OpenTRNG: an open-source initiative for ring-oscillator based TRNGs

F. PEBAY-PEYROULA, L. BENEA,
M. CARMONA, R. WACQUEZ

OpenTRNG

FRANCE 2030 PROGRAMME DE RECHERCHE CYBERSÉCURITÉ

CARNOT Research for industrial innovation

Agenda

1. **Objectives**
2. **TRNG basics**
   - From transistor noise…
   - … to RO phase noise
   - Entropy source
3. **OpenTRNG key components**
   - Emulator
   - Implementation
   - Validation tools

OpenTRNG objectives

Framework comprising reference designs, emulation and analysis tools

OpenTRNG is **not** a product and is not certified

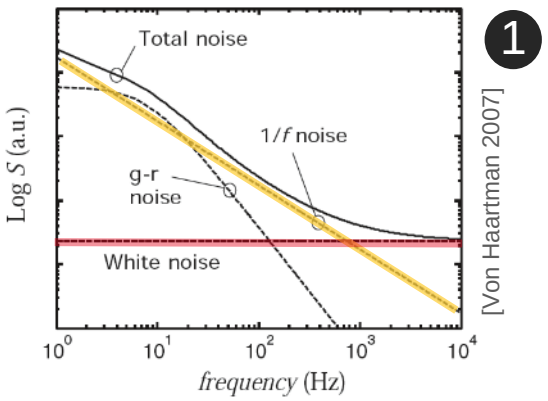
- Help **industrials** to develop and validate TRNG in their products
  - Access to state-of-the-art entropy sources and tools
  - Characterize noise for a given hardware target
  - Helping with certification requirements
- Help **academics** in their research and teaching
  - Characterize custom entropy sources
  - Use the framework in practical work with students

2. TRNG basics

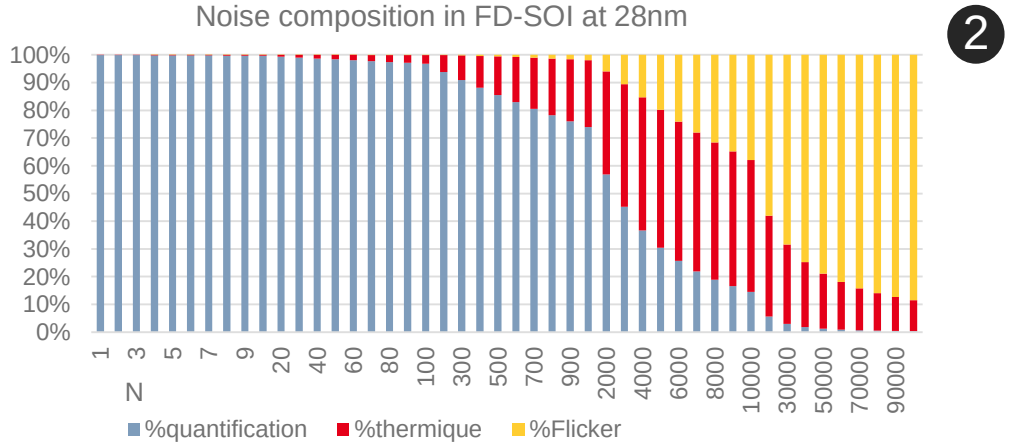
Noises and entropy sources

From transistor noise…

- **Thermal noise**: Gaussian “white” noise
- **Flicker noise**: colored “pink” noise [DSD2022]
  - Higher flicker noise for advanced technological nodes
  - Flicker is a correlated noise, **but**
    - No correlations in the generated randomness
    - Increase the measured entropy

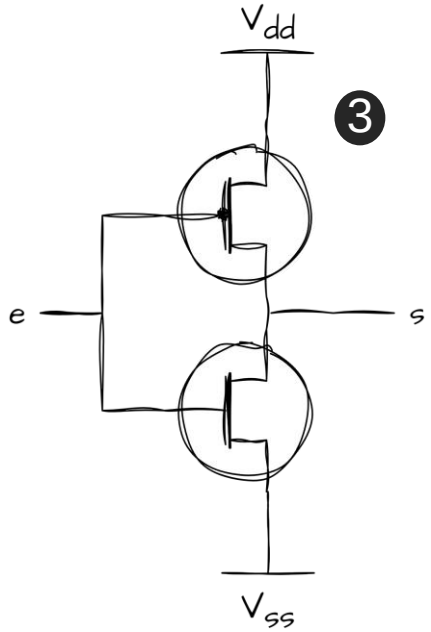


1



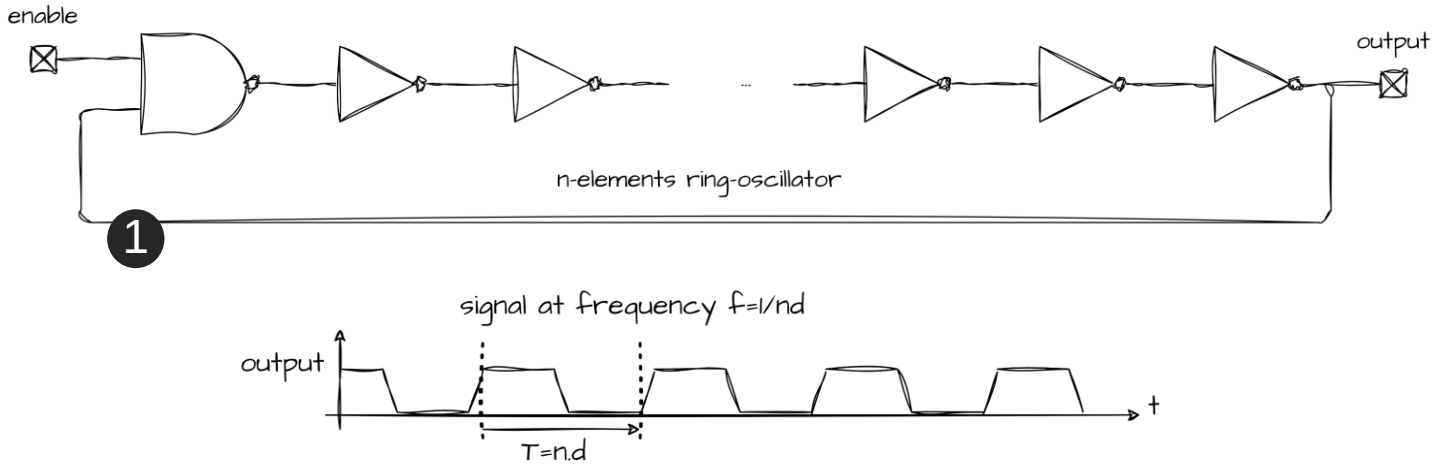
2

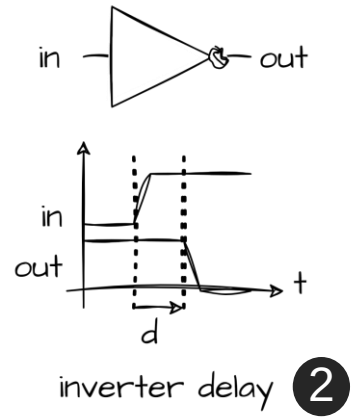
[DSD22] On the Characterization of Jitter in Ring Oscillators using Allan variance for True Random Number Generator Applications, L. Benea, M. Carmona, F. Pebay, R. Wacquez



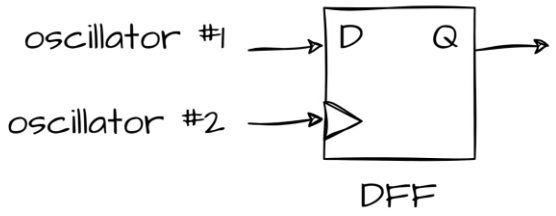
3

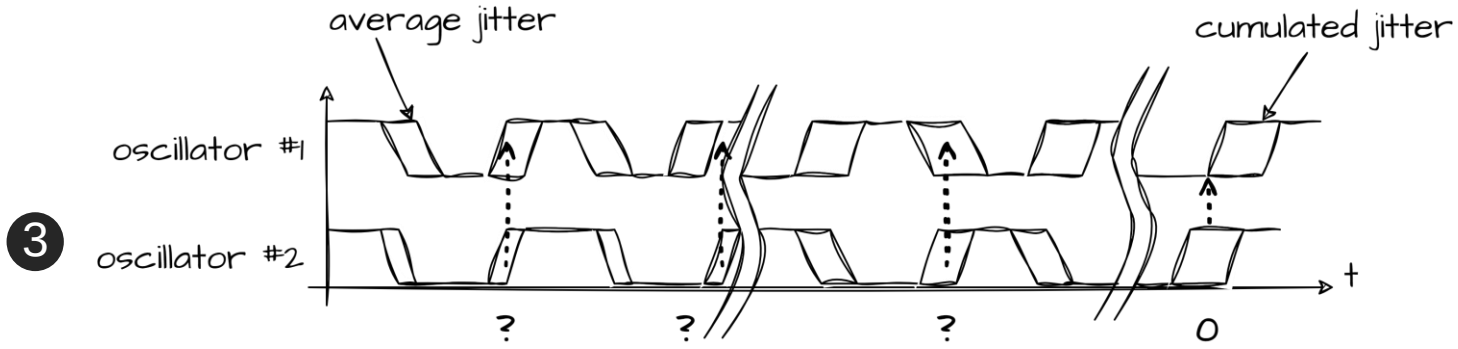
...to ring-oscillator phase noise





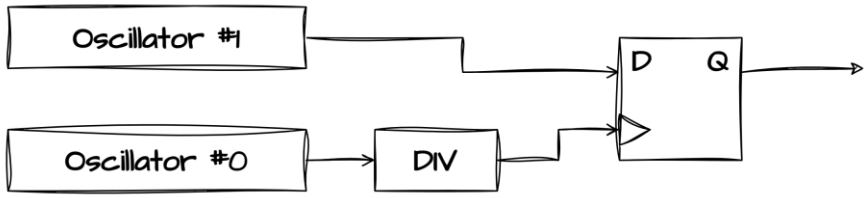
- Basic principle: DFF samples one RO with another
  - Digital randomness is the image of the RO differential phase noise
  - Immunity to common variations (voltage, temperature…)

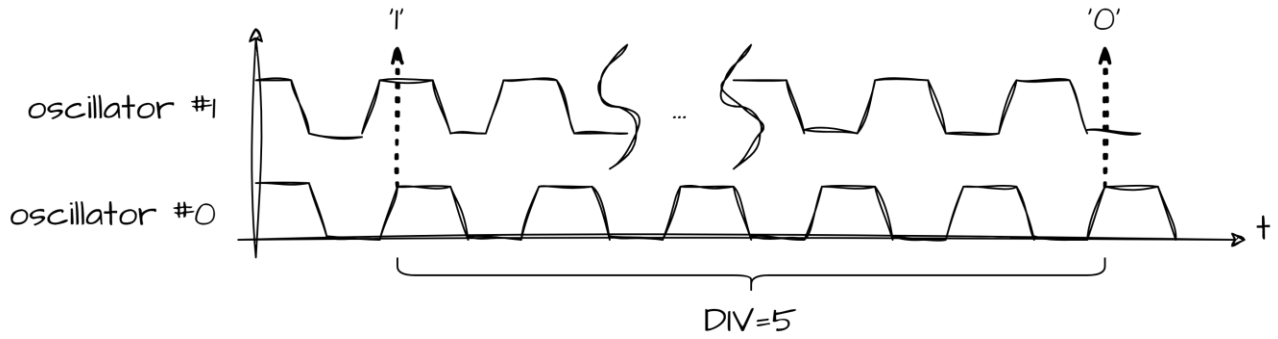




ERO Elementary RO based TRNG

- Oscillator #0 frequency is divided by DIV and samples oscillator #1





Pros

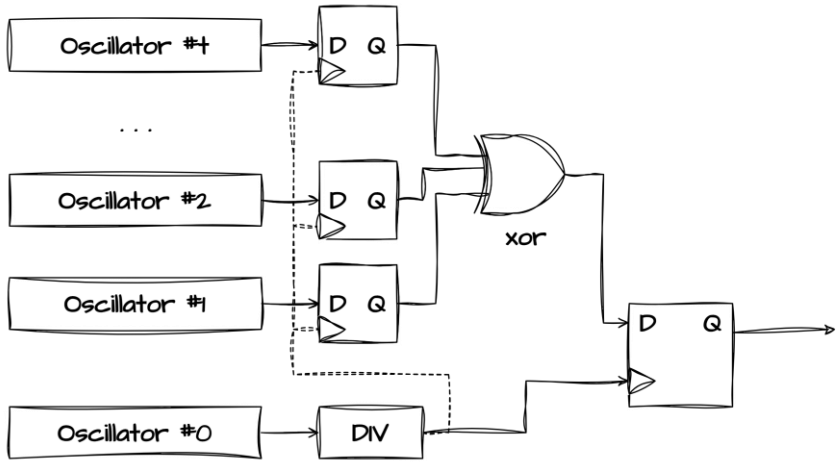
- Simple design
- Available stochastic model
- Few risk of locking

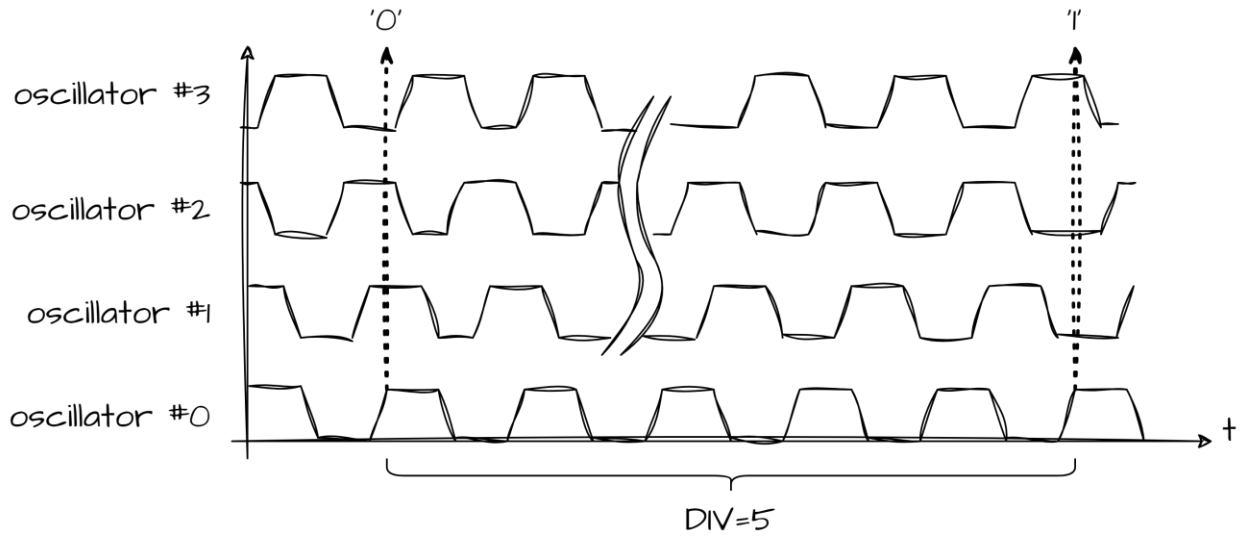
Cons

- Lack of entropy
- Need long integration time (high clock division factor)
- Low throughput

MURO Multi RO based TRNG

- Oscillator #0 frequency is divided by DIV and samples oscillator #1 to 3, results are XORed





Pros

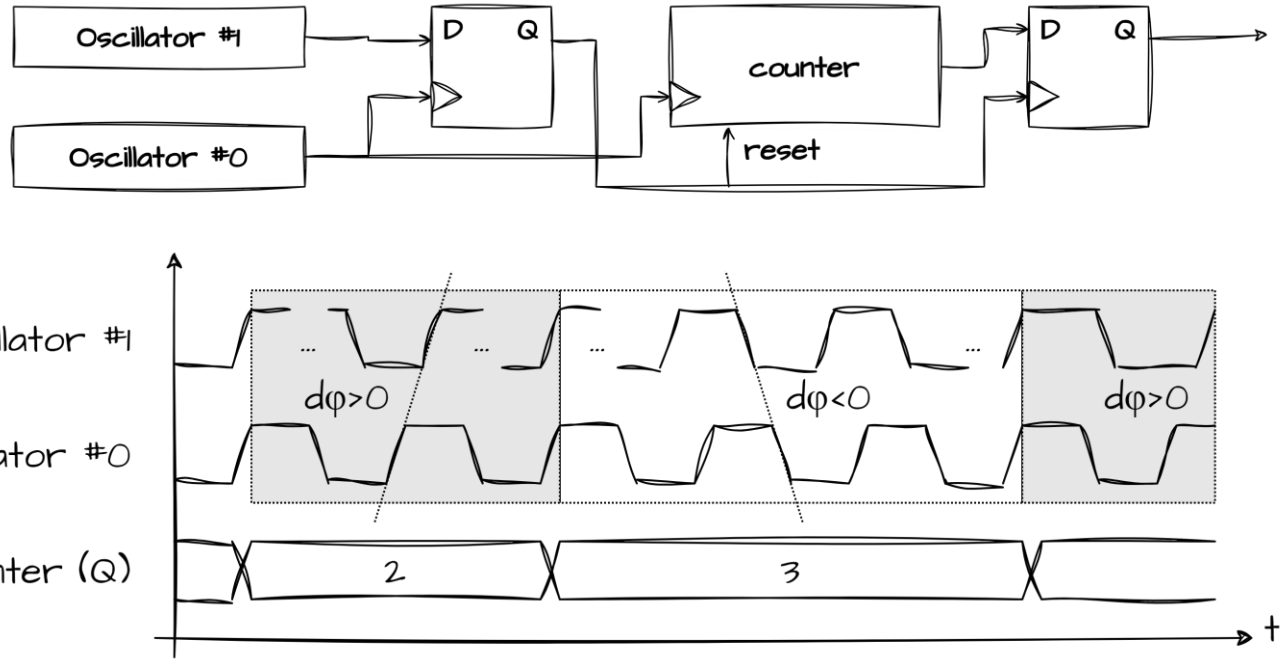
- More entropy
- More throughput at constant entropy

Cons

- More power consumption
- More surface
- Increased risk of locking
- Complex and partially available model

COSO Coherent Sampling RO based TRNG

- Oscillator #0 samples oscillator #1
- On sample rising edge
  - register its current value
  - and reset the counter
- Random bit is output LSB
- Output also gives an internal metric of the entropy source



- **Pros**
  - High throughput (no clock divided)
  - Acceptable entropy
  - Internal metric for online tests
  - Total failure alarm (by design)

- **Cons**
  - Requires freq. conditions on both RO
  - And that's it ☺

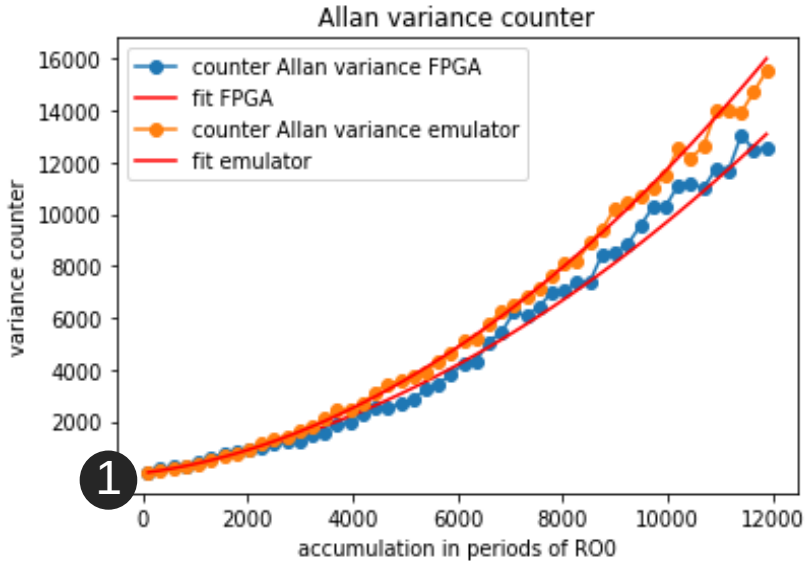
3. OpenTRNG

1. Emulator
2. Implementation
3. Validation tools

Emulator: phase noise

- Emulates RO with thermal and flicker noises [CHES24]

$$dN_i = N + \sqrt{\frac{a_1 \cdot N}{factor_{th}}} \cdot \delta i_{th}{}^{RO} + \sqrt{\frac{a_2 \cdot N^2}{factor_{fl}}} \cdot \delta i_{fl}{}^{RO}$$



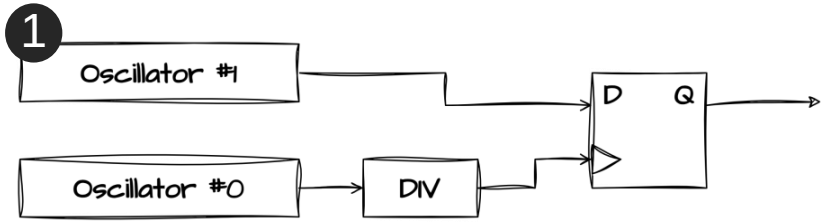
[CHES24] Impact of the Flicker Noise on the Ring Oscillator-based TRNGs, L. Benea, M. Carmona, V. Fischer, F. Pebay, R. Wacquez

- Generates RO time-series
  - at a given RO frequency
  - with thermal and flicker noise figure measured on real hardware
- As instance, generate 10 million periods of an RO at 100MHz on Xilinx Artix 7 (FPGA)

```
$ python ro.py –size 10e6 –freq 100e6 –a1 1.42e-13 –a2 1.15e-25 ro.txt
```

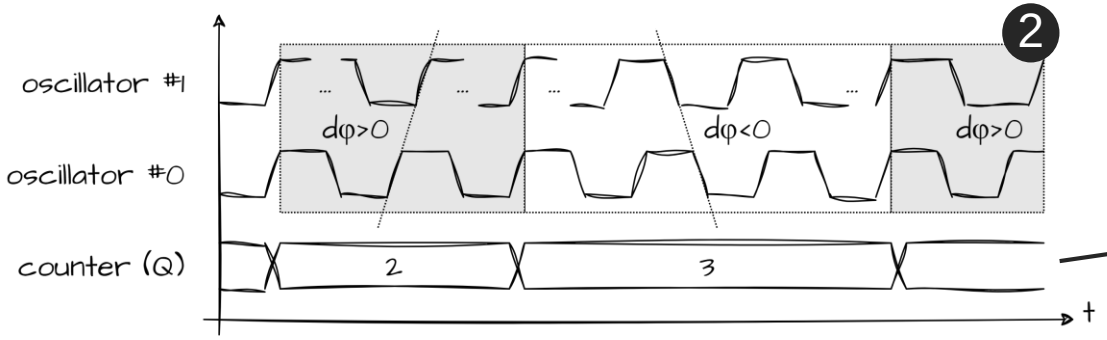
Emulator: Raw Random Number

- With two (or more) emulated RO we are able to
  - Emulate Raw Random Numbers (RRN)
  - With full timing behaviors (setup, hold…)

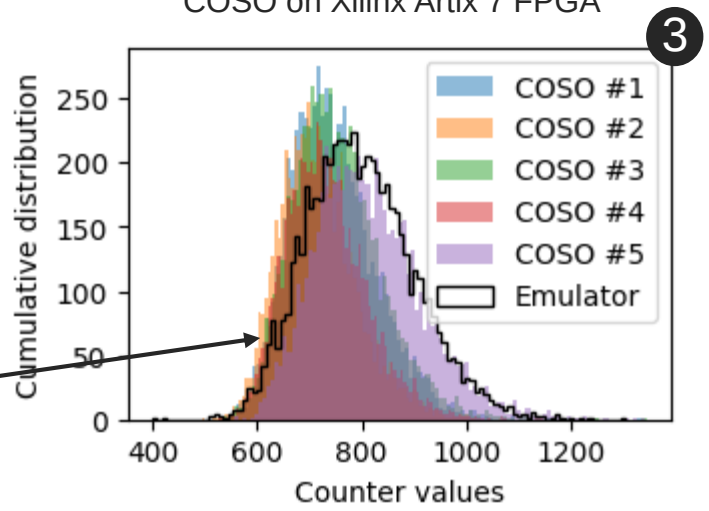



```
$ python ero.py –size 10e6 –freq0 100e6 –freq1 100e6 –div 500 ero.txt
$ python muro.py –size 10e6 –freq 100e6 98e6 99e6 101e6 –div 500 muro.txt
$ python coso.py –size 10e6 –freq0 167e6 –freq1 167e6 coso.txt
```

- As instance for the COSO [DTTIS24]



Emulated COSO vs. measured COSO on Xilinx Artix 7 FPGA



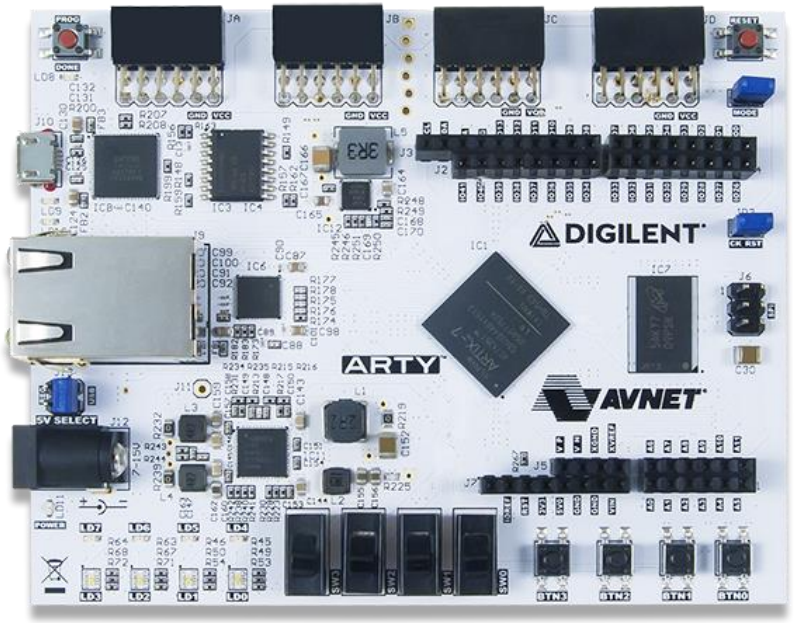
[DTTIS24] OpenTRNG: an open-source initiative for ring-oscillator based TRNGs, F. Pebay, L. Benea, M. Carmona, R. Wacquez

Implementation: supported targets

- **On the shelf support**
  - **FPGA**: Digilent Xilinx Arty A7
  - **ASIC**: FD-SOI 28nm (emulator only)

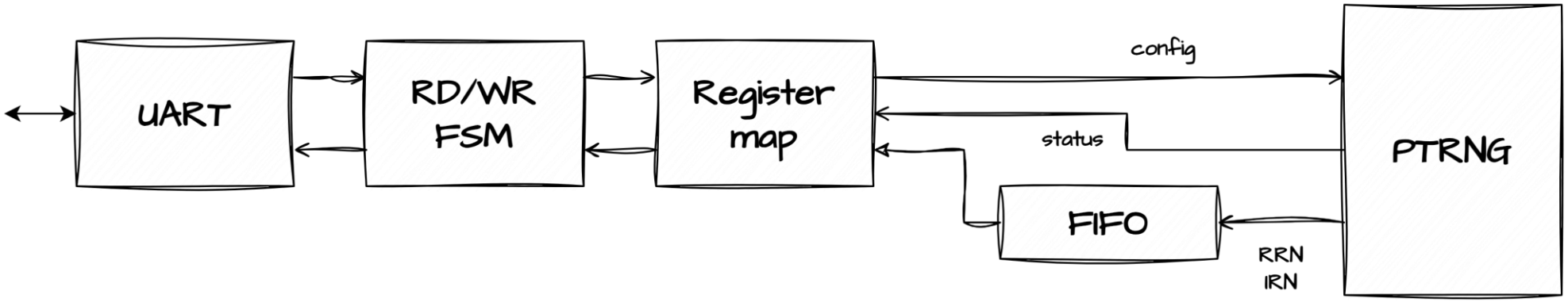
- **Future ports**
  - **FPGA**: other Xilinx devices, Intel, Microsemi, Lattice
  - **ASIC**: FD-SOI 22FDX

- Design is fully **portable**



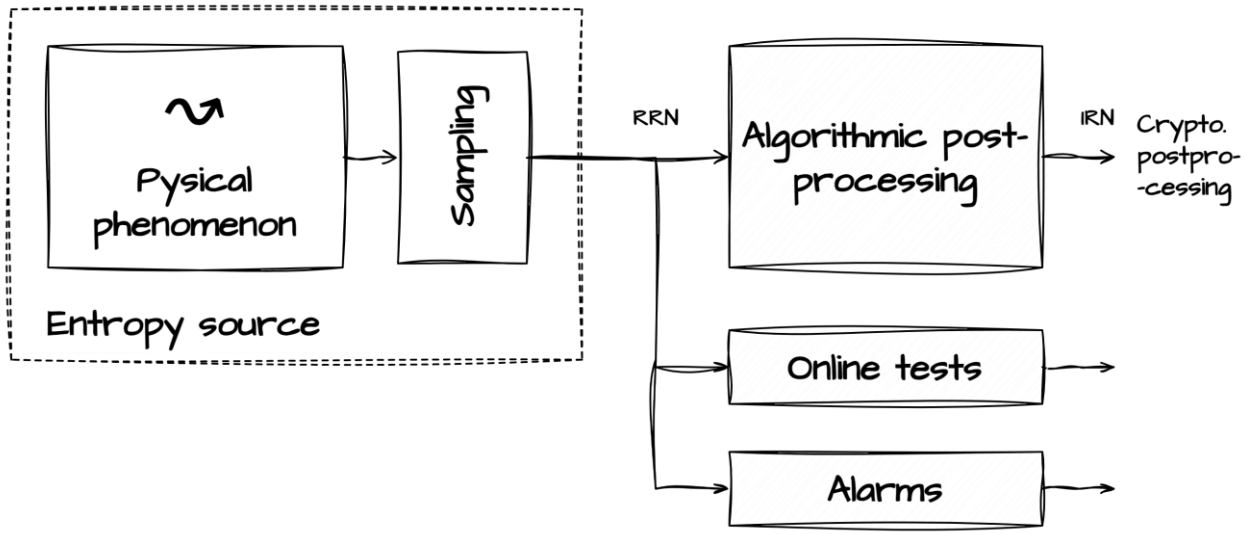
Implementation: global view

- **Direct access to TRNG** from PC over USB-UART
- HDL is currently VHDL, but System Verilog port is considered
- Python script read/write in register map
  - Set configuration registers
  - Get status registers
  - Get generated RRN/IRN



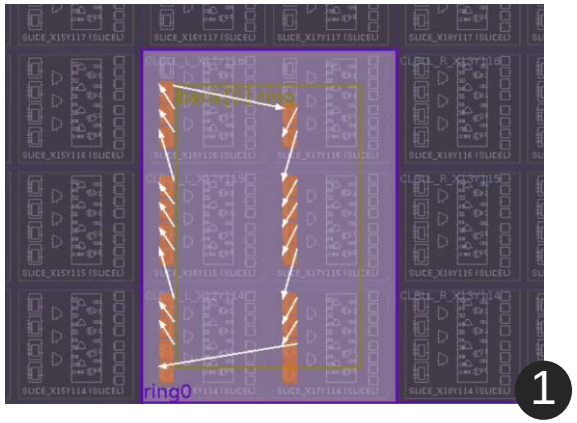
Implementation: TRNG

- **Entropy sources**:
  - ERO
  - MURO
  - COSO



- Algorithmic post-processing implements VN
- Online test and total failure alarm depend on source
- Fits with PTG.2 functionality class as defined in AIS 20/31

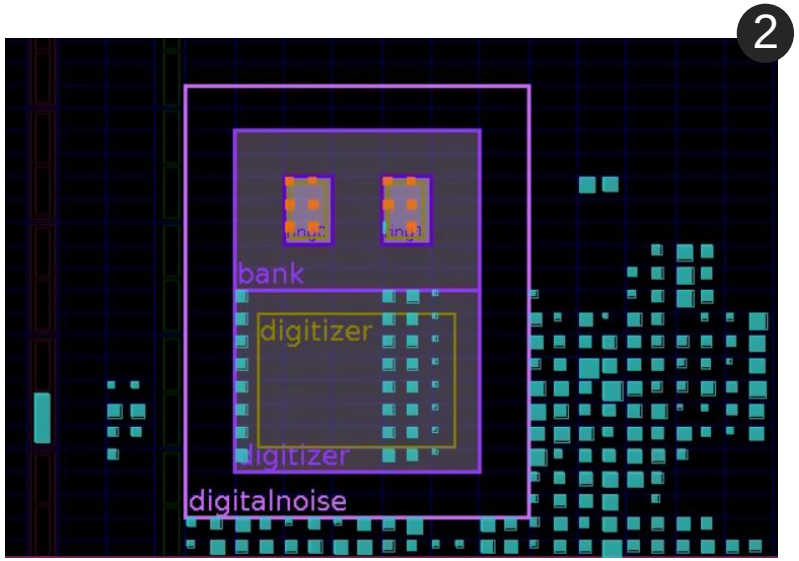
Implementation: automated tooling



RO with 20 elements automated place and route in Xilinx A7

- Auto-generated place and route constraints for RO
  - Takes as input number of elements and slice parameters
  - Deterministic and stable output for each P/R iteration

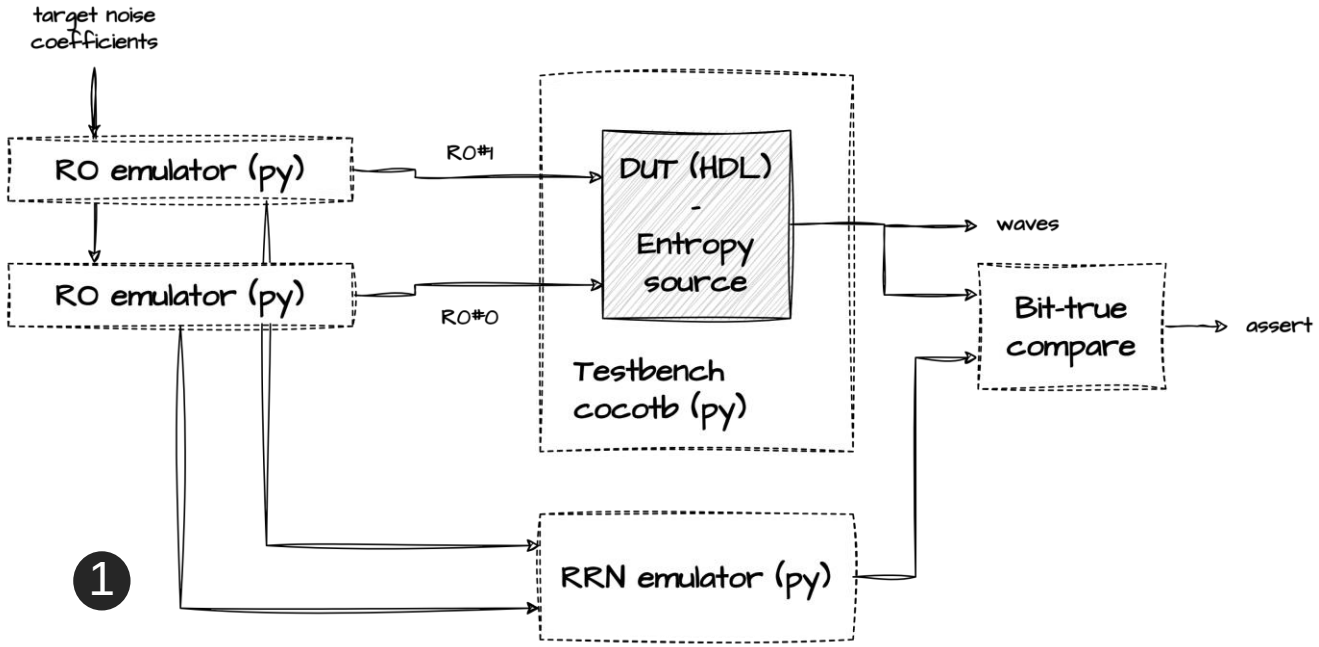
- Automated bloc placement
  - Takes as input source type and number of RO
  - Isolates each RO, avoid crosstalk between RO domains and other system clocks



RO and sampling blocs physical isolation in Xilinx A7

Validation tools: simulation & verification

- Language and simulator agnostic with cocotb (QuestaSim, GHDL…)
- Bit-true simulation and verification with RO and RRN emulators



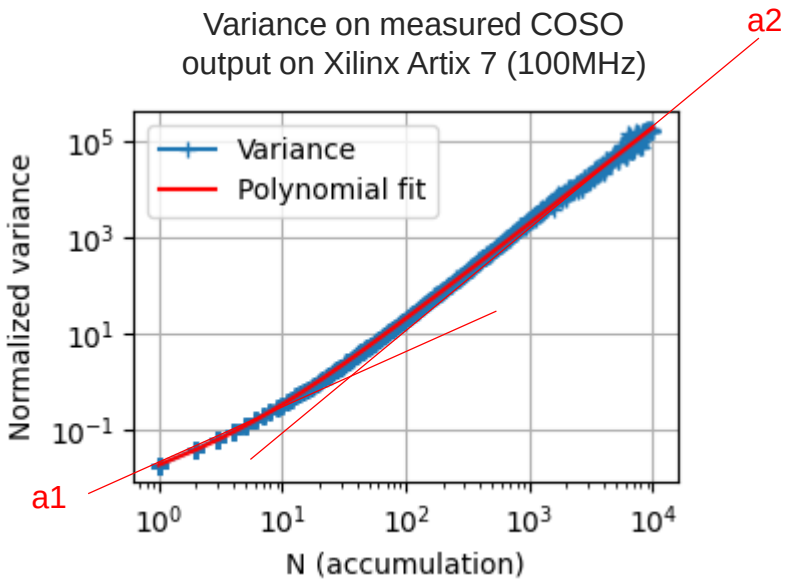
2

```
$ cd hardware/sim/test_coso
$ make
```

```
**************************************************************
** TEST                                STATUS  SIM TIME (ns) **
**************************************************************
** test_coso.test_total_failure_alarm   PASS       10010.00  **
** test_coso.test_gen_random_100        PASS       63706.73  **
**************************************************************
** TESTS=2 PASS=2 FAIL=0 SKIP=0                    73716.73  **
**************************************************************
```

Validation tools: noise figure extraction

- **COSO** helps to extract noise parameters
- Plot Allan variance of measured COSO output
- Polynomial fit gives amplitudes
  - a1: thermal noise
  - a2: flicker noise
- Coefficients are injected back in RO emulator for target specific accurate behavior

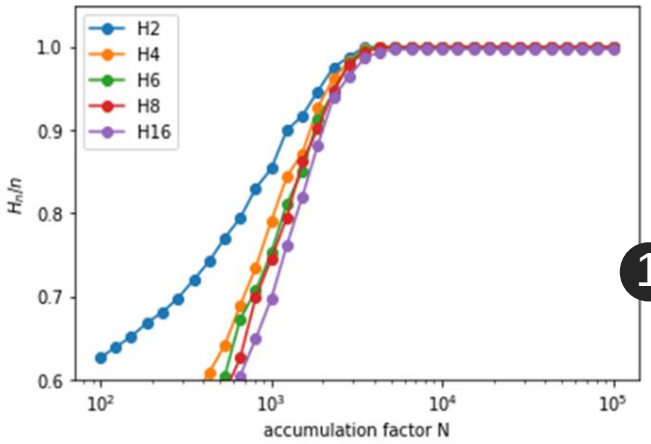


[DTTIS24] OpenTRNG: an open-source initiative for ring-oscillator based TRNGs, F. Pebay, L. Benea, M. Carmona, R. Wacquez

Validation tools: randomness KPI

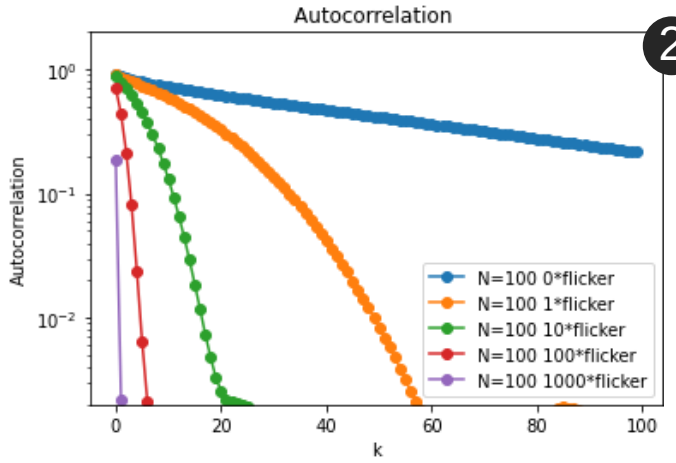
- Python scripts provide access to
- Entropy estimators
  - Shannon
  - Most Common Value (NIST SP800-90B)
  - Markov (NIST SP800-90B)
  - T8 as in AIS 20/31 procedure B
- Auto-correlations
  - Variable word size and lag

Entropy estimations for ERO measured data into Xilinx A7 vs accumulation time



1

Autocorrelations for emulated ERO with different flicker noise amplitude vs time

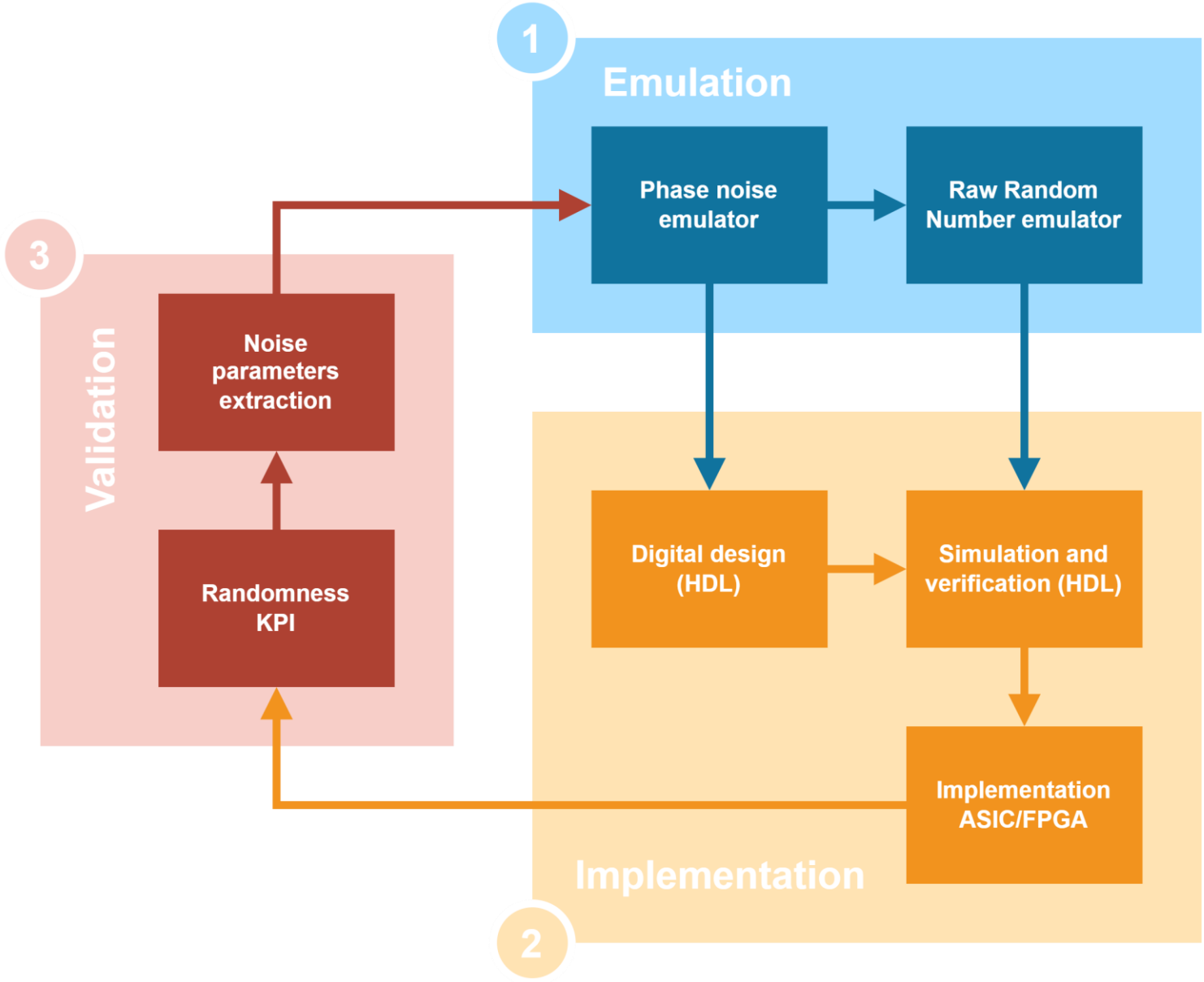


2

Development cycle

OpenTRNG helps in the TRNG development cycle

- **Key benefits**
  - Validate HDL with emulators
  - Validate entropy model
  - Measure and extract noise for your hardware target
  - Reduce time-to-product



Takeaway message

- **Open-source** framework for RO based TRNG
- Includes: **emulator, implementation** and **validation** tools
- Helps **industrials** to develop and validate TRNG in their products and academics in their **research** and **teaching**
- Opened to contributions ☺