SAND2024-11637C

How to use stochastic devices in probabilistic calculations

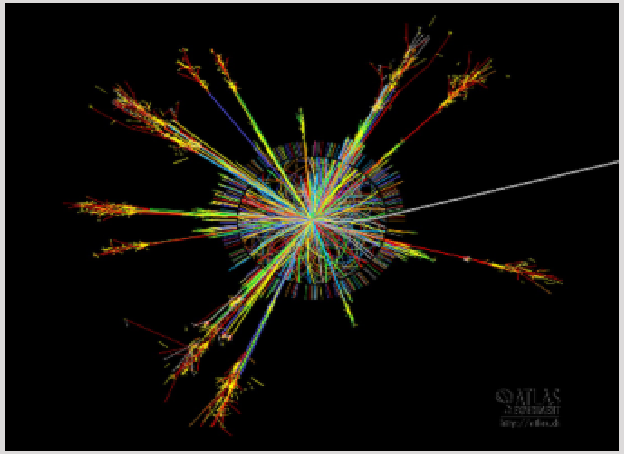
Shashank Misra[1], **Christopher R. Allemang**[1], Christopher D. Arose[1], **Brady G. Taylor**[2], Andre Dubovskiy[3], Ahmed Sidi El Valli[3], Laura Rehm[3], Andrew Haas[3], Andrew D. Kent[3], Leslie C. Bland[4], Suma G. Cardwell[1], **J. Darby Smith**[1], J. Bradley Aimone[1]

[1]Sandia National Laboratories [2]Duke University [3]New York University [4]Temple University

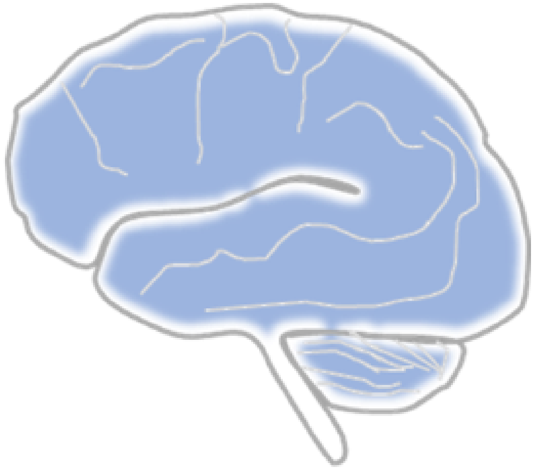
SAND2024-11637C

The authors acknowledge financial support from the DOE Office of Science (ASCR / BES) for our Microelectronics Co-Design project COINLFIPS. SNL is managed and operated by NTESS under DOE NNSA contract DE-NA0003525. This paper describes technical results and analysis. Any subjective views or opinions that might be expressed in the paper do not necessarily represent the views of the U.S. Department of Energy or the United States Government

Probabilistic computing



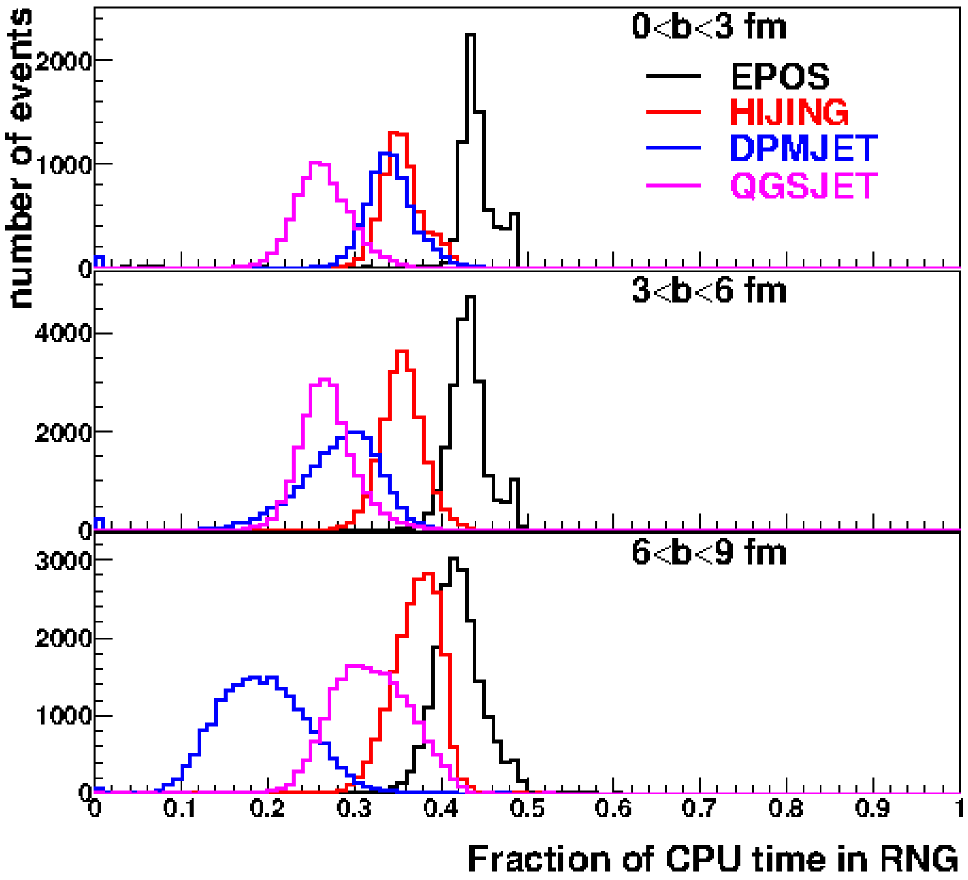
Modeling and Simulation



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

COINFLIPS: Codesign stochastic devices and brain-inspired approaches to scientific problems

Event generator for cosmic rays



Some calculations consume random numbers faster than they can be produced

S. Misra, Adv. Mater. (2022)

Unrealized advantage of switching to stochastic hardware

Sandia National Laboratories

Potentially three orders of magnitude efficiency moving from pseudo random number generator to a true random number generator…

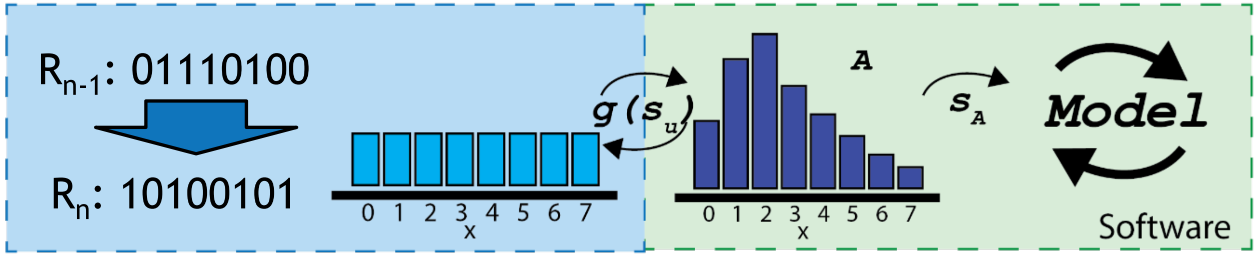
- PRNGs: ~ 1 nJ
- TRNG (MTJ): < 1 pJ

Djupdal ,CARRV (2023)
A. Shukla, IEEE ISQED (2023)

… but unclear how to use TRNGs in practice.

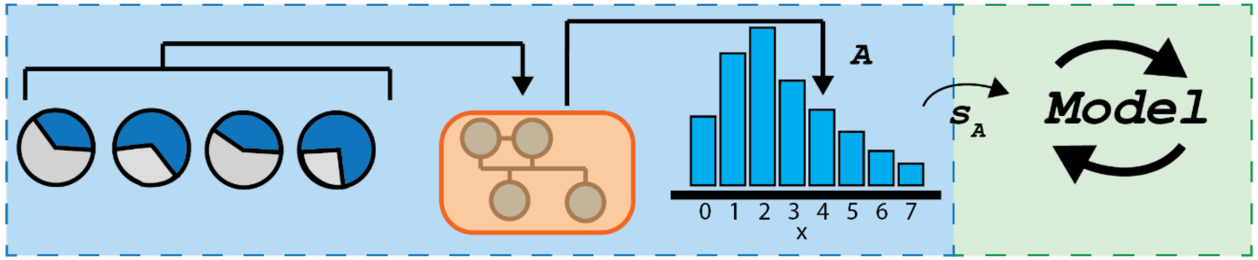
How this is done now:

- CPU generates a uniform pseudo-random number
- Numerical transformation to distribution needed

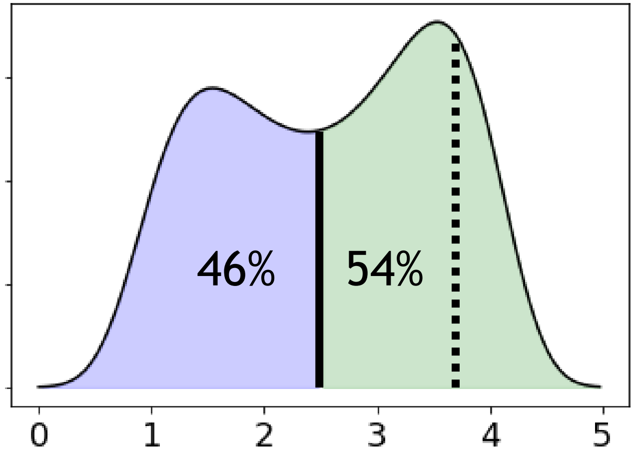


This talk:

- TRNG directly samples distribution

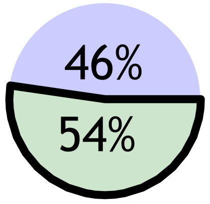


What does this system look like?



Goal: 10^8 32-bit samples of a nonuniform probability distribution function on finite domain.

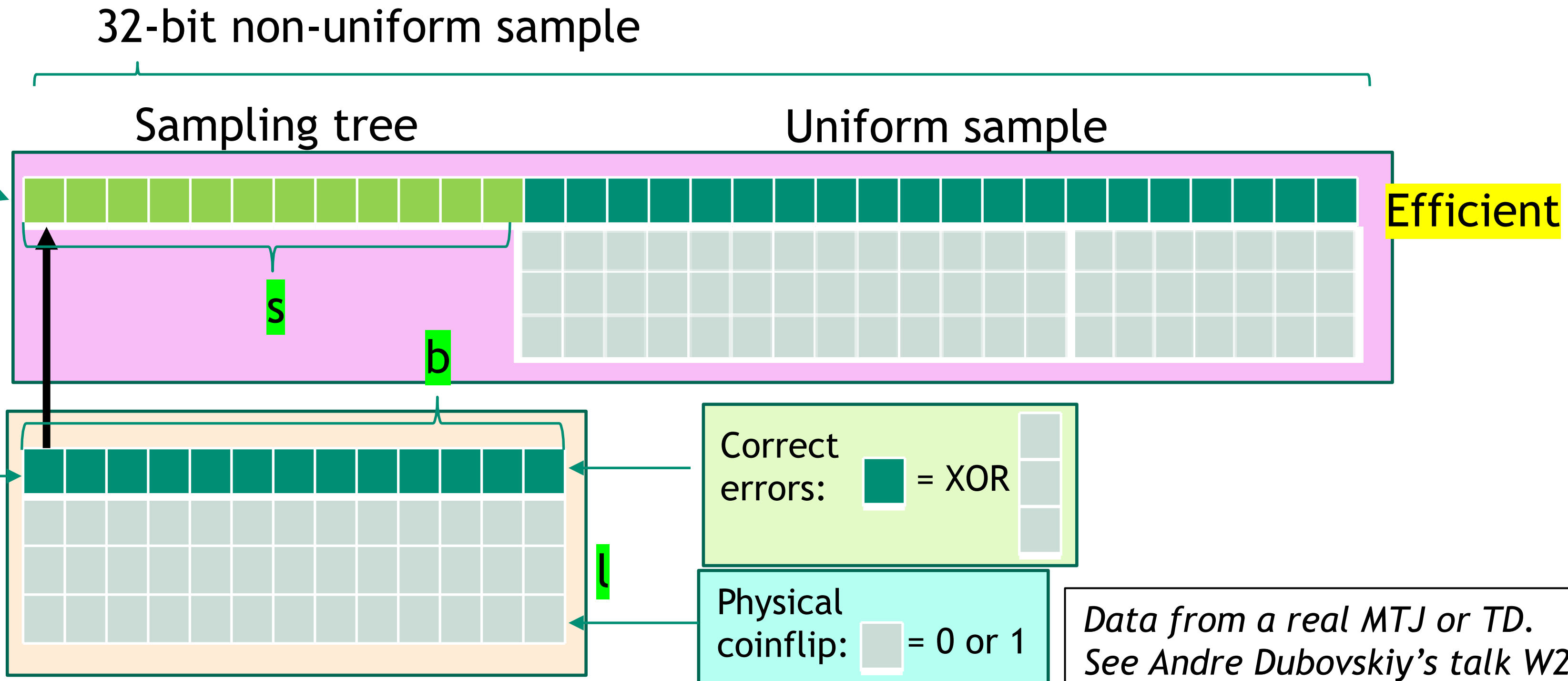
This talk:
- Justify accurate, memory efficient
- Justify choices of l, b, s
- Compare to rejection sampling w/ PRNG



Vs.

Uniform random number

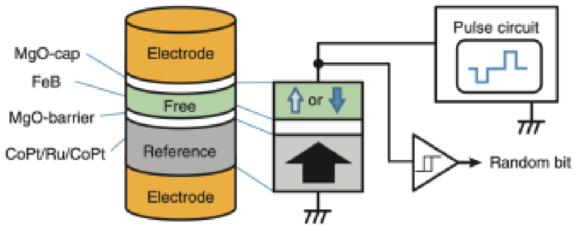
Accurate



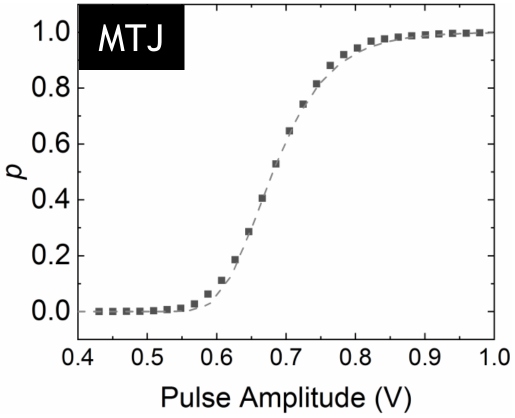
Data from a real MTJ or TD. See Andre Dubovskiy's talk W21.

How to evaluate device bitstreams

Use magnetic tunnel junction (MTJ) or tunnel diode (TD) to generate random bitstream



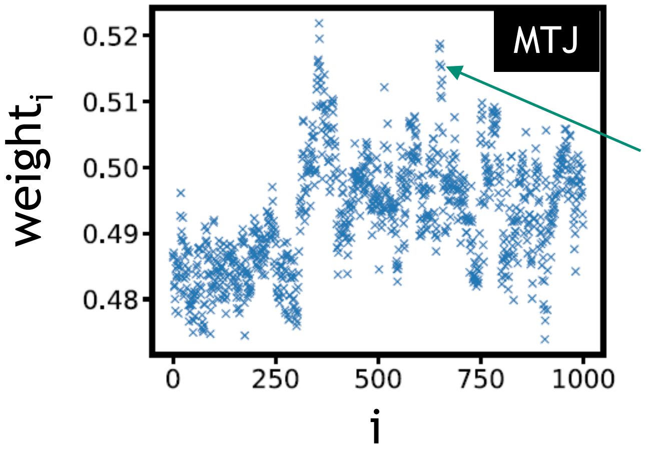
Fukushima, Applied Physics Express (2014)



L. Rehm, Phys. Rev. Applied (2023)

How fair (weight close to 0.5) can we tune MTJ and TD bitstream devices?

<u>Weight drift</u>

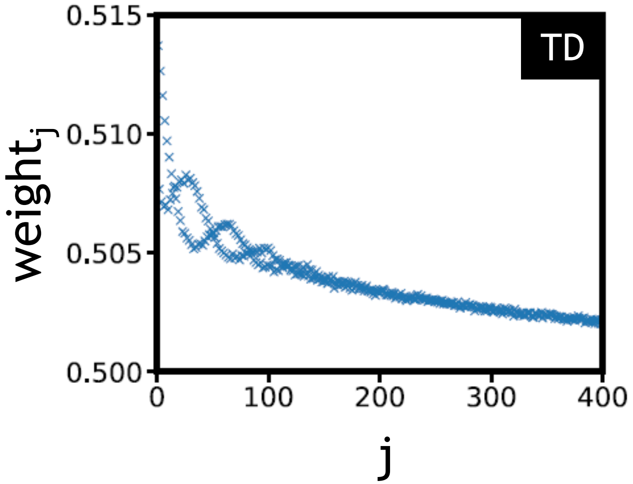


Each point is the average of 10^8 coinflips

Infidelity

$\delta_i = w_i - 0.5$

<u>Dependence</u>



If the 0^{th} coinflip is a 1, what is the weight of the j^{th} flip

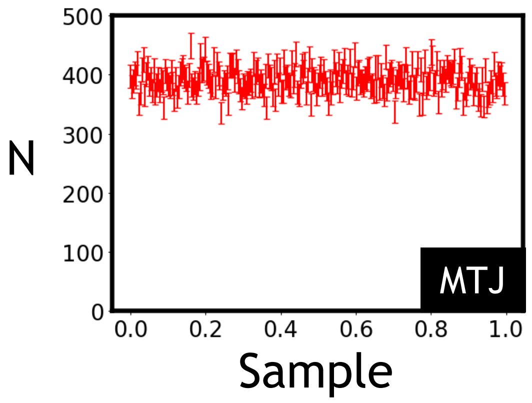
Dependence

$\varepsilon = w_1 - 0.5$

δ and ε impact sampling a uniform distribution

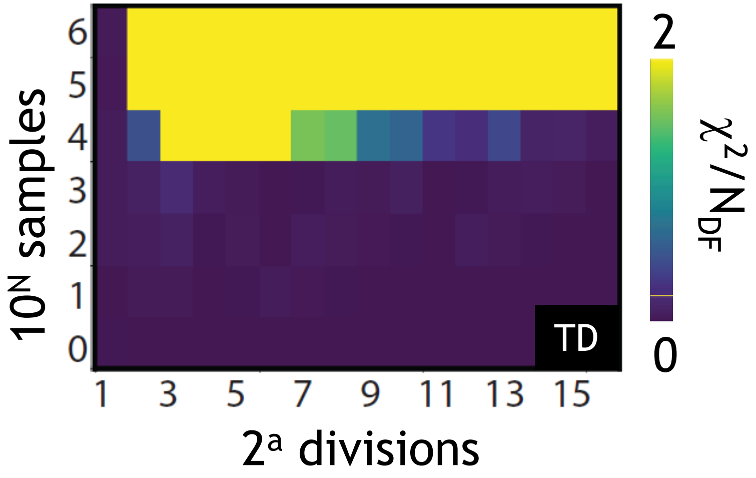
0/1 Uniform random sample

<u>Discretized uniform random sample</u>



Test uniform sample with χ^2 fit.

<u>How much does ε matter?</u>



Heuristic:
$N \max(\delta,\varepsilon)^2 \sim 1$

N uniform samples
δ infidelity
ε dependence

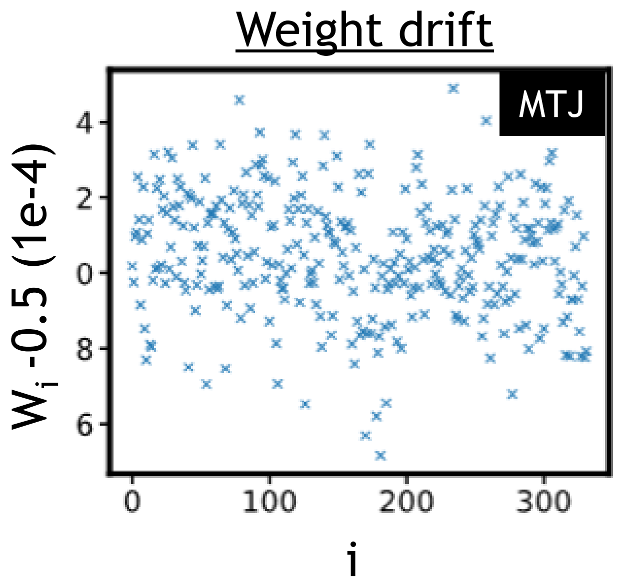
Sample distribution is significantly different from uniform for just 10000 samples when infidelity or dependence are 1%

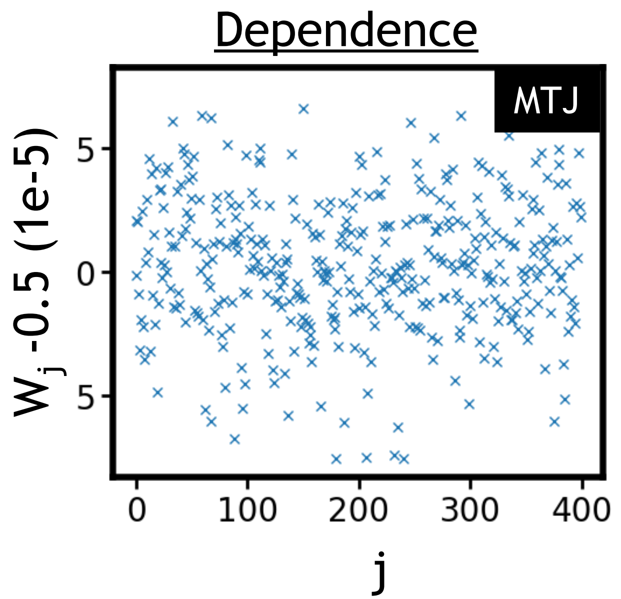
Can we improve accuracy?

Sandia National Laboratories

infidelity δ
dependence ε

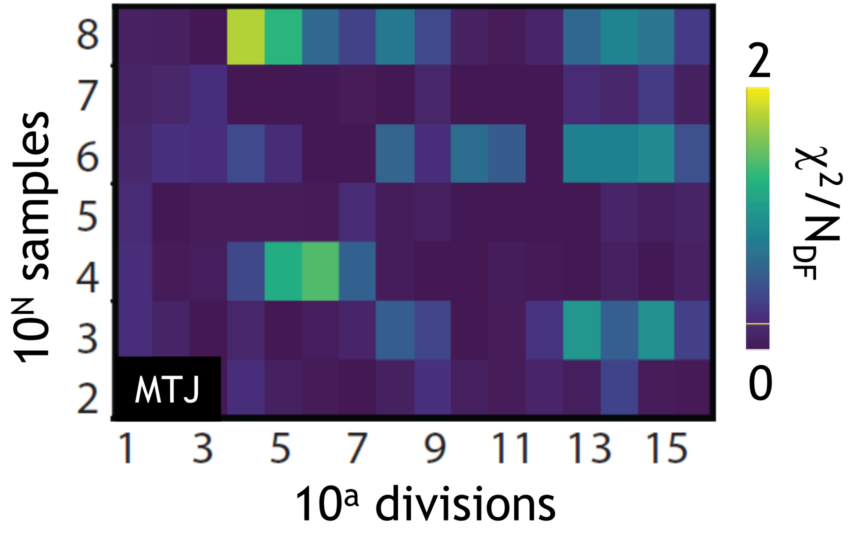
First	Second	Raw	XOR2	XOR3
1	1	$\frac{1}{4}-\frac{\varepsilon}{2}+\frac{\delta}{2}$	$\frac{1}{4}+\frac{\varepsilon}{2}-\delta^2$	$\frac{1}{4}-2\epsilon\delta$
0	1	$\frac{1}{4}+\frac{\varepsilon}{2}+\frac{\delta}{2}$	$\frac{1}{4}+\frac{\varepsilon}{2}-\delta^2$	$\frac{1}{4}-2\epsilon\delta$
0	0	$\frac{1}{4}-\frac{\varepsilon}{2}-\frac{\delta}{2}$	$\frac{1}{4}-\frac{\varepsilon}{2}+\delta^2$	$\frac{1}{4}+2\epsilon\delta$
1	0	$\frac{1}{4}+\frac{\varepsilon}{2}-\frac{\delta}{2}$	$\frac{1}{4}-\frac{\varepsilon}{2}+\delta^2$	$\frac{1}{4}+2\epsilon\delta$



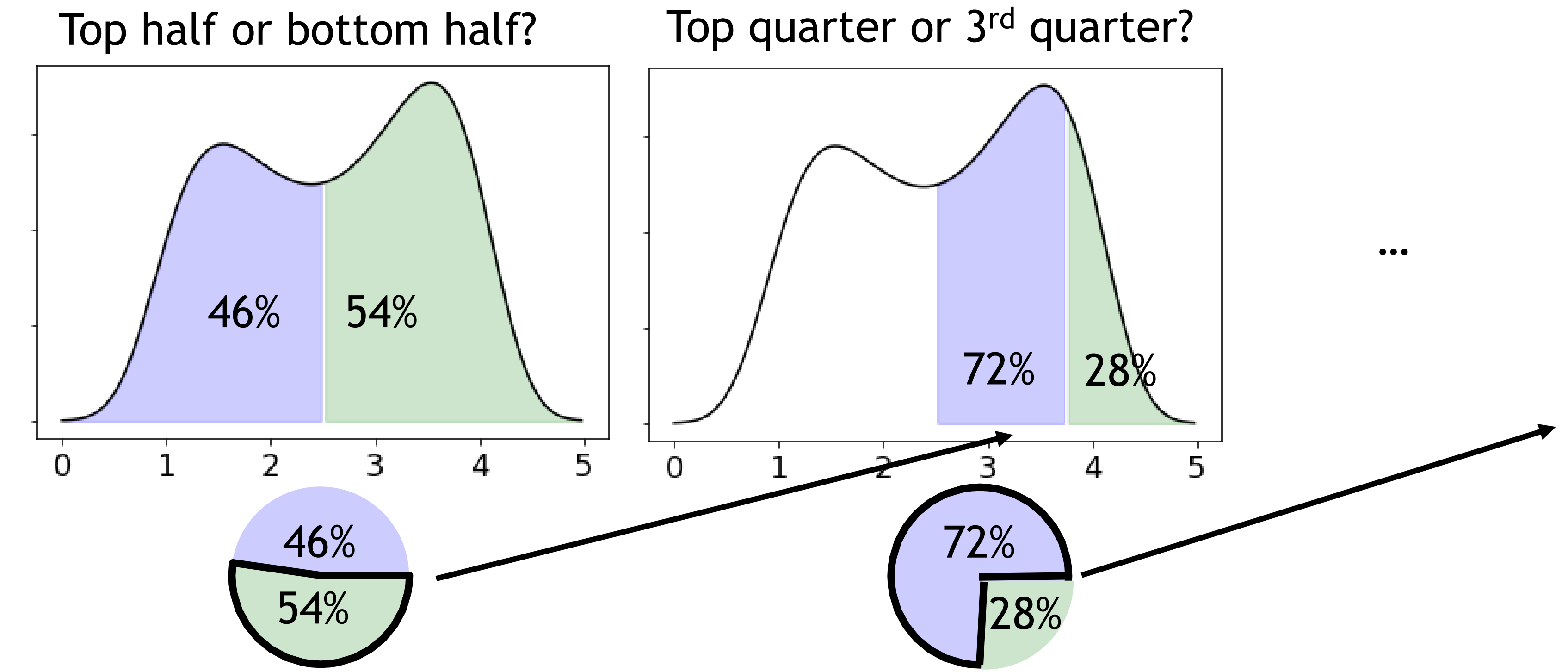


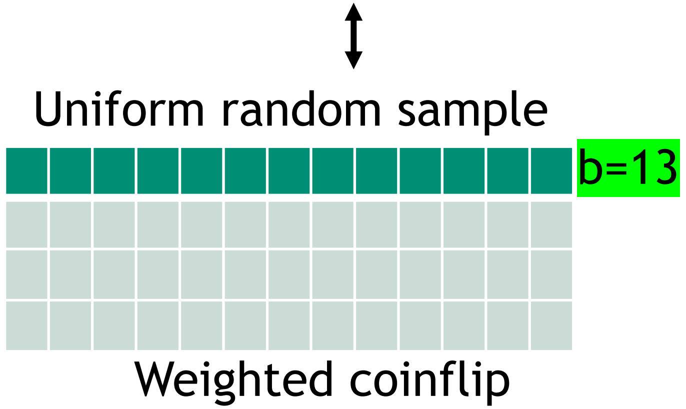
Logical exclusive or of 3 consecutive bits produces low error rates, allows for more samples

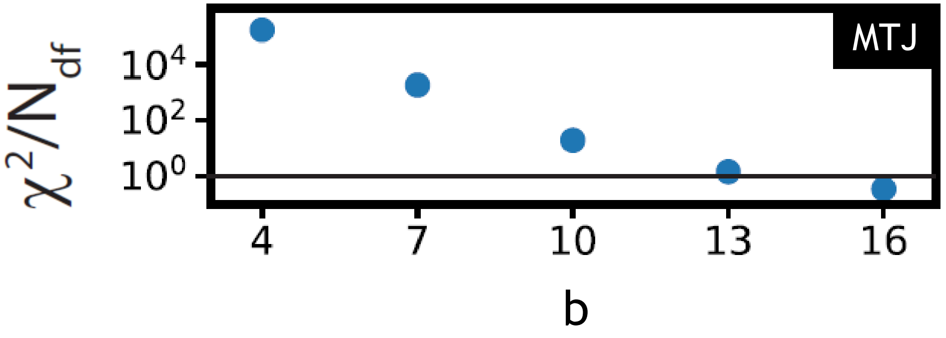
= 0 or 1 = XOR l=3



How to sample a non-uniform distribution







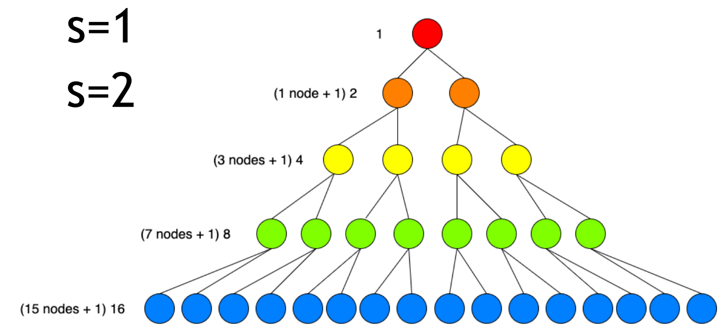
Problem: Say we want 10^8 samples - requires $\delta, \varepsilon \sim 10^{-4}$

Impractical for a weighted coinflip device.

Solution: use fair coins to draw a uniform random sample with 13 bits of precision

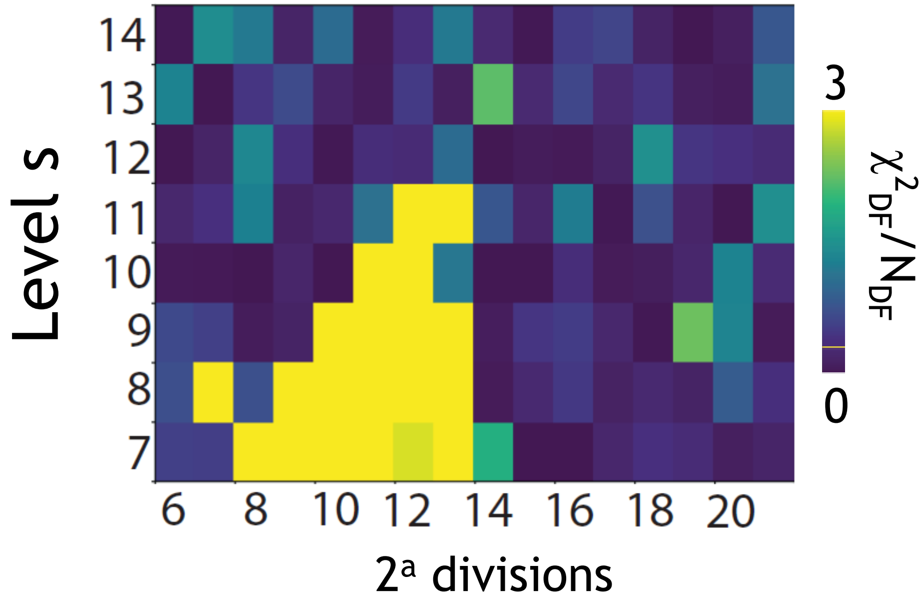
Heuristic:
$N \max(1/2^b)^2 \sim 1$

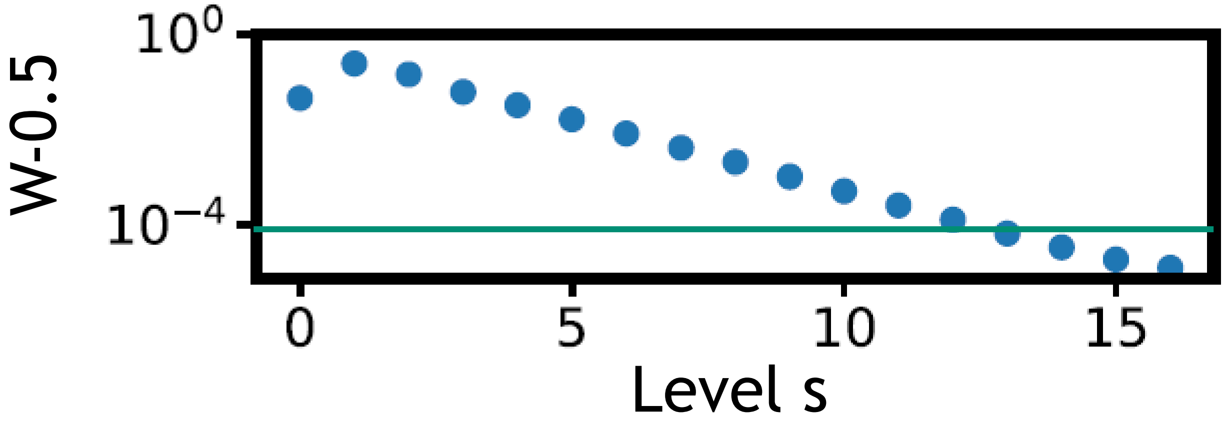
Cutoff sampling tree for efficiency



Only need sampling tree for top 12 bits – remaining bits can be uniform random sample

Tree weights should fit in 64 kB





How well does this actually work?

Uniform distribution

PRNG

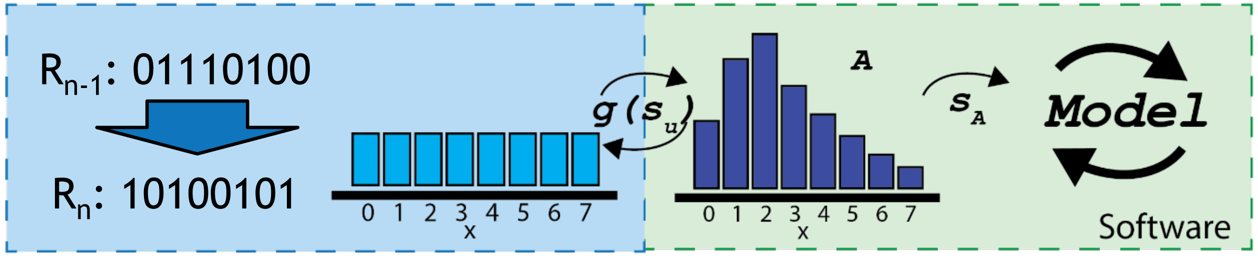
10 simple operations

TRNG

96 coinflips

2 simple operations

5x advantage



Non-uniform distribution

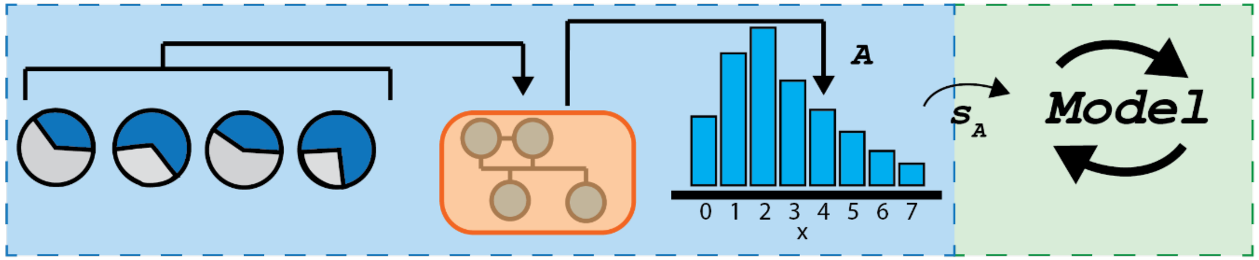
PRNG (rejection)

10 operations/ PRNG
100 operations acceptance
1 conditional
2x executed on average

TRNG (tree)

526 coinflips
26 XOR
12 conditionals
12 cache access

4x advantage

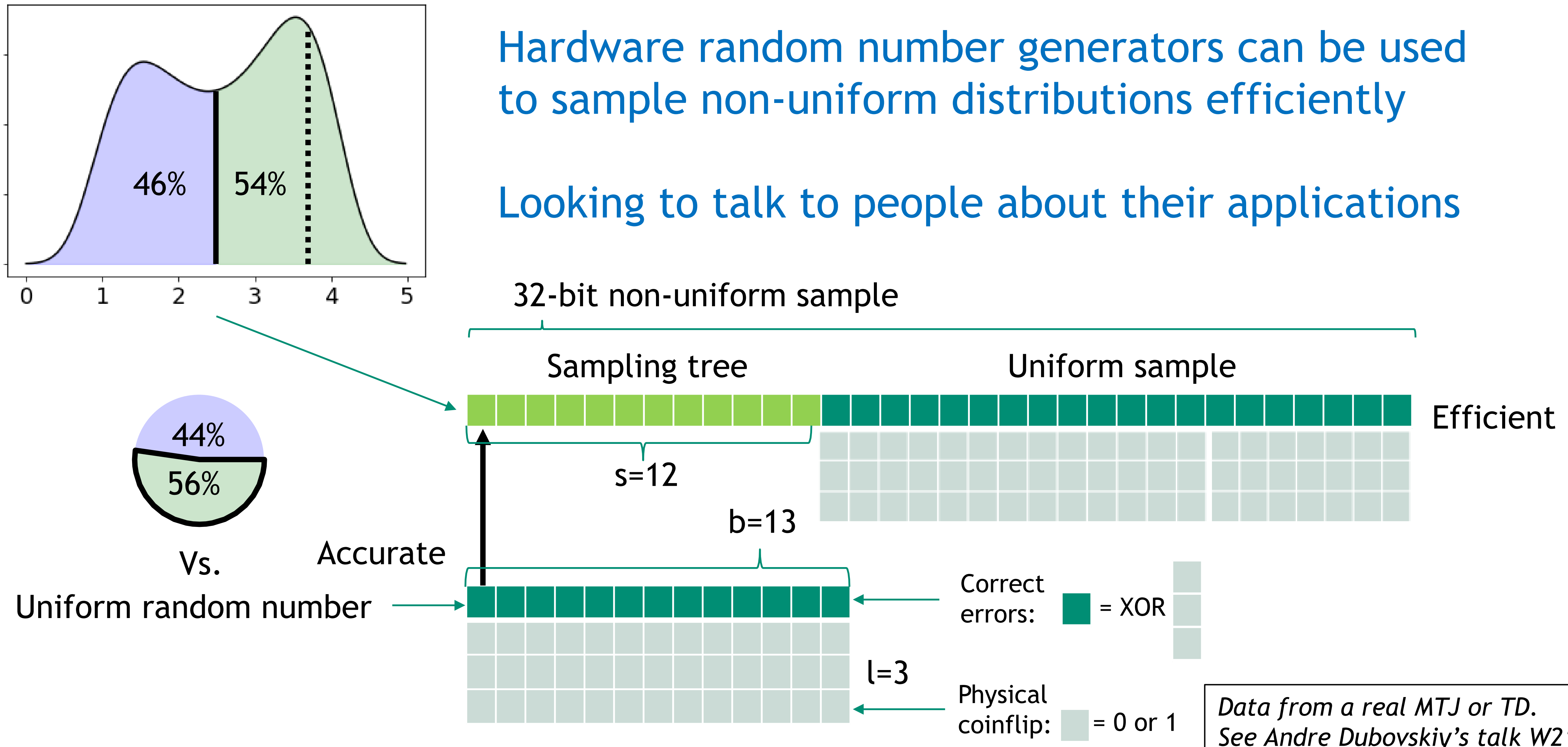


To have real application impact, we need to engage on how to move more of model into sampling, and requirements for accuracy

Conclusion

Hardware random number generators can be used to sample non-uniform distributions efficiently

Looking to talk to people about their applications



Data from a real MTJ or TD.
See Andre Dubovskiy's talk W21.