Exploring and Exploiting Runtime Reconfigurable Floating-Point Precision in Scientific Computing: a Case Study for Solving PDEs

Cong “Callie” Hao, Georgia Institute of Technology

Scientific Computing

- **Definition**: solving complex real-world problems using mathematical models and computational algorithms

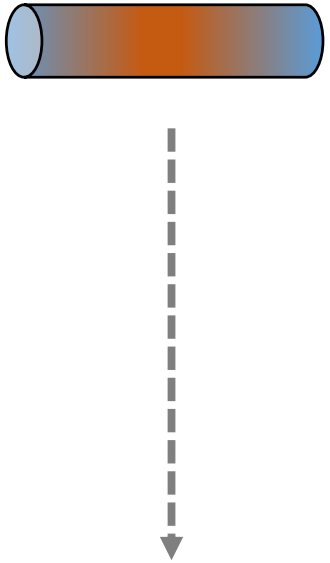
- **Core Techniques: numerical methods**
  - Partial Differential Equations (PDE)

- **Applications**
  - Physics: Simulations of fluid dynamics and heat transfer
  - Engineering: Structural analysis and optimization
  - Climate Science: Weather prediction and environmental modeling
  - Biology: Protein folding and genome analysis

Example of Heat Equation

- **Simulate heat diffusion along a 1D rod over time**
  - $u(x,t)$: temperature at position x and time t
  - t: time (in seconds)
  - x: position along the length of the rod

$$\frac{\partial u}{\partial t} = \alpha \frac{\partial^2 u}{\partial x^2}$$



- **Finite difference approximation:**
  - Discretize space into N points and time into steps Δt
  - Compute step by step

$$u_i^{n+1} = u_i^n + \frac{\alpha \Delta t}{(\Delta x)^2}(u_{i+1}^n - 2u_i^n + u_{i-1}^n)$$

Scientific Computing v.s. Machine Learning

Scientific Computing

Computationally intensive

Supercomputers in HPC centers

Very sensitive in data precision

32-bit floating point

64-bit floating point

Machine Learning

Computationally intensive

Clouds and edges

Low-precision is prevailing

8-bit floating point

8-bit or 16-bit fixed point

8-bit or 4-bit integer

Approximate Computing (AxC)

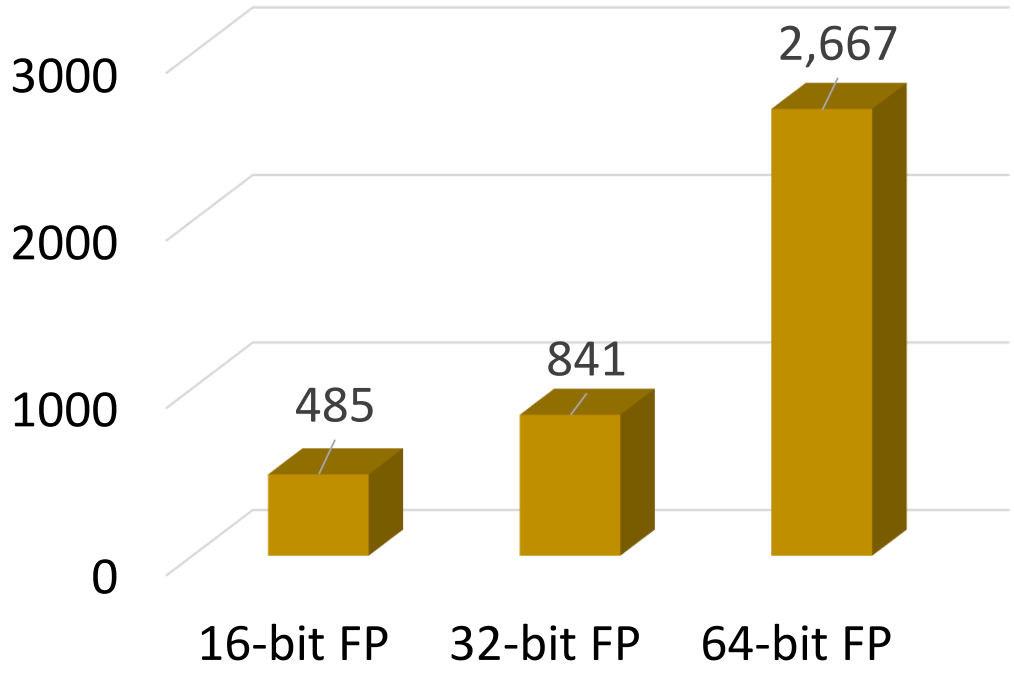
…

Scientific Computing Challenges

- **Massive computation power**
- **Long simulation latency**
  - Can take weeks or months
- **More bandwidth**
- **Large memory requirement**

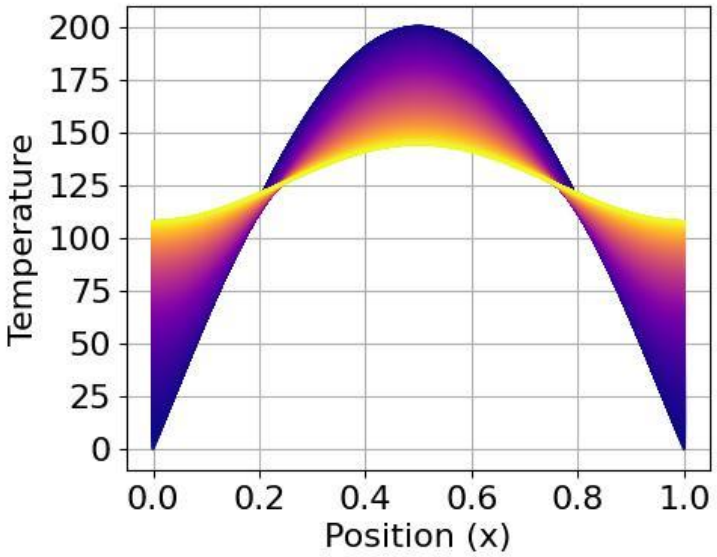
- **Can we reduce precision?**
  - Even reducing by half would be significant: 1000 nodes → 500 nodes

- **Dangerous…!**

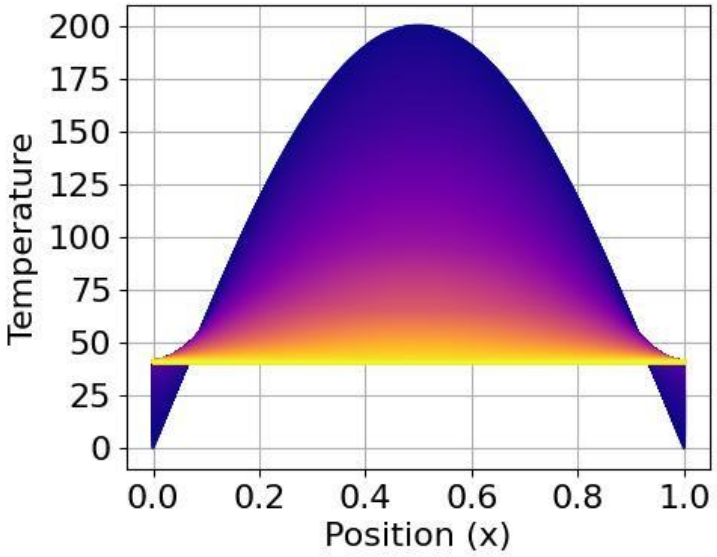


Area of floating-point multipliers
(Synthesized on FPGA)

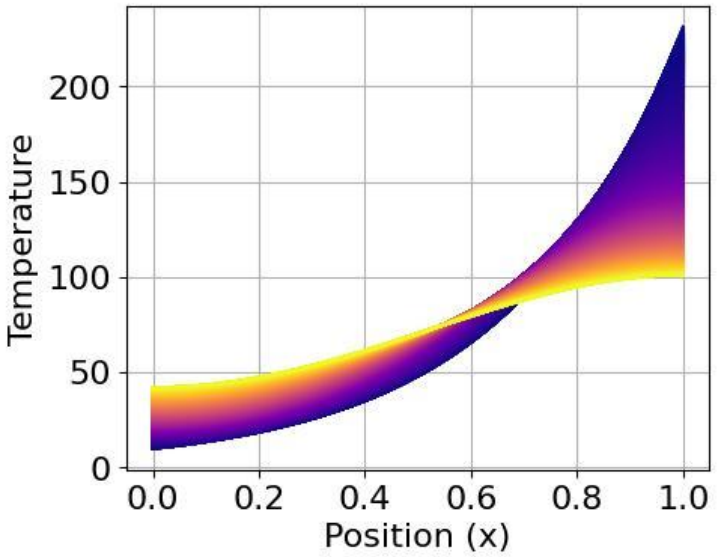
Reducing Precision is Risky – 1D Heat Equation



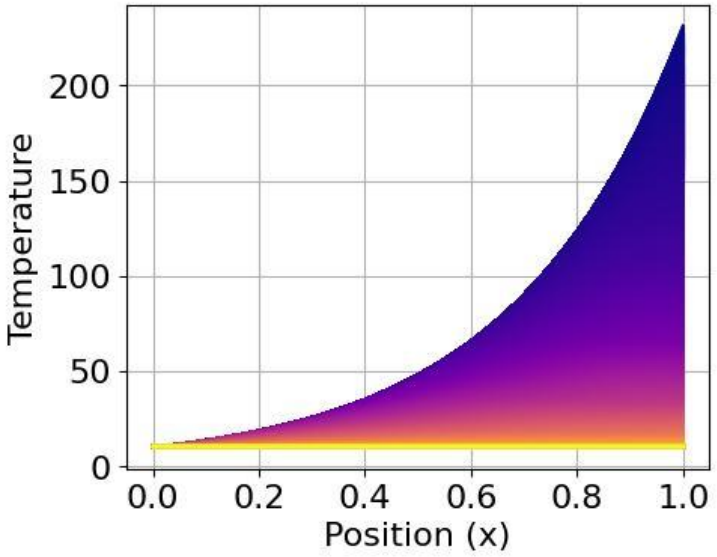
1D heat Sin init. 32-bit FP



1D heat Sin init. 16-bit FP

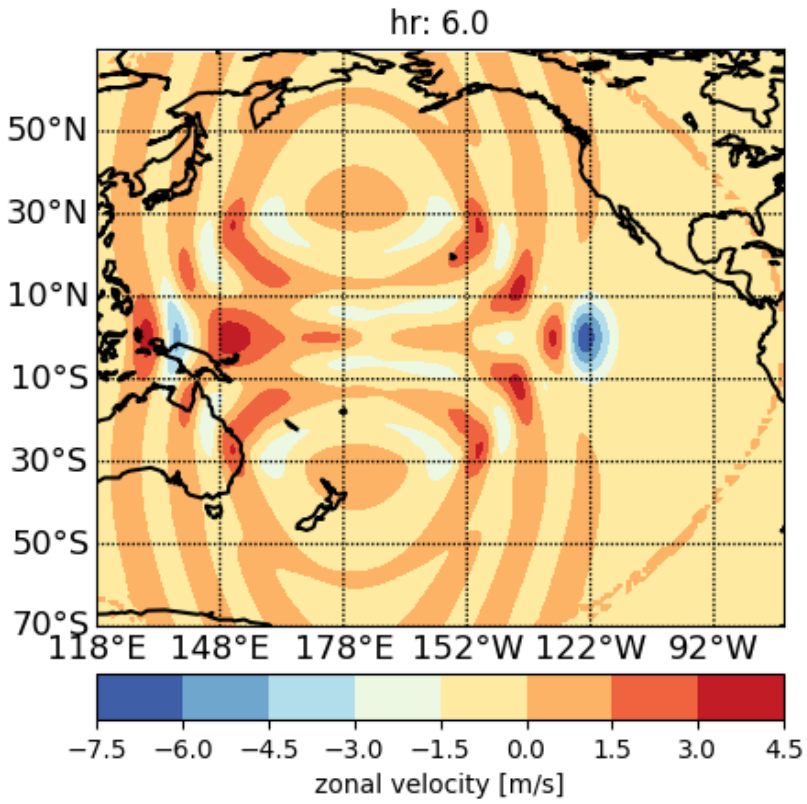


1D heat exp init. 32-bit FP

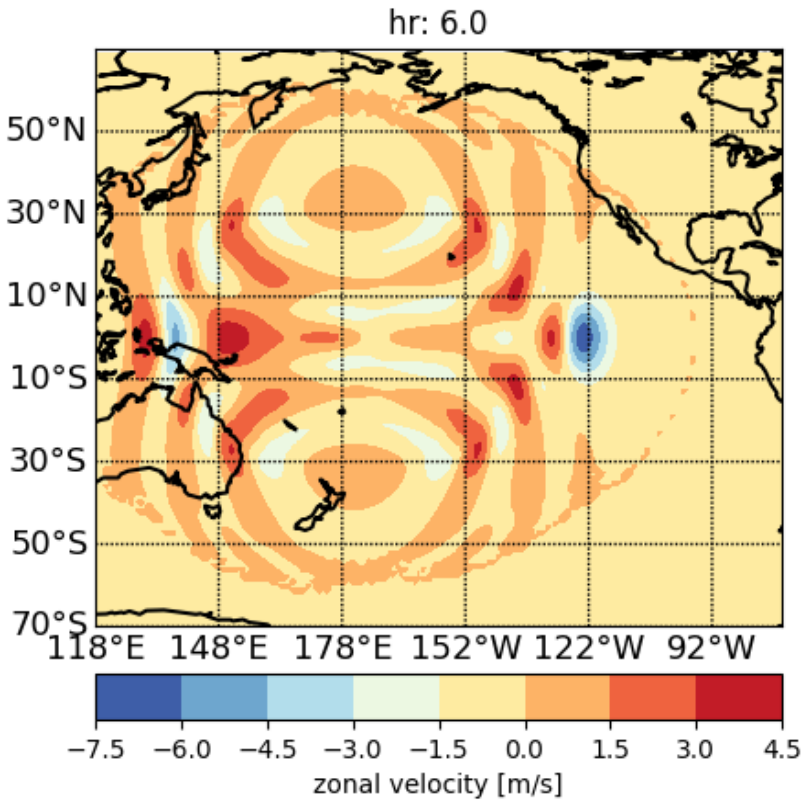


1D heat exp init. 16-bit FP

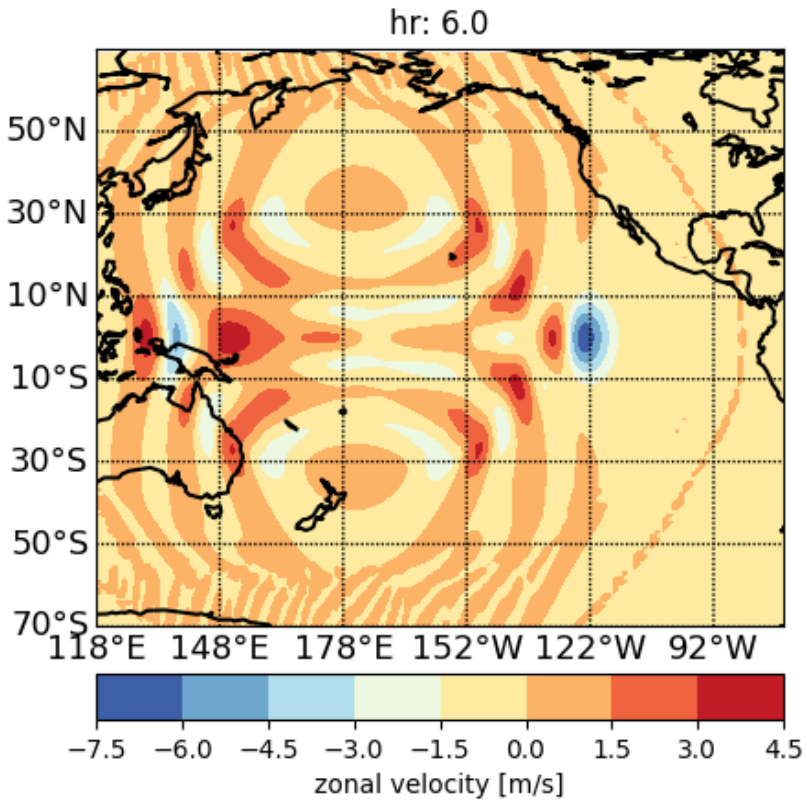
Reducing Precision is Risky – 2D Shallow Water



64-bit double precision



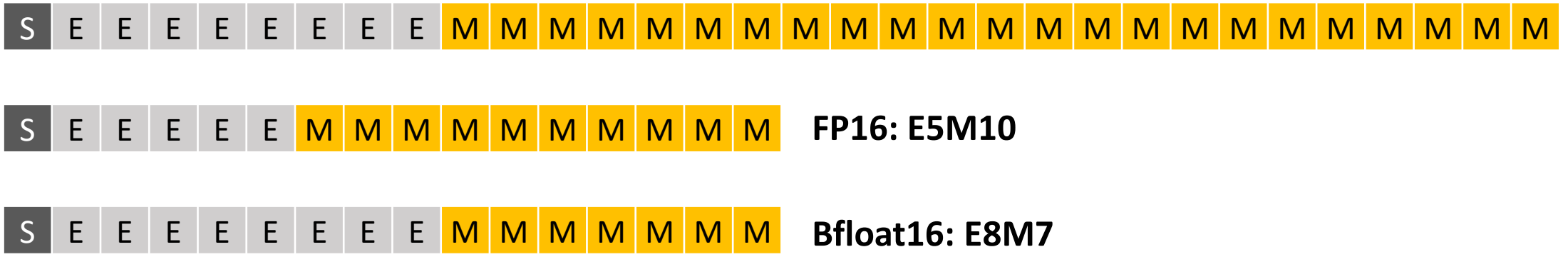
Mixed 32/64-bit
(paradigm 1)

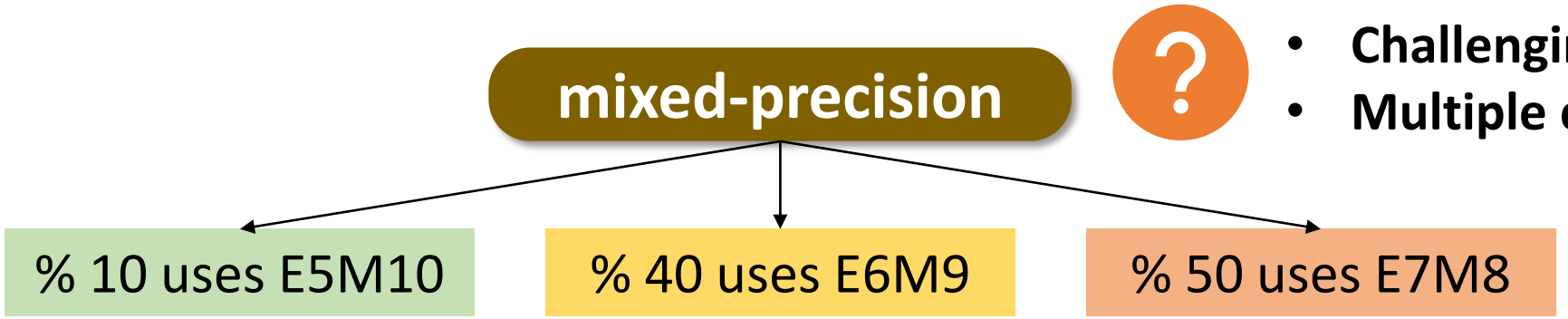


Mixed 32/64-bit
(paradigm 2)

Traditional Mixed-Precision

- **Static mixed-precision**

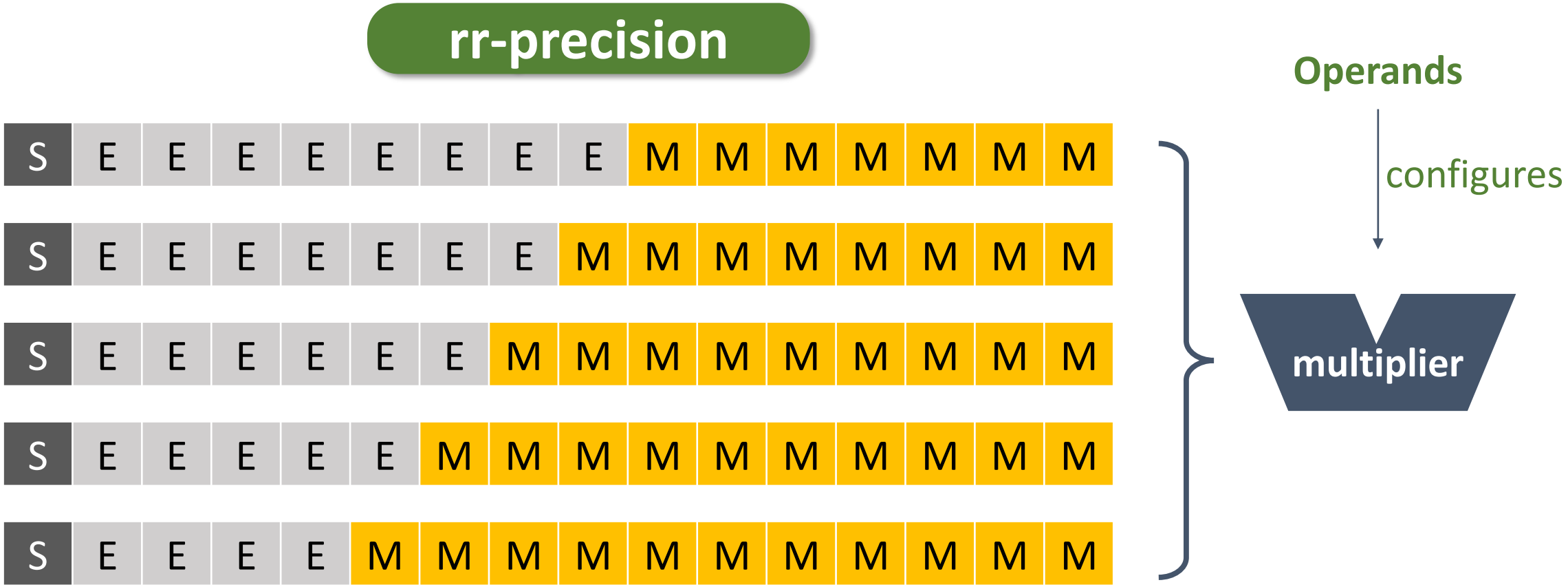




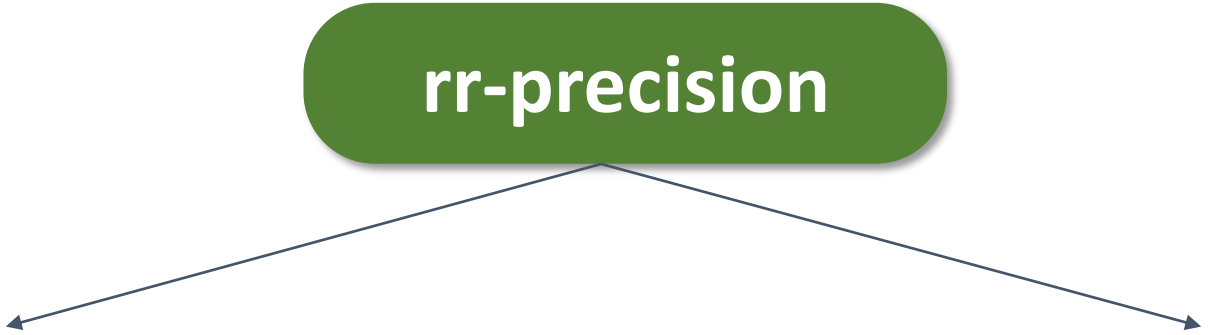
- **Challenging to decide beforehand**
- **Multiple copies of hardware**

Run-time Reconfigurable Reduced Precision

- **What if the reduced precision can be dynamically adjusted at runtime?**



RR-Precision Initial Studies



Exploration

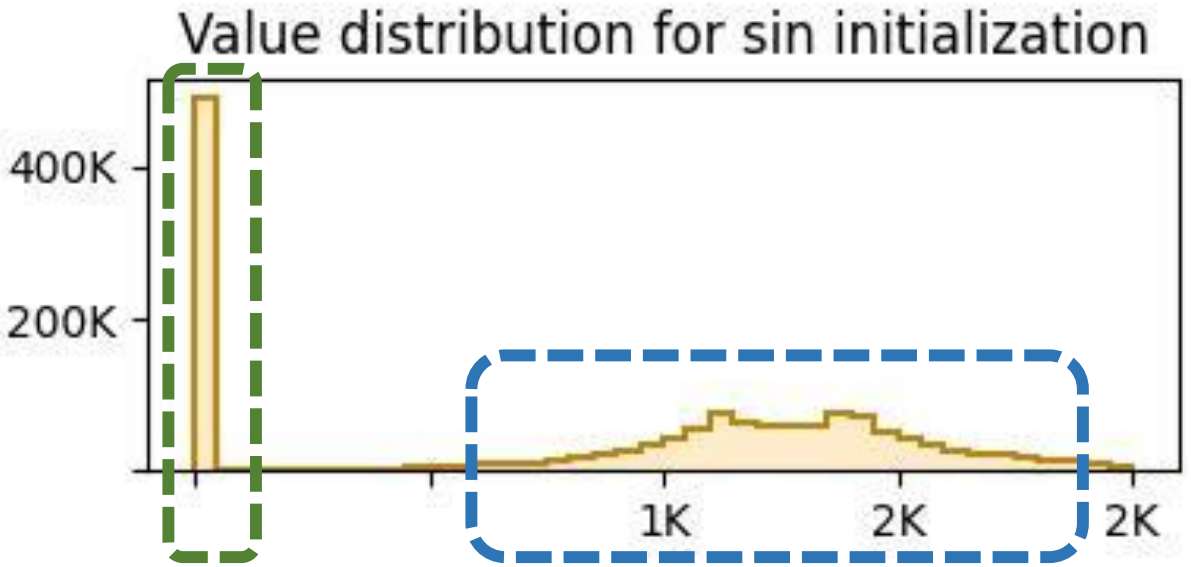
- Are there opportunities to use lower bitwidth with **rr-precision**?
- Is **rr-precision** indeed promising?

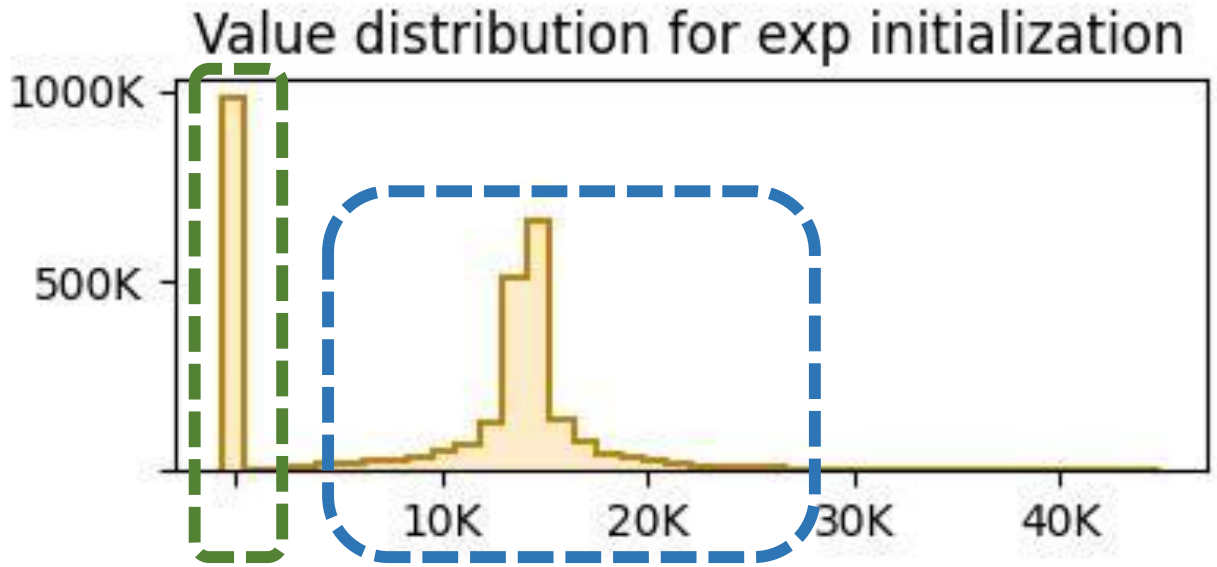
Exploitation

- If such opportunities exist, how to exploit the benefit on hardware?

RR-Precision Exploration

- Any special properties in the data that may provide opportunities?
- Look at data distribution for heat equation



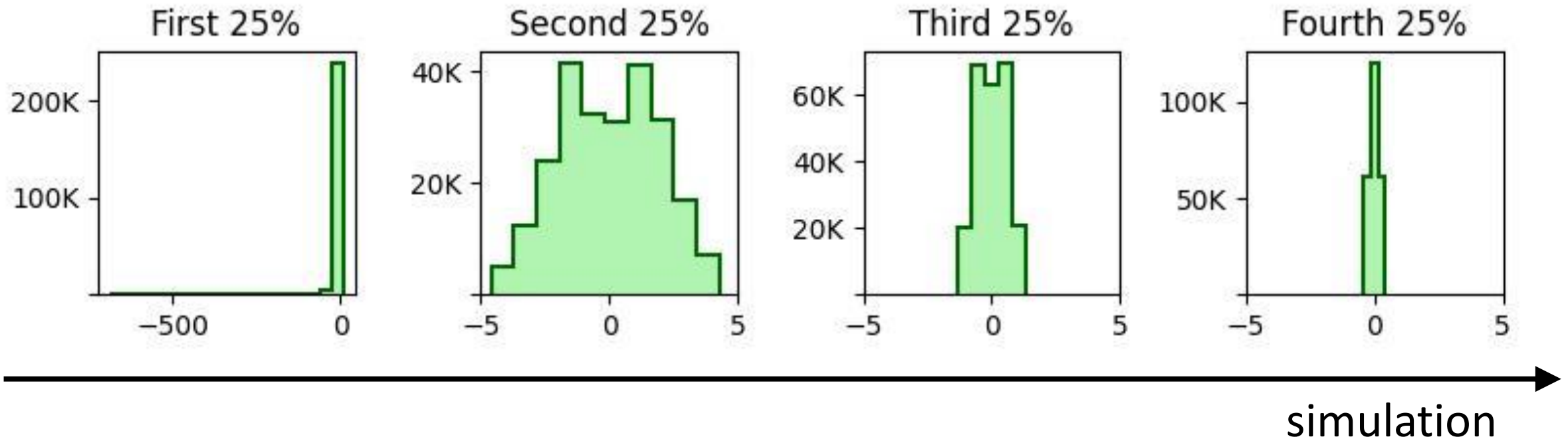


RR-Precision Exploration

Small values

Value range still wide:
more bits for exponent, fewer for mantissa?

Value range becomes narrow:
more bits for mantissa, fewer for exponent?

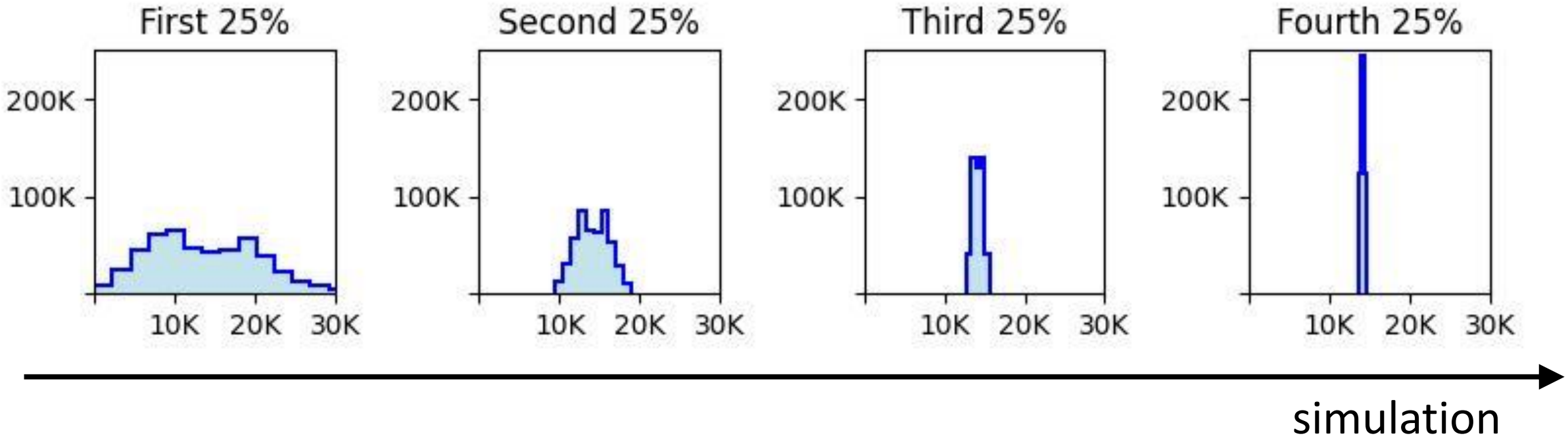


RR-Precision Exploration

Large values

Value range still wide:
more bits for exponent, fewer for mantissa?

Value range becomes narrow:
more bits for mantissa, fewer for exponent?



RR-Precision Exploration

- Any special properties in the data that may provide opportunities?
- Look at data distribution for heat equation

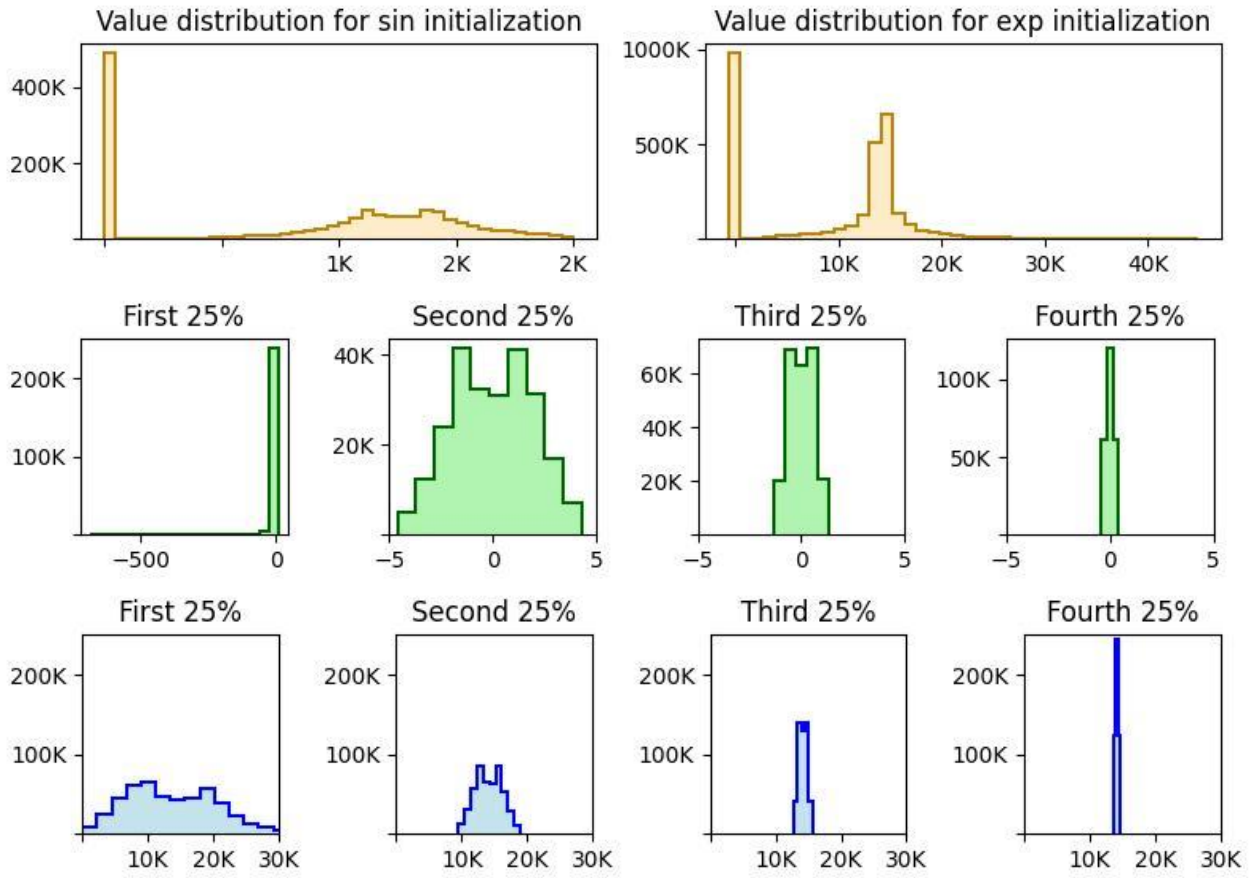
Globally wide

Locally narrow

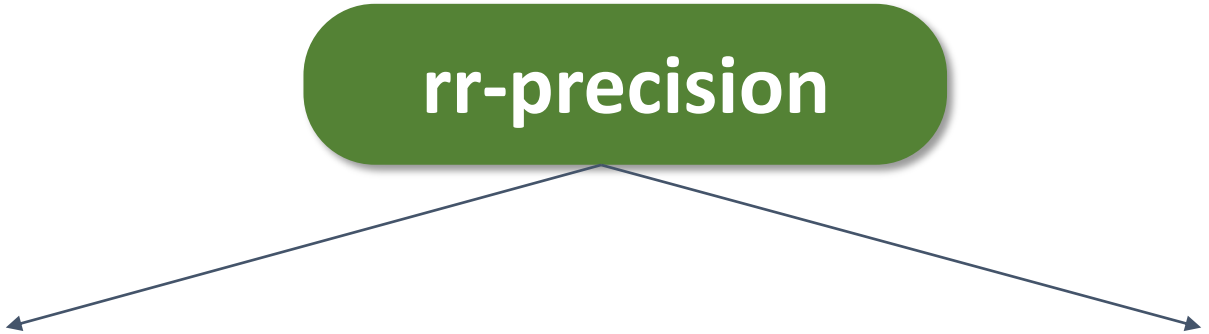
Dynamically changing

rr-precision

spatially & temporally



RR-Precision Initial Studies



Exploration

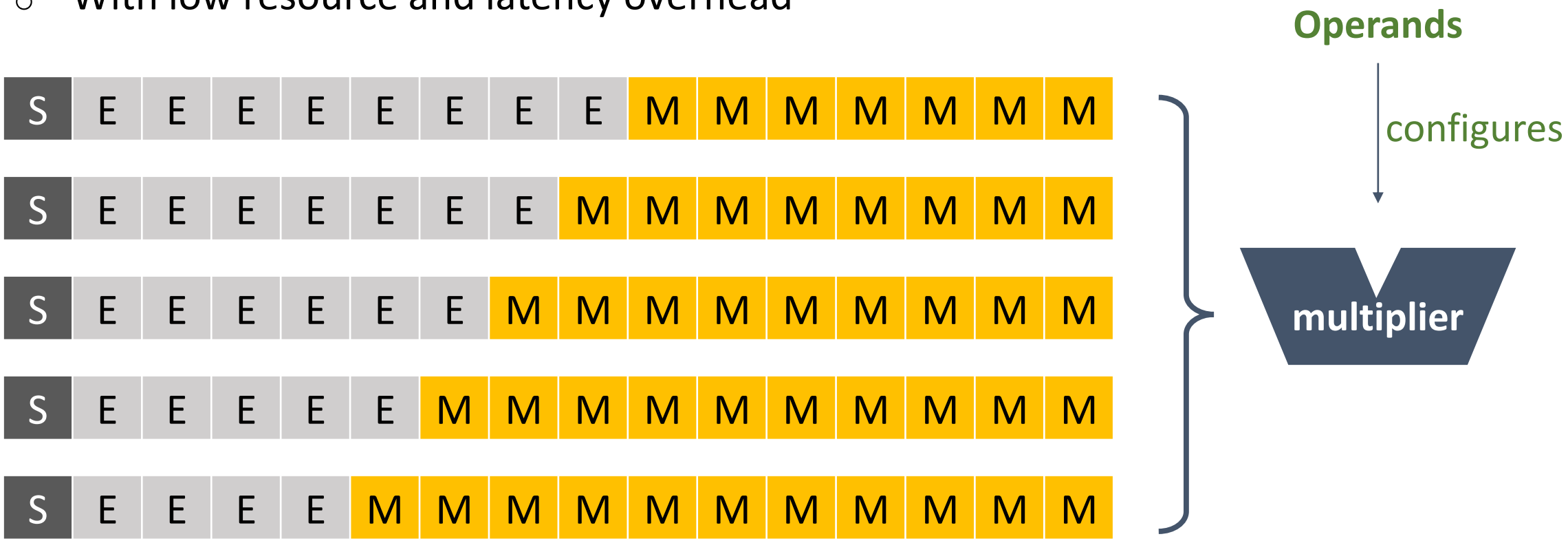
- Are there opportunities to use lower bitwidth with **rr-precision**?
- Is **rr-precision** indeed promising?

Exploitation

- If such opportunities exist, how to exploit the benefit on hardware?

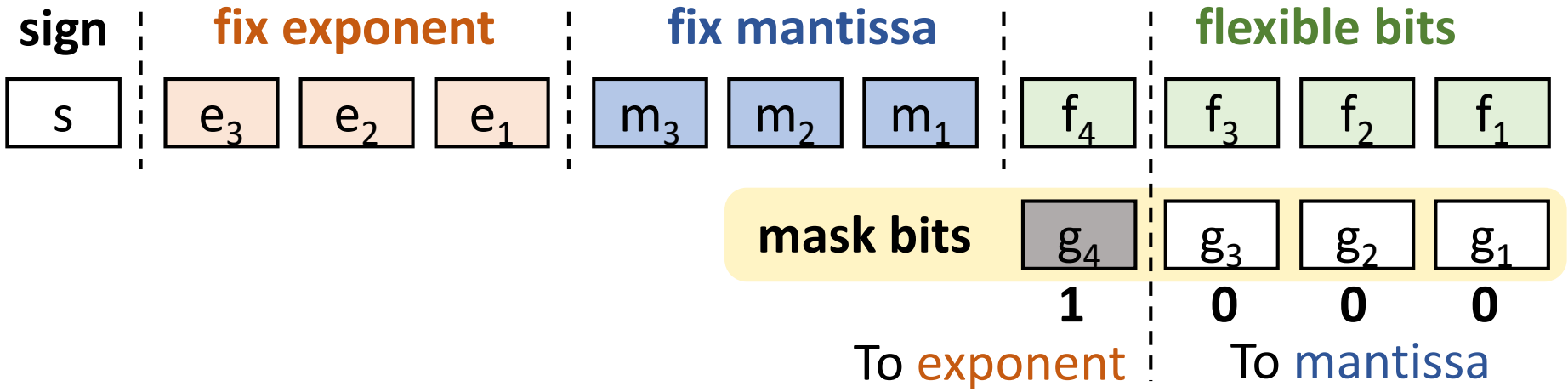
RR-Precision Exploitation

- **How to design floating-point arithmetic hardware units that can dynamically reconfigure the precision at runtime?**
  - With low resource and latency overhead

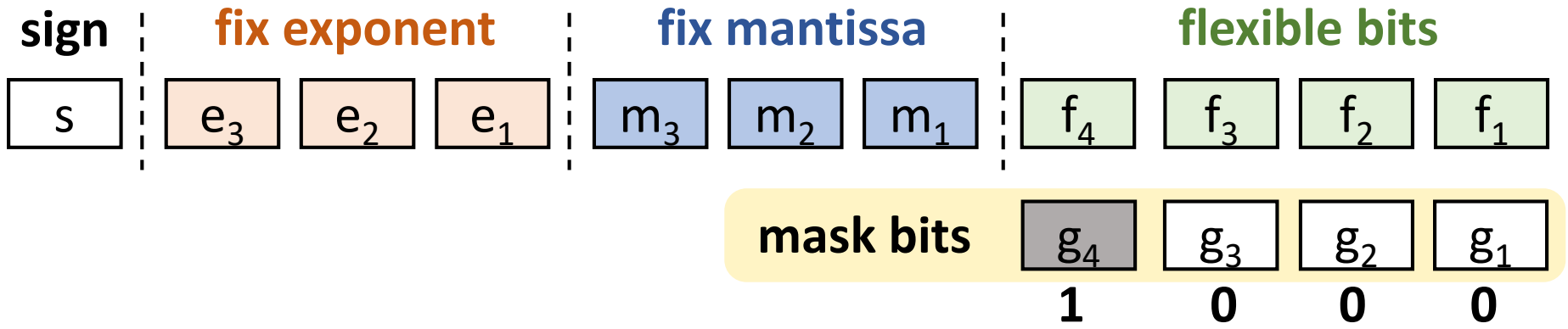


R2F2 Multiplier

- **Runtime Reconfigurable Flexible Floating-point multiplier**
- **Assumptions (limitations)**
  - Two operands have the same precision
  - Total bit width is 16: E + M = 15
- **Representation: <FE, FM, FLEX>**



R2F2 Multiplier

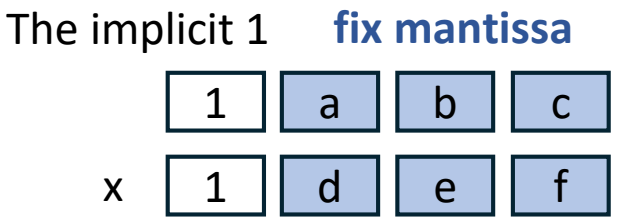


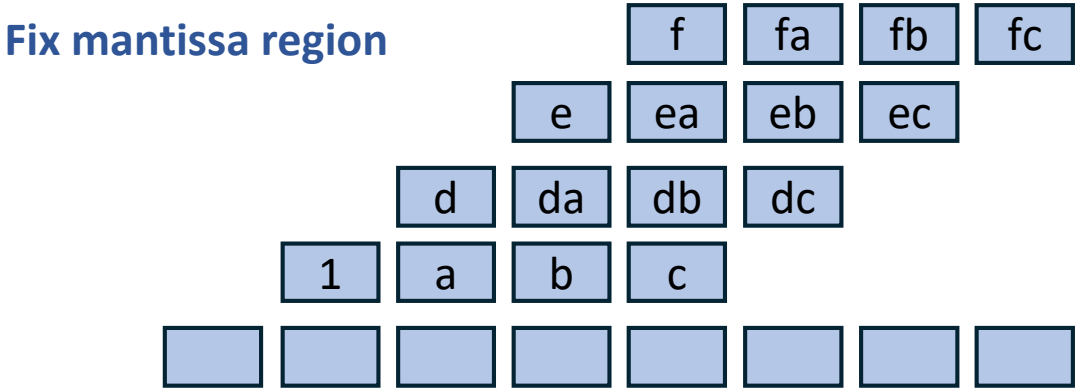
Fixed exponent addition ?

Fixed mantissa multiplication ?

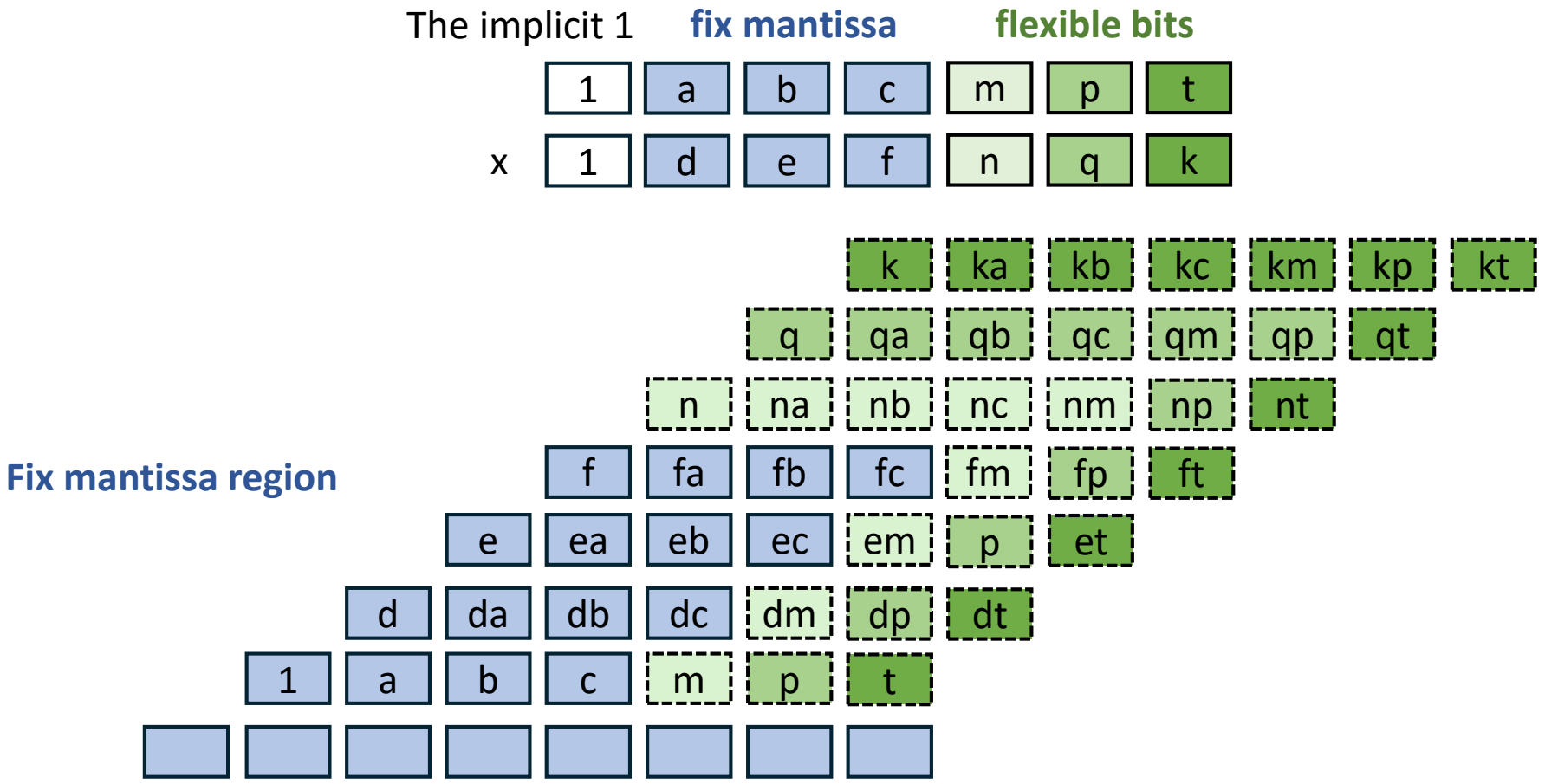
Flexible mantissa multiplication ?

R2F2 Multiplier – Mantissa Multiplication

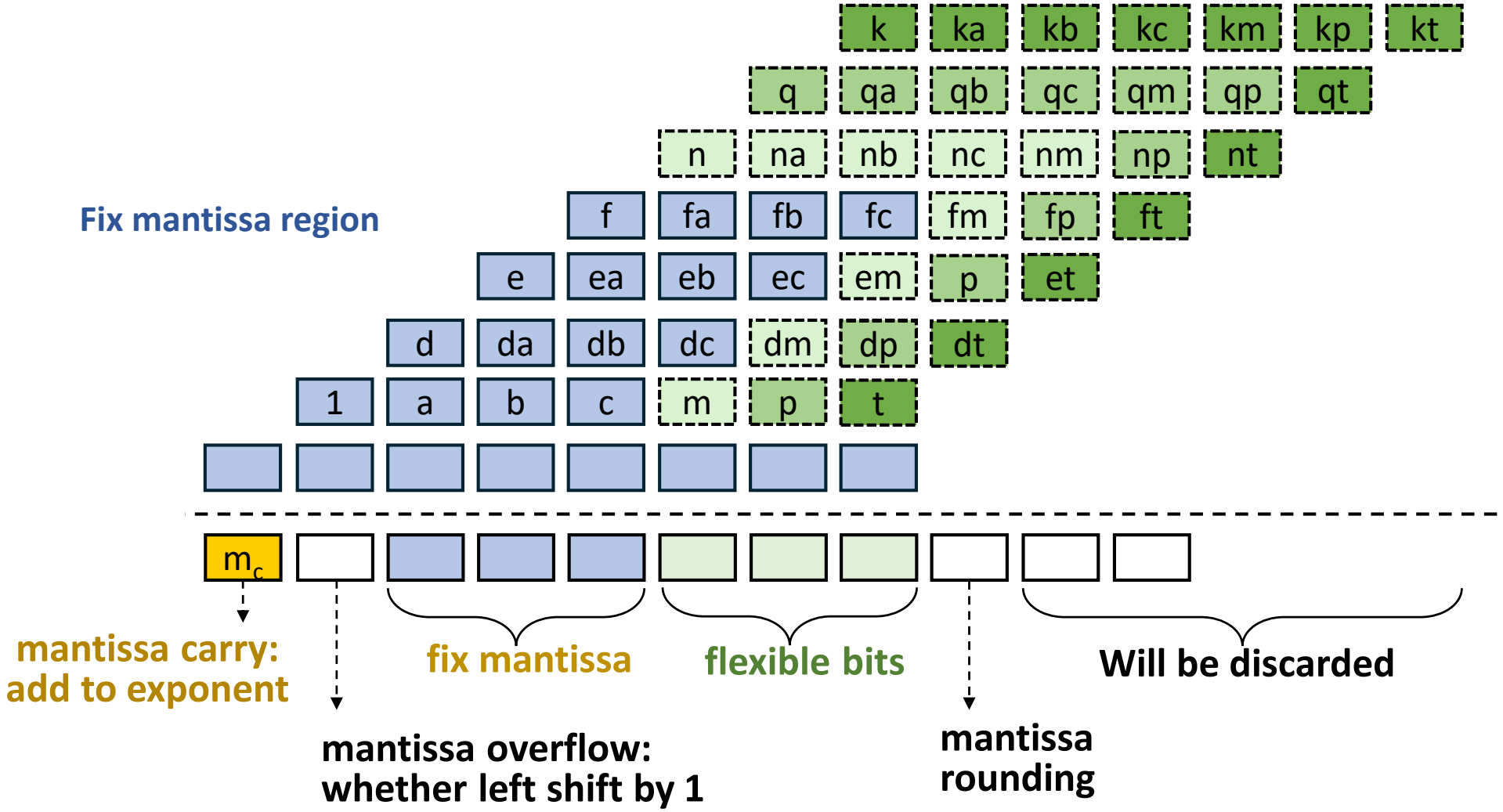




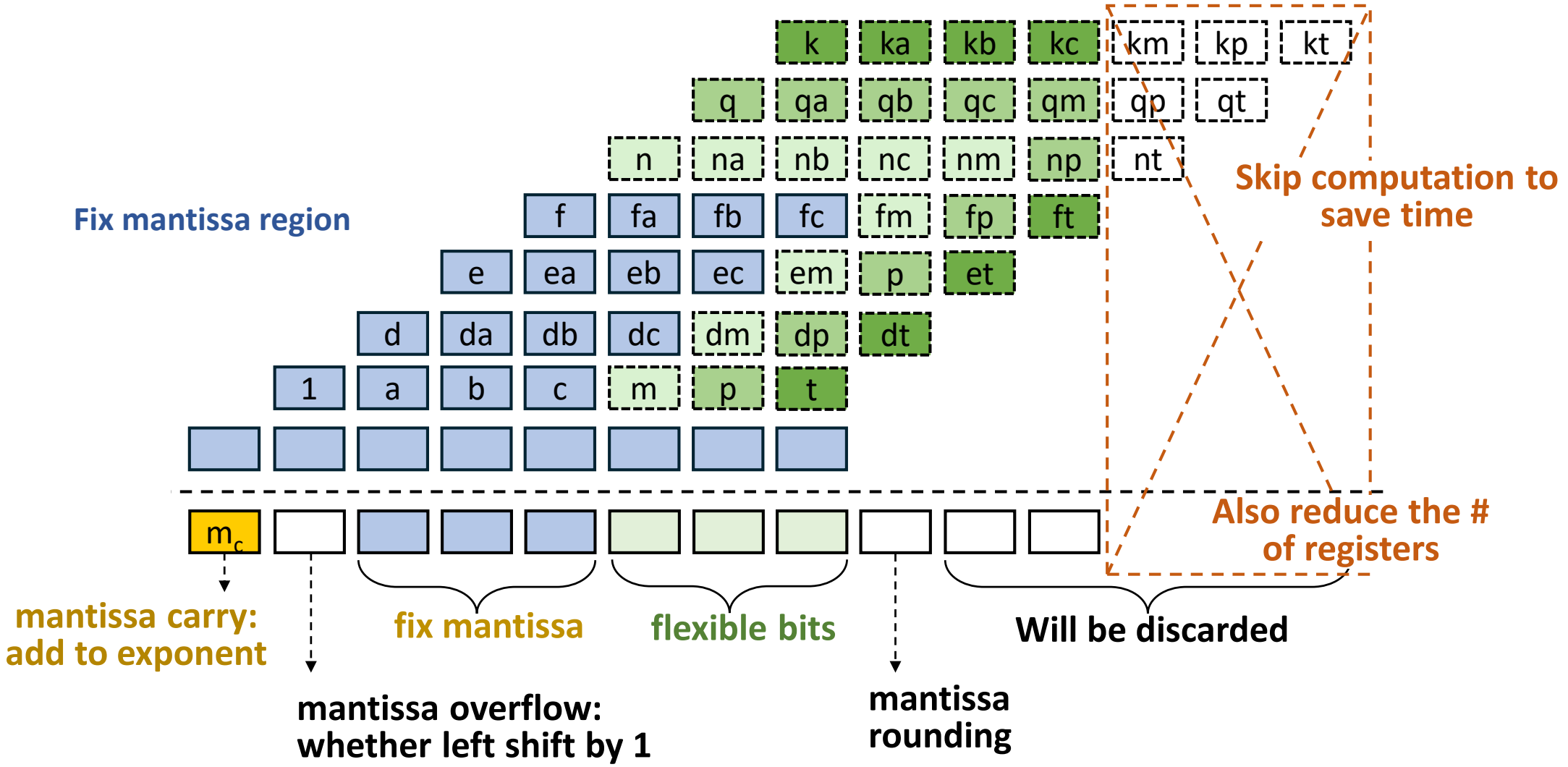
R2F2 Multiplier – Mantissa Multiplication



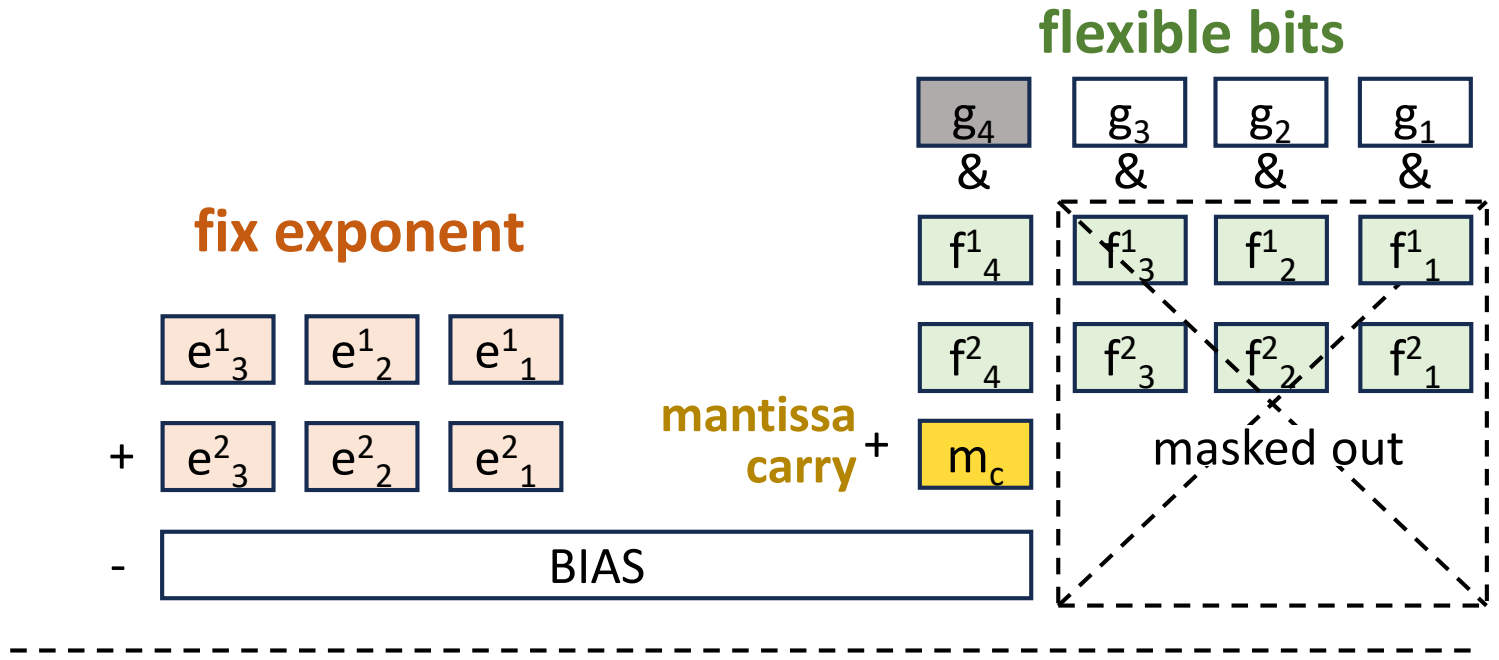
R2F2 Multiplier – Mantissa Multiplication



R2F2 Multiplier – Mantissa Multiplication



R2F2 Multiplier – Exponent Addition



$$e_{res} = e_1 + e_2 - BIAS$$

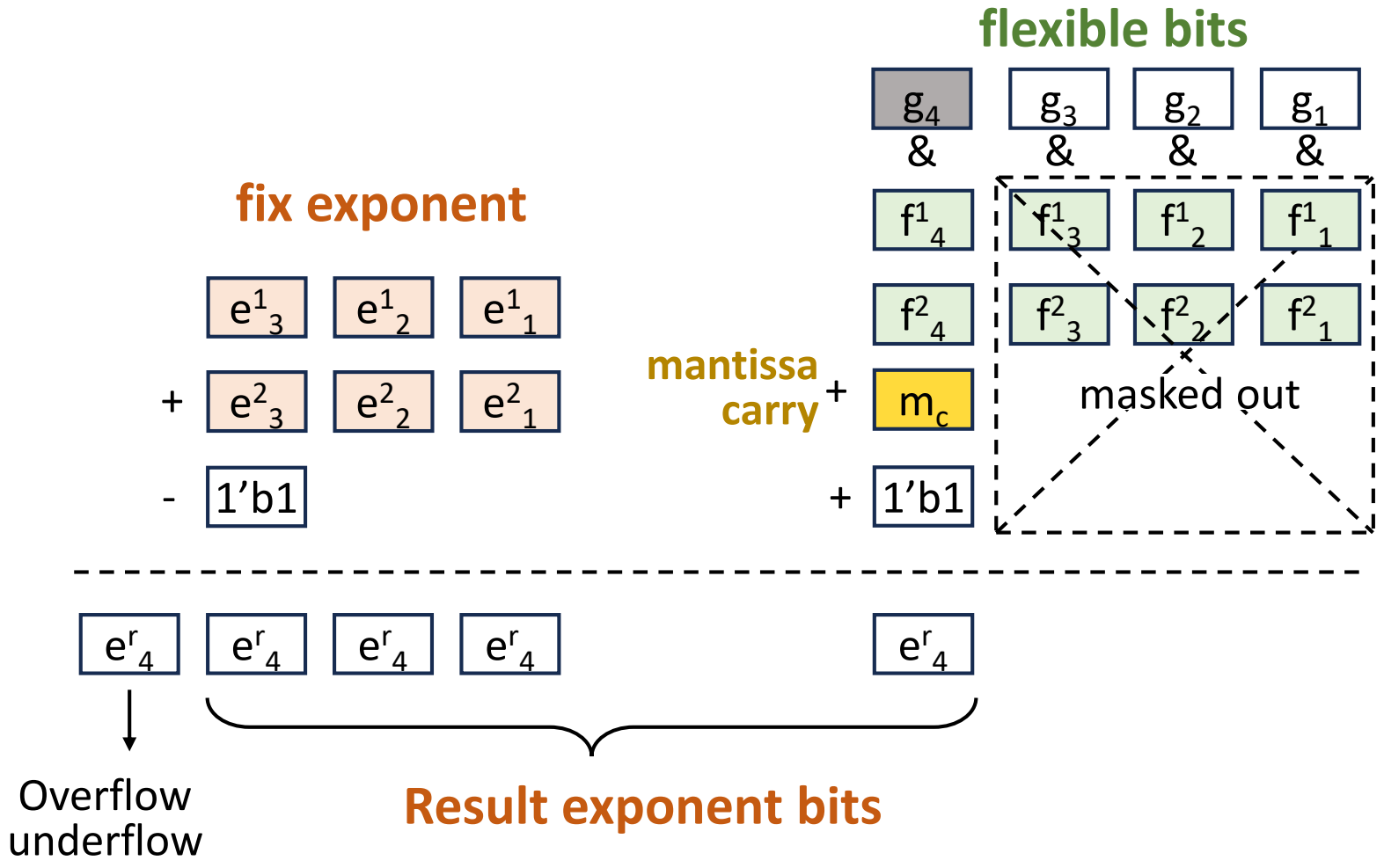
$$BIAS = 2^{|e|-1} - 1 = 1111111$$

Subtracting this is complicated

$$= 10000000 - 1$$

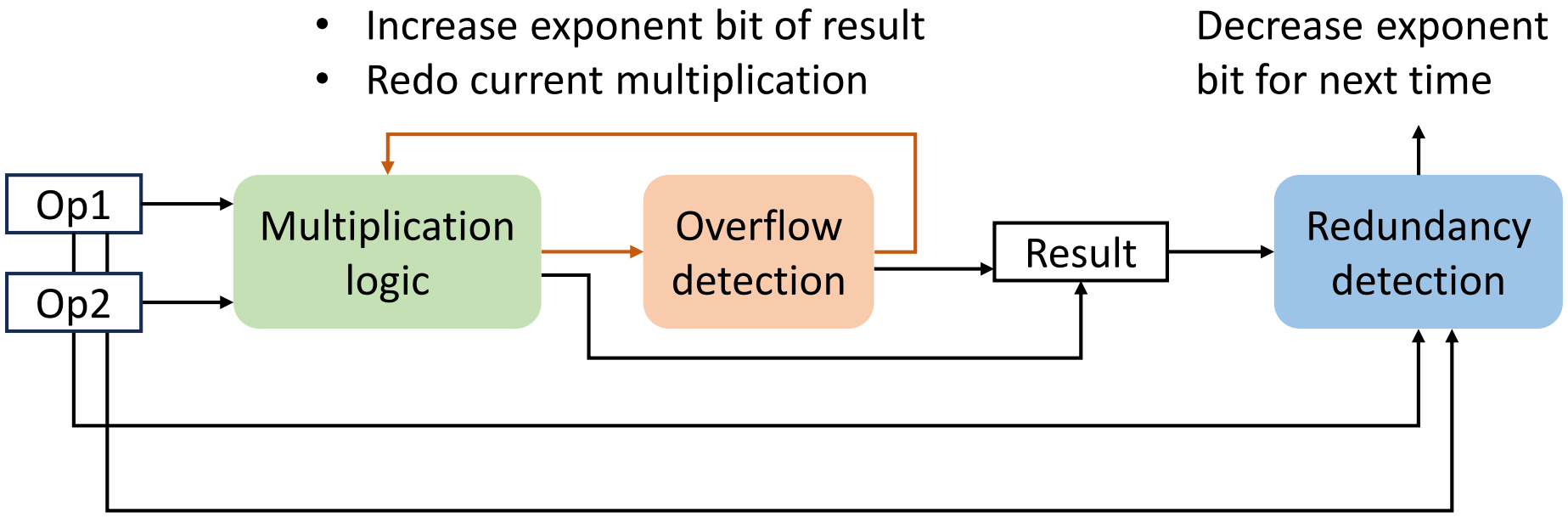
Subtracting this is easy

R2F2 Multiplier – Exponent Addition



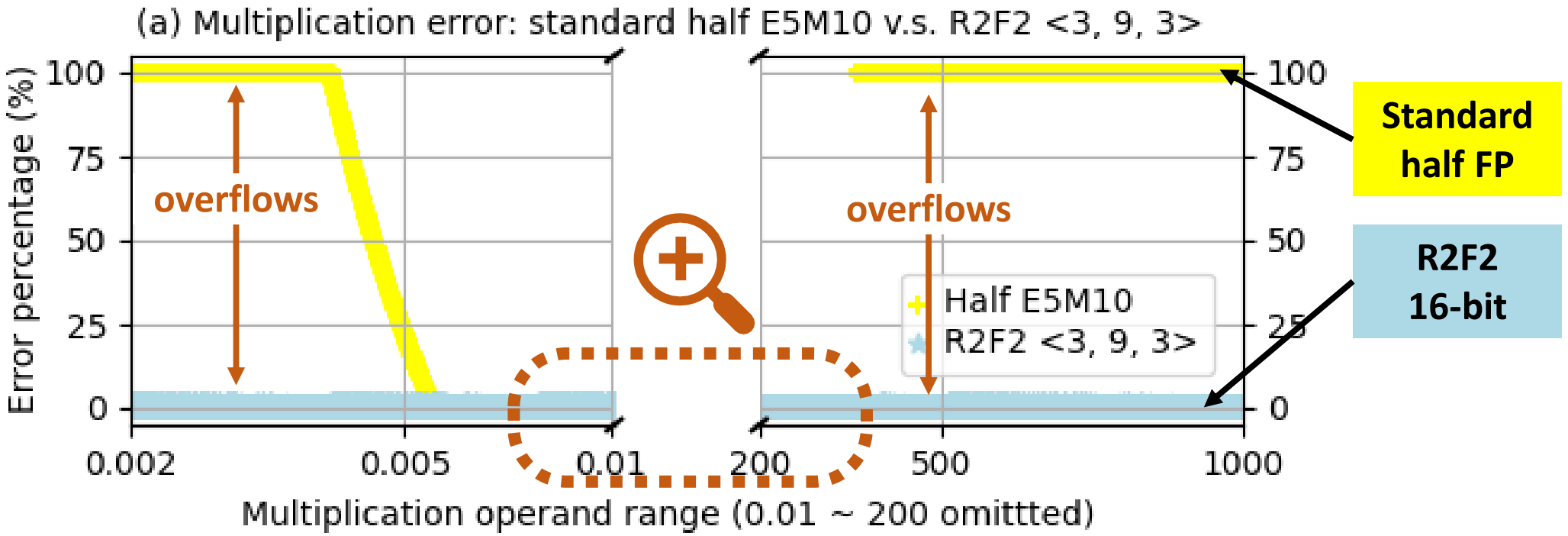
R2F2 Precision Adjustment

- **Precision is controlled by Mask bits, specified by software**
  - Note: this can be improved as future work!



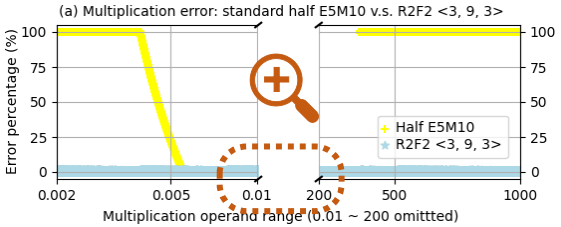
R2F2 Precision Comparing with Standard Types

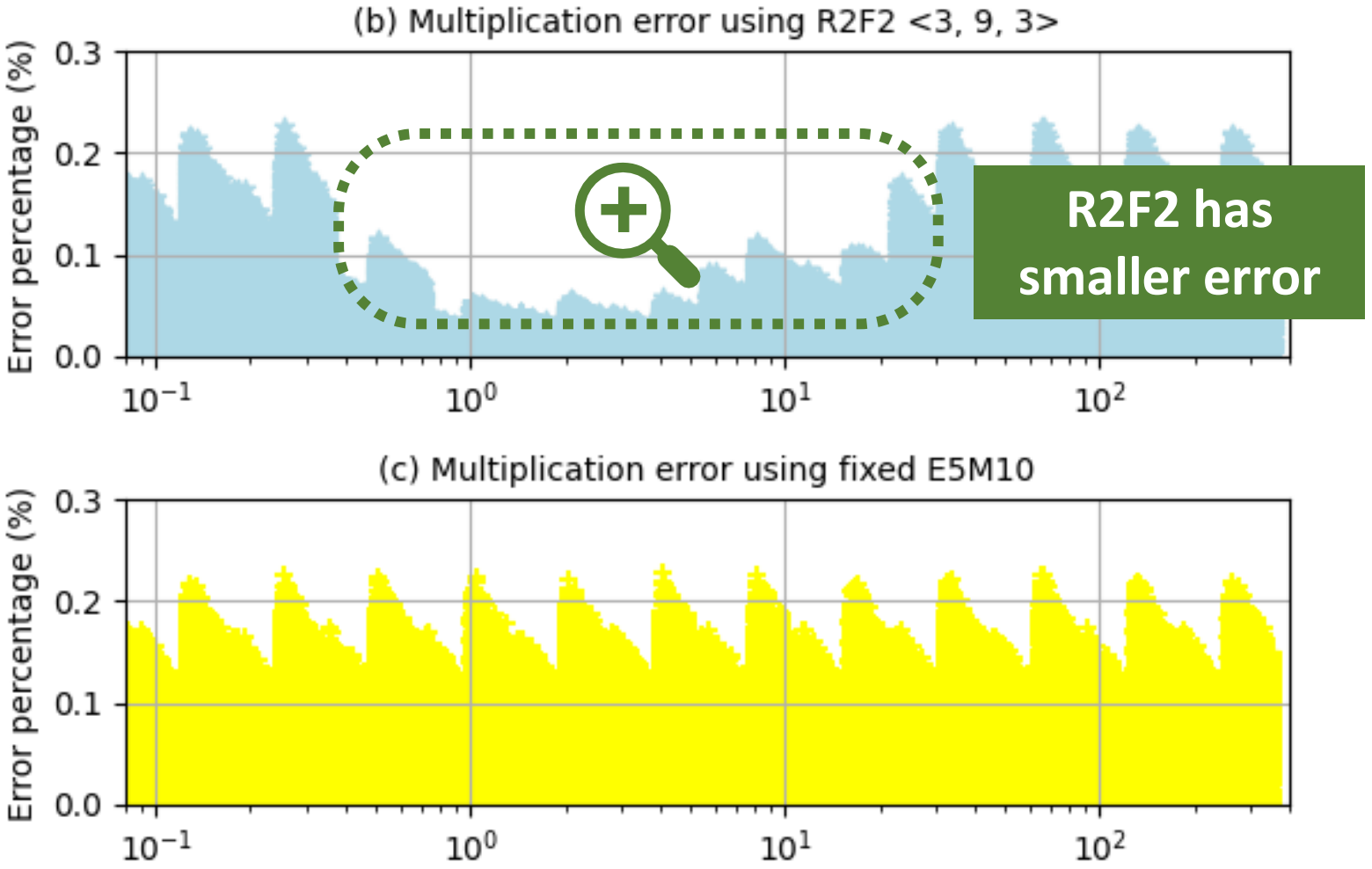
- **Run-time Reconfigurable precision reduces error**
  - Adjustable exponent → avoid overflow
  - When fewer exponent bits and more mantissa bits → higher precision



R2F2 Precision Comparing with Standard Types

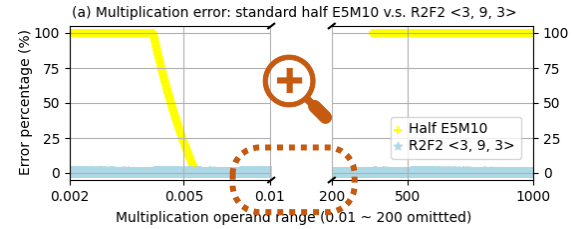
- **Fewer exponent bits and more mantissa bits → higher precision**

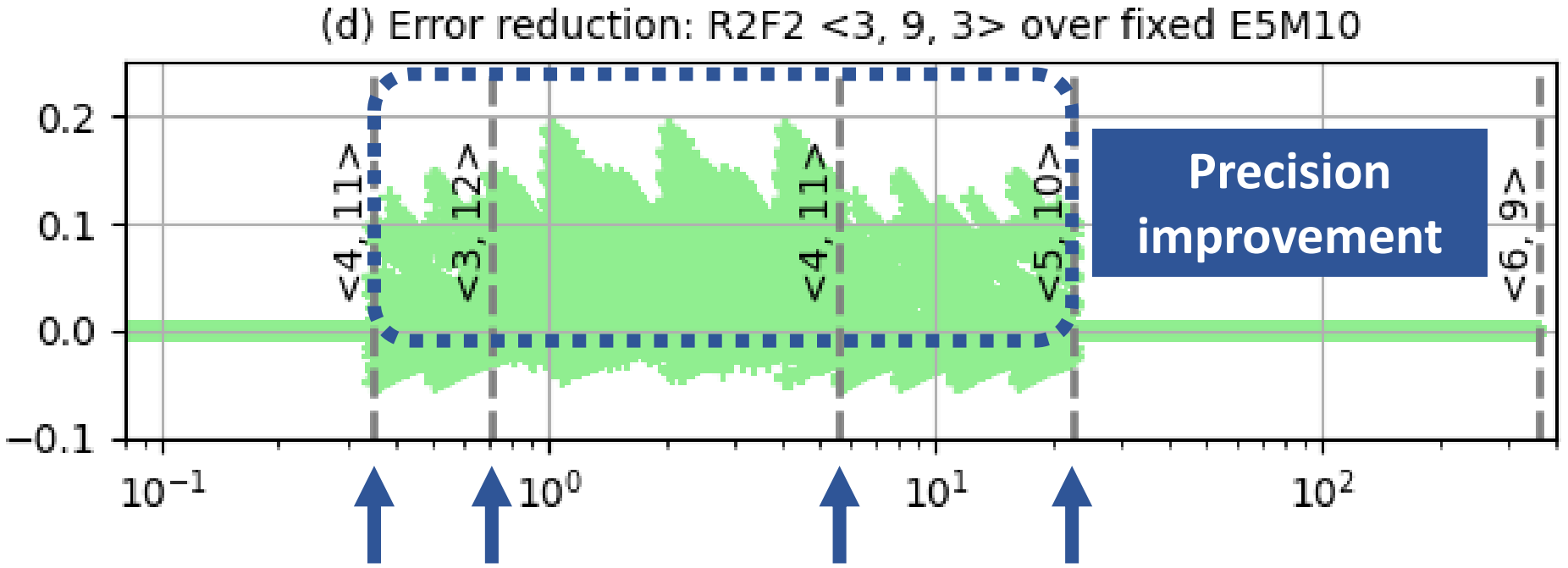




R2F2 Precision Comparing with Standard Types

- **Fewer exponent bits and more mantissa bits → higher precision**

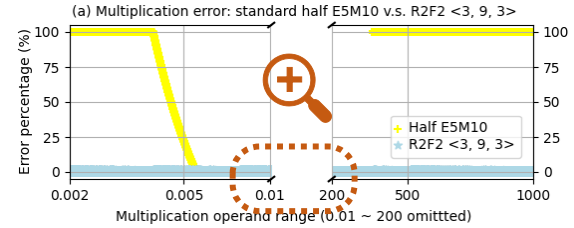


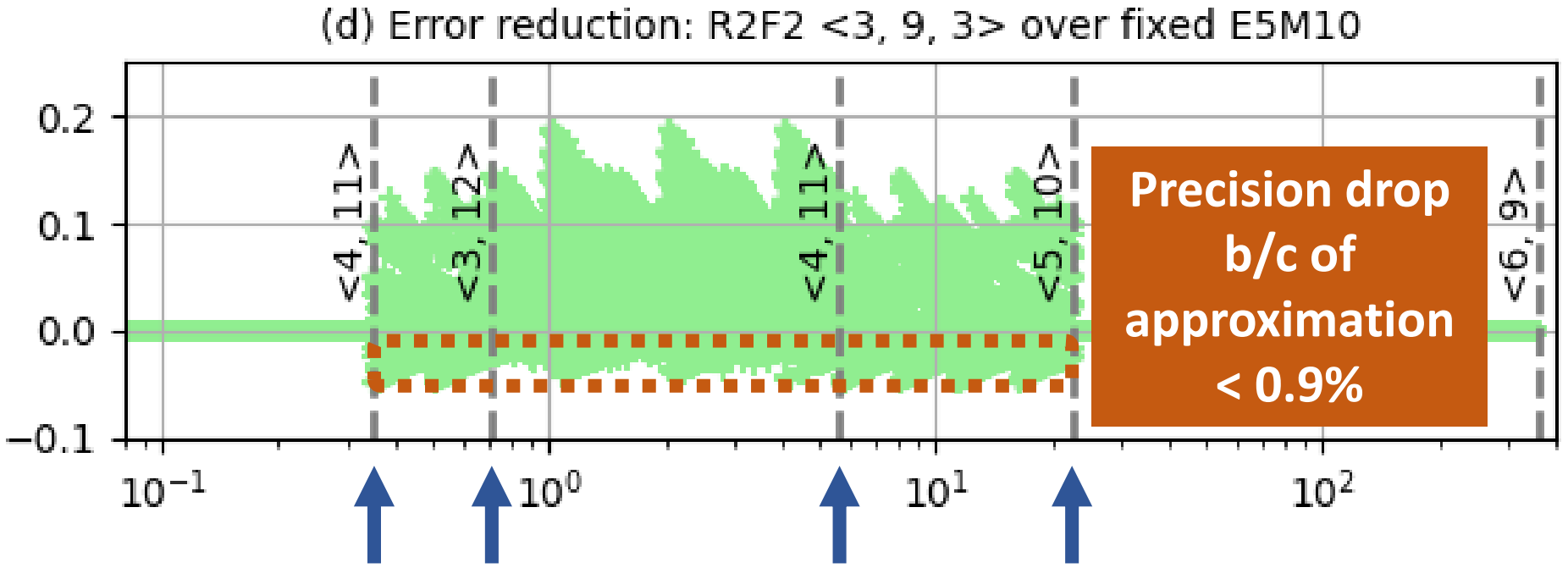


Precision adjustment results in more mantissa bits

R2F2 Precision Comparing with Standard Types

- **Fewer exponent bits and more mantissa bits → higher precision**

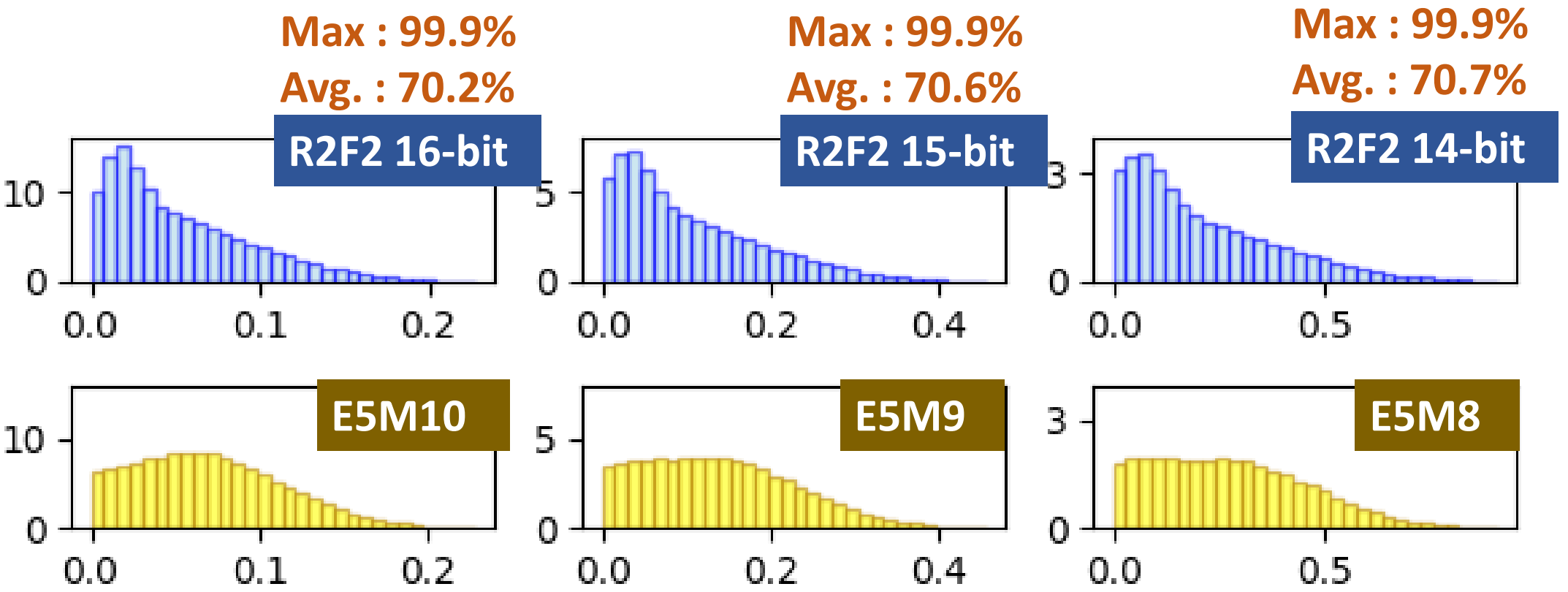




Precision adjustment results in more mantissa bits

R2F2 Precision Comparing with Standard Types

- **Error distribution of R2F2 → mostly smaller errors**



R2F2 Resource/Area Overhead

	Flip Flops		Look-Up Tables		Cycles	
	Cnt	OH	Cnt	OH	Lat.	II
Lib. 64-bit FP (HLS)	2180	-	3264	-	30	11
Lib. 32-bit FP (HLS)	492	-	1438	-	24	5
Lib. 16-bit FP (HLS)	318	-	740	-	26	5
Impl. 64-bit FP	2032	2.82×	15650	3.20×	13	4
Impl. 32-bit FP	1025	1.42×	8093	1.66×	13	4
Impl. 16-bit FP	720	**1.0**	4888	**1.0**	**12**	**4**
R2F2 16-bit <3,9,3>	710	**0.99×**	5161	**1.06×**	12	4
R2F2 16-bit <3,8,4>	720	**1.00×**	5132	**1.05×**	12	4
R2F2 16-bit <3,7,5>	731	**1.02×**	5152	**1.05×**	12	4
R2F2 15-bit <3,8,3>	696	**0.97×**	5091	**1.04×**	12	4
R2F2 15-bit <3,7,4>	713	**0.99×**	5082	**1.04×**	12	4
R2F2 14-bit <3,7,3>	685	**0.95×**	5028	**1.03×**	12	4
R2F2 14-bit <3,6,4>	702	**0.96×**	5249	**1.07×**	12	4

Standard Types

R2F2 Types

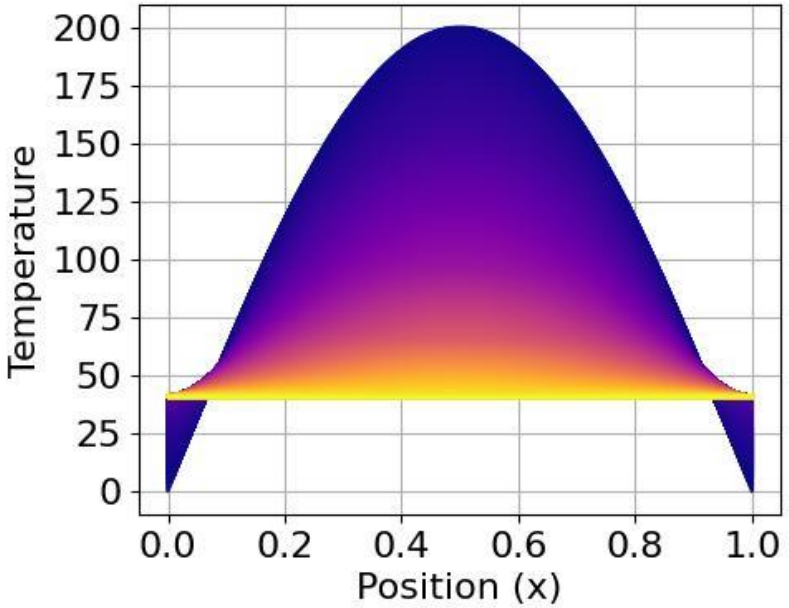
R2F2 in Real PDEs – Case Studies

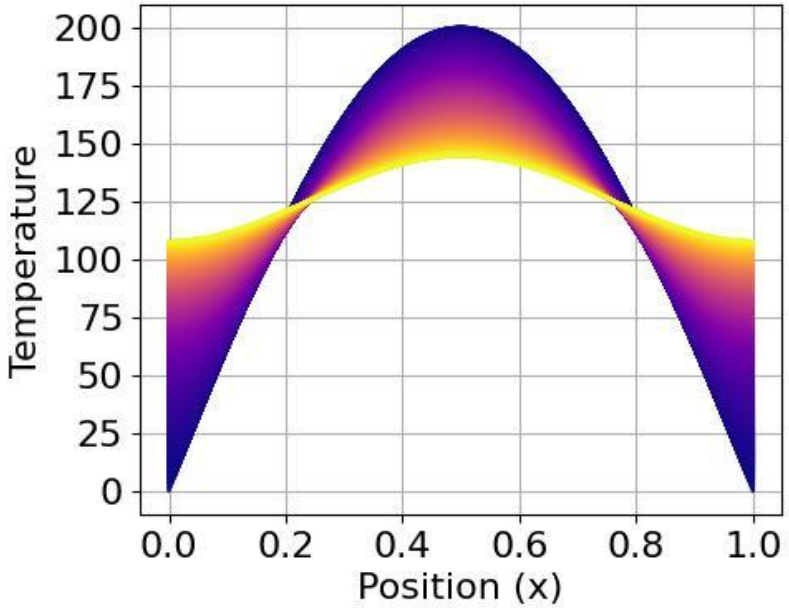
- **Replacing only one multiplier in the entire computation**
  - The multiplier will be used every time step and every spatial step
  - Exponent bits redundancy detected: reduce exponent and increase mantissa
  - Overflow detected: increase exponent bits and redo the computation

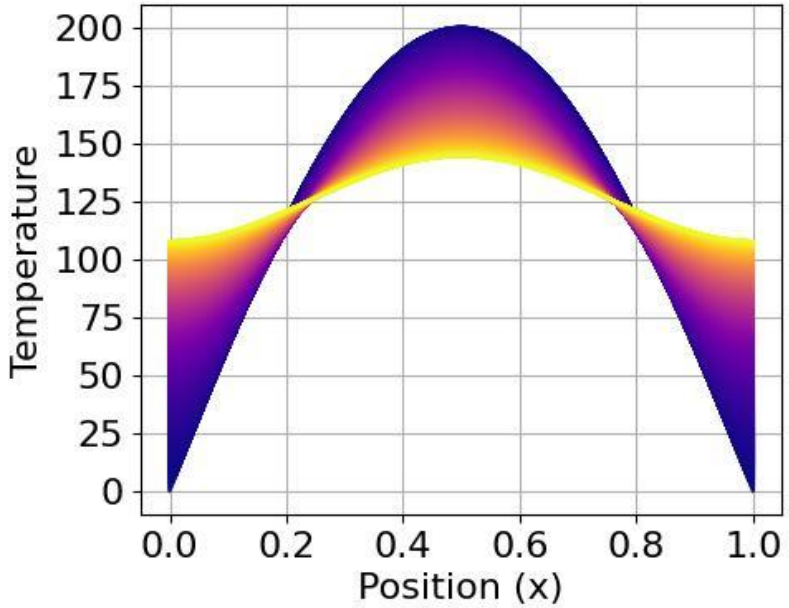
- **Case 1: 1D Heat Equation**
  - 5 multiplications in one iteration – 1 out 5 is replaced

- **Case 2: 2D Shallow Water Equation**
  - 53 multiplications – 1 out 53 is replaced

R2F2 in Real PDEs – 1D Heat Equation







Standard E5M10

16-bit

R2F2 <3, 9, 3>

16-bit

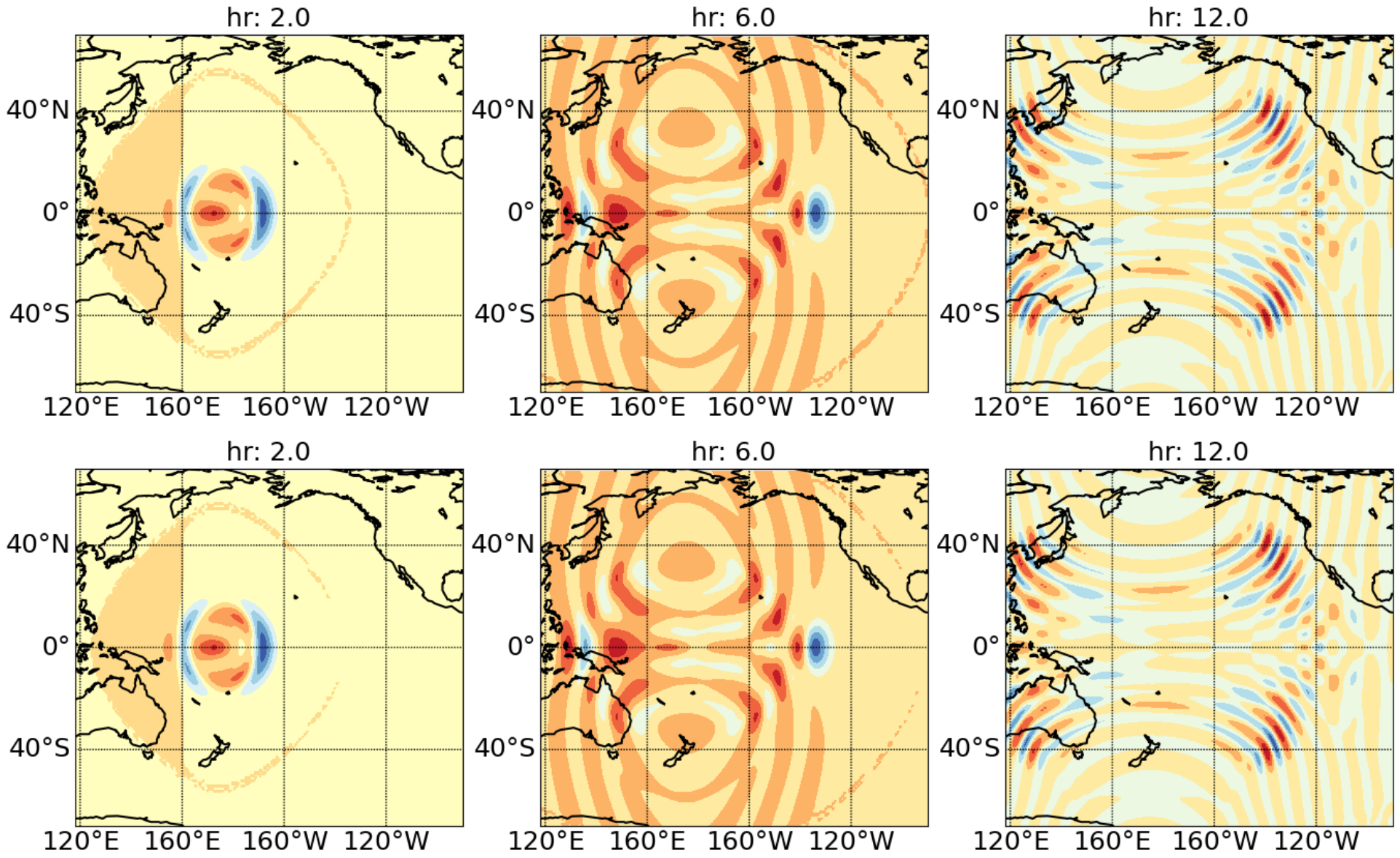
R2F2 <3, 8, 3>

15-bit

R2F2 in Real PDEs – 2D Shallow Water Equations

Standard 64-bit FP

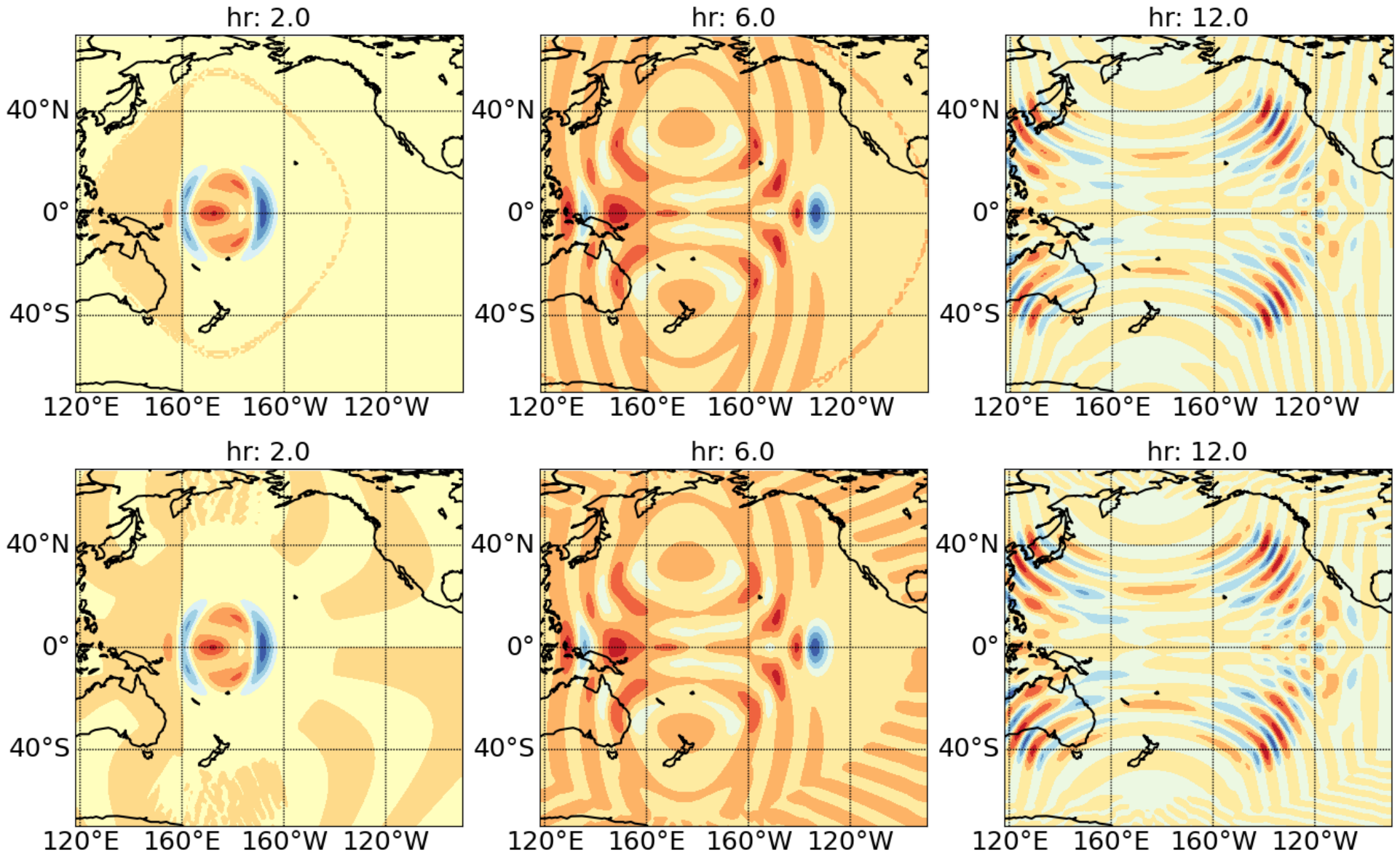
R2F2 16-bit (one eq.)



R2F2 in Real PDEs – 2D Shallow Water Equations

Standard 64-bit FP

Standard 16-bit (one eq.)



Limitations

- **Still limited flexibility:**
  - Fixed total bitwidth, has to be 16-bit
  - Two operands have the same precision
  - Mantissa increases only when exponent is redundant (not that often in reality)
    - Increase mantissa by allowing more flexible bits?

- **No quantitative analysis of:**
  - Overall precision of the algorithm: still very empirical
  - Overall performance gain of the whole algorithm: only one multiplier is evaluated

- **Implementation details have a lot to improve:**
  - When used in a pipeline
  - Format conversion overhead is not considered nor evaluated

Summary

- **Exploration**: can dynamic floating-point data precision be beneficial for scientific computing?
  - If the data range is **globally wide**, **locally narrow**, and **dynamically changing**
- **Exploitation**: efficient hardware to support dynamic precision?
- **R2F2**: a **R**un-time **R**econfigurable, **F**lexible **F**loating-point multiplier
  - Reduces multiplication error by 70% on average comparing with standard types
  - Can run simulation with high fidelity when standard type fails
  - Up to 5% resource overhead and no latency overhead

More information

- **Source code: https://github.com/sharc-lab/R2F2**
  - Arbitrary-precision floating point multipliers
  - HLS implementation of R2F2
  - PDE source code for the two case studies

Software/Hardware Co-Design for Intelligence and Efficiency

Email: callie.hao@ece.gatech.edu
Homepage: http://sharclab.ece.gatech.edu