True random number generator based on metastability in FPGA

Sebastien Chassot (sinux)

<2019-10-23 Wed>

Introduction

FPGA is a compromise between ASIC and CPU
a versatile reprogrammable circuit

Introduction

FPGA is a compromise between ASIC and CPU
a versatile reprogrammable circuit

ASIC

very specific task
expensive to produce but cheap to use

Introduction

FPGA is a compromise between ASIC and CPU
a versatile reprogrammable circuit

ASIC

very specific task
expensive to produce but cheap to use

CPU

Turing machine can *do anything*
highly reprogrammable

Introduction

FPGA is a compromise between ASIC and CPU
a versatile reprogrammable circuit

ASIC

very specific task
expensive to produce but cheap to use

CPU

Turing machine can *do anything*
highly reprogrammable

FPGA

reprogrammable logic

FPGA are able to...

- implement *any* logic

FPGA are able to...

- implement *any* logic
- perform tasks in parallel

FPGA are able to...

- implement *any* logic
- perform tasks in parallel
- be reprogrammed in a second

FPGA are able to...

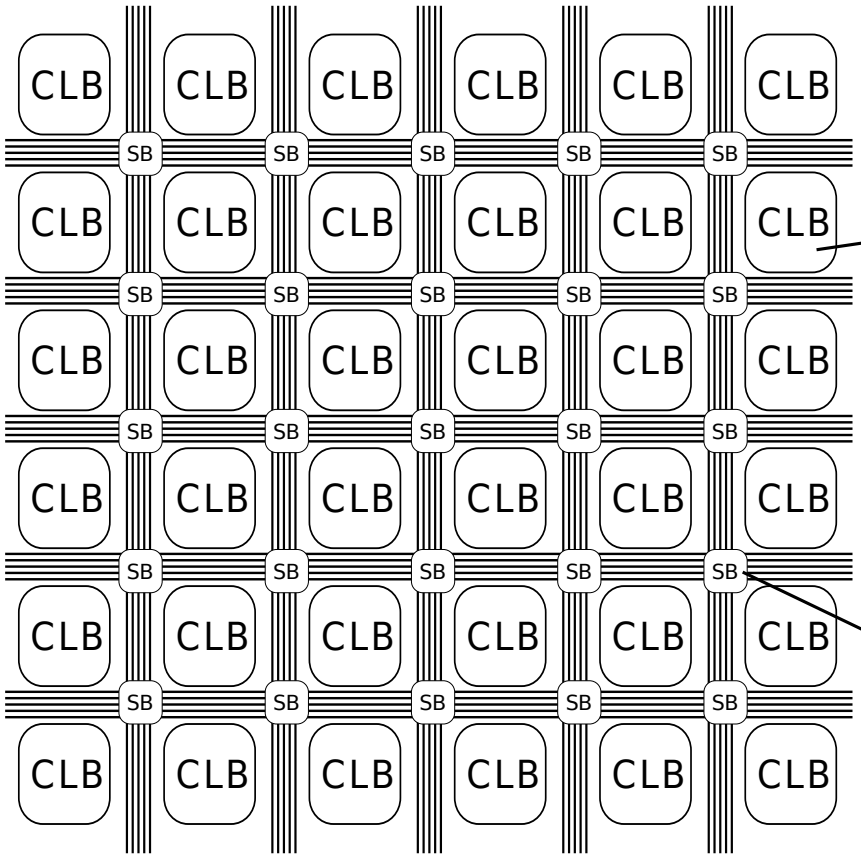
- implement *any* logic
- perform tasks in parallel
- be reprogrammed in a second
- embed is own CPU (hardcore or softcore)

FPGA are able to...

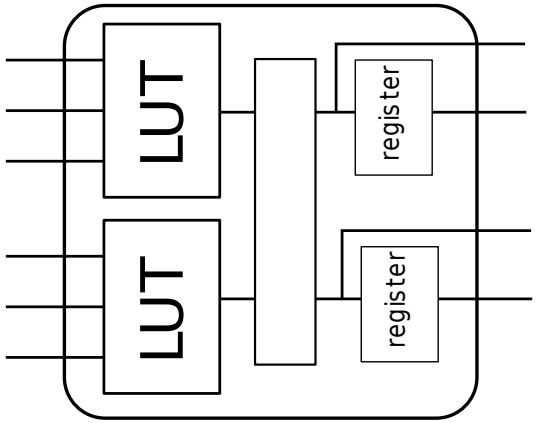
- implement *any* logic
- perform tasks in parallel
- be reprogrammed in a second
- embed is own CPU (hardcore or softcore)
- communicate at high speed typically >6Gbs (PCIe, SATA, USB 3.0, ethernet,...)

FPGA architecture

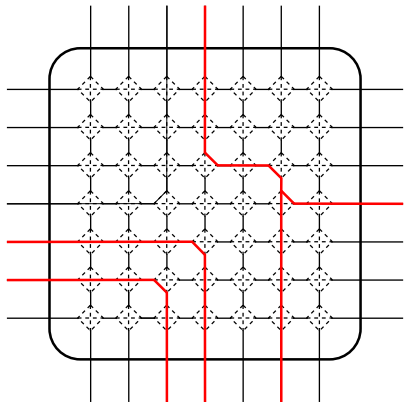
made of thousand of CLB (logic)
interconnected by SB (routing)



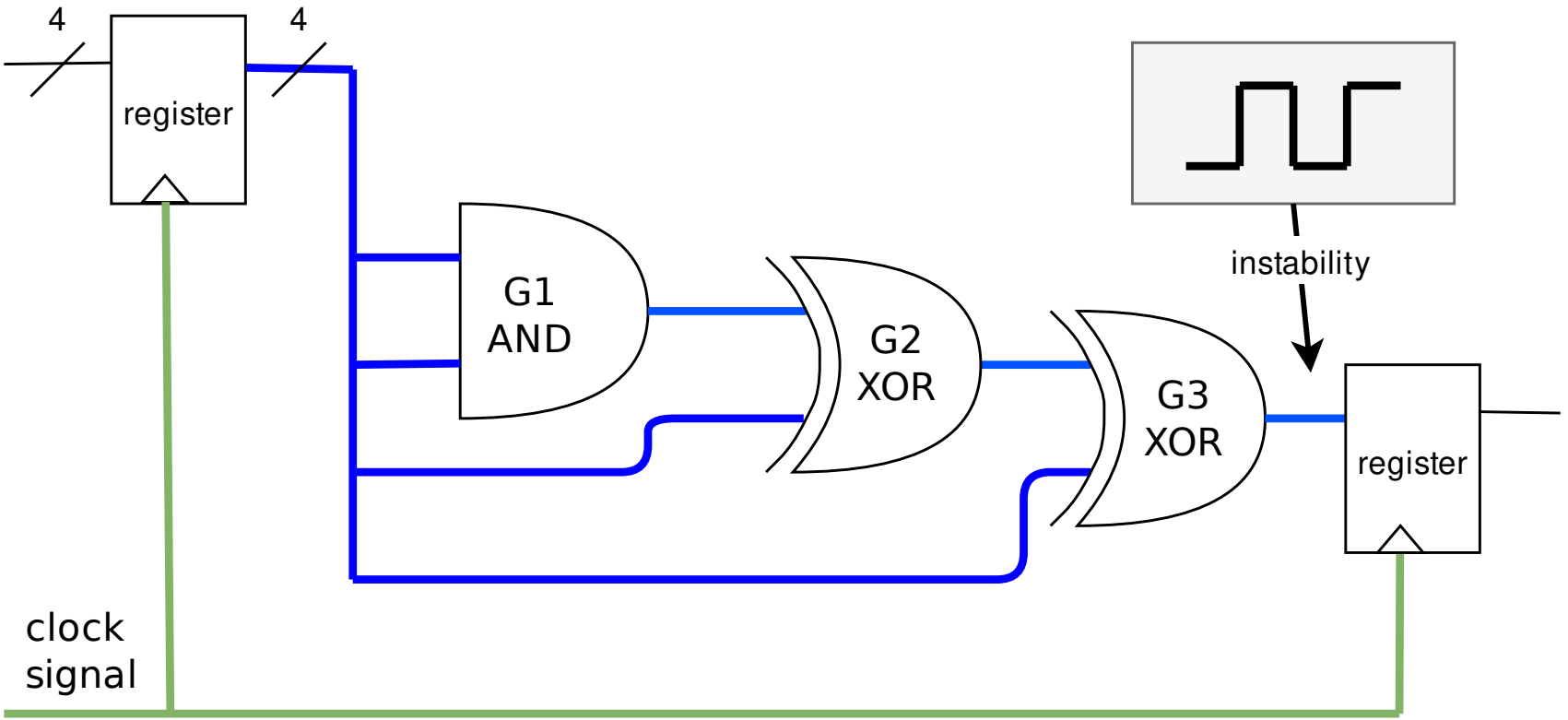
Configurable Logic Block



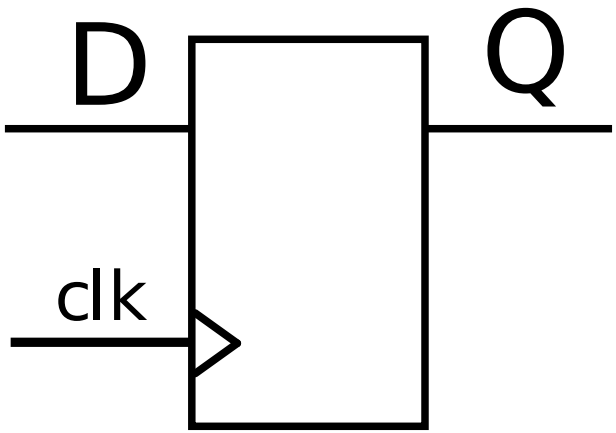
Switch Block

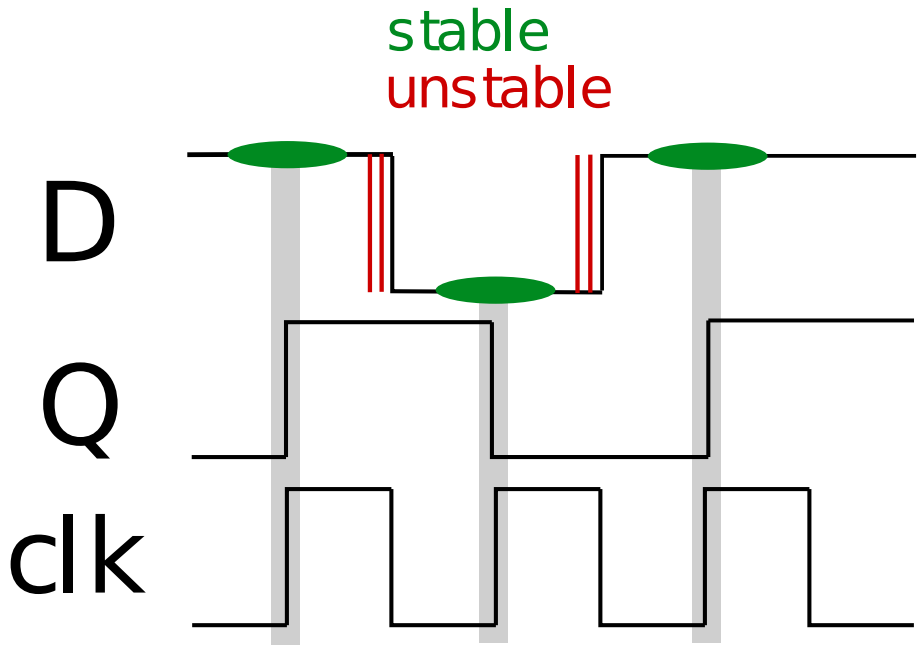


Signal propagation and instability

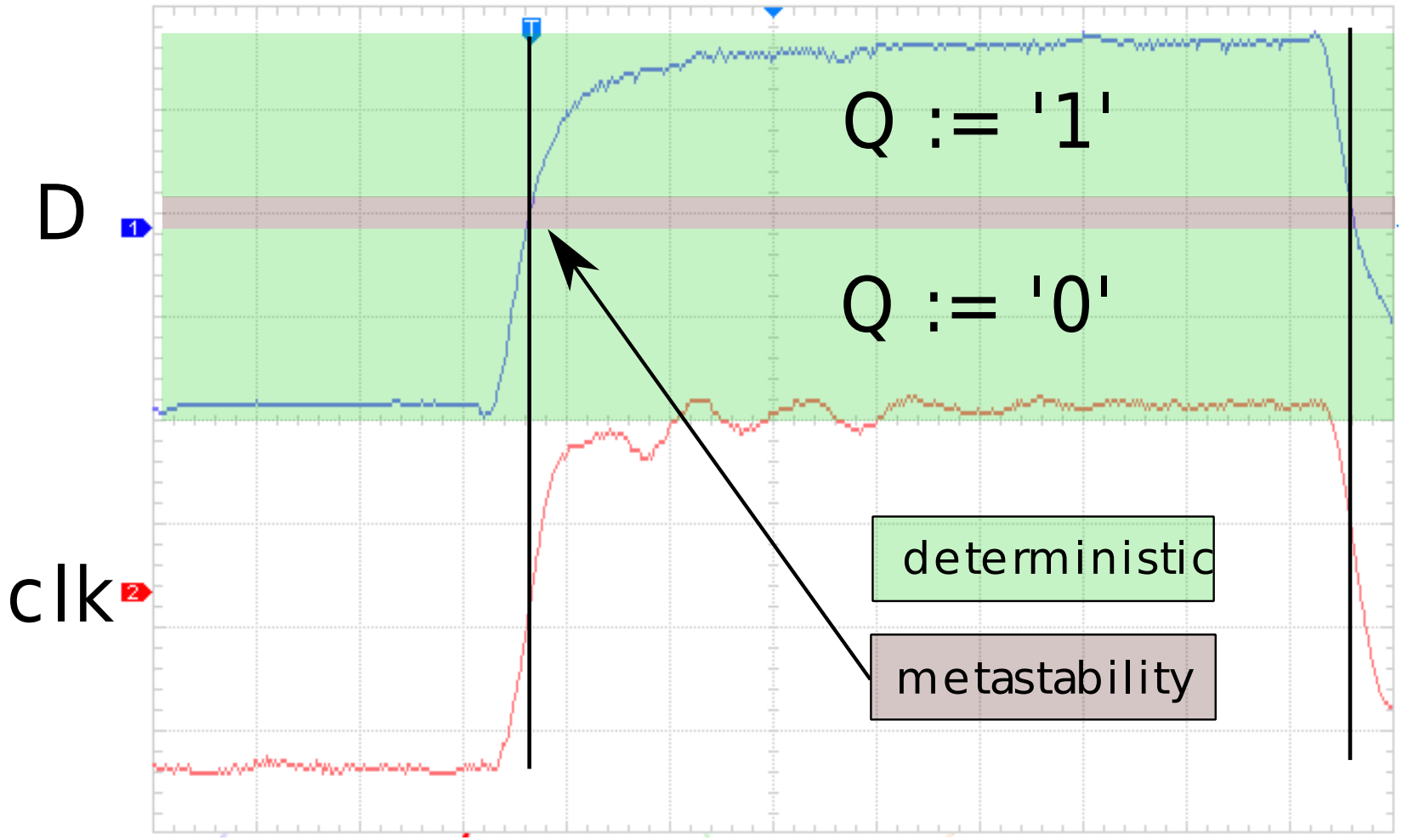


Reminder about registers





Metastability as source of randomness



NIST Statistical Test Suite for Random and Pseudorandom Number Generators

NIST test
monobits frequency test (distribution of 0s and 1s)
block frequency test (periodicity)
spectral (DFT) test

NIST Statistical Test Suite for Random and Pseudorandom Number Generators

NIST test

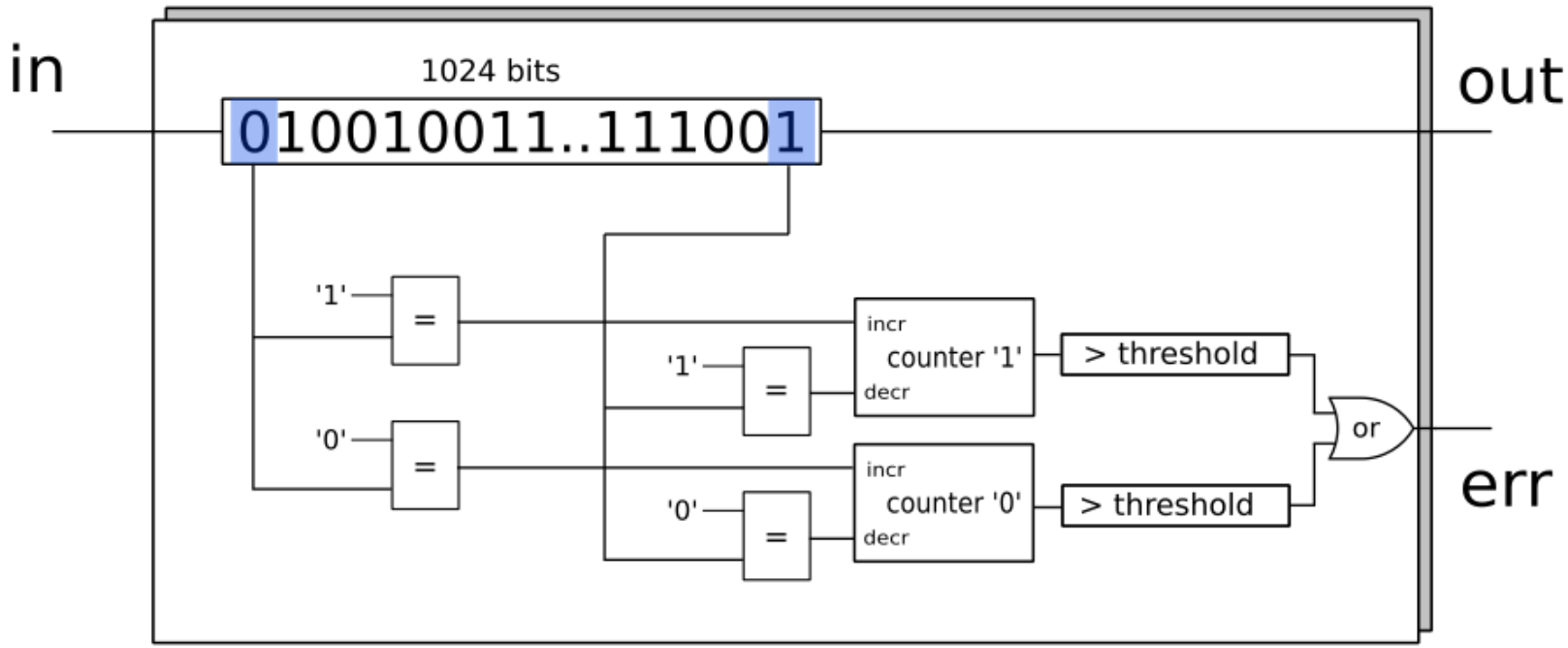
monobits frequency test (distribution of 0s and 1s)
block frequency test (periodicity)
spectral (DFT) test

FPGA fails test so a correction is needed

Adaptive Proportion Test (Von Neumann algorithm)
Cryptographic hash function as randomness extractor (keccak)

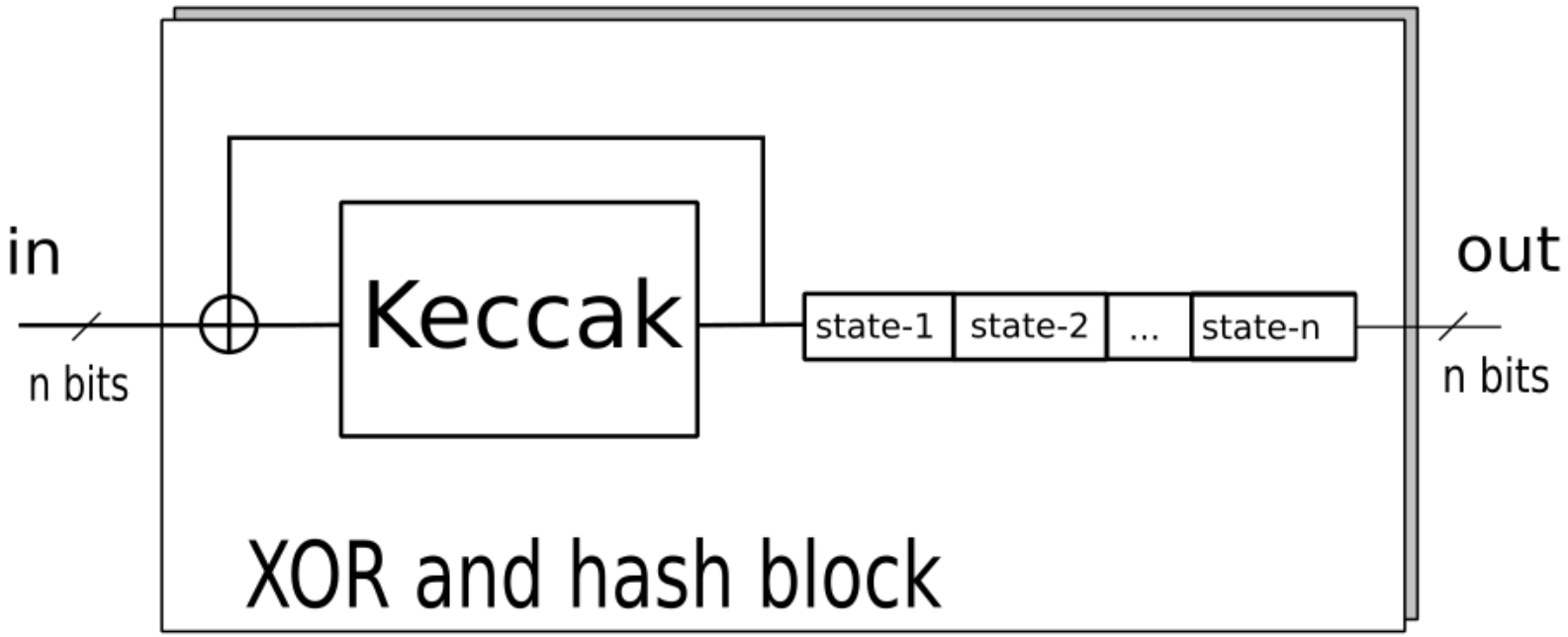
Adaptive Proportion Test

Accept output only if fairly balanced (bit level)



keccak extractor

XOR hash(state) with previous state (block level)



Conclusions

Not a valid solution for the moment but. . .

Conclusions

Not a valid solution for the moment but. . .

Advantage

- minimal surface attack
- don't use any OS resource
- can be coupled with more complex cryptography algorithm

Conclusions

Not a valid solution for the moment but. . .

Advantage

- minimal surface attack
- don't use any OS resource
- can be coupled with more complex cryptography algorithm

Problems and security risk

- hard to implement (hardware specific)
- thermal attack
- power supply attack

Questions ?

links

- Génération de vrais nombres aléatoires en FPGA
  https://sitehepia.hesge.ch/diplome/ITI/2019/Perez-467
- FPGA pour la générationde de vrais nombres aléatoires
  https://sitehepia.hesge.ch/diplome/ITI/2017/Damien-59
- Provably Robust Sponge-Based PRNGs and KDFs
  https://dl.acm.org/citation.cfm?id=3081774
- Quantum Random Number GeneratorsMiguel
  https://arxiv.org/pdf/1604.03304.pdf
- NIST test suite (one of many)
  https://github.com/ycmjason/nist-randomness-test-suite