Probabilistic computing using noisy devices

SAND2024-11633C

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

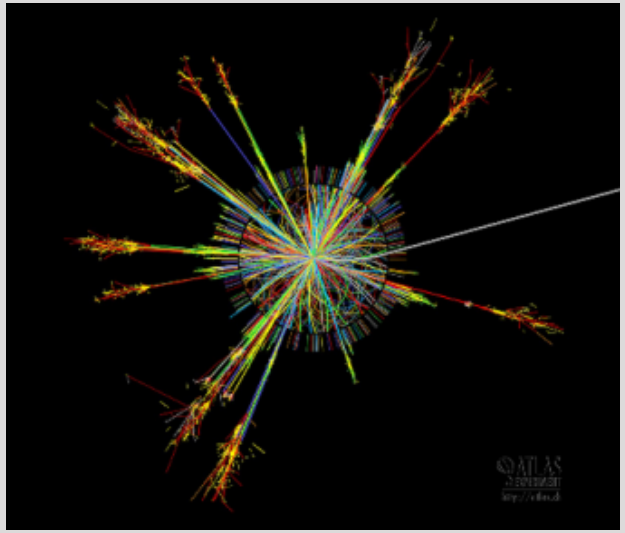
Shashank Misra[1], **Christopher R. Allemang**[1], Christopher D. Arose[1], **Brady G. Taylor**[2], Andre Dubovskiy[3], Ahmed Sidi El Valli[3], Laura Rehm[3], Andrew Haas[3], Andrew D. Kent[3], Leslie C. Bland[4], Suma G. Cardwell[1], **J. Darby Smith**[1], J. Bradley Aimone[1]

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

Statistical computations and the hardware lottery

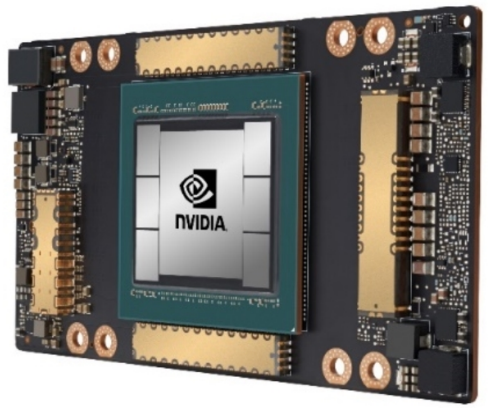
Artificial Intelligence

Bayesian neural networks are appealing yet often computationally intractable

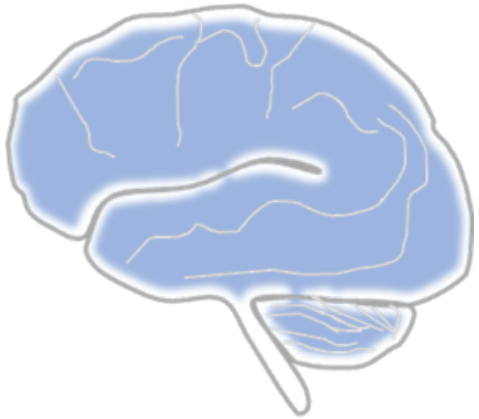


Modeling and Simulation

Many applications are inherently stochastic in their physics and are best modeled using probabilistic methods

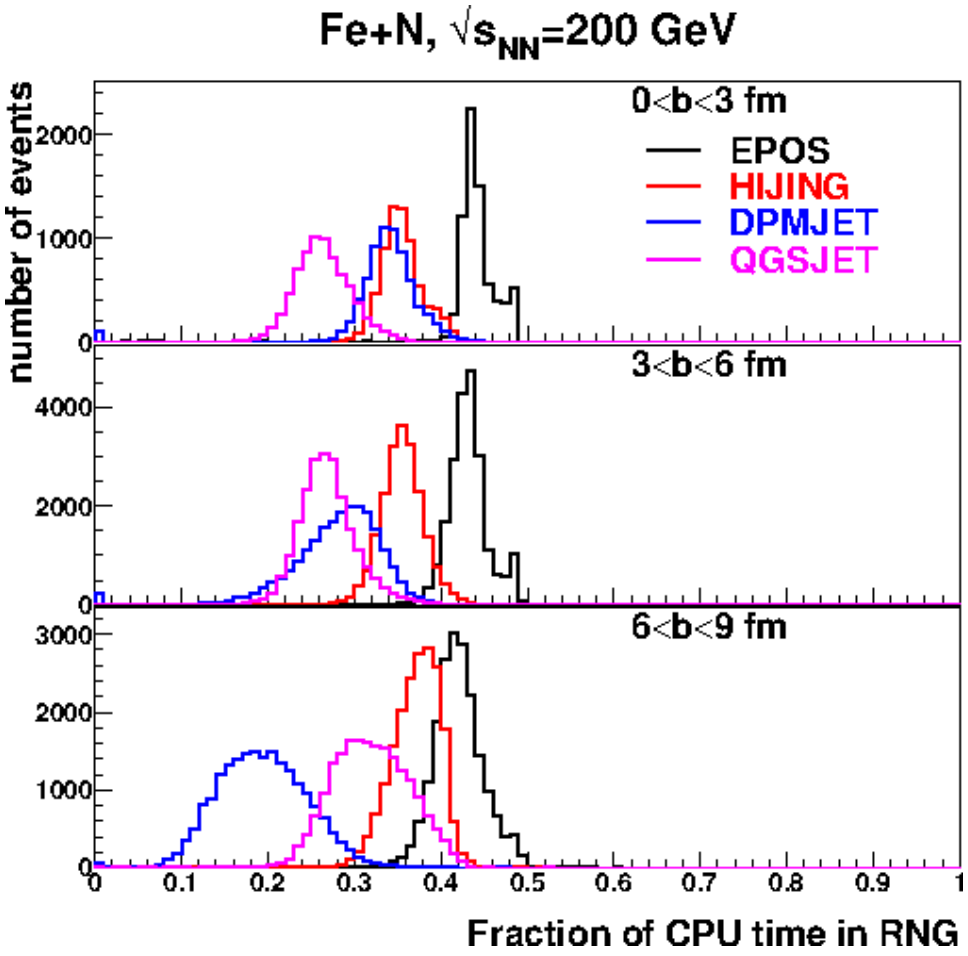


~400 W
~10^{13}-10^{14} FLOPS
Fully deterministic



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

Exemplar problem



Misra, Adv. Mater. (2022)

Pythia – event generator used here to simulate particle showers from high energy cosmic rays

Profile amount of time used by the pseudo-random number generator (PRNG)

Uniform random sample converted to sample other distributions using elementary functions

Some calculations consume random numbers faster than they can be produced

How can we use noisy devices?

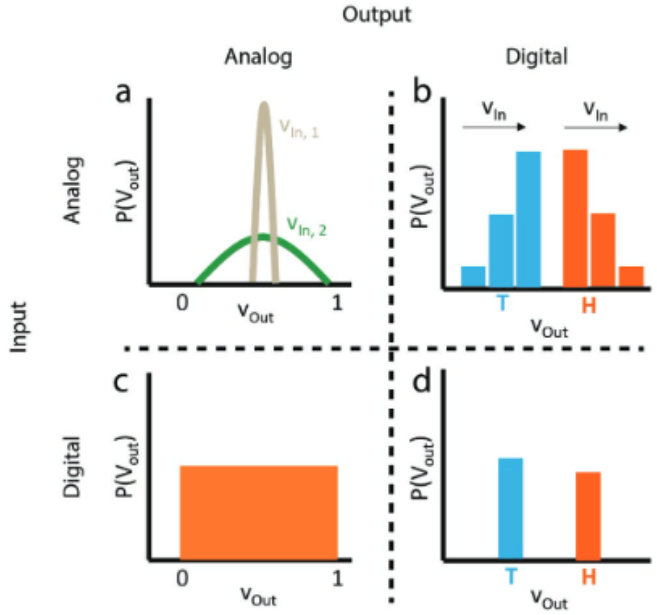
Potentially three orders of magnitude efficiency moving from pseudo random number generator (PRNG - software) to a true random number generator (TRNG - hardware)…

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

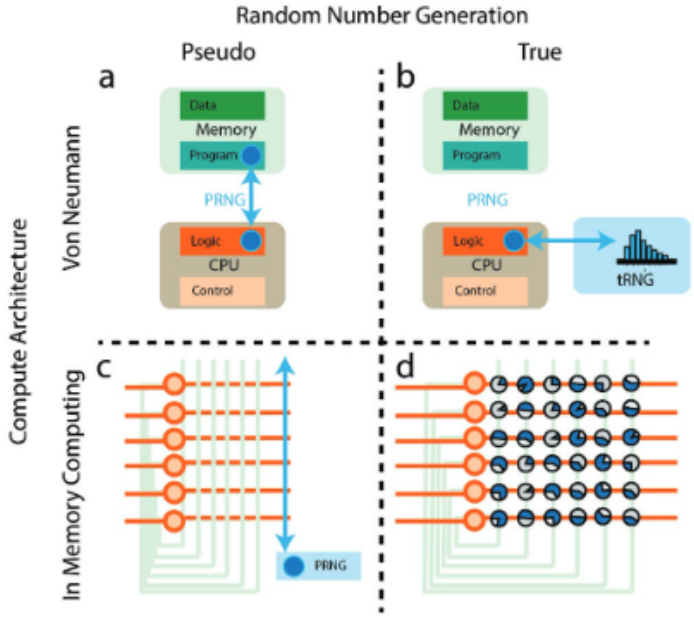
Djupdal ,CARRV (2023)
A. Shukla, IEEE ISQED (2023)

… but unclear how to use TRNGs in practice.

Fair vs. weighted?



Accelerator vs. integrated?



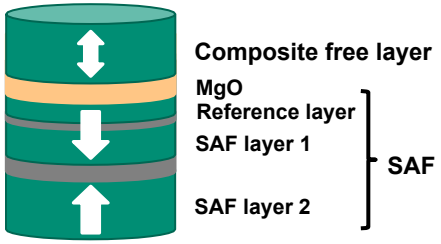
Misra, Adv. Mater. (2023)
Misra & Aimone, IEEE EMD (2023)

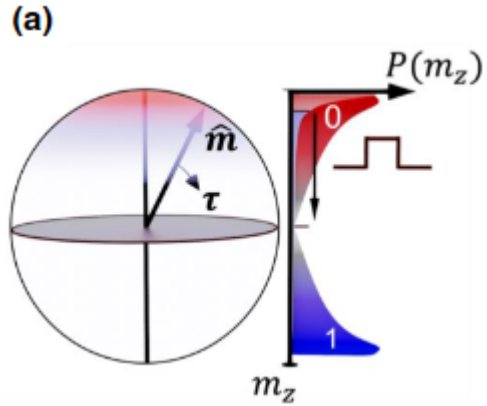
Outline

1. Hardware bitstreams
2. Sampling distributions
3. System implications

Coinflip device - a random bitstream generator

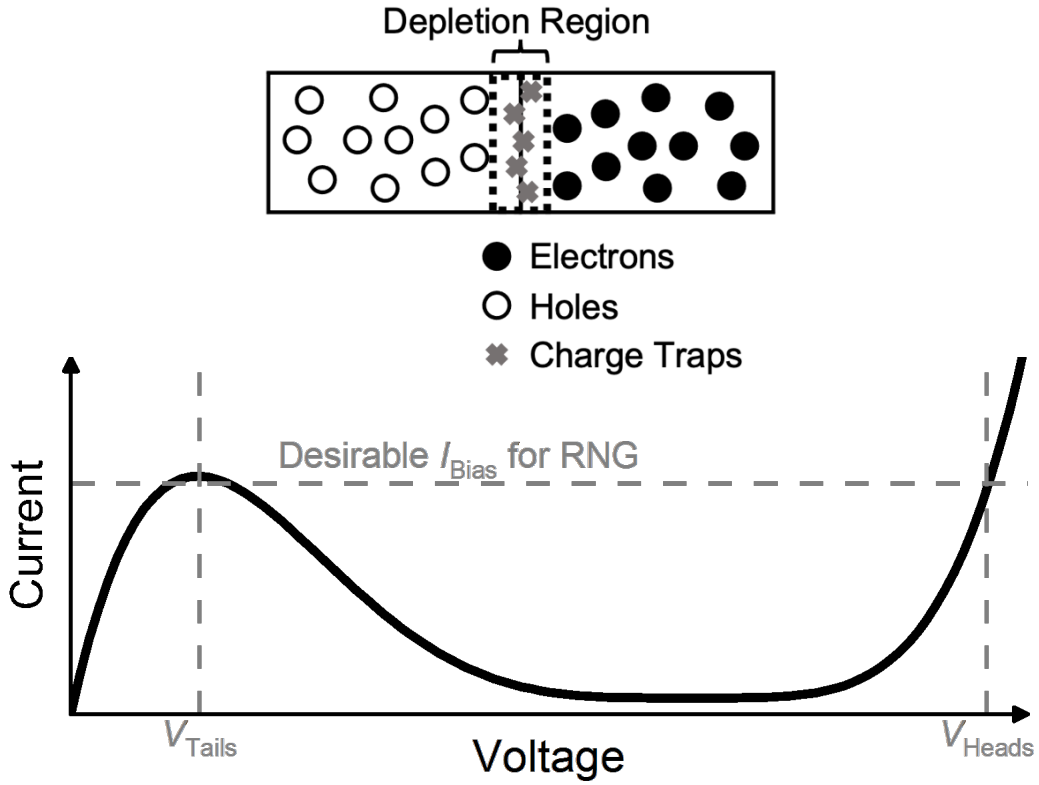
Stochastic magnetic actuated random transducer (SMART)





Rehm, Phys. Rev. Appl. (2023)

Tunnel diode



Why?

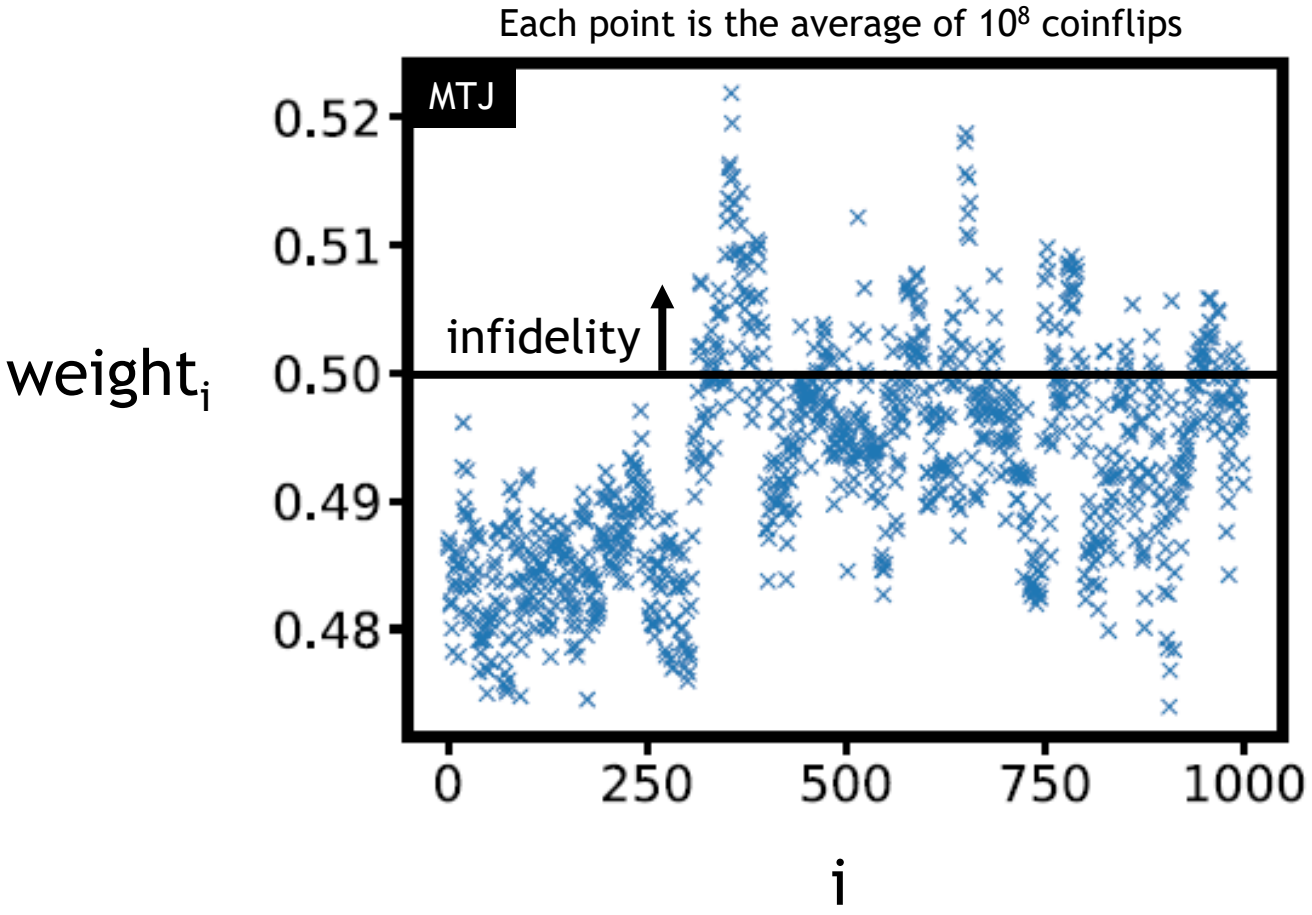
- Output states are well-resolved
- Input pulse controls probability of ‘high’ output
- Easy to understand simple case with uniform sampling using fair coin

How do we evaluate the bitstream (1)?

NIST tests are useful litmus test, but provide little insight into sampling

How well can we set the weight?

Define infidelity $\delta_i = w_i - 0.5$



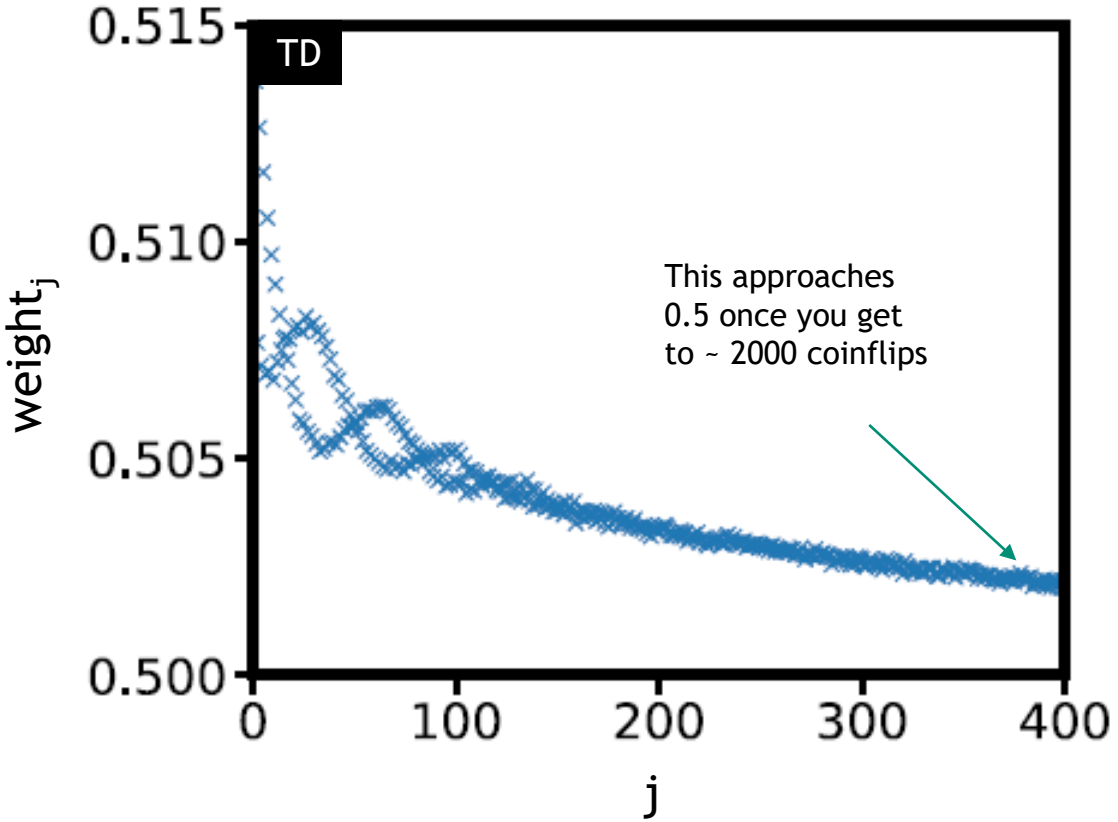
Infidelity drifts with external factors – e.g. temperature

How do we evaluate the bitstream (2)?

Are the coinflips independent of one another?

Define dependence $\varepsilon = w_1 - 0.5$

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



Dependence can be intrinsic (heating from last pulse), or extrinsic (pickup)

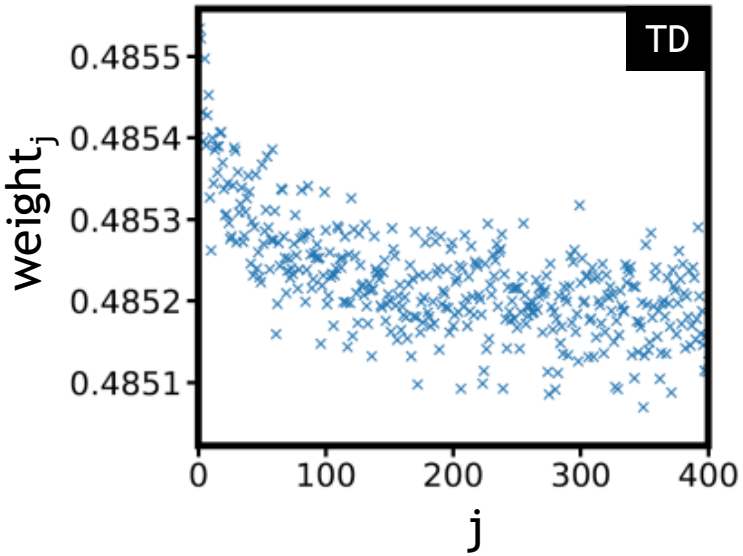
Can we improve accuracy?

infidelity δ
dependence ε

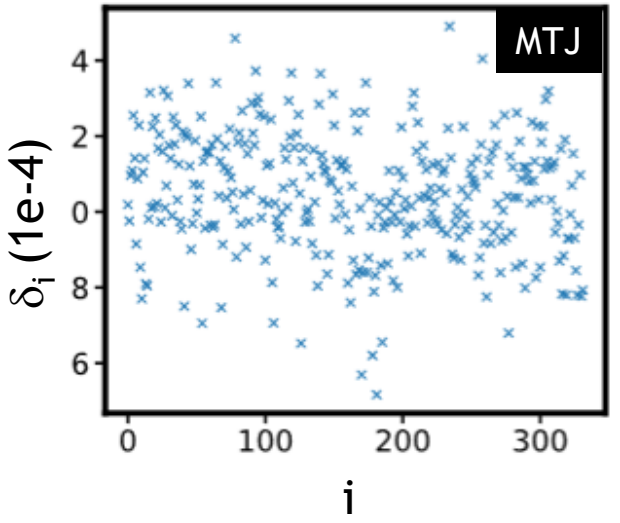
First	Second	Raw	XOR2	XOR3
1	1	$\frac{1}{4}-\frac{\varepsilon}{2}+\frac{\delta}{2}$	$\frac{1}{4}+\frac{\varepsilon}{2}-\delta^2$	$\frac{1}{4}-2\epsilon\delta$
0	1	$\frac{1}{4}+\frac{\varepsilon}{2}+\frac{\delta}{2}$	$\frac{1}{4}+\frac{\varepsilon}{2}-\delta^2$	$\frac{1}{4}-2\epsilon\delta$
0	0	$\frac{1}{4}-\frac{\varepsilon}{2}-\frac{\delta}{2}$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

Dependence (XOR2)



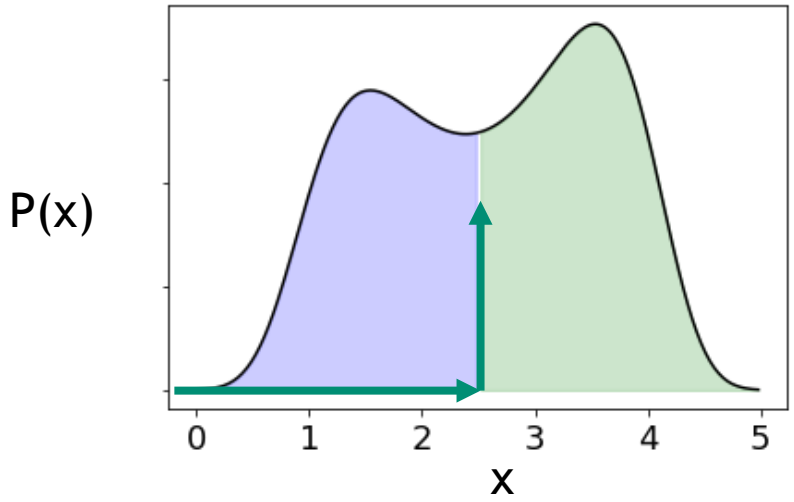
Weight drift (XOR3)



Outline

1. Hardware bitstreams
2. Sampling distributions
3. System implications

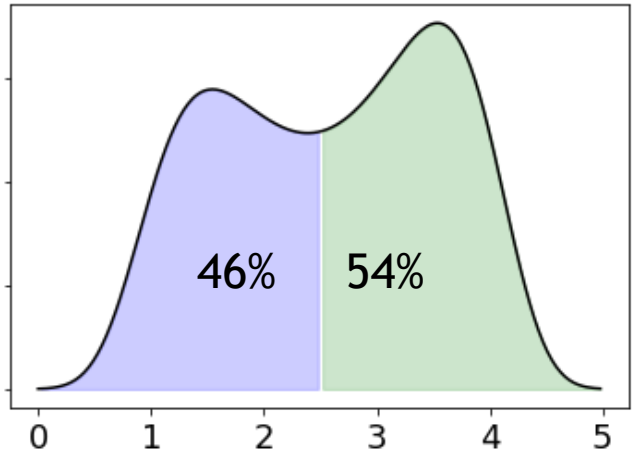
Sampling



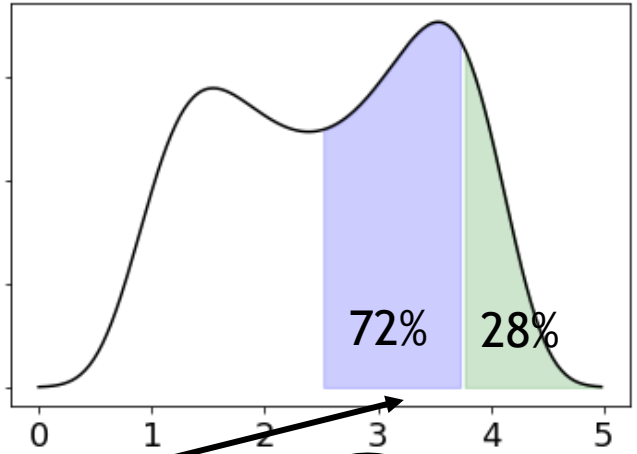
Rejection sampling:

- Random number between 0 and 5 → x
- Random number between 0 and 1 → accept/reject

Top half or bottom half?



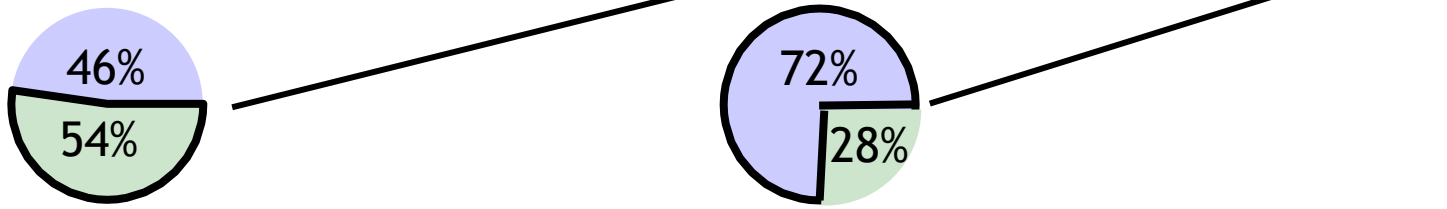
Top quarter or 3rd quarter?



Sampling tree

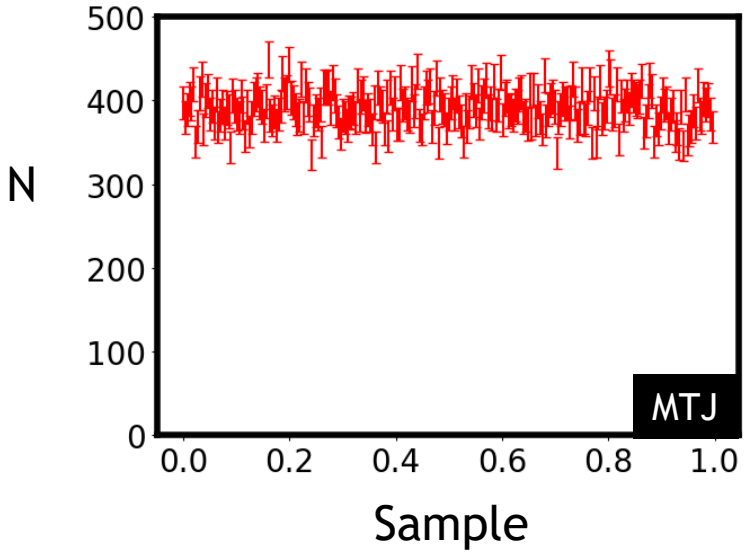
- Weigted decision determines successive bits

Gryzka, Periodica Mathematica Hungarica (2021)



Sampling a uniform distribution

Discretized uniform random sample

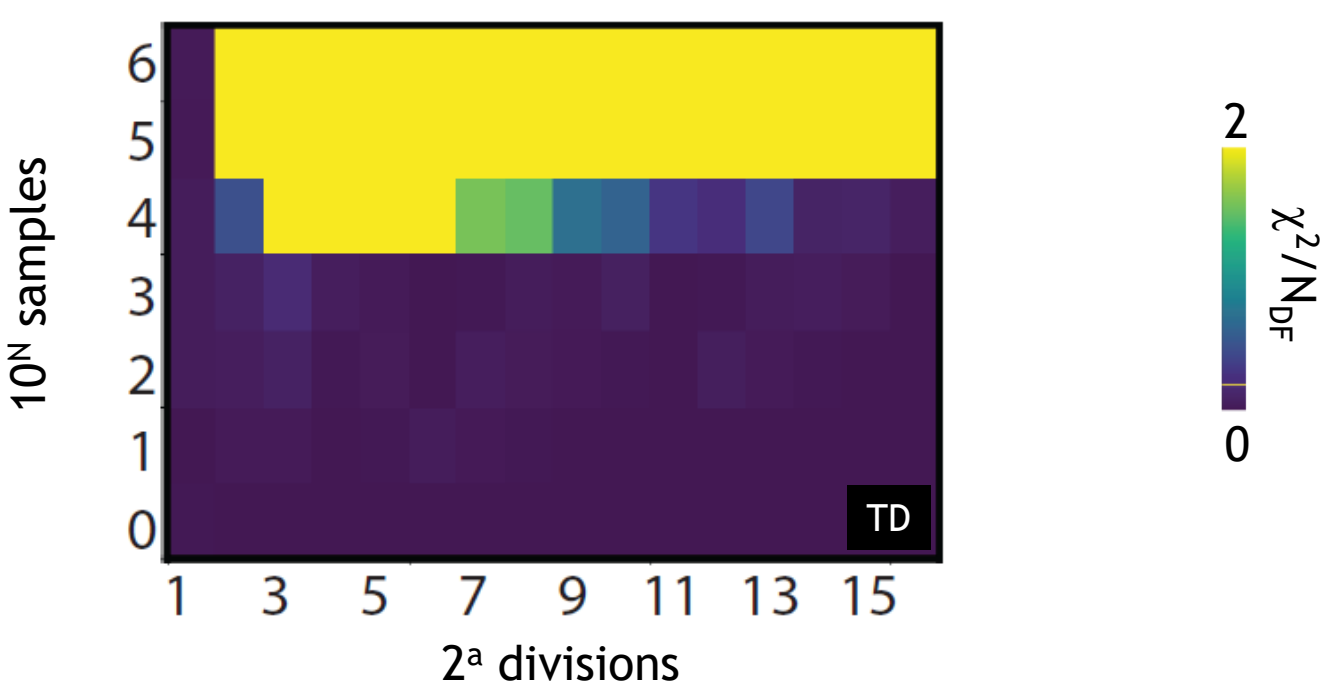


χ^2 statistics tests whether the sample and target distributions are distinguishable

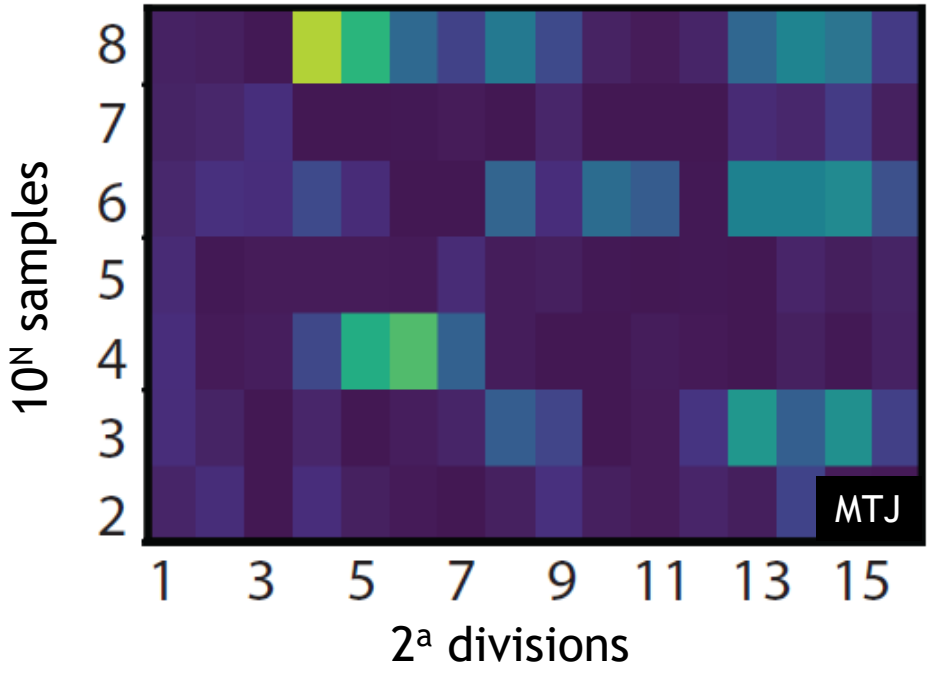
Statistical uncertainty of sqrt(N_i)
How you sum χ^2 is important.

δ and ε impact sampling a uniform distribution

How much does ε matter?

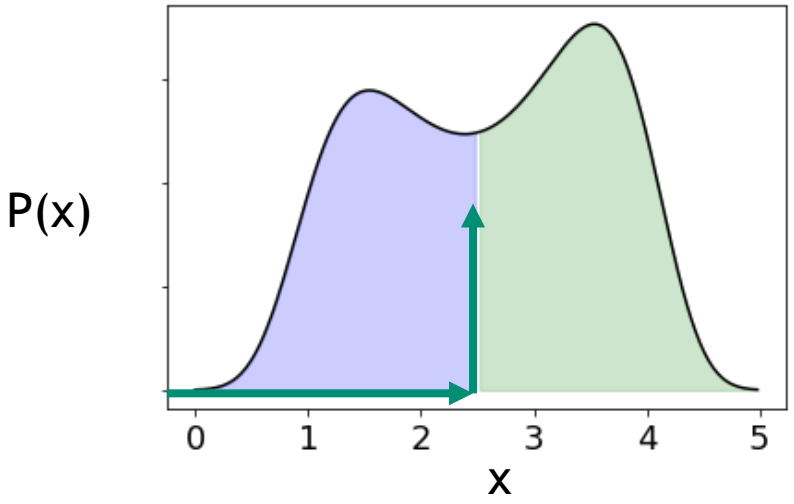


XOR3 improves sampling



Heuristic: N max(δ,ε)² ~ 1

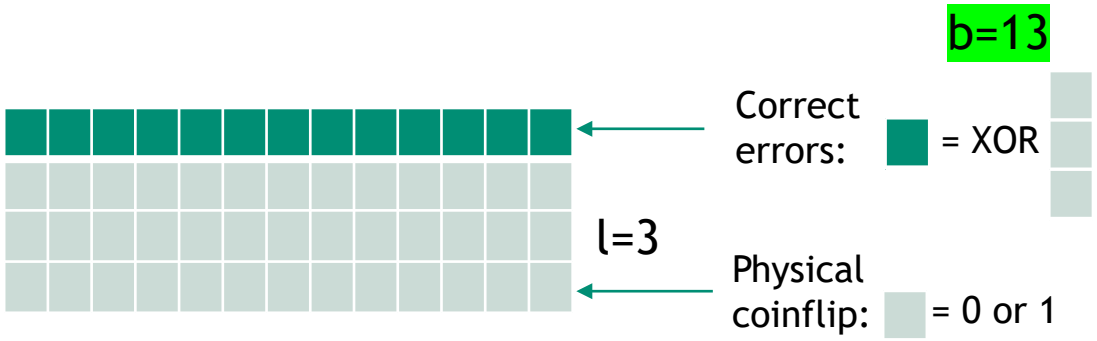
Rejection sampling

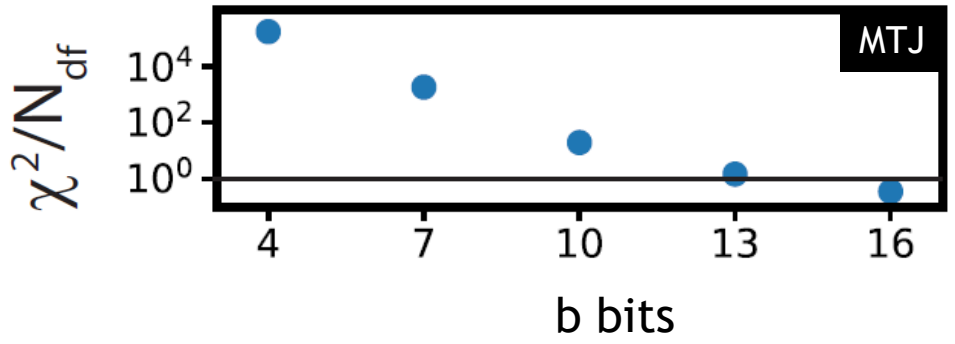


Problem: Say we want 10^8 samples - requires $\delta, \varepsilon \sim 10^{-4}$

<u>Rejection sampling:</u>

- Random number between 0 and 5 → x (32 bits)
- Random number between 0 and 1 → accept/reject (b bits)

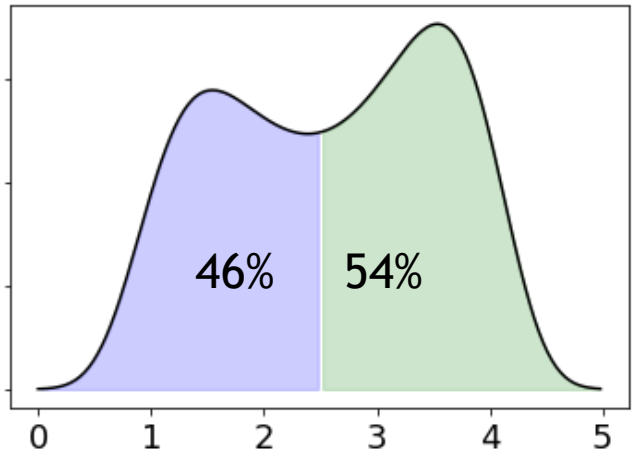




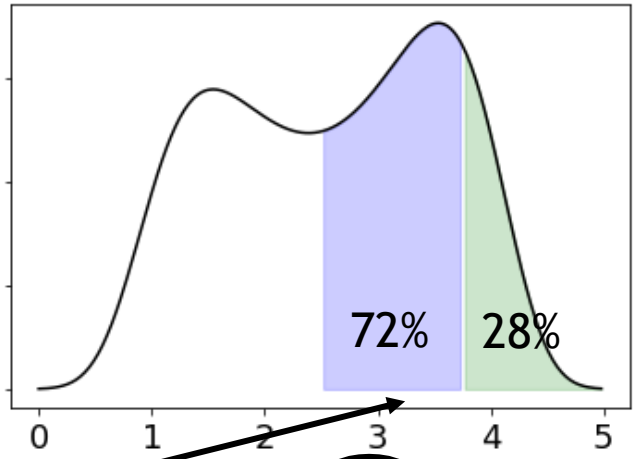
Heuristic: $N \max(1/2^b)^2 \sim 1$

Sampling tree

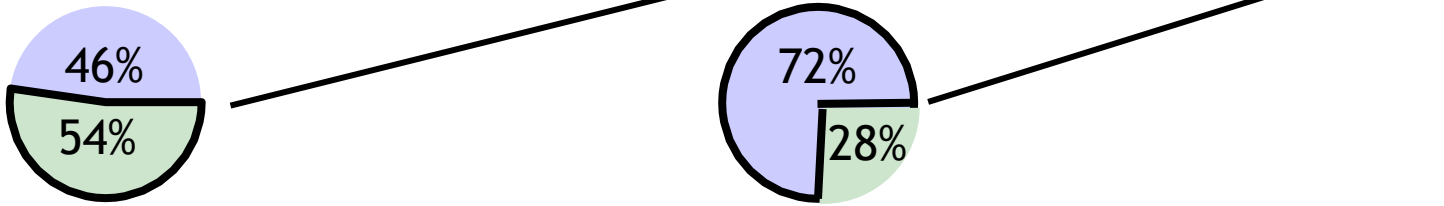
Top half or bottom half?



Top quarter or 3rd quarter?



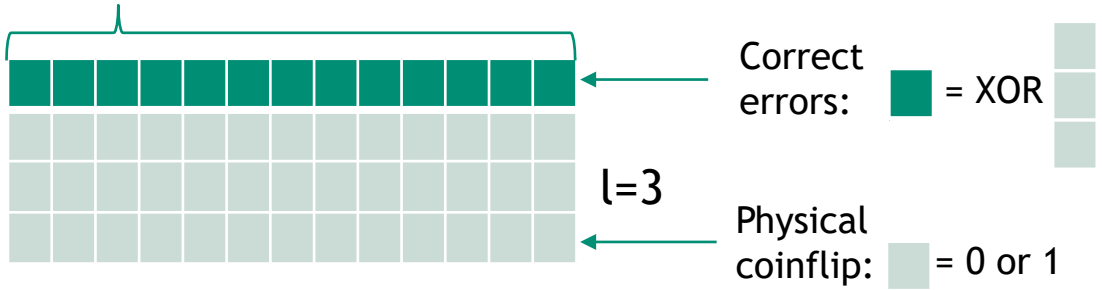
...



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

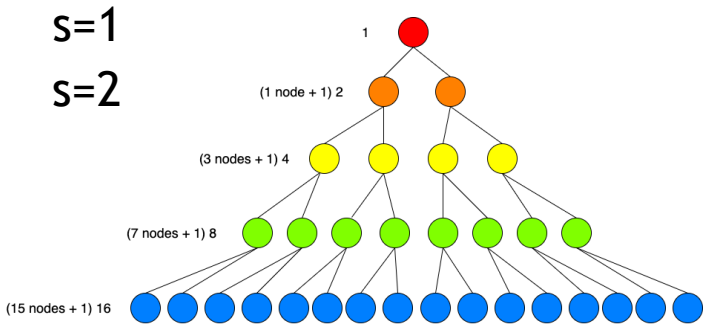
Impractical for a weighted coinflip device.

Logical weighted coinflip



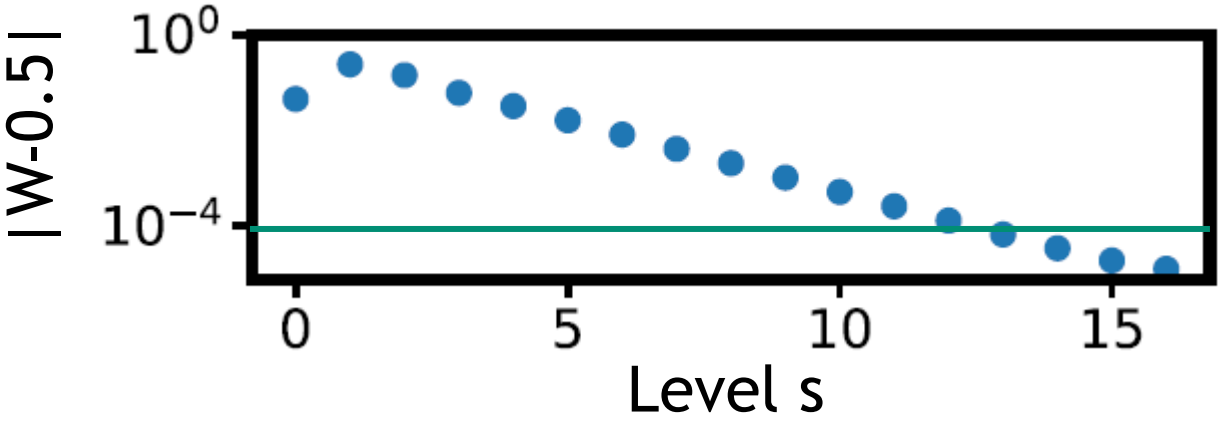
Solution: use fair coins to draw a uniform random sample with 13 bits of precision

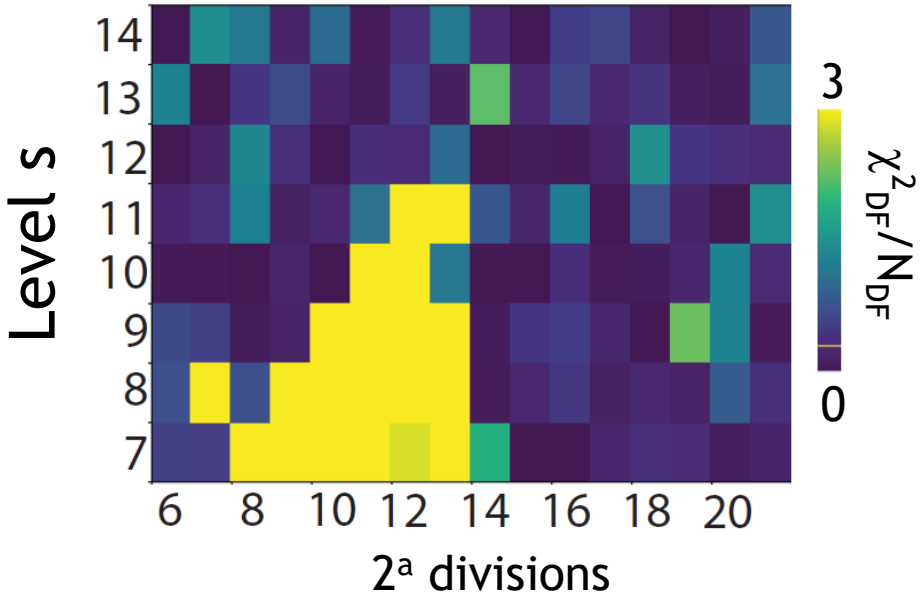
Cutoff sampling tree for efficiency



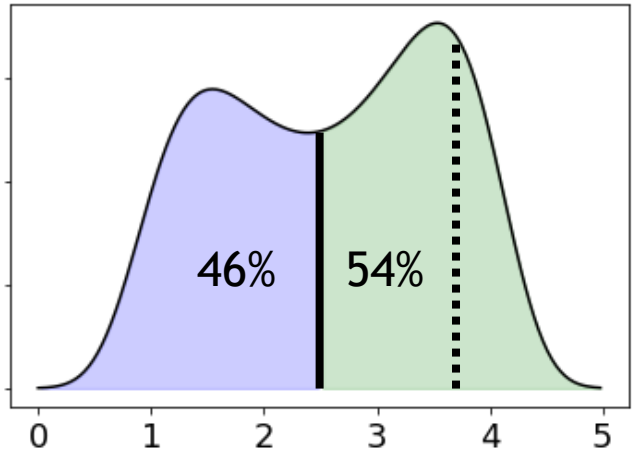
Need to store 2^{32}-1 sixteen bit weights in memory, but nearly all of the weights are 0.5 (after s=12)

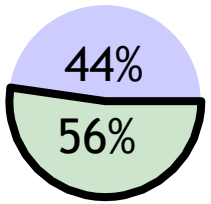
Only need sampling tree for top 12 bits – remaining bits can be uniform random sample





Scheme





Vs.

Uniform random number

32-bit non-uniform sample

Sampling tree

Uniform sample

Efficient

s=12

b=13

Accurate

l=3

Correct errors: = XOR

Physical coinflip: = 0 or 1

Misra, in preparation

Outline

1. Hardware bitstreams
2. Sampling distributions
3. System implications

What device is better?

Component	Description	Technology	Power consumption	# Needed	Reference
V_{in}	Square wave generator using operational transconductance amplifiers	130 nm CMOS	457 µW	Potentially 1	[4]
Current Mirror	High-compliance ultra-high output resistance with positive shunt feedback	130 nm CMOS	570 µW @ 96 µA	1/TD	[5]
Tunnel Diode	Three-terminal silicon surface junction tunneling device	1 µm CMOS	~1 nW in “tails” ~1.75 nW in “heads”	1/TD	[6]
Comparator	Low-noise, low-power, dynamic latched comparator using cascode	130 nm CMOS	100 µW	1/TD	[7]
XOR	2-input pass-transistor XOR	130 nm CMOS	231 pW	Potentially 1/RN	[8]

At a few ns timing, device (aJ - fJ) < circuit (pJ) < INT (nJ)

- Device is irrelevant
- Have flexibility at the circuit level to incorporate, e.g., analog feedback
- It is not that integer operations (INT) need to get more efficient… you just want to minimize them

[4] I. A. A. Al-Darkazly *et al.*, IEEE Access, vol. 4, pp. 3169–3181, 2016, doi: 10.1109/ACCESS.2016.2557843.
[5] M. H. Maghami *et al.*, International Journal of Circuit Theory and Applications, vol. 43, no. 12, pp. 1935–1952, 2015, doi: 10.1002/cta.2049.
[6] J. Koga *et al.*, Appl. Phys. Lett., vol. 70, no. 16, pp. 2138–2140, Apr. 1997, doi: 10.1063/1.118970.
[7] A. M. Maghraby *et al,* 2020 12th International Conference on Electrical Engineering (ICEENG), Jul. 2020, pp. 335–338. doi: 10.1109/ICEENG45378.2020.9171746.
[8] N. Ahmad *et al.*, Active and Passive Electronic Components, vol. 2013, p. e148518, Mar. 2013, doi: 10.1155/2013/148518.

Is TRNG better than PRNG? What sampling scheme is better?

Minimize integer operations (INT)

Uniform distribution

PRNG

10 INT

TRNG

96 coinflips
2 XOR

Non-uniform distribution

Rejection

2 RNG
100 INT acceptance
1 conditional
2x executed on average

Tree

13 RNG
12 conditionals
12 cache access

	TRNG	PRNG
Rejection	202	242
Tree	24	154

Misra, in preparation

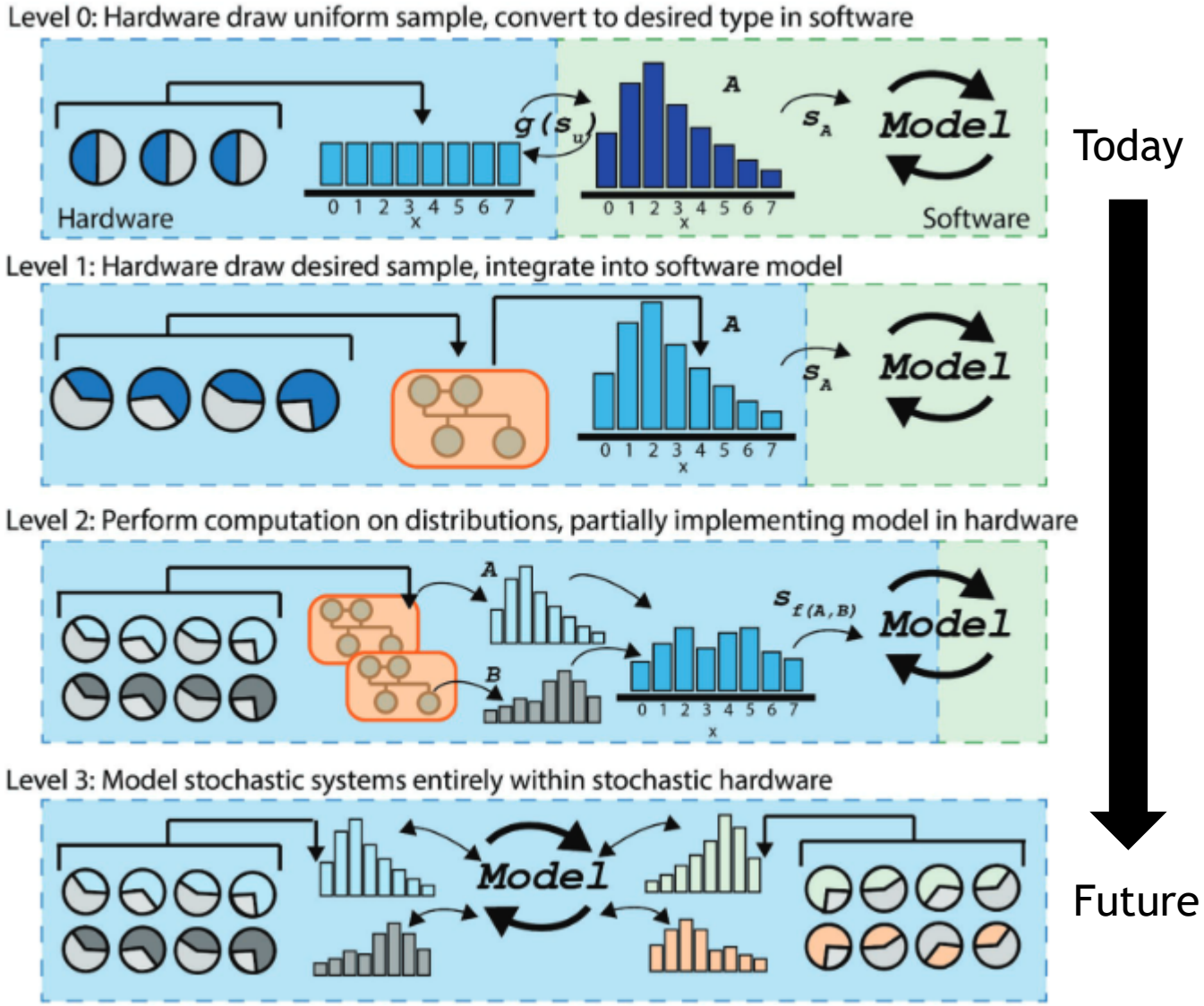
Codesign *matters*

A vision for probabilistic computation

Sampling is expensive (time, energy)
Sampling < 50% of CPU time

Goal: Make sampling cheap (devices, circuits)

Move more of computational burden from deterministic computations into sampling (new algorithms, architecture, etc.)



Misra, Adv. Mater. (2023)