UNIVERSITY OF LEEDS

Implementation and Standardization of Stochastic Rounding

Mantas Mikaitis

School of Computing, University of Leeds, Leeds, UK

TARAN Team Seminar, Inria Centre at Rennes University

Rennes, France, Apr. 9, 2024

Slides: `https://mmikaitis.github.io/talks`

Introduction

- In binary floating-point hardware **round-to-nearest** (RN) is a default mode (standardized by IEEE 754).
- Deterministic, optimal accuracy per operation.
- Closest machine number to real answer—cannot improve.
- Over many rounding ops may accumulate error of factor n, where n a problem size.

What we get from today's talk

Learn about the implementation of **stochastic rounding** (SR) which enforces probabilistic error bound with factor $\sqrt{n}$.

Floating-point (FP) number representation

A floating-point system $F \subset \mathbb{R}$ is described with $\beta, t, e_{min}, e_{max}$ with elements

$$x = \pm m \times \beta^{e-t+1}.$$

Virtually all computers have $\beta = 2$ (binary FP).

Here t is precision, $e_{min} \leq e \leq e_{max}$ an exponent, $m \leq \beta^p - 1$ a significand (m, t, $e \in \mathbb{Z}$).

Standard model [Higham, 2002]

Given $x, y \in \mathbb{R}$ that lie in the range of F it can be shown that

$$\mathrm{fl}(x \text{ op } y) = (x \text{ op } y)(1 + \delta), \quad |\delta| \leq u,$$

where $u = 2^{-t}$, $\text{op} \in \{+, -, \times\}$ and **round-to-nearest** mode.

Rounding error analysis

Rounding errors δ accumulate. For example, consider computing $s = x_1y_1 + x_2y_2 + x_3y_3$.

We compute $\widehat{s}$ with

$$\begin{aligned}\widehat{s} &= \Big(\big(x_1y_1(1+\delta_1) + x_2y_2(1+\delta_2)\big)(1+\delta_3) + x_3y_3(1+\delta_4)\Big)(1+\delta_5)\\ &= x_1y_1(1+\delta_1)(1+\delta_3)(1+\delta_5) + x_2y_2(1+\delta_2)(1+\delta_3)(1+\delta_5)\\ &\quad + x_3y_3(1+\delta_4)(1+\delta_5).\end{aligned}$$

Therefore we deal with a lot of terms of the form $\prod_{i=1}^{n}(1+\delta_i)$.

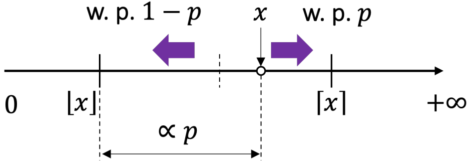
Worst case backward error bound (exact result for perturbed inputs)

$\prod_{i=1}^{n}(1+\delta_i) = 1 + \theta_n, \quad |\theta_n| \leq \gamma_n$, with $\gamma_n = \frac{nu}{1-nu}$ and assuming $nu < 1$.

What is stochastic rounding

With **stochastic rounding** (**SR**), we are not rounding a number to the same direction, but to either direction with probability.

Given some x and FP neighbours $\lfloor x \rfloor$, $\lceil x \rceil$, we round to $\lceil x \rceil$ with prob. p and $\lfloor x \rfloor$ with $p - 1$.



Mode 1 SR (nearness): $p = \frac{x - \lfloor x \rfloor}{\lceil x \rceil - \lfloor x \rfloor}$ **Mode 2 SR**: $p = 0.5$

Mode 2

With **Mode 1 SR** we round x depending on its distances to the nearest two FP numbers, **cancelling out errors of different signs**.

Rounding error analysis with SR

Standard error model for SR

With SR we replace u by $2u$ since it can round to the second nearest neighbour in F.

Rounding error analysis

Worst-case error analysis determines the **upper bounds of errors**, while probabilistic error analysis describes **more realistic bounds**.

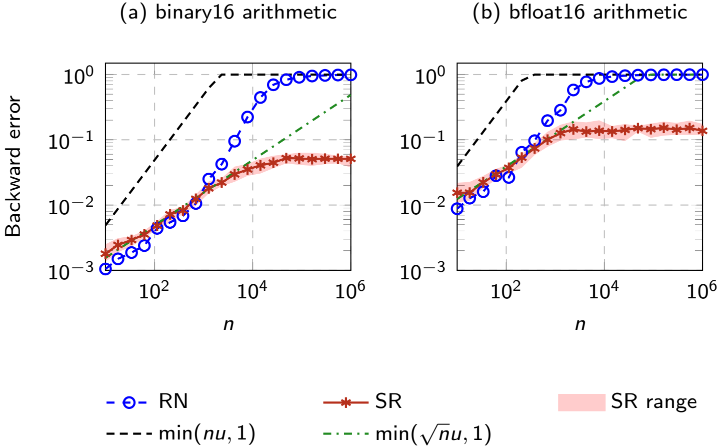
- Worst-case b-err bound with **RN**: $\frac{nu}{1-nu}$.
- Probabilistic bound with **RN**: $\lambda\sqrt{n}u + \mathcal{O}(u^2)$ w. p. $1 - 2ne^{-\lambda^2/2}$. Requires an assumption that δ_n are *mean independent zero-mean* quantities—often satisfied [Connolly, Higham, Mary, 2021].

Wilkinson rule of thumb

$\sqrt{n}u$ error growth is a rule of thumb with **RN**, but always holds with **SR**.

Example error growth with SR in mat-vec prod

Backward error in $y = Ax$ where $A \in \mathbb{R}^{100\times n}$ with entries from uniform dist over $[0, 10^{-3}]$ and $x \in \mathbb{R}^n$ over $[0, 1]$: $\max_i \frac{|\widehat{y}-y|_i}{(|A||x|)_i}$.



Stagnation

Take binary floating-point numbers a and b, such that $a \gg b$ and $\mathrm{fl}(a + b) = a$ (round-to-nearest).

In sums of arbitrary length, $s_n = x_1 + x_2 + \cdots + x_n$, *stagnation* appears if, for example, $\mathrm{fl}(x_1 + x_i) = x_i$ for $i \leq n$ and therefore $\widehat{s_n} = x_1$.

If $x_i > 0$, the total error is $x_2 + \cdots + x_n$, which is a growth of factor $(n - 1)u$.

The assumptions in probabilistic bound for RN do not hold.

Stagnation/swamping

Whole, part, or parts of a running sum do not change the intermediate value, and addends contribute wholly to the error.

Stagnation

With SR, stagnation *is not as severe* as with RN.

Take again a and b, such that $a \gg b$ and with RN $\mathrm{fl}(a+b) = a$.

With SR $\mathrm{fl}(a+b)$ will yield a or the next floating-point value with probability $\frac{b}{\mathrm{ulp}(a)}$ where $\mathrm{ulp}(a)$ is the gap between a and next fl. val.

Stagnation with SR

Stagnation can still occur if b is so small that its significand gets shifted past the random bits in SR; probabilistic bounds will not hold and drop back to factor *nu*.

Other theoretical results

[Arar, Sohier, Castro, Petit, 2022, 2023] extend the probabilistic bound results to Horner's scheme and tighten some of the bounds.

Follow references therein for various other results: Ipsen & Zhou (probabilistic bounds), Croci & Giles (PDE solvers).

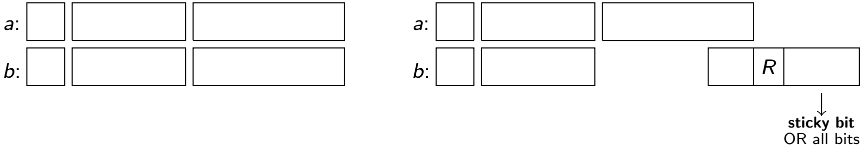
Key takeaways from theory about SR

- Probabilistic bounds hold unconditionally.
- Experimentally $\sqrt{n}$ growth is observed.
- Stagnation that breaks assumptions for probabilistic bounds with RN, does not appear in SR.

But beware of stagnation in limited-precision SR—it can still occur and break probabilistic bound assumptions.

How do we implement this? First, consider standard modes

Consider $a, b \in \mathbb{F}$ with $a, b > 0$ and $a > b$.



round-sticky	RD	RU	RN
00	D	D	D
01	D	U	D
10	D	U	D/U
11	D	U	U

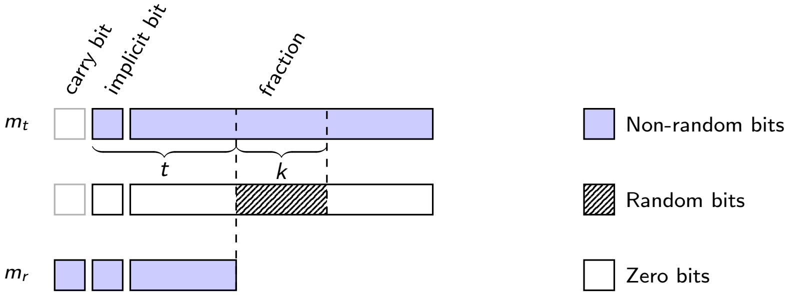
Guard bit

Guard bit is a complication that arises when we consider non-normalized floating-point significands, to compute the R bit correctly.

Implementation of SR

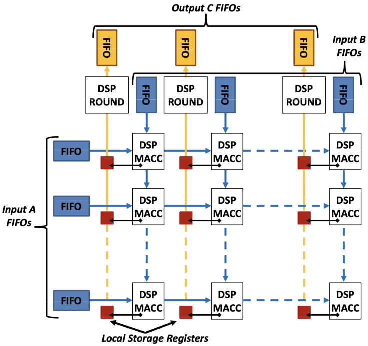
Take m_t to be a high precision unrounded significand from an operation.

Take t to be source precision and k the precision of random numbers.

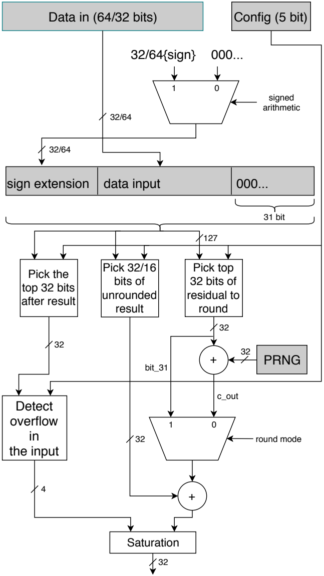


Fixed-point inner product implementation

- Probably one of the first hardware implementations of SR by [Gupta et al. 2015].
- 18-bit fixed-point inputs/outputs.
- Exact dot products in 48-bit internal format.
- SR applied once, at the end on an exact 48-bit result matrix.
- LSFR for PRNG.
- SR: 4% HW overhead.



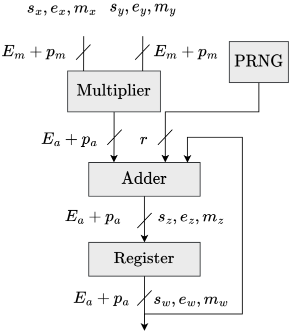
Hybrid fixed/floating-point hardware implementation



- Design and synthesis study available [Mikaitis, 2021].
- RN and SR in one.
- 32- or 64-bit fixed-point inputs.
- Programmable destination precision: round 1 to 32 bits.
- binary32 → bfloat16 rounding (16 bits).
- 32-bit uniform PRNG with 4 separate streams (seeds can come from TRNG).
- Accelerator integrated to each core in a 152-core chip (ARM M4F).
- Operation: Write to a memory location, read back rounded.

Eager rounding implementation in accumulation

- [Ali, Filip, Sentieys, 2024] propose an approach to lower critical path.
- Applied in deep learning: 8-bit FP products accumulated in 12-bit FP.
- Accumulator is rounded stochastically. No rounding in multiplier—exact.
- *Lazy* implementation: perform SR after normalization of sum (implement algorithm step-by-step).
- *Eager* implementation: perform SR after the alignment of significands, correct later if needed.

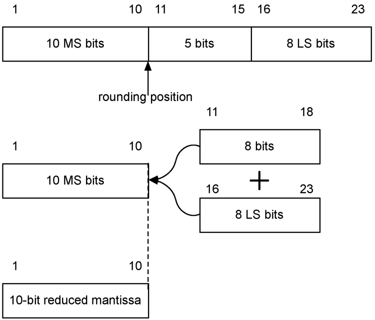


Patents from industry

There are numerous patents for SR from industry giants: NVIDIA, AMD, IBM. See our SR survey [Croci et al, 2022].

Here we focus on NVIDIA's ([NVIDIA, 2019]).

Below binary32 $\rightarrow$ binary16 example.



- Does not use PRNG.
- Take 8 bottom discarded bits and add to the top 8.
- Deterministic and cheaper to implement.
- Effect on numerical results not known.

SR in hardware

Commercial hardware that implements SR is for machine learning:

- **Graphcore IPU**
- **Intel Loihi**
- **Tesla Dojo**
- **Amazon Trainium**

Custom precision simulators with SR

- Various packages available: `chop`, `FLOATP`, `QPyTorch`.
- Usual approach is to perform ops in binary32/64 HW.
- Round down to sub-32-bit precision: careful with double rounding.
- We believe ours is most customizable and fastest: `CPFloat` [Fasi & Mikaitis, 2023].
- Can be used in MATLAB, Octave or C.

Example with CPFloat in MATLAB

```
>> options.format = 'bfloat16';
>> options.round = 5;
>> cpfloat(pi, options)
ans =
   3.142578125000000
>> options.format = 'fp8-e5m2';
>> cpfloat(pi, options)
ans =
   3.500000000000000
>> cpfloat(pi, options)
ans =
   3
>> cpfloat(pi*pi, options)
ans =
    10
```

Proposed IEEE 754 style properties

There is no standard way to implement SR.

We proposed a set of rules ([Croci et al, 2022]):

- If $x \in F$, $\mathrm{SR}(x) = x$.
- If x is in the range of F, round as though x is held in $t + k$ bits and rounded to t bits.
- **Overflows**: numbers between maximum value and $\pm\infty$: round as though exponent is not limited.
- When x is smaller than the smallest representable number, round stochastically to zero or that smallest number.
- If **subnormals** are disabled, round to zero or smallest normalized value.
- $\pm\infty$ and ± 0 should not be changed. NaNs should not be rounded.
- Exceptions signalled as standard.

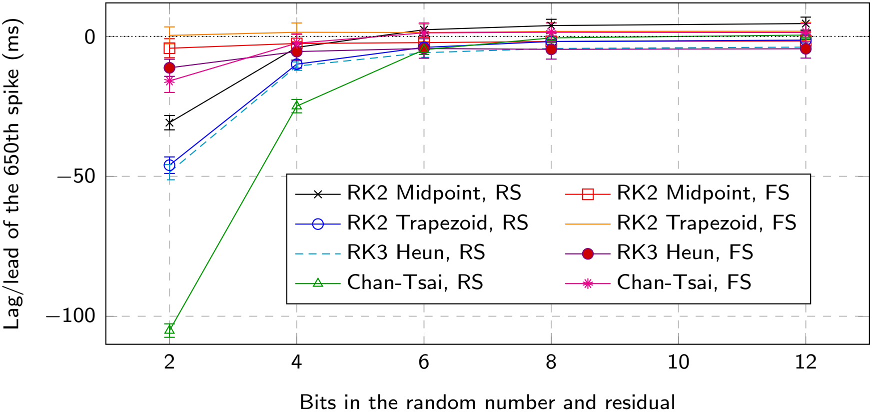
Proposed IEEE 754 style properties: outstanding questions

- What PRNG requirements should be specified: algorithms, quality?
- What are requirements around k (precision of PRNG)?
- What to do with argument bits past $t + k$ pre-SR application: drop or round?

Random number precision in SR

The question of k, precision of random numbers in SR, still open.

We did some experiments with ODE solvers in fixed-point arithmetic (Hopkins et al, 2020).



Summary

Key points

- Theory promises clear advantages with SR where probabilistic bounds do not hold for RN.
- Beware that stagnation can still occur with SR.
- Implementations are known, but key questions on random number generation remain.
- Precision and quality of random numbers.
- No official standard.

More details in the stochastic rounding survey paper

M. Croci, M. Fasi, N. J. Higham, T. Mary, and M. Mikaitis. *Stochastic rounding: implementation, error analysis and applications*. **R. Soc. Open Sci.**. Mar. 2022.
`https://bit.ly/3Kzw7mA`.

Leeds Mathematical Software and Hardware Lab

New informal group in the School of Computing, Univ. Leeds.

Massimiliano Fasi
Lecturer
Research&Teaching

Mantas Mikaitis
Lecturer
Research&Teaching

- Focusing on computer arithmetic, numerical linear algebra, high-performance computing.
- Working with IEEE P3109 and IEEE 754-2029.
- Serving on PC committees of ARITH.
- Planning MSc module on computer arithmetic.
- PhD studentships available.

Acknowledgements

Professor Nicholas J. Higham (1961–2024).

I am grateful to my collaborators on this work: Matteo Croci, Max Fasi, Michael Hopkins, and Theo Mary.

References I

N. J. Higham
Accuracy and Stability of Numerical Algorithms.
2nd ed. SIAM. 2002.

M. P. Connolly, N. J. Higham, T. Mary
Stochastic rounding and its probabilistic backward error analysis.
SIAM J. Sci. Comput., 43. 2021.

E.-M. El Arar, D. Sohier, P. de Oliveira Castro, E. Petit
The Positive Effects of Stochastic Rounding in Numerical Algorithms.
29th IEEE Symposium on Computer Arithmetic (ARITH). 2022.

E.-M. El Arar, D. Sohier, P. de Oliveira Castro, E. Petit
Stochastic Rounding Variance and Probabilistic Bounds: A New Approach.
SIAM J. Sci. Comput., 45. 2023.

References II

S. Gupta, A. Agrawal, K. Gopalakrishnan, P. Narayanan
Deep Learning with Limited Numerical Precision.
Proceedings of Machine Learning Research (PMLR). 2015.

M. Mikaitis
Stochastic Rounding: Algorithms and Hardware Accelerator.
International Joint Conference on Neural Networks (IJCNN). 2021.

S. B. Ali, S.-I. Filip, O. Sentieys.
A Stochastic Rounding-Enabled Low-Precision Floating-Point MAC for DNN Training.
27th IEEE/ACM Design, Automation and Test in Europe (DATE). 2024.

J. M. Alben, P. Micikevicius, H. Wu, M. Y. Siu.
Stochastic Rounding of Numerical Values.
2019. Patent Status: Active.

References III

M. Fasi, M. Mikaitis
CPFloat: A C library for emulating low-precision arithmetic.
ACM Trans. Math. Soft., 49. 2023.

M. Hopkins, M. Mikaitis, D. R. Lester, S. Furber
Stochastic rounding and reduced-precision fixed-point arithmetic for solving neural ordinary differential equations.
Phil. Trans. R. Soc., 378. 2020.