results for FP32

Design	Max. RED	MRED	NMED	ER
Proposed	2.00×10^{-1}	3.63×10^{-3}	1.28×10^{-5}	18.1%
RLOA I	5.00×10^{-1}	4.51×10^{-3}	2.24×10^{-5}	17.7%
RLOA II	3.33×10^{-1}	1.35×10^{-3}	7.62×10^{-6}	**11.3%**
ACFPA Acc=0	4.88×10^{-4}	4.95×10^{-6}	1.53×10^{-8}	18.4%
ACFPA Acc=7	$\mathbf{4.77 \times 10^{-7}}$	$\mathbf{1.33 \times 10^{-8}}$	$\mathbf{4.12 \times 10^{-11}}$	11.8%

Results for FP32

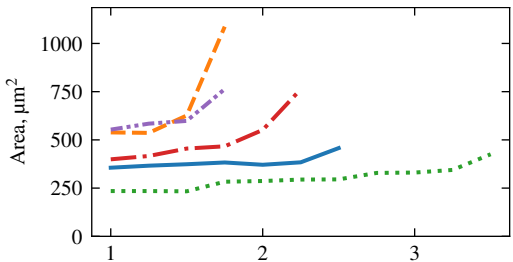
Design	Max. RED	MRED	NMED	ER
Proposed	2.00×10^{-1}	3.63×10^{-3}	1.28×10^{-5}	18.1%
RLOA I	5.00×10^{-1}	4.51×10^{-3}	2.24×10^{-5}	17.7%
RLOA II	3.33×10^{-1}	1.35×10^{-3}	7.62×10^{-6}	**11.3%**
ACFPA Acc=0	4.88×10^{-4}	4.95×10^{-6}	1.53×10^{-8}	18.4%
ACFPA Acc=7	$\mathbf{4.77 \times 10^{-7}}$	$\mathbf{1.33 \times 10^{-8}}$	$\mathbf{4.12 \times 10^{-11}}$	11.8%

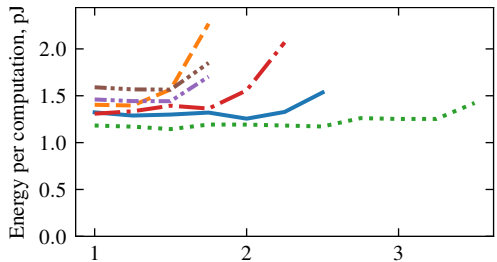
Design	Delay, ns	LUTs
Proposed	**4.987**	**111**
RLOA I	5.450	117
RLOA II	8.415	139
ACFPA	9.816	233
FloPoCo	9.537	170

Results for FP32

Design	Max. RED	MRED	NMED	ER
Proposed	2.00×10^{-1}	3.63×10^{-3}	1.28×10^{-5}	18.1%
RLOA I	5.00×10^{-1}	4.51×10^{-3}	2.24×10^{-5}	17.7%
RLOA II	3.33×10^{-1}	1.35×10^{-3}	7.62×10^{-6}	**11.3%**
ACFPA Acc=0	4.88×10^{-4}	4.95×10^{-6}	1.53×10^{-8}	18.4%
ACFPA Acc=7	$\mathbf{4.77 \times 10^{-7}}$	$\mathbf{1.33 \times 10^{-8}}$	$\mathbf{4.12 \times 10^{-11}}$	11.8%

Design	Delay, ns	LUTs
Proposed	**4.987**	**111**
RLOA I	5.450	117
RLOA II	8.415	139
ACFPA	9.816	233
FloPoCo	9.537	170





Proposed FloPoCo RLOA I RLOA II ACFPA Acc=0 ACFPA Acc=7

Takeaway: Balances numerical performance and hardware cost

- Compared to designs with less delay on ASIC
  - More accurate
- Compared to more accurate designs
  - Less area on ASIC
  - Shorter delays on both ASIC and FPGA

Processor platforms

- Usable alongside existing soft FP libraries

Processor platforms

- Usable alongside existing soft FP libraries
- Platforms with FPU:s can benefit since idle integer units can be used

Processor platforms

- Usable alongside existing soft FP libraries
- Platforms with FPU:s can benefit since idle integer units can be used
- Branches avoided by
  - Conditional assignments on CPUs
  - Max instructions on CPUs
  - Masking instructions on GPUs

Processor platforms

- Usable alongside existing soft FP libraries
- Platforms with FPU:s can benefit since idle integer units can be used
- Branches avoided by
  - Conditional assignments on CPUs
  - Max instructions on CPUs
  - Masking instructions on GPUs

$\max(X, Y, (X + Y + C) \gg 1)$ for BF16 on RV64IB:

```
1  addiw   a5, a1, 513     ; a5 = Y + C
2  addw    a5, a5, a0      ; a5 = a5 + X
3  srliw   a5, a5, 1       ; a5 = a5 >> 1
4  maxu    a0, a0, a1      ; a0 = max(X, Y)
5  maxu    a0, a5, a0      ; a0 = max(a5, a0)
```

Summary

- Novel approximation to FP addition
  - Geometric mean
  - Integer computations in logarithmic domain
  - Suited for sums of squares and exponentials

Summary

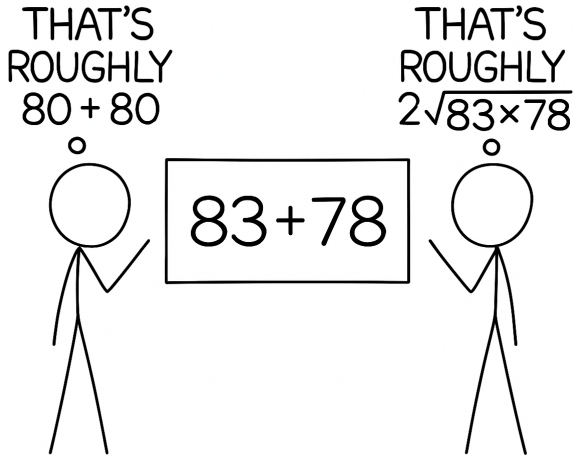
- Novel approximation to FP addition
  - Geometric mean
  - Integer computations in logarithmic domain
  - Suited for sums of squares and exponentials
- Derived maximum relative errors

Summary

- Novel approximation to FP addition
  - Geometric mean
  - Integer computations in logarithmic domain
  - Suited for sums of squares and exponentials
- Derived maximum relative errors
- Favorable trade-off: hardware cost vs. numerical performance

Summary

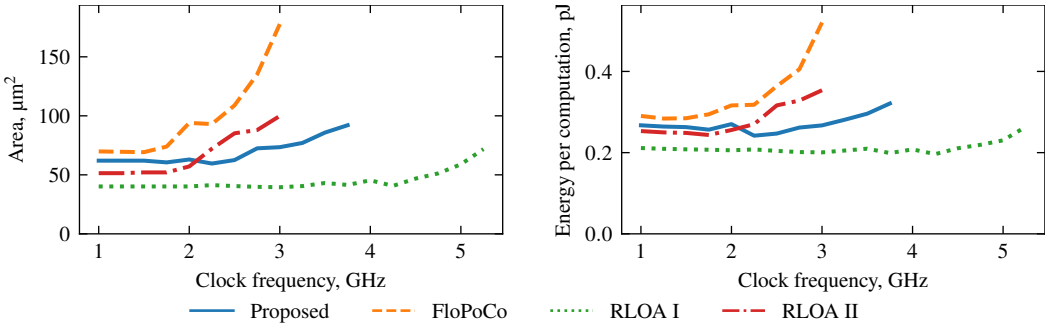
- Novel approximation to FP addition
  - Geometric mean
  - Integer computations in logarithmic domain
  - Suited for sums of squares and exponentials
- Derived maximum relative errors
- Favorable trade-off: hardware cost vs. numerical performance
- Attractive for processors


THAT'S ROUGHLY 80 + 80
83+78
THAT'S ROUGHLY 2√83×78

Results for E4M3

Design	Max. RED	MRED	NMED	ER
Proposed	$\mathbf{2.00 \times 10^{-1}}$	5.42×10^{-2}	4.96×10^{-3}	43.9%
RLOA I	5.00×10^{-1}	8.98×10^{-2}	9.35×10^{-3}	48.3%
RLOA II	3.60×10^{-1}	$\mathbf{4.39 \times 10^{-2}}$	$\mathbf{4.75 \times 10^{-3}}$	**30.6%**

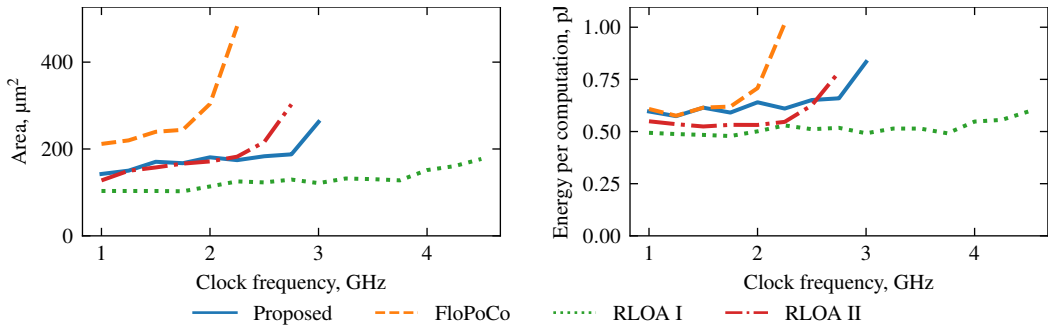
Design	Delay, ns	LUTs
Proposed	4.103	27
RLOA I	**2.981**	**15**
RLOA II	3.967	28
FloPoCo	4.652	37



Results for BF16

Design	Max. RED	MRED	NMED	ER
Proposed	$\mathbf{2.00 \times 10^{-1}}$	3.62×10^{-3}	1.30×10^{-5}	6.3%
RLOA I	5.00×10^{-1}	4.60×10^{-3}	2.18×10^{-5}	5.9%
RLOA II	3.35×10^{-1}	$\mathbf{1.49 \times 10^{-3}}$	$\mathbf{8.28 \times 10^{-6}}$	**3.9%**

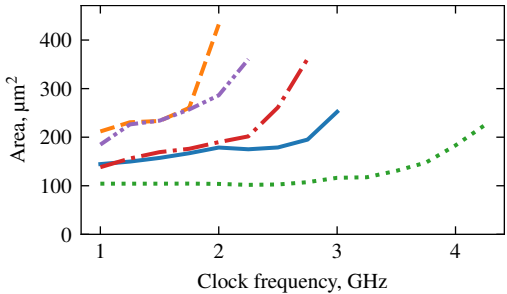
Design	Delay, ns	LUTs
Proposed	4.521	55
RLOA I	**4.396**	**46**
RLOA II	6.556	56
FloPoCo	7.913	75

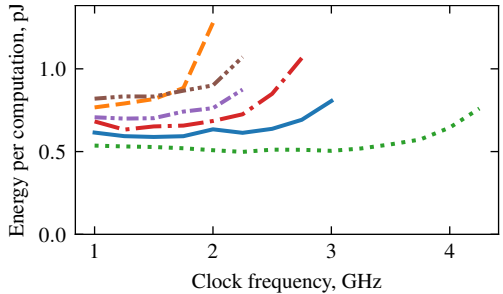


Results for FP16

Design	Max. RED	MRED	NMED	ER
Proposed	2.00×10^{-1}	2.88×10^{-2}	9.91×10^{-4}	61.0%
RLOA I	5.00×10^{-1}	3.78×10^{-2}	1.64×10^{-3}	58.7%
RLOA II	3.34×10^{-1}	1.16×10^{-2}	6.01×10^{-4}	39.6%
ACFPA Acc=0	2.00×10^{-1}	1.49×10^{-2}	4.77×10^{-4}	62.5%
ACFPA Acc=7	$\mathbf{7.75 \times 10^{-3}}$	$\mathbf{5.19 \times 10^{-4}}$	$\mathbf{1.86 \times 10^{-5}}$	**37.8%**

Design	Delay, ns	LUTs
Proposed	4.850	55
RLOA I	**3.829**	**41**
RLOA II	7.086	57
ACFPA	7.010	94
FloPoCo	7.209	79





Proposed FloPoCo RLOA I RLOA II ACFPA Acc=0 ACFPA Acc=7

Why is correct rounding obtained for inputs sharing exponent?

Consider

$$
\begin{aligned}
x + y &= 2^{e_x}(1 + m_x) + 2^{e_x}(1 + m_y) && (1)\\
&= 2^{e_x}(2 + m_x + m_y) && (2)\\
&= 2^{e_x+1}\left(1 + \frac{m_x + m_y}{2}\right) && (3)
\end{aligned}
$$

Why is correct rounding obtained for inputs sharing exponent?

Consider

$$
\begin{aligned}
x + y &= 2^{e_x}(1 + m_x) + 2^{e_x}(1 + m_y) && (1)\\
&= 2^{e_x}(2 + m_x + m_y) && (2)\\
&= 2^{e_x+1}\left(1 + \frac{m_x + m_y}{2}\right) && (3)
\end{aligned}
$$

and

$$
\begin{aligned}
(X + Y + C) \gg 1 \Rightarrow \left(E_x + m_x + E_x + m_y + 2 + 2^{-t}\right) \gg 1 && (4)\\
= E_x + 1 + \left\lfloor \frac{m_x + m_y + 2^{-t}}{2} \right\rfloor && (5)
\end{aligned}
$$