En-Tropic Thunder

Analyzing Randomness and Entropy

Team Members & Mentors

Team Members:

Boris Strots

Simon

Joshua Hatfield

Gauthier

Kyrill Serdyuk

Lui

Nicole Baker

Mentors:

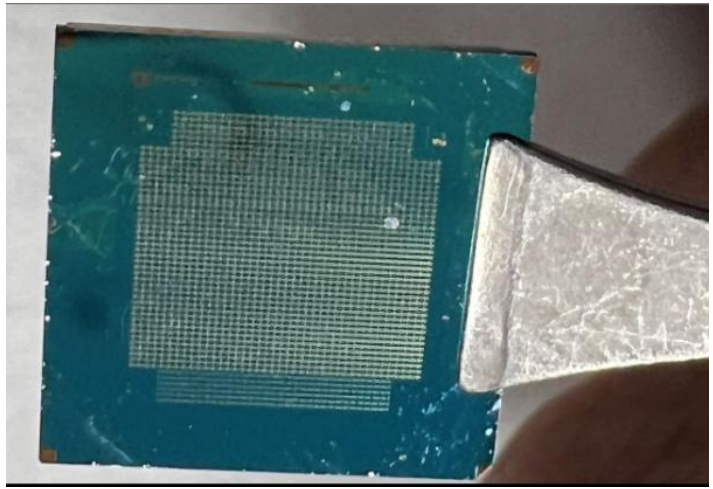
Dr. Tyler

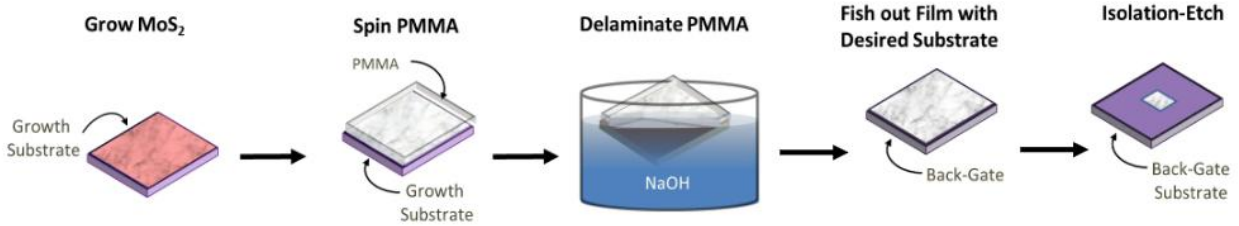
Daniel

Wesley

Our Quest For True Randomness

- Our Mission: Evaluate different sources of entropy on hardware as true random number generators.
  - Theory of Entropy and Randomness
  - Applications of random numbers
  - Observing and testing physical phenomena as source of randomness
- There is the **Dark Crystal** developed by Penn State:
  - True random number generator
  - A single power efficient chip
- How effective the **Dark Crystal** is as a true random number generator?
- Can we do better?





Theory
(Entropy and Randomness)

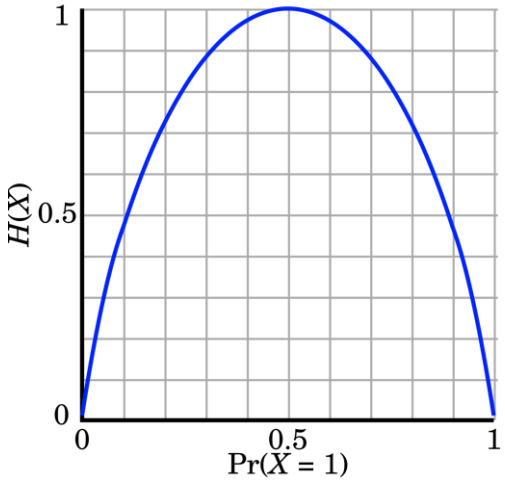
“Anyone who attempts to generate random numbers by deterministic means is, of course, living in a state of sin.”

- John von Neumann

LANL - http://www.lanl.gov/history/atomicbomb/images/NeumannL.GIF (archive copy at the Wayback Machine)

What is Entropy?

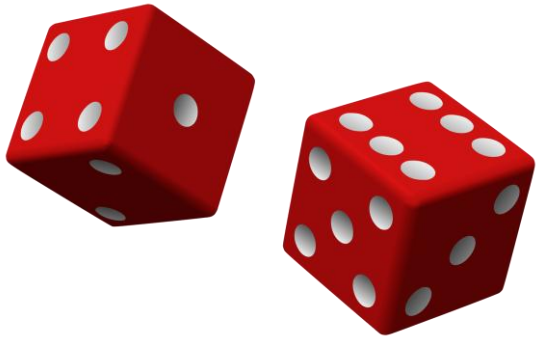
- Entropy is a measure of randomness or predictability.
- Entropy can be used to measure the efficiency of an encoding and compression scheme.
  - Ent (rand.bin): 7.999 bits per byte (~2.0 MiB)
  - Ent (rand.txt): 3.999 bits per byte (~3.9 MiB)
- Shannon's Entropy Equation:
  - H = - (p log p + q logq)
  - When p = q = 0.5, H = 1

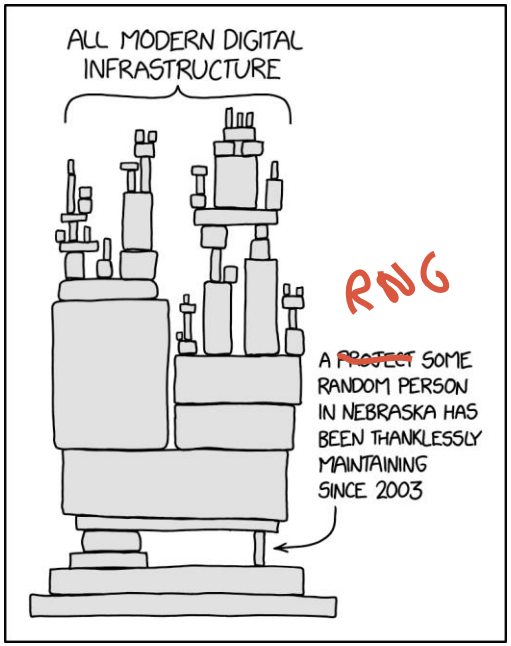


"DEADBEEF"	"\xDE\xAD\xBE\xEF"
8 bytes total	4 bytes total
"A-F|0-9" = 16 4 bits per byte	"\x00 - \xFF" = 255 8 bits per byte

What is Randomness?

- What is randomness?
  - Reliably unpredictable
- Base of the pyramid of trust
- Pseudo-random vs True random
  - TRNG utilizes physical phenomena
- Tested via NIST and Dieharder test suites
  - Monobit, Longest run of 1s, Craps simulation, etc…
  - 18 NIST tests, 188 permutations





STS-CILENT SERVER FOR RANDOM BITS

MENU

Import Bit File

Test Bit File

Compare Bit Files

? Help

Click Me!

Server Status: Operational

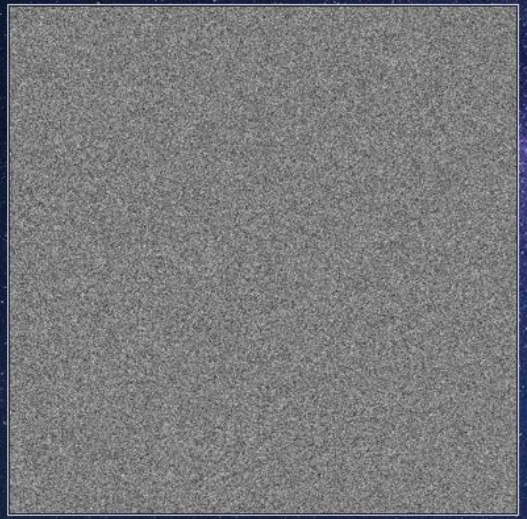
R A N D O M

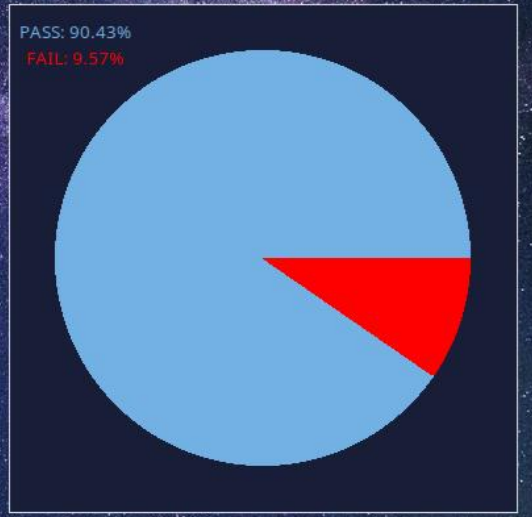
Your md5 hash is: 43dc0fba2a77154c154a23a703e1b686

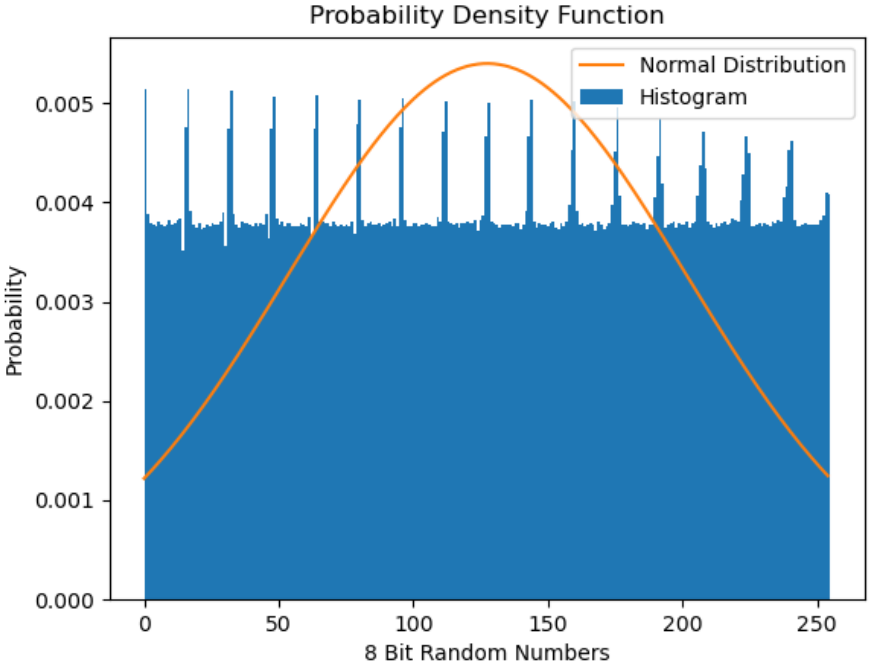
Get Entropy

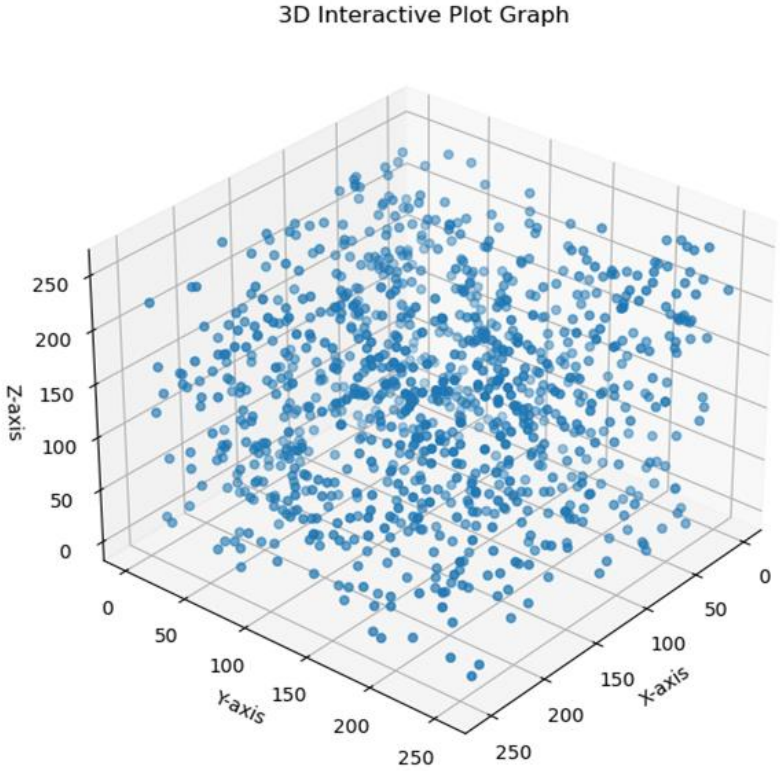
Get Full Report

FileID	Time	Total	Pass	Fail	%Pass	%Fail
43dc0fba2a77154c154a23a703e1b686	2024-07-10 13:30:39	188	170	18	0.90	0.10



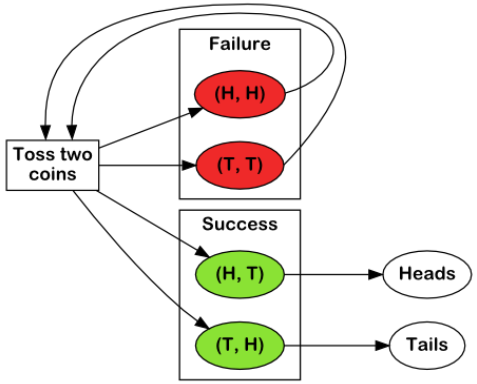



Probability Density Function
Normal Distribution
Histogram
Probability
0.005
0.004
0.003
0.002
0.001
0.000
0
50
100
150
200
250
8 Bit Random Numbers


3D Interactive Plot Graph
Z-axis
250
200
150
100
50
0
Y-axis
0
50
100
150
200
250
X-axis
0
50
100
150
200
250

Improving Randomness

Von Neumann Extractor



P(0) = 0.9
P(1) = 0.1

P(00) = 0.9 * 0.9 = 0.81
P(01) = 0.9 * 0.1 = 0.09
P(10) = 0.1 * 0.9 = 0.09
P(11) = 0.1 * 0.1 = 0.01

XOR

Input A	Input B	XOR Output
0	0	0
0	1	1
1	0	1
1	1	0

$P_A(1) = 0.7$
$P_B(1) = 0.6$

$P(1) = 0.3 * 0.6 + 0.7 * 0.4$
$= 0.46$

$P(1) = P_A(0) * P_B(1) + P_A(1) * P_B(0)$
$= P_A(0) * P_B(1) + P_A(1) * (1 - P_B(0))$

Weighted average - in between $P_A(0)$ and $P_A(1)$
Entropy maximized closer to P(0) = P(1) = 0.5

Applications

Monte Carlo Problems

Freivalds' Algorithm

Eve (Authentication Scheme)

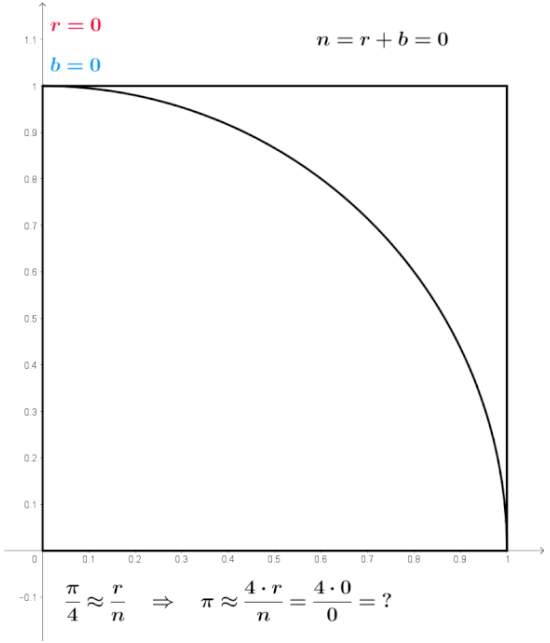
What is the Monte Carlo Method?

- Monte Carlo method is a general technique that uses repeated sampling as a way to obtain a numerical result.
- For example Monte Carlo Markov Chains are used for sampling bayesian priors.
- Problems can have both analytical and Monte Carlo solutions.

Gamow-Stern Elevator Problem:

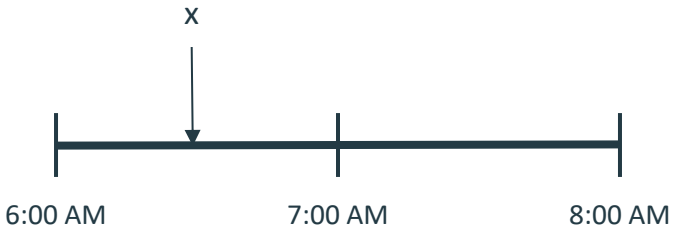
Analytical Solution: ½ + ½(⅔)^n

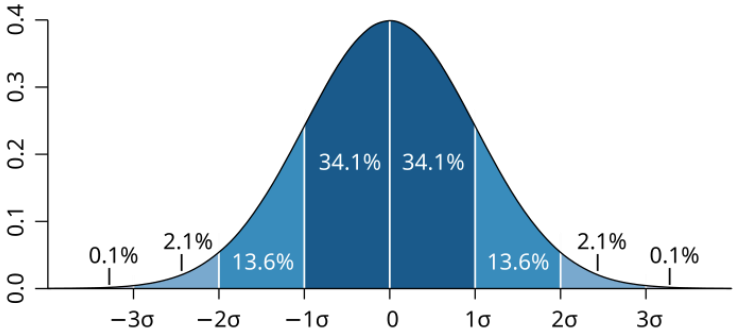
Monte Carlo Solution: Simulate it!



Example: Waiting For Buses

- One bus line arrives at 6 a.m., 7 a.m., 8 a.m., etc.
- Other bus lines arrive at (6 + x) a.m., (7 + x) a.m., etc. where x is [0, 1].
- A person arrives at a random time to a bus stop, on average how long will they have to wait?





Expected	30	20	15	12	10
Data #1	60	30	17	10	6.8
Data #2	29.8	20	14.8	11.8	9.97

Freivalds' Algorithm

- Fast Probabilistic Matrix Multiplication Algorithm
  - Deterministic Algorithm: $O(n^{2.3739})$
  - Freivalds' Algorithm: $O(kn^2)$ with 2^{-k} probability of failure.
- Want to know whether A * B = C
  - Repeat the calculation k times using random numbers
  - Only accept A * B = C if all k iterations were all zero
- Failure rates are low (2^{-k}) only with a good distribution of bits and many iterations.

vec(r) = <0, 1, 1, 0, 1, … , 0>

vec(P) = A * (Bvec(r)) - Cvec(r)

vec(P) = <0, 0, 0, 0, 0, …, 0>

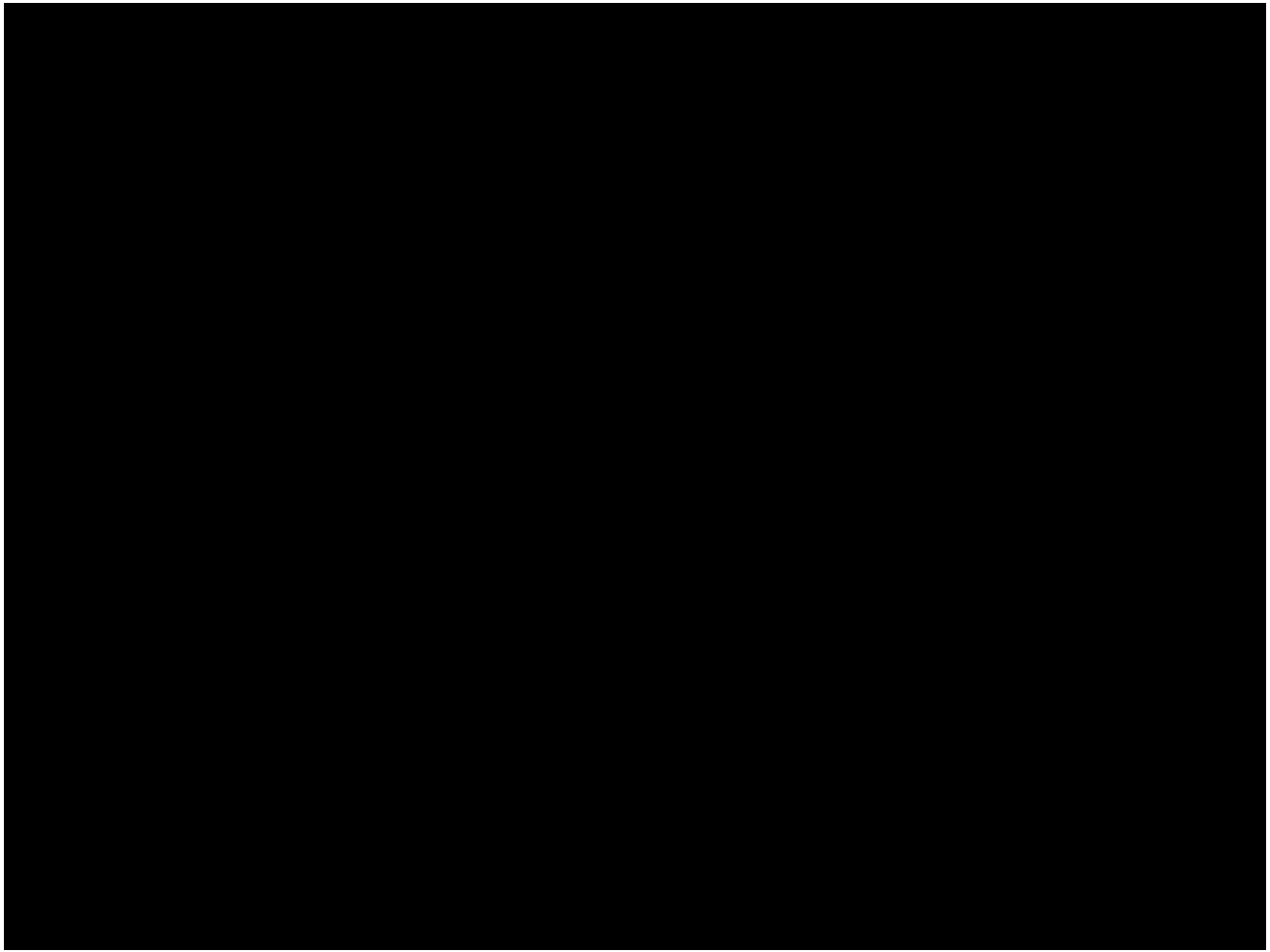
More Examples of Randomized Algorithms:

- Quicksort
- Fisher Yates Shuffle
- Karger's Algorithm (min cut)

EVE (Authentication Scheme)

- Tested our random number generators using a password guessing scheme
- The user inputs a username and a password
  - We know the length of the password and the password only consists of lowercase letters
- The server contains a mapping of each password character to a binary vector, with ‘a’ corresponding to 0 = 0b00000 and ‘z’ corresponding to 25 = 0b11001.
- The server then finds the hamming distances from the binary vector and checks it against the list of hamming distances calculated from the correct password. If they match then the password is correct if not then the server returns which characters didn’t match.

```
nmbaker@nmbaker-XPS-13-9310:~/Documents/eve$ python3 client.py -u Demo -g appar
Connecting to ('rigel.lps.umd.edu', 6100)
Connected
Sending Demo, appar
----response start----
3: [0, 1, 2, 3, 4]
4: [0, 1, 2, 3, 4]
----response end----
Connection closed
```

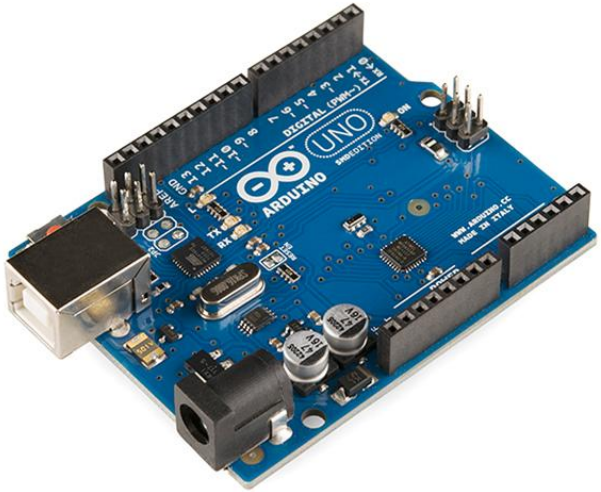


Micro-controllers

Observing Physical Phenomena Through Sensors

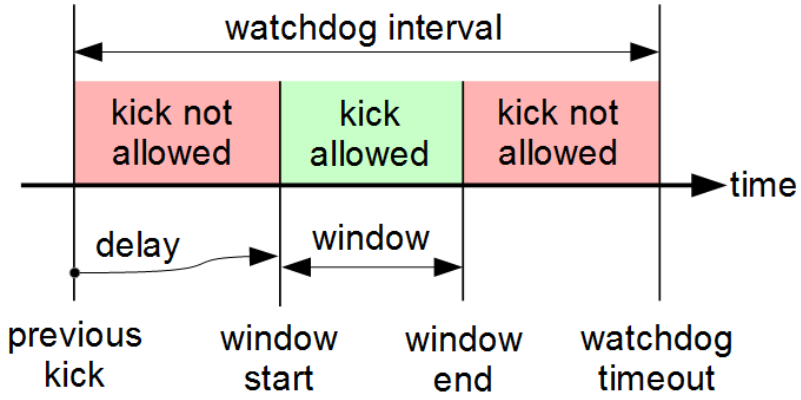
Microcontrollers and Sensors: Our weapon for generating entropy

- Microcontroller is a small embedded system that interfaces well with hardware
- Many sources physical phenomena
  - Onboard Timer/Clock
  - Sound Sensor
  - Laser Sensor
  - Temperature Sensor
  - Etc.



Watchdog Timer

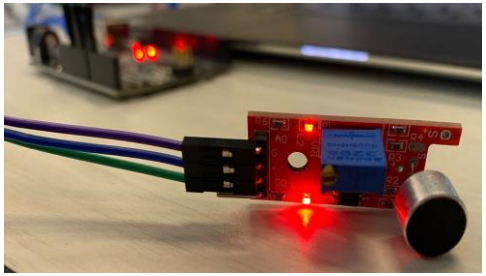
- Entropy through jitter
- 10 timing configurations
  - 16ms, 32ms, 64ms, 125ms, 250ms, 500ms, 1000ms, 2000ms, 4000ms, and 8000ms
- Not a good source of randomness

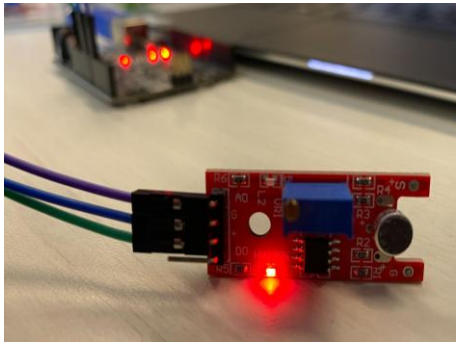


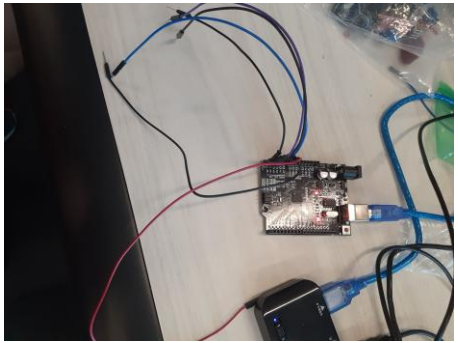
1000 iterations for each configuration

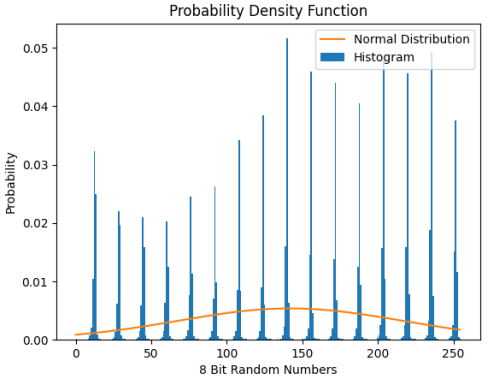
Configurations	Average MS	Average MS Difference
16ms	16.6	0.62
32ms	33.1	0.76
64ms	66.1	0.98
125ms	132.2	0.47
250ms	264.4	0.70
500ms	528.9	0.98
1000ms	1056.6	0.91
2000ms	2114.0	1.89
4000ms	4227.8	4.13
8000ms	8454.2	5.55

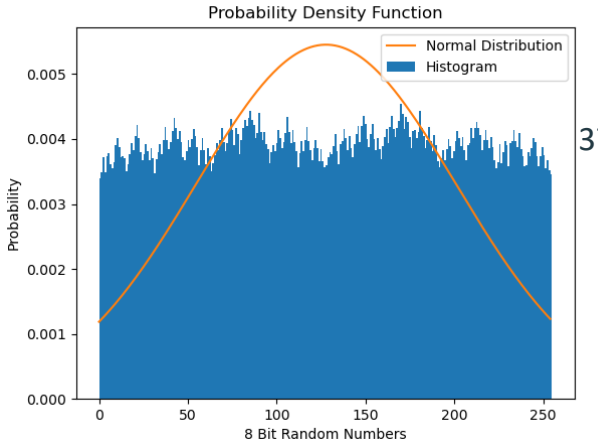
Arduino Sensors





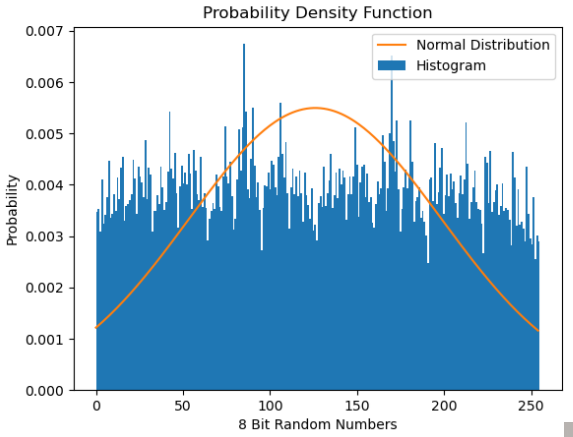




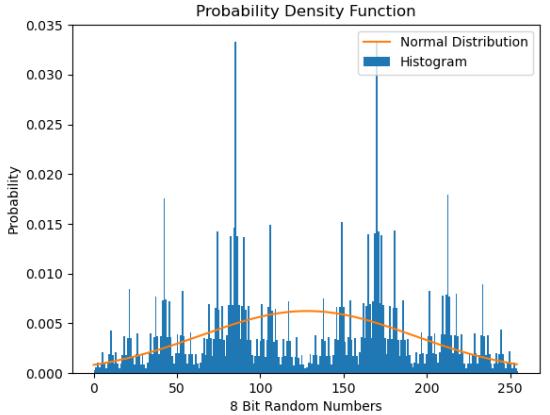


37 bps

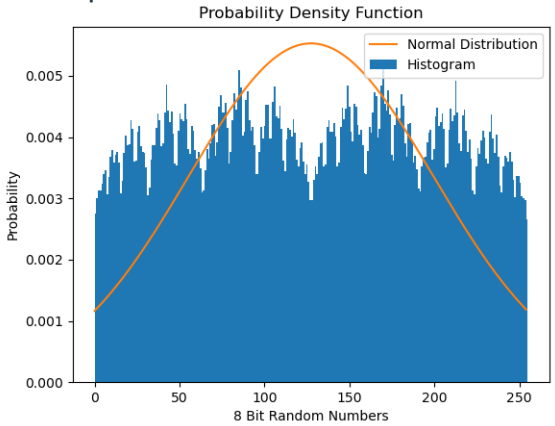
Arduino Sensors

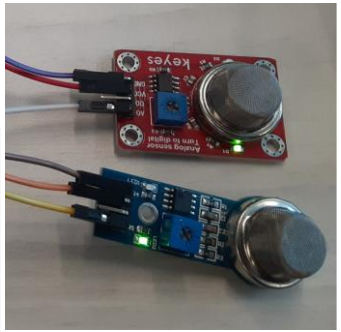


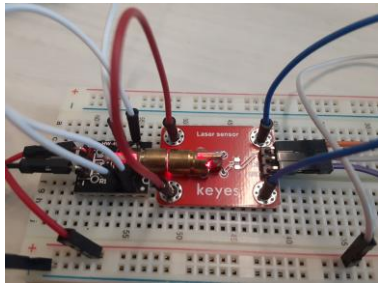
Graphs symmetric - computer synchronization effects?



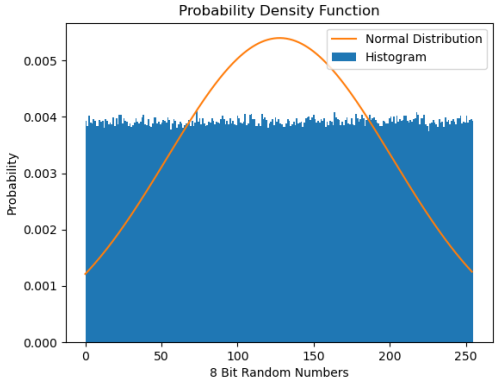
40 bps
68 bps





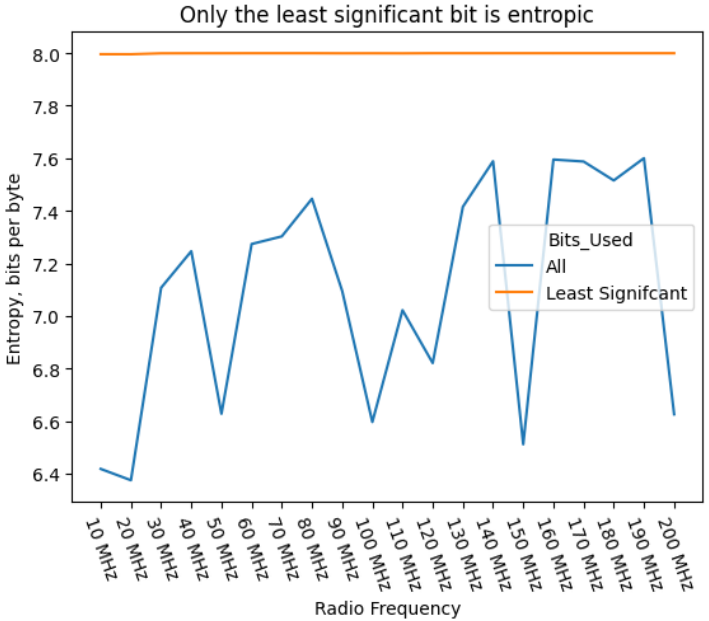


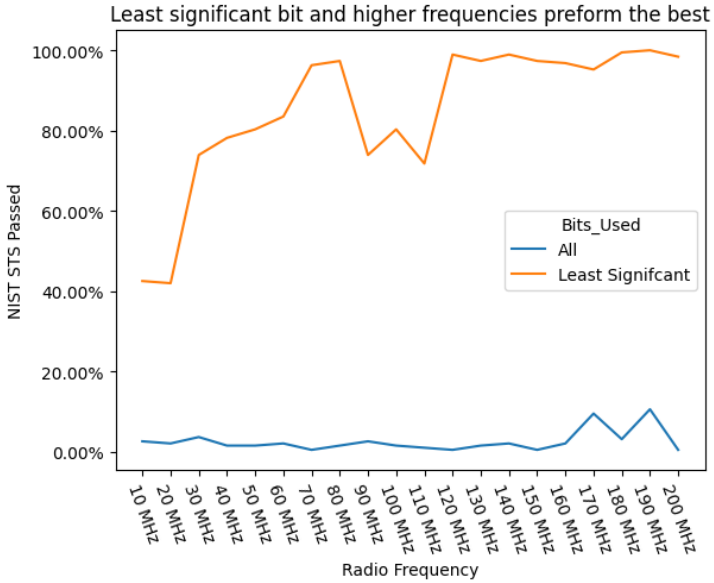
151 bps
26 bps



Radio Antenna

- Radio stations broadcast in 90 - 110 MHz range
- 5992 bps for least significant
- 95839 bps for raw data

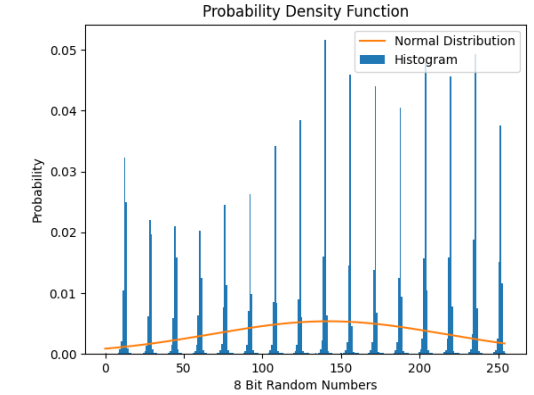


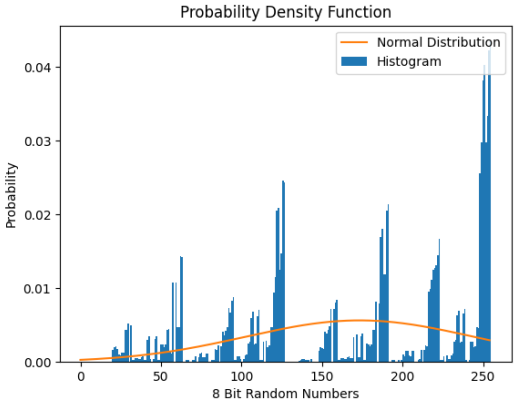


The need for a better weapon

Unfortunately, the microcontroller and sensors had flaws:

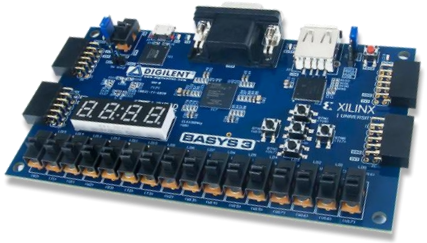
- Very slow to collect data:
  - Average 46.8 Kbits per second
  - About 3 minutes for 1 MiB (1024^2 bytes)
- Sensors were not very precise
  - Most were digital with some threshold
- Many sensors required someone else to generate the physical phenomena
- Did not reach good pass rates at a reasonable speed
  - NIST Pass Rate: From 0% to 80%
  - We want: 90% to 95% with Megabit speed





FPGAs (Field Programmable Gate Arrays)

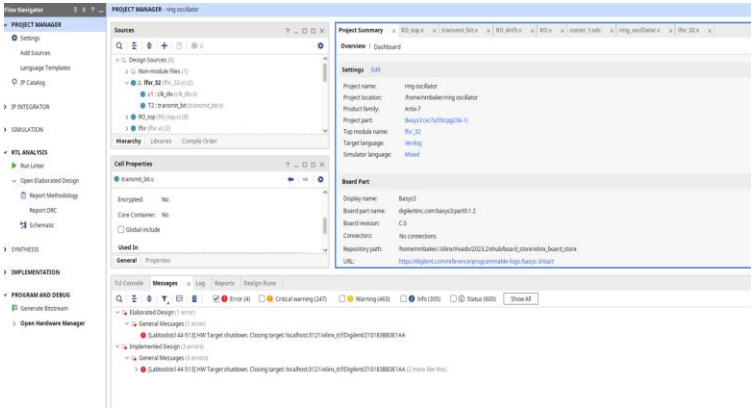
FPGA vs Microcontrollers

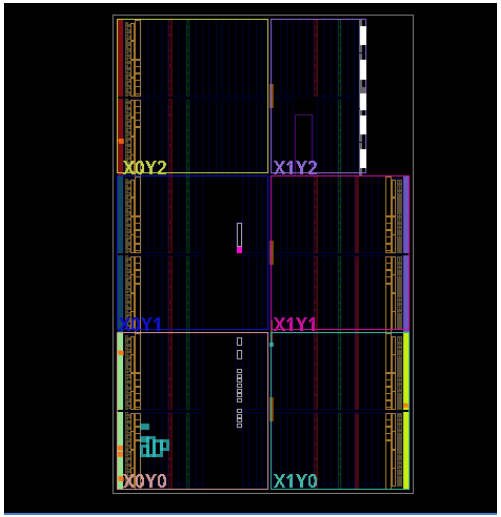


- The approach we took with microcontrollers was using noise from sensors as a source of entropy
  - Selected sensors that we thought would consistently pick up noise
  - Wrote code to process data from the sensors
  - Like a mini-computer: has memory, processor, I/O
- With FPGAs we intended to design hardware that would consistently produce random numbers
  - Designed Linear feedback shift registers, and ring oscillators
  - Our code was implemented on the FPGA as wires, registers, and gates
  - FPGAs are made up of configurable logic blocks that act like logic elements + processor to interface with I/O

Tools for FPGAs

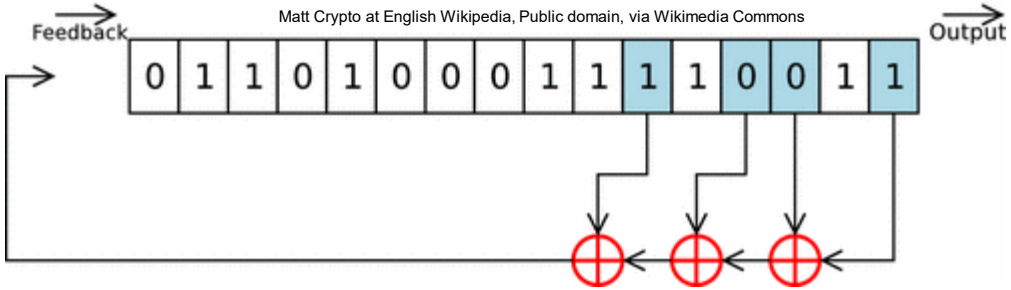
- What is Verilog?
  - Hardware Description language
  - Write code that can be implemented as logical elements on an FPGA
- What is Vivado?
  - Design suite - create, simulate, synthesize, implement
  - Gives us information about utilization, power, and visual representations of how our design is implemented on the FPGA



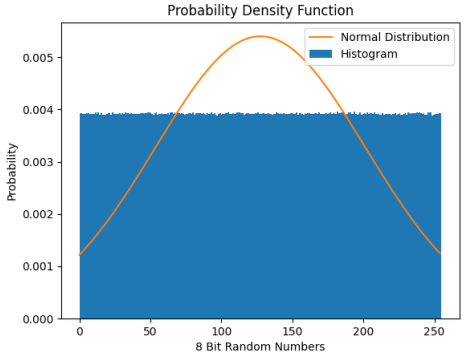


LFSR

(Linear Feedback Shift Register)



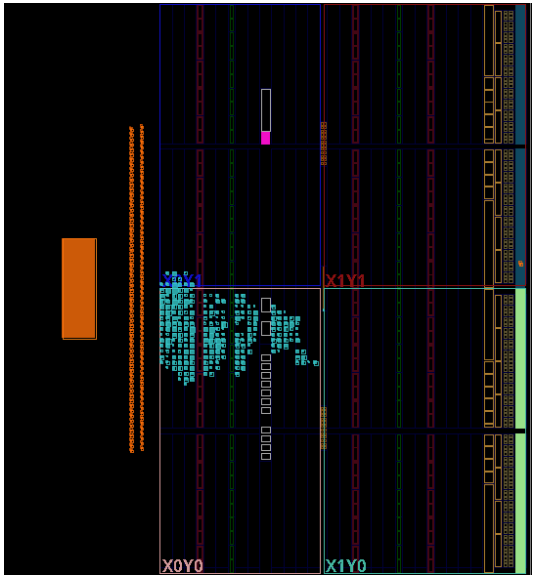
- How it works
  - When the output is shifted out the leftmost bit that is shifted in is an xor of several bits in the shift register
  - If the bits are chosen correctly then you can create a max length LFSR
  - Taps represented by feedback polynomial that must be primitive over GF2
- Our results
  - passes 90-100% of NIST tests
- PRNG
- Predictable
- Easy to implement
- 2^n -1 bits
- Speed limited by clock speed of FPGA

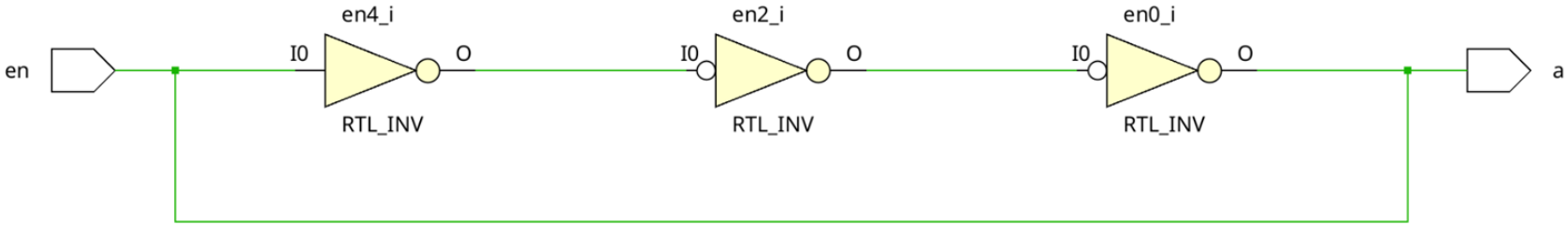


Distribution of bits generated by LFSR

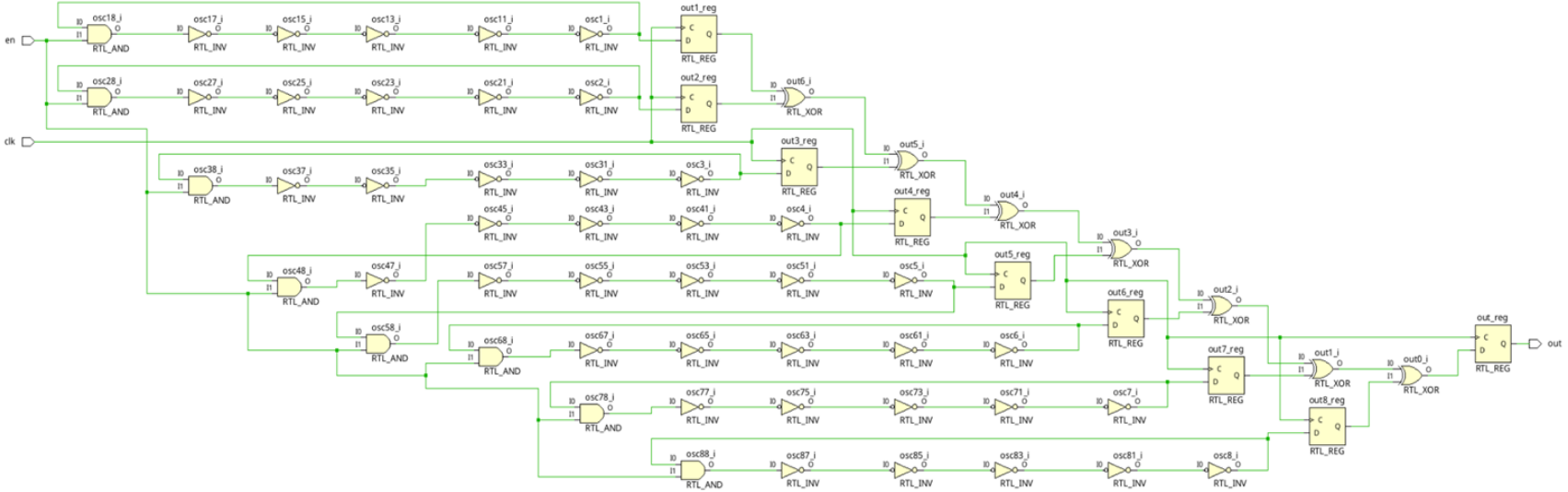
Ring Oscillator

- What is a ring oscillator?
  - Simple circuit w/ not gates
- Shot noise & jitter
  - Inconsistent circuit time
- TRNG
  - Source of true randomness
  - 94% - 100% NIST tests and 7.999+ entropy
  - 1200 bytes per second (a little slow)

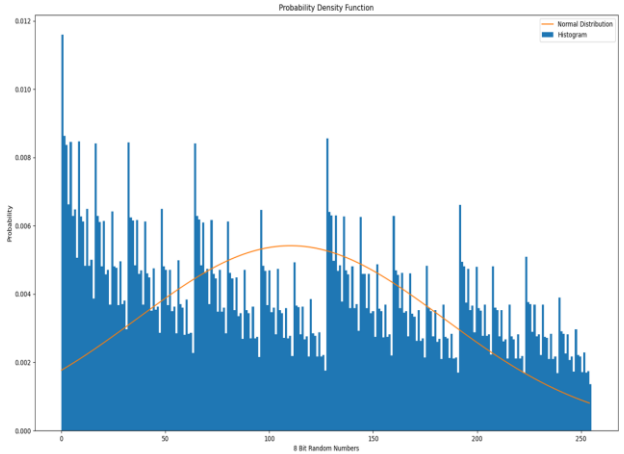


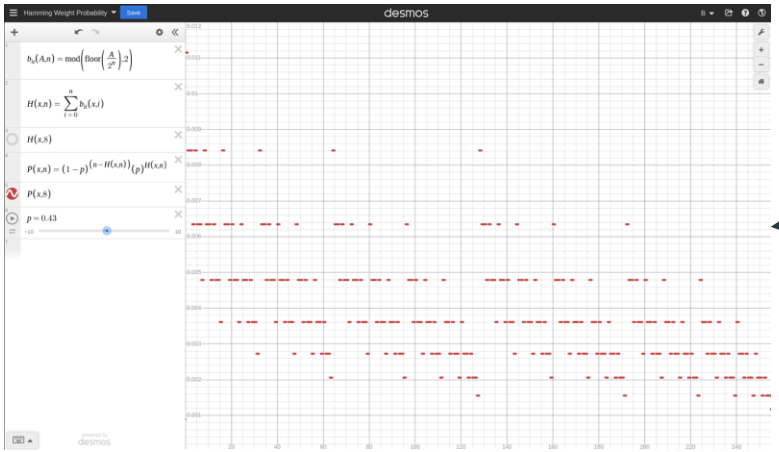


XOR Ring Oscillator



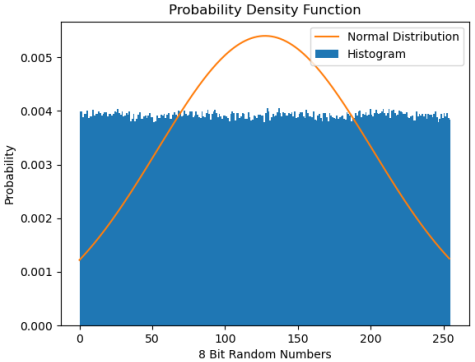
XOR Ring Oscillator Data

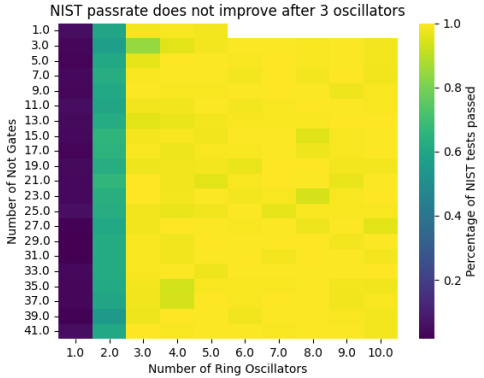




No D Flip-Flops

- Each bit independent
- XOR threshold likely not in exact middle of LOW and HIGH





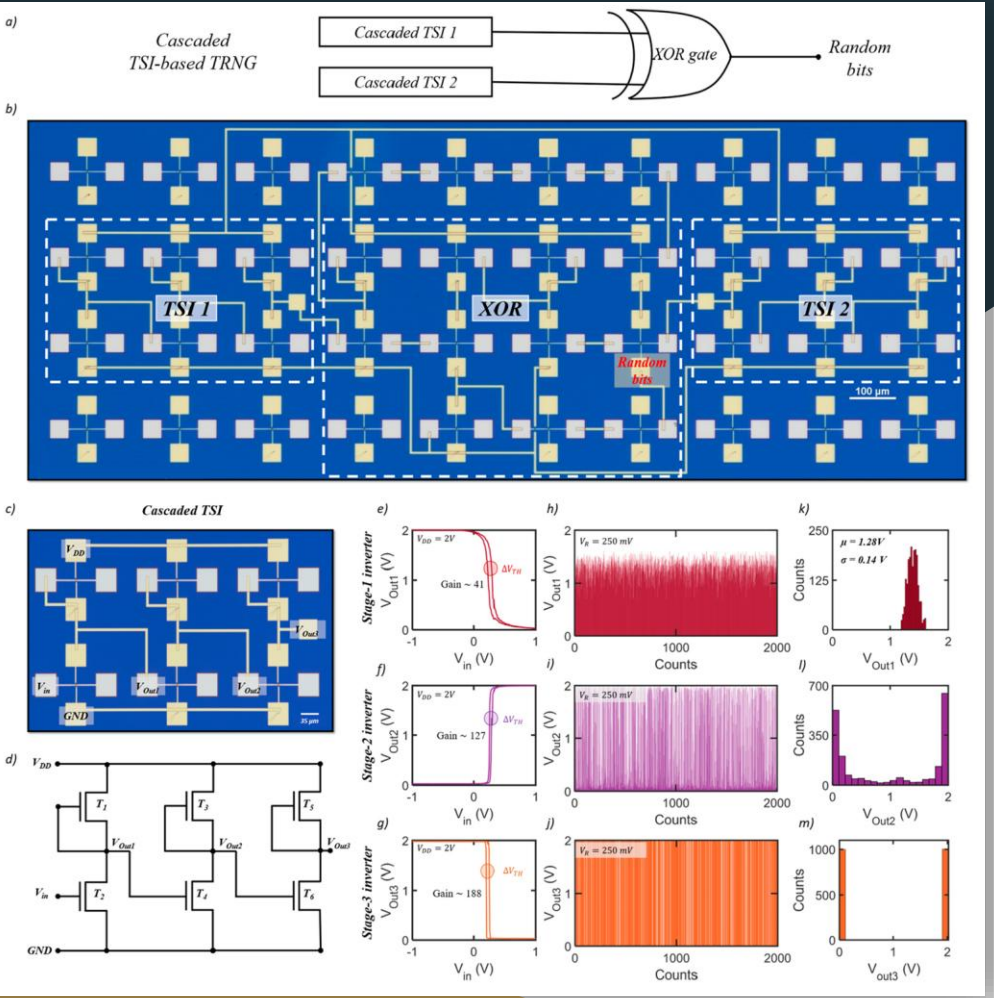
With D Flip Flops

Penn State TRNG

	Penn State	FPGA XOR Ring Oscillator
Speed	1 Kbps	7.2 Mbps
Energy	~30 pJ/Bit	≥650 pj/Bit
NIST Pass Rate	91%	98-100%
Entropy	7.999859	7.9999

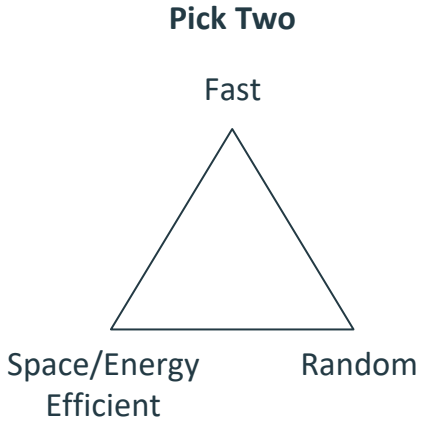
Penn State plans:

- Increase TSI count
- Parallelize for multiple bits at once



Big Takeaways

- It is difficult to generate true random numbers
- ‘Perfect’ randomness does not always produce the same results
- Data can “look” random but that doesn’t mean that it is random
  - May not pass randomness tests or may be generated deterministically
  - Important to use different tests for randomness (NIST, Dieharder, Monte Carlo, Probabilistic Algorithms)
- The number of not gates don’t make a significant difference towards randomness
- Xoring random sources made them more random (ring oscillator, tri-stage inverter)
- True random is cryptographically more secure compared to the PRNG counterpart



Acknowledgements

Two Smart Guys and a Certificate

Tyler Simon and **Taner Pirim**, Ole Miss graduate students and staff members in the Mississippi Center for Supercomputing Research (MCSR), show off the award they recently received at the Mississippi Academy of Sciences (MAS) Conference held in Biloxi. They co-wrote a paper entitled, "Performance Analysis of Linear Algebra and Numerical Aerodynamic Simulation Benchmarks on the MCSR Beowulf Linux Cluster," which took first place in the student paper competition of the Mathematics, Computer Science, and Statistics Division of MAS. Simon presented the paper as part of a special high performance computing sub-session of the conference co-sponsored by MCSR.

References

- By Original: Brona Vector: Alessio DamatoNewer version by Rubber Duck - Own work based on: Binary entropy plot.png by Brona, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=1984868
- By Kmhkmh - Own work, CC BY 4.0, https://commons.wikimedia.org/w/index.php?curid=140013480
- By IkamusumeFan - This drawing was created with LibreOffice Draw., CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=27378699
- By M. W. Toews - Own work, based (in concept) on figure by Jeremy Kemp, on 2005-02-09, CC BY 2.5, https://commons.wikimedia.org/w/index.php?curid=1903871
- Smithsonian Institution Archives, Accession 90-105, Science Service Records, Image No. SIA2008-0295 https://www.si.edu/object/robert-r-coveyou-1915-1996%3Asiris_arc_290783
- By Stephen Silver - https://en.wikipedia.org/wiki/Random_number_generation#/media/File:Two_red_dice_01.svg
- By Lambtron - Own work, CC BY 4.0, https://commons.wikimedia.org/wiki/File:WatchdogWindow.png
- By SparkFun Electronics from Boulder, USA - Arduino Uno - R3, CC BY 2.0, https://commons.wikimedia.org/w/index.php?curid=26785892
- By Sánchez - Own Work, CC BY 4.0 - https://commons.wikimedia.org/wiki/File:Pascual_S%C3%A1nchez_-_Ese_instante.jpg
- Shannon, C. (1948). A Mathematical Theory of Communication. The Bell System Technical Journal, 27, 379–423.
- Kneusel, R. (2018). Random Numbers and Computers. Springer.
- Nahin, P. (2008). Digital Dice: Computational Solutions to Practical Probability Problems. Princeton University Press.
- Ravichandran, H., Sen, D., Wali, A., Schranghamer, T., Trainor, N., Redwing, J., Ray, B., & Das, S. (2023). A Peripheral-Free True Random Number Generator Based on Integrated Circuits Enabled by Atomically Thin Two-Dimensional Materials. ACS Nano, 17(17), 16817-16826.
- Lawrence Bassham, Andrew Rukhin, Juan Soto, James Nechvatal, Miles Smid, Stefan Leigh, M Levenson, M Vangel, Nathanael Heckert, & D Banks. (2010). A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications.
- Choi, S., Shin, Y., & Yoo, H. (2021). Analysis of Ring-Oscillator-based True Random Number Generator on FPGAs. In 2021 International Conference on Electronics, Information, and Communication (ICEIC) (pp. 1-3).

Questions?