SAND2023-00215C

Sandia National Laboratories

Probabilistic computing and stochastic devices




Shashank Misra[1], Christopher R. Allemang[1], J. Darby Smith[1], Laura Rehm[2], Andrew D. Kent[2], Jean Anne C. Incorvia[3], Leslie C. Bland[4], Catherine Schuman[5], Suma G. Cardwell[1], and Bradley Aimone[1]

[1]Sandia National Laboratories
[2]New York University
[3]University of Texas – Austin
[4]Temple University
[5]University of Tennessee - Knoxville

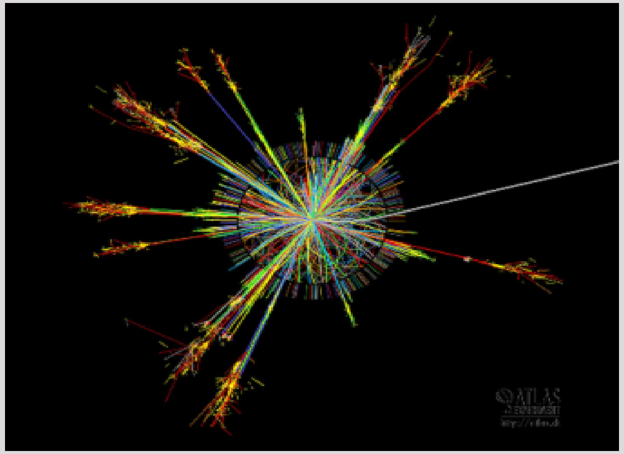
Sandia National Laboratories is a multimission laboratory managed and operated by National Technology and Engineering Solutions of Sandia LLC, a wholly owned subsidiary of Honeywell International Inc. for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

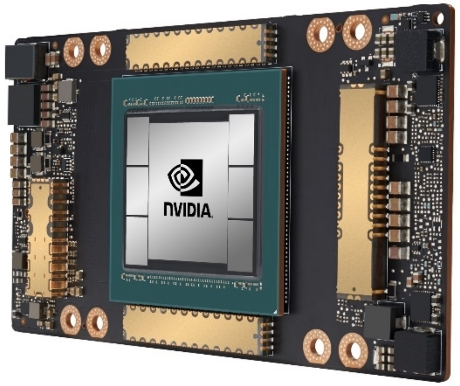
Probabilistic computing

<u>**Artificial Intelligence**</u>

Which approach is best to interpret an ambiguous input?

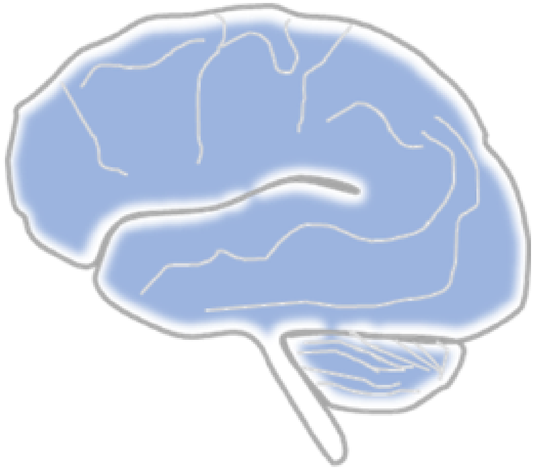


Modeling and Simulation



~400 W
~10^{13}-10^{14} FLOPS
Run simulation many times

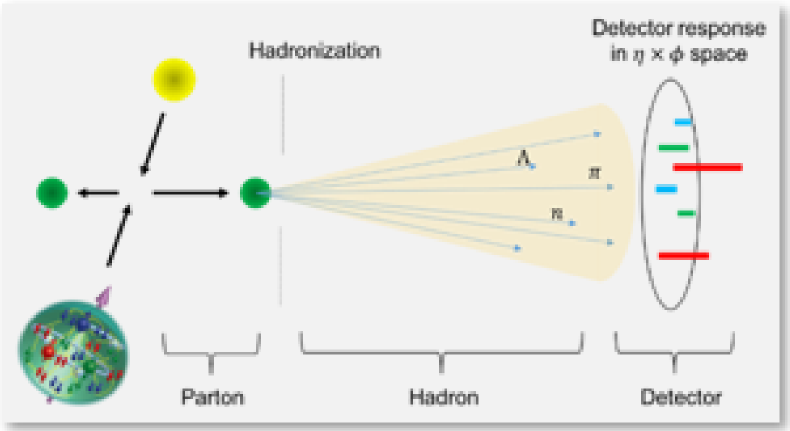
~20 W
~10^{15} events / second
Fully stochastic



Combine stochastic devices with neuromorphic approaches to solve problems where probabilistic outcomes are important

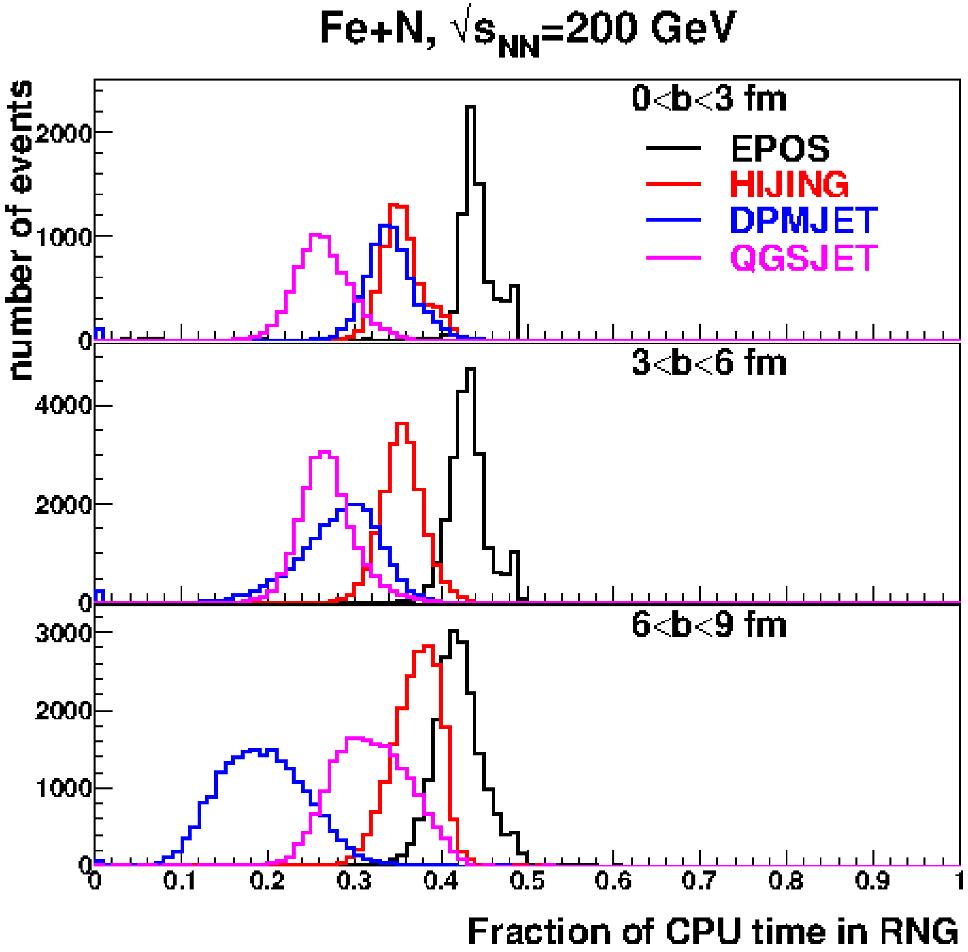
Monte Carlo integration

Event generator for cosmic rays



Need 10^{12} samples

~25-50% spent on PRNG, more including non-uniform sampling



Courtesy of Les Bland

Six orders of magnitude efficiency moving PRNG → TRNG

- PRNGs from standard library: ~ 1 µJ
- TRNG (MTJ): < 1 pJ

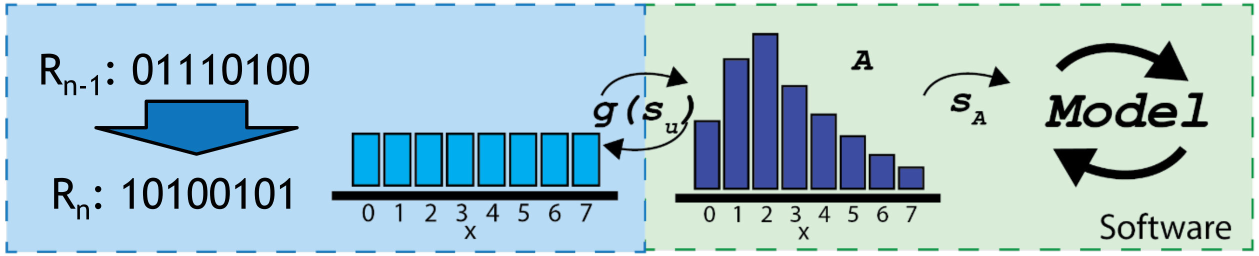
*L. Rehm, Phys. Rev. Applied **19**, 024035 (2023)*

S. Misra, Adv. Mater. 2022, 2204569 (2022)

This talk

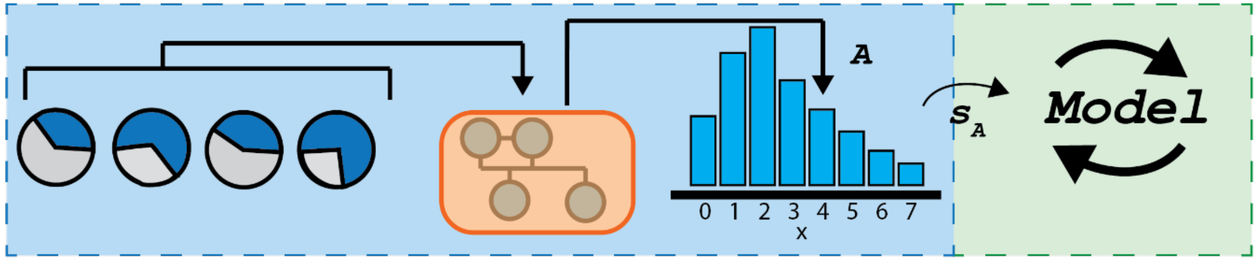
How this is done now:

- CPU generates a uniform pseudo-random number
- Numerical transformation to distribution needed
- Model runs calculation based on sample



This talk:

- TRNG directly samples distribution
- Model runs calculation based on sample



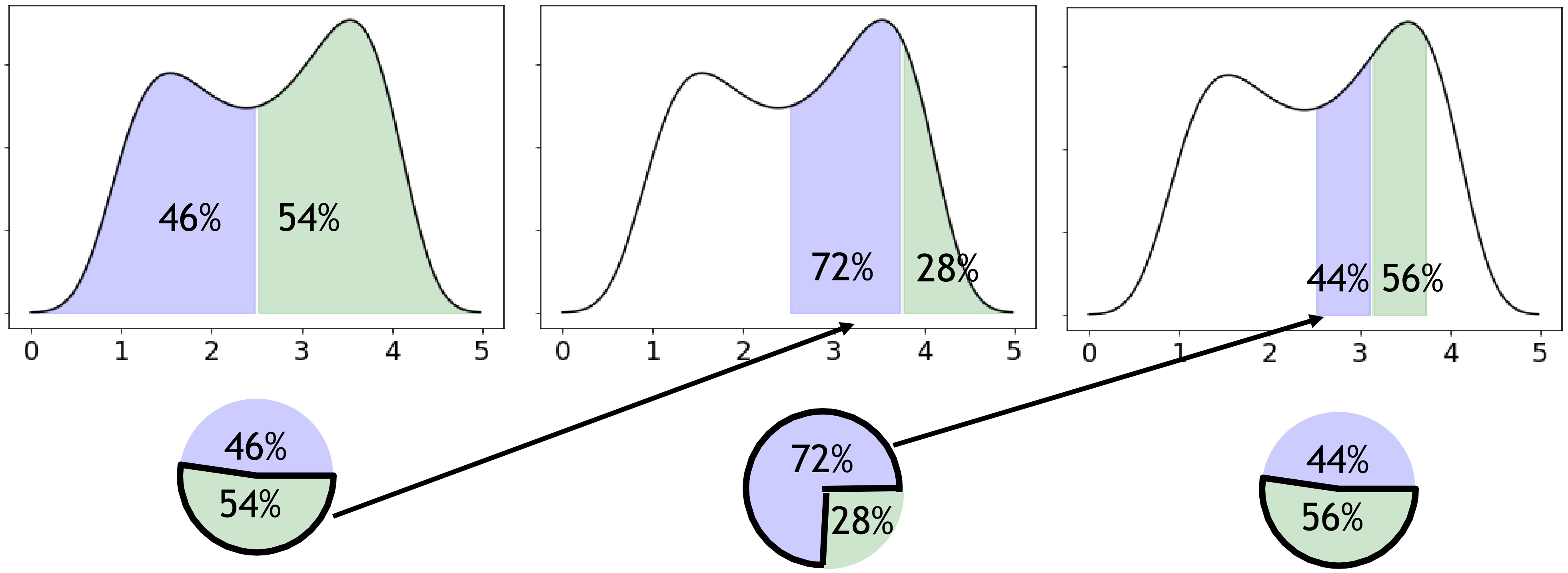
1) How can we directly sample a distribution?
2) How 'good' does a weighted coin have to be?
(magnetic tunnel junction, tunnel diode)

Use weighted stochastic devices to ‘search’ distribution

Top half or bottom half? Top quarter or 3rd quarter? …

Where in the distribution should this sample come from?

Sequence of weighted coin tosses

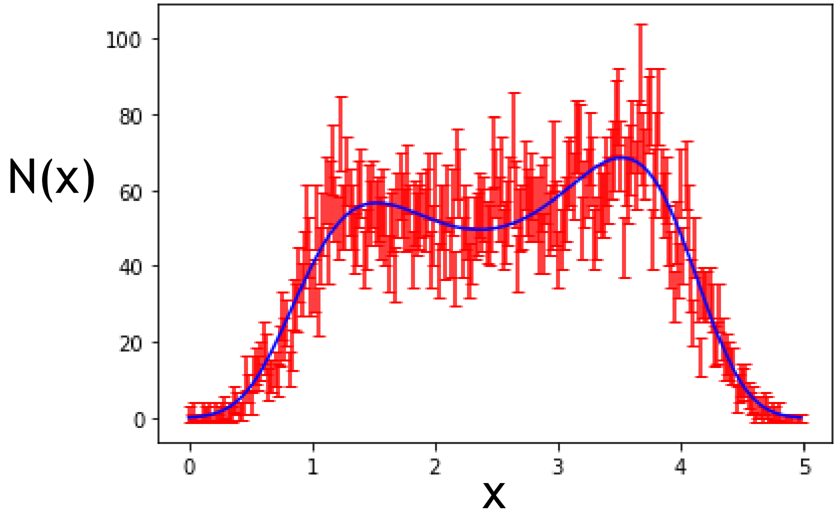


Sample well-behaved distribution on discretized domain using weighted coinflips

Quality of samples from non-uniform distribution

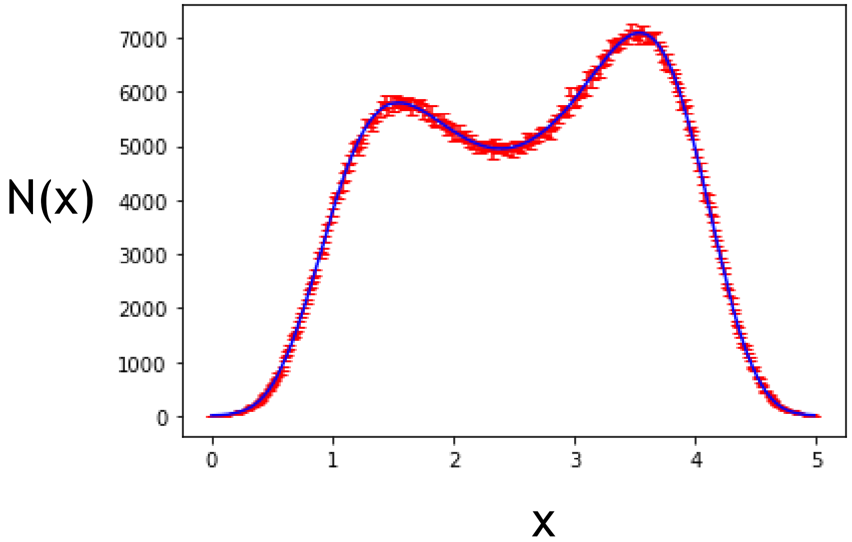
Evaluate quality of sample distribution using curve fitting

PRNG: p_i, $N_{TOT}=10^4$



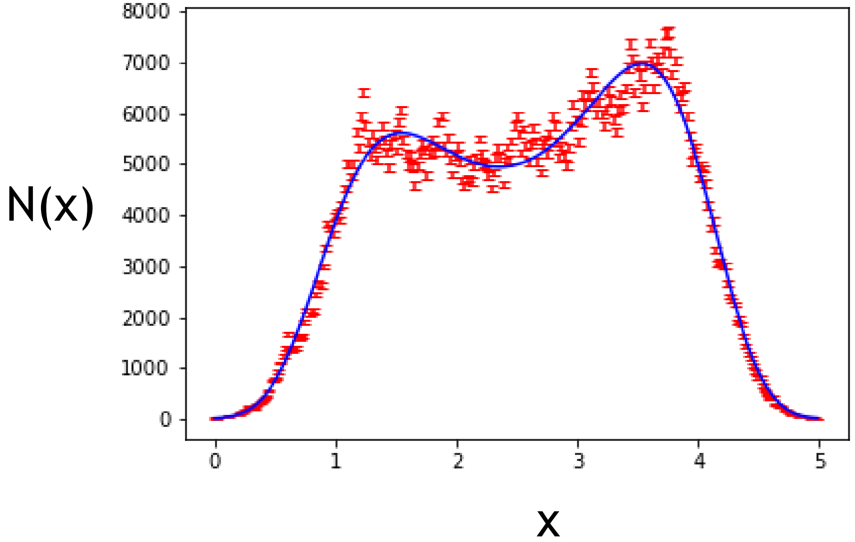
✓ X^2~1

PRNG: p_i, $N_{TOT}=10^6$



✓ X^2~1

PRNG: $p_i + \delta$, $N_{TOT}=10^6$

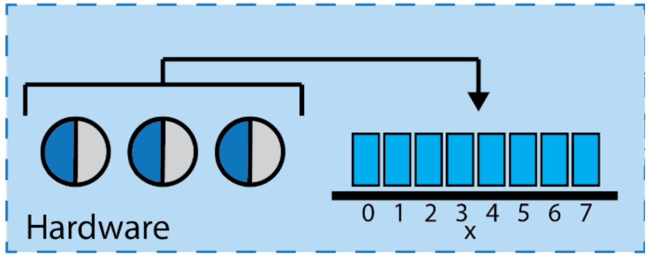


✗ X^2>1

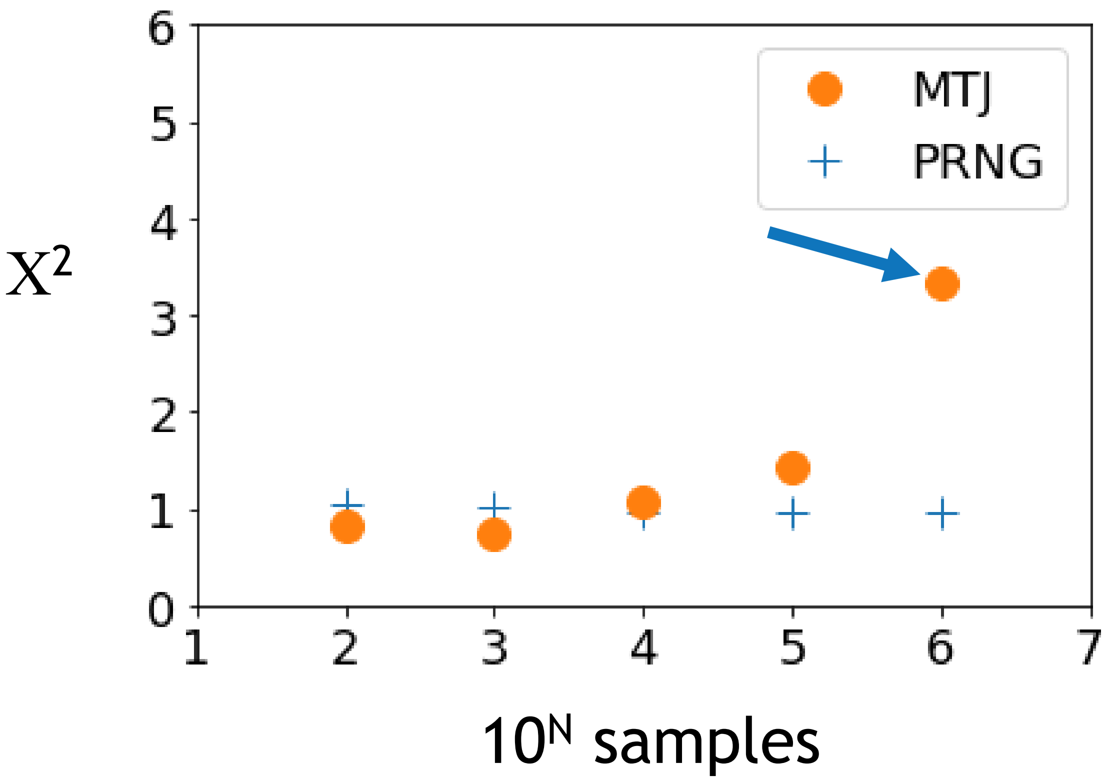
Evaluate uniform samples generated with NYU MTJ data

Evaluation scheme

1. Tune device to $p_i = 0.5$
2. Generate 6-bit uniform sample

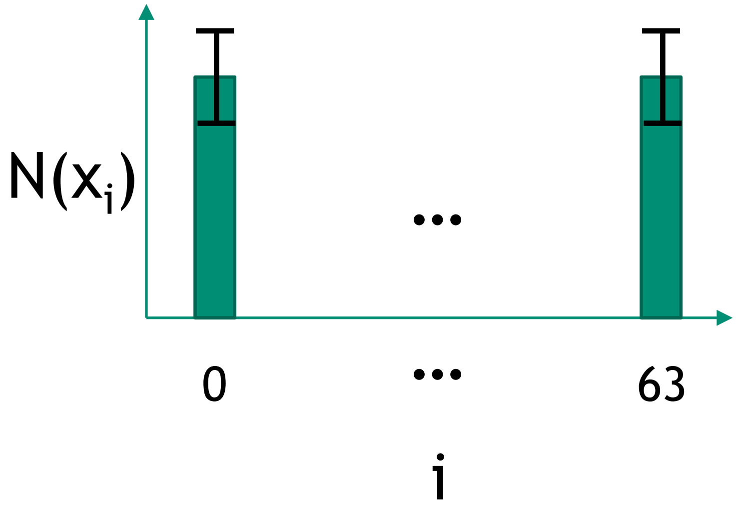


3. Fit distribution created using 10^N samples
4. Determine if $X^2 \sim 1$ for larger and larger N



Significant difference between the distribution generated by the MTJ bitstream and a uniform distribution above 10^6 samples

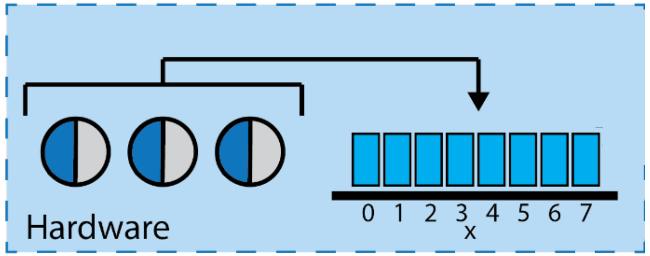
When does $X^2>1$?



When is your statistical error bar...

$$\sqrt{\frac{N}{2^B}}$$

Number of samples in a bin



p_i for MTJ is 0.5025 = 0.5 + δ

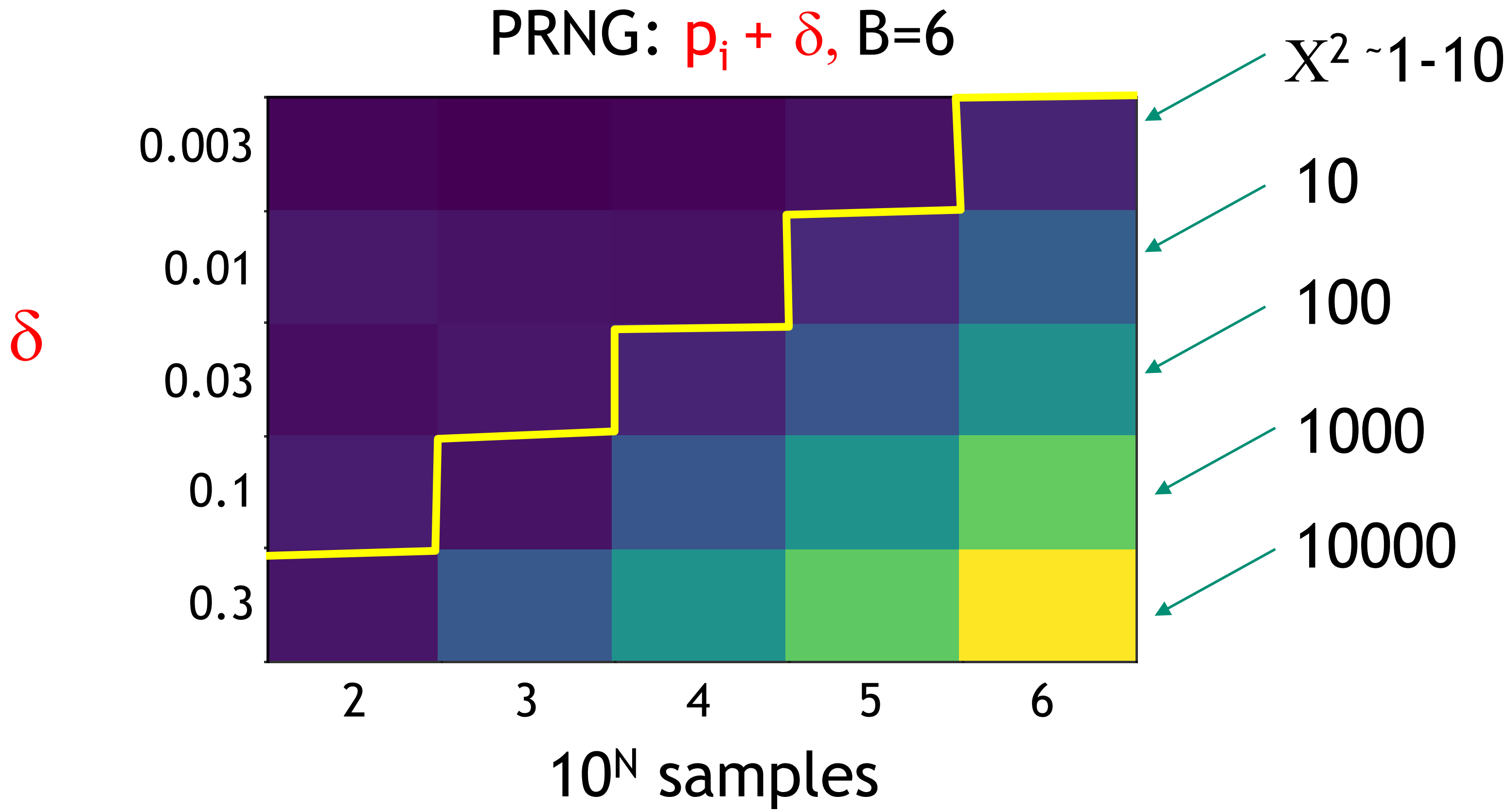
... as big as the source of an error?

$$(\delta B)\frac{N}{2^B}$$

Probability of an erroneous sample

Number of samples in a bin

General expression N samples of B bits with error δ



Point at which X^2 will indicate sample distribution is different from uniform distribution

$$N\delta^2 = \frac{2^B}{B^2}$$

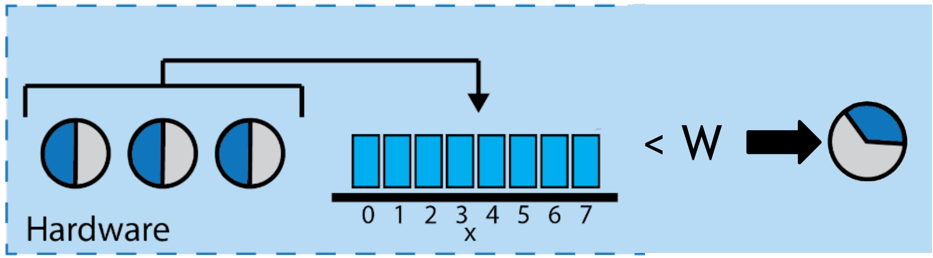
Monte Carlo example ($N = 10^{12}$, $B = 24$) requires $\delta < 0.0002$
That's untenable!!!

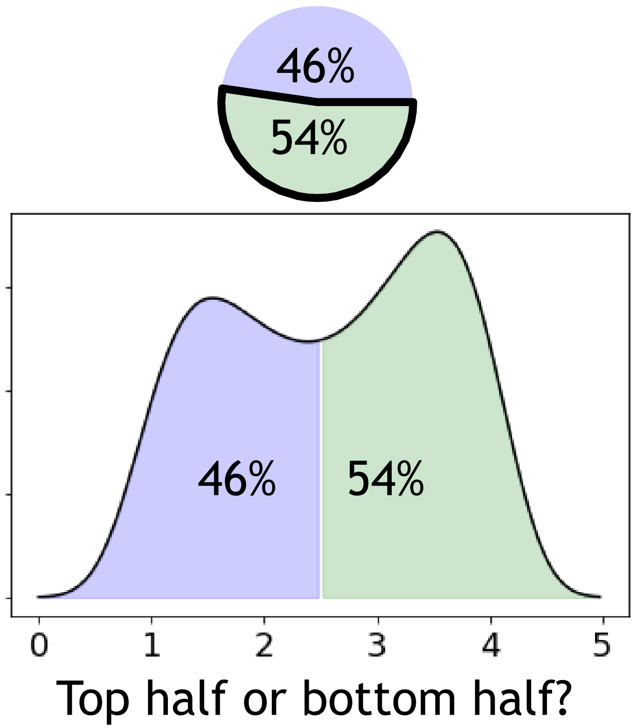
Generate accurate weighted coinflips for non-uniform distributions

1) Use multiple physical coinflips to produce a higher accuracy logical coinflip

- Single coin: $p_i = 0.5 + \delta$
- XOR two coins: $p_i = 0.5 + 2\delta^2$

2) Use multiple fair coinflips to produce a weighted coinflip

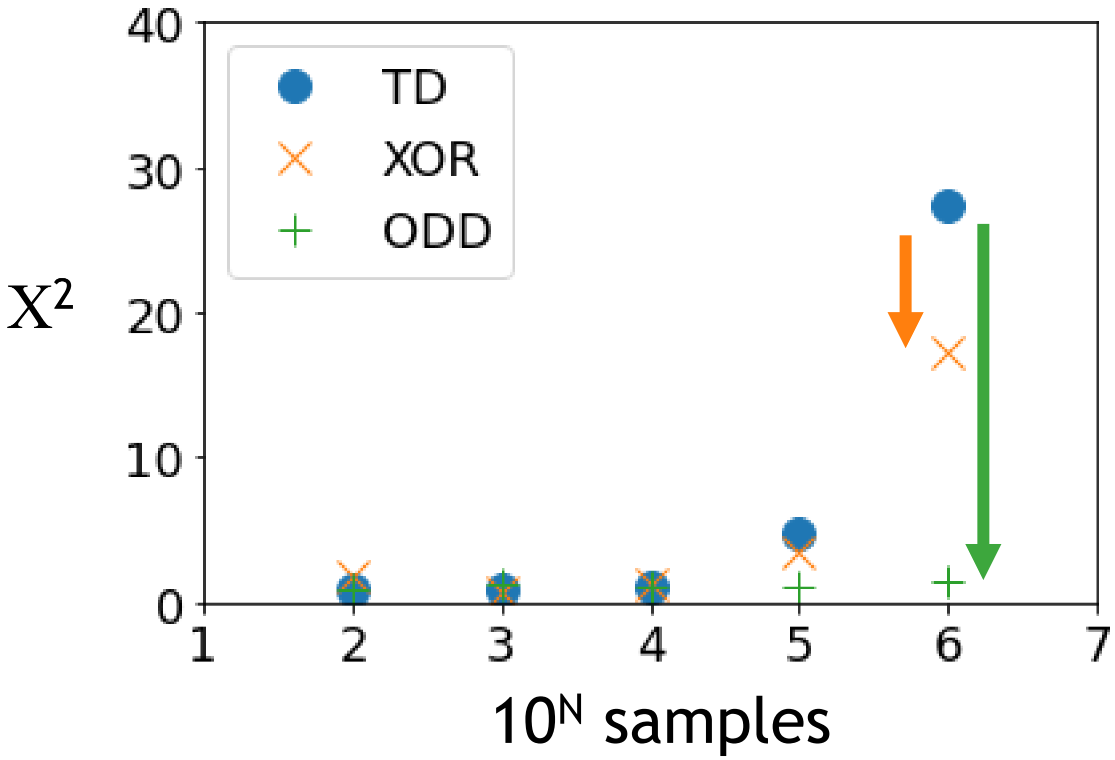




Conclusion

Sandia National Laboratories

Using many fair physical coins to generate a non-uniform sample directly works!



We are looking for postdocs:
- STM-based fabrication
- Cryogenic measurement
- Superconducting devices

Email: smisra@sandia.gov

Monte Carlo calculation of particle collisions: $N=10^{12}$ and $B=24$

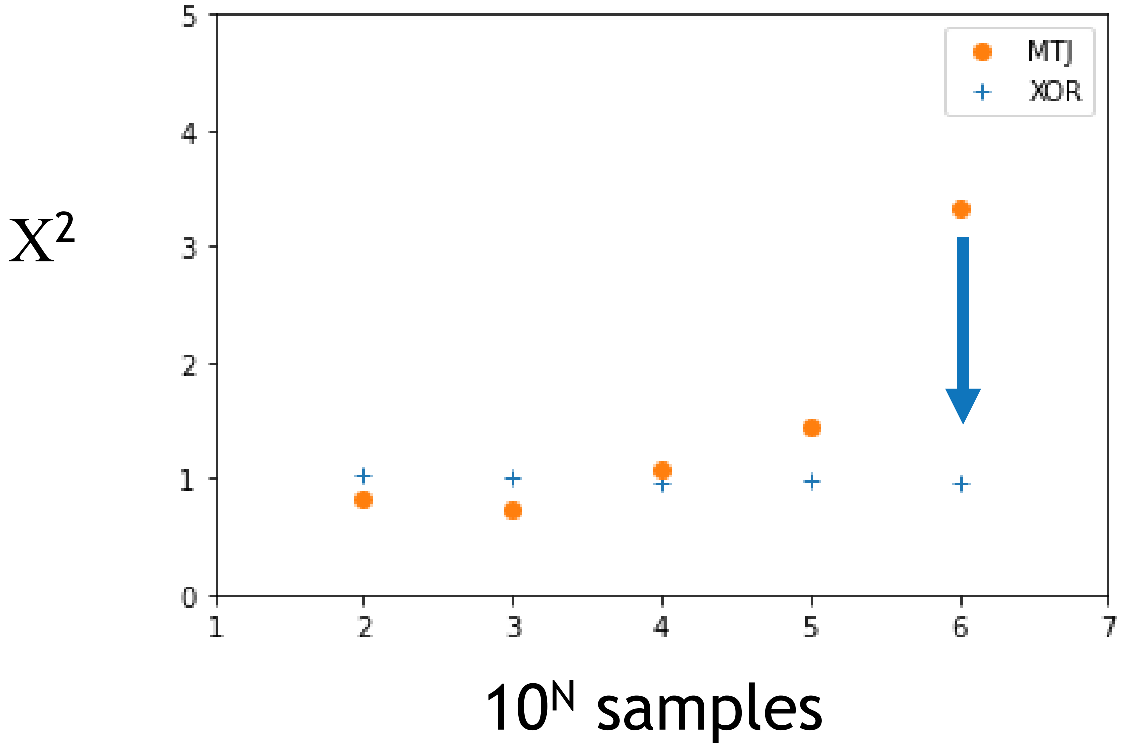
Solution: 2000 physical coins directly generates a highly accurate non-uniform sample

- XOR 2 coins with 1% error
- ODD of 3 coins with 1% autocorrelation
- 14 fair physical coins to 1 weighted logical coin for precision
- 3 orders of magnitude of potential energy efficiency remaining

Use XOR of two bits to improve p_i

Coin	Probability
0	$0.5 + \delta$
1	$0.5 - \delta$

Coins	Probability	XOR	Probability
0 1	$0.25 - \delta^2$	1	$0.5 - 2\delta^2$
1 0	$0.25 - \delta^2$		
0 0	$0.25 + 2\delta + \delta^2$	0	$0.5 + 2\delta^2$
1 1	$0.25 - 2\delta + \delta^2$		



XOR reduces error from δ to δ^2 - significantly relaxes demands on device

Previous physical bit was 0

	first	second	third
001	q	Q	1-q
010	q	1-q	1-q
111	1-q	Q	Q
100	1-q	1-q	Q
101	1-q	1-q	1-q
110	1-q	Q	1-q
000	q	Q	Q
011	q	1-q	Q

Previous physical bit was 1

	first	second	Third
001	1-q	Q	1-q
010	1-q	1-q	1-q
111	Q	Q	Q
100	Q	1-q	Q
101	Q	1-q	1-q
110	Q	Q	1-q
000	1-q	Q	Q
011	1-q	1-q	Q

Rows	P(ODD sum)	For $q=0.5+\delta$, error is quadratic order.
Bit 0: 001, 010, 111, 100	$2q-2q^2$	$0.5-2\delta^2$
Bit 0: 101, 110, 000, 011	$1-2q+2q^2$	$0.5+2\delta^2$
Bit 1: 001, 010, 111, 100	$1-2q+2q^2$	$0.5+2\delta^2$
Bit 1: 101, 110, 000, 011	$2q-2q^2$	$0.5-2\delta^2$