
# True Random Number Generation
# TRNG

Jean-Luc Danger

December 2019


# Outline

- Introduction to TRNG
- Architectures
- Randomness
- Security
- Conclusion

# RNG Applications

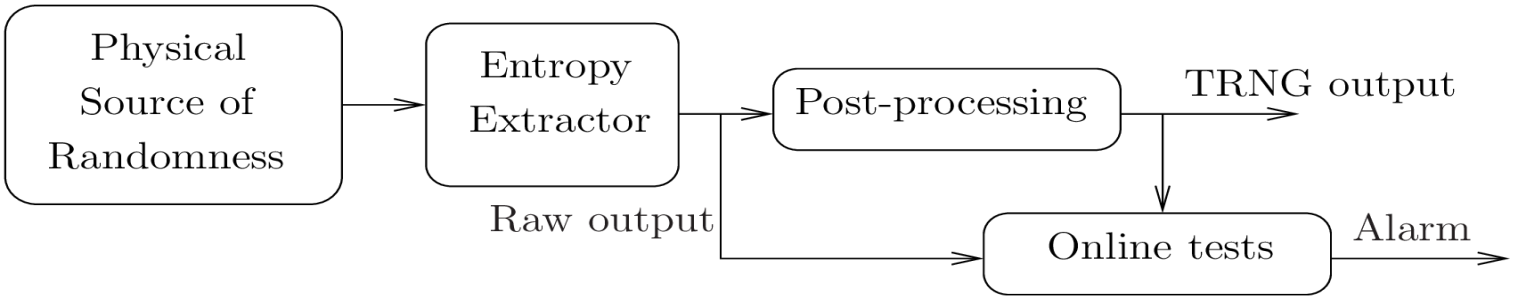
- **Cryptography**
  - Key generation
    - Symmetric/asymetric crypto
    - Stream ciphering (if deterministic RNG)
  - Initialization vector
    - For cryptographic operating modes
  - Authentication
    - Challenges
    - Nonce
  - Protection
    - Masks to thwart Side-Channel Attacks
- **System validation**
  - Monte-Carlo simulation
  - Statistical analysis
  - Channel emulation
- **Games, gambling**

# RNG types

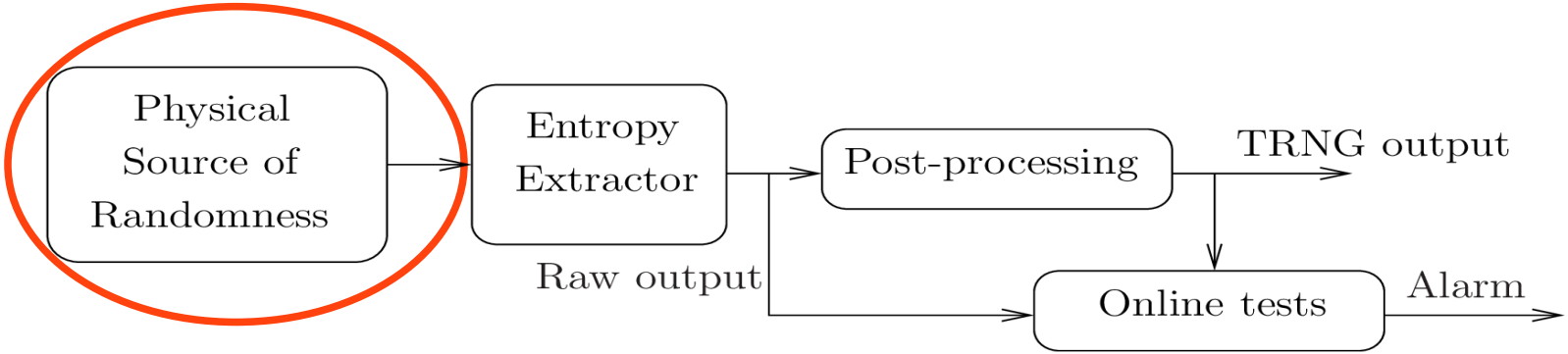
- **Pseudo Random number (PRNG)**
  - Deterministic
  - Fast
  - Periodic (based generally on LFSRs)
  - Adapted for stream ciphering
- **True Random Number generator (TRNG)**
  - Unpredictable
  - Use physical sources
- **Hybrid Random Number generator (HRNG)**
  - TRNG used as a seed of PRNG

# TRNG Architecture

- **2 main blocks**
  - Entropy Source
  - Entropy extractor
- **2 optional blocks**
  - Postprocessing
  - Embedded Tests



# Source of randomness

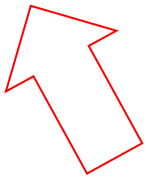



**Two types of randomness source:**

- Non Physical noise
  - Keyboard and mouse activity, Processor load, network data rate,…
- Physical noise
  - Noise in electronics circuits
  - Others: desintegration of radioactive atomic kernel,…

# Noise in electronics circuit

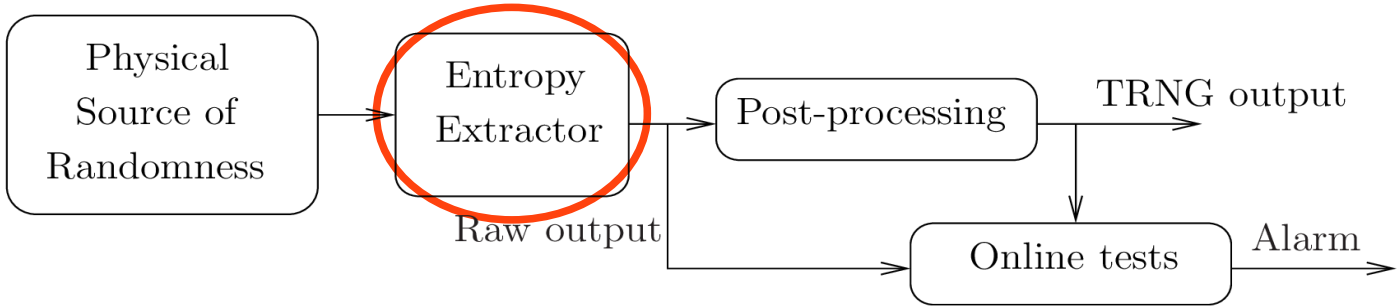
- **Noise = Sum of different phenomenon:**
  - Thermal noise
  - 1/F noise
  - Shot noise
  - Popcorn noise
  - Crosstalk
  - Interference



Can be source of attacks !

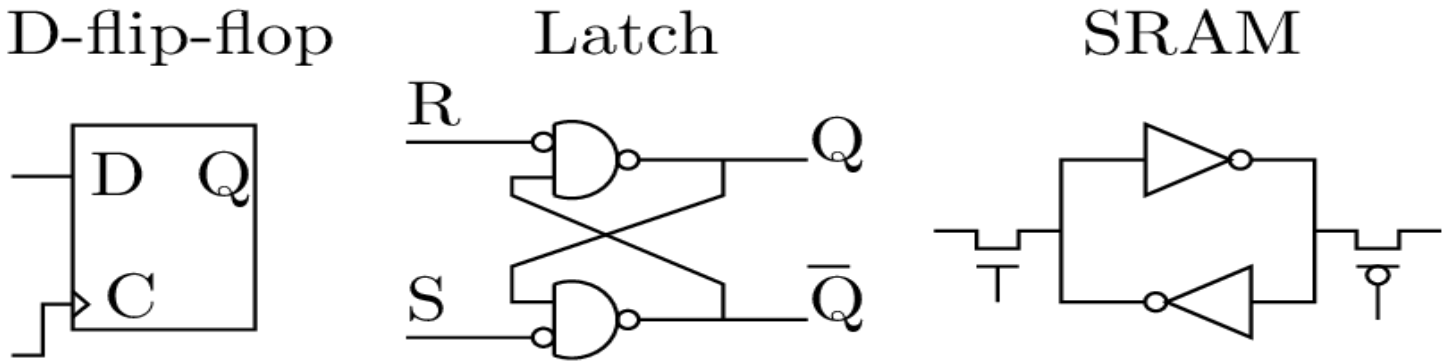
# Randomness extraction

- **To sample the noise at source level**
  - Use of a sample/hold circuit, or memory element



# Randomness extraction

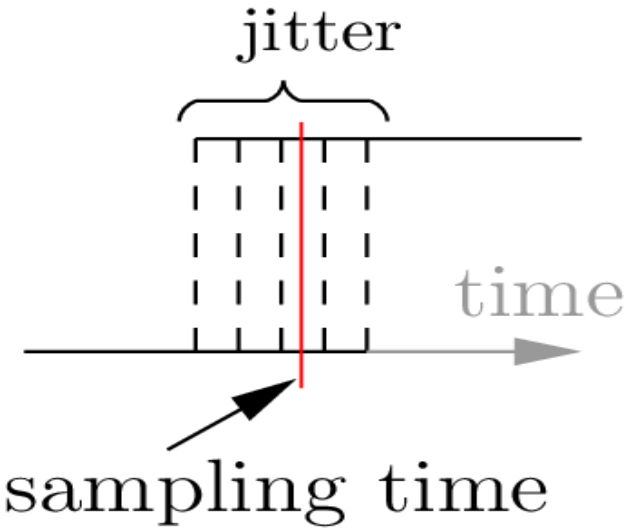
- **Memory elements**




- **Noise Axis**
  - Horizontal: **phase** noise (frequency) or **jitter** (time)
  - Vertical: **amplitude** noise

# Phase noise

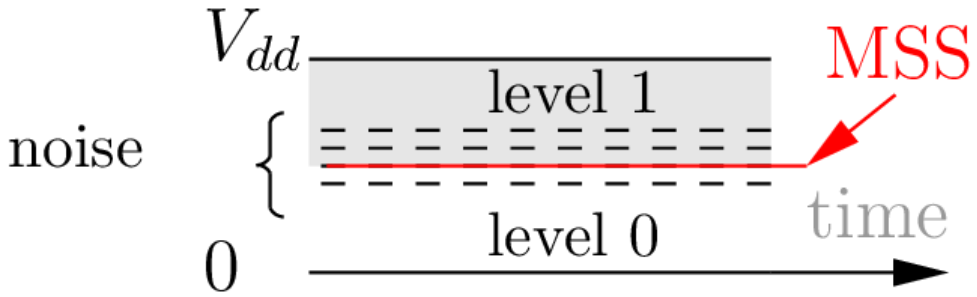
- **Combination of random jitter and deterministic jitter (unwanted)**



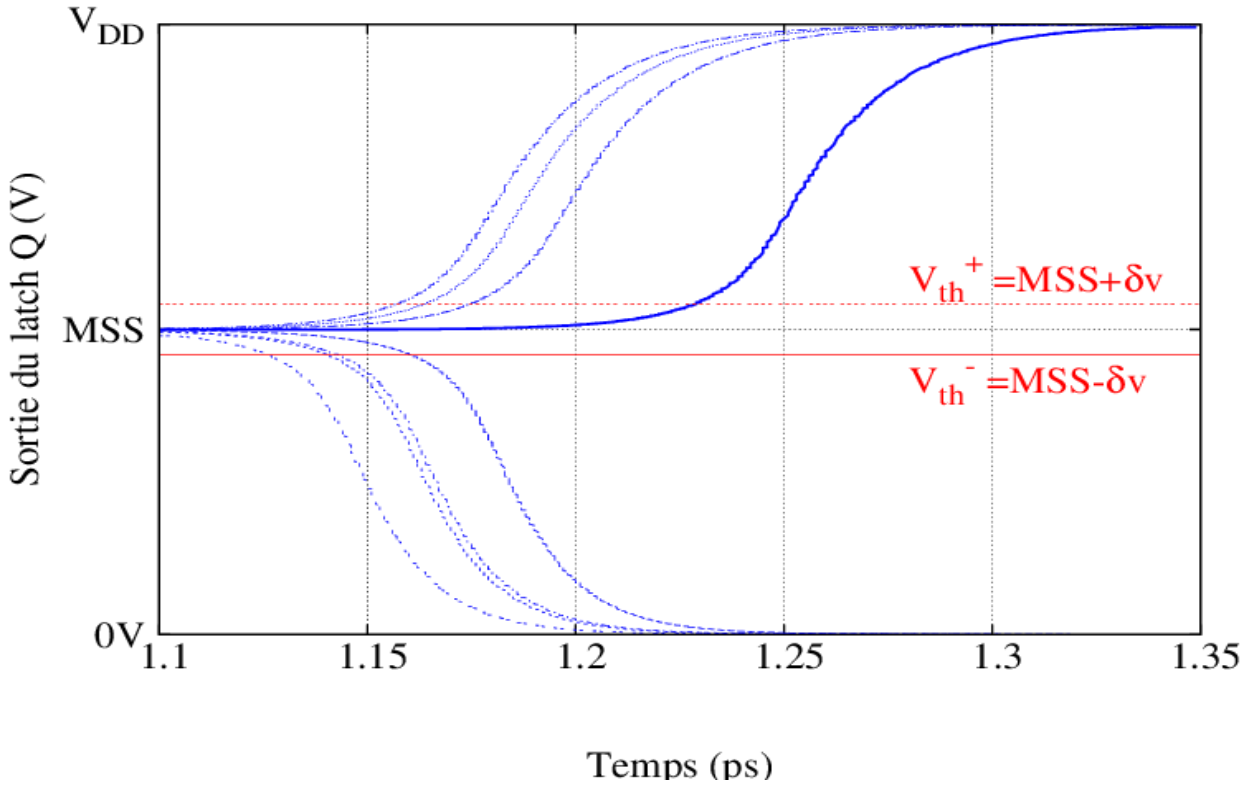
# Amplitude noise

**Also called vertical noise**

The signal is at the boundary of 0 and 1 level, around the "**metastable** state" MSS (~Vdd/2)



# Metastability phenomenon



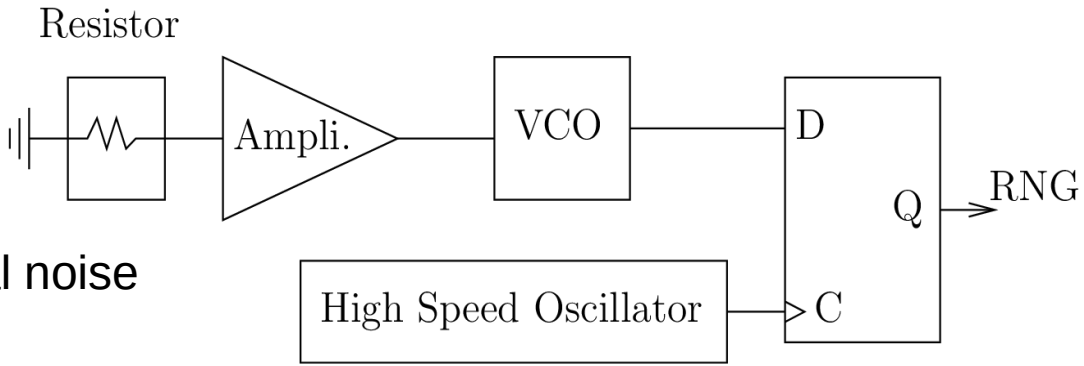
# TRNG types

- **Jitter-based TRNG**
  - One jittery clock samples another jittery clock
- **Metastability-based TRNG**
  - The bi-stable is placed in the MSS state, its output depends on the vertical noise
- **Mixed TRNG**
  - Exploits both phenomemon

# Outline

- Introduction to TRNG
- Architectures
- Randomness
- Security
- Conclusion

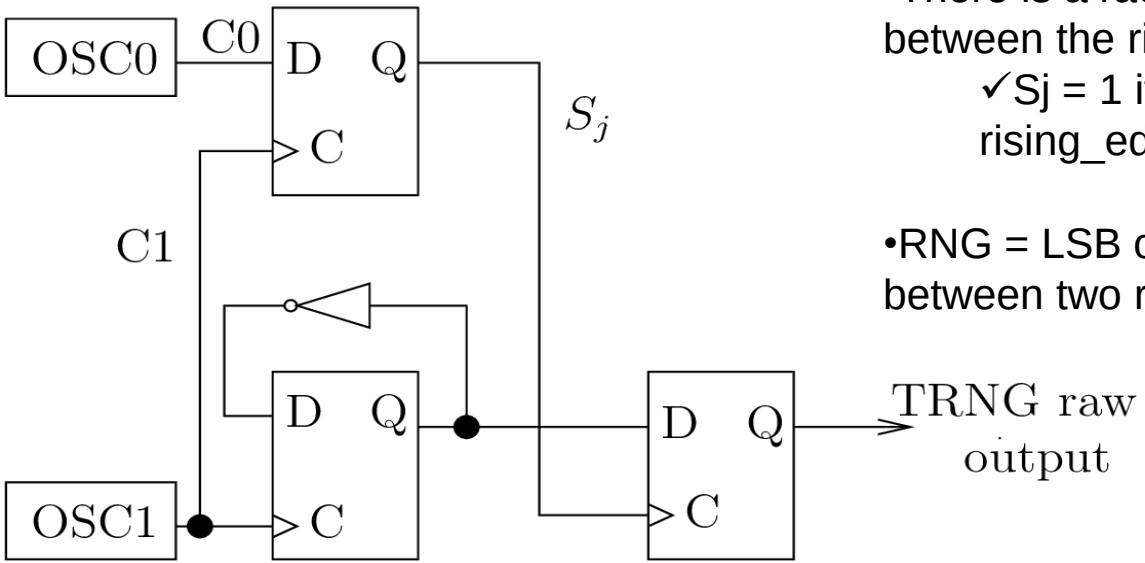
# Two ring oscillators
# Intel first generation




Thermal noise

Benjamin Jun and Paul Kocher. The Intel Random Number Generator, 1999. http://www.cryptography.com/intelRNG.pdf.

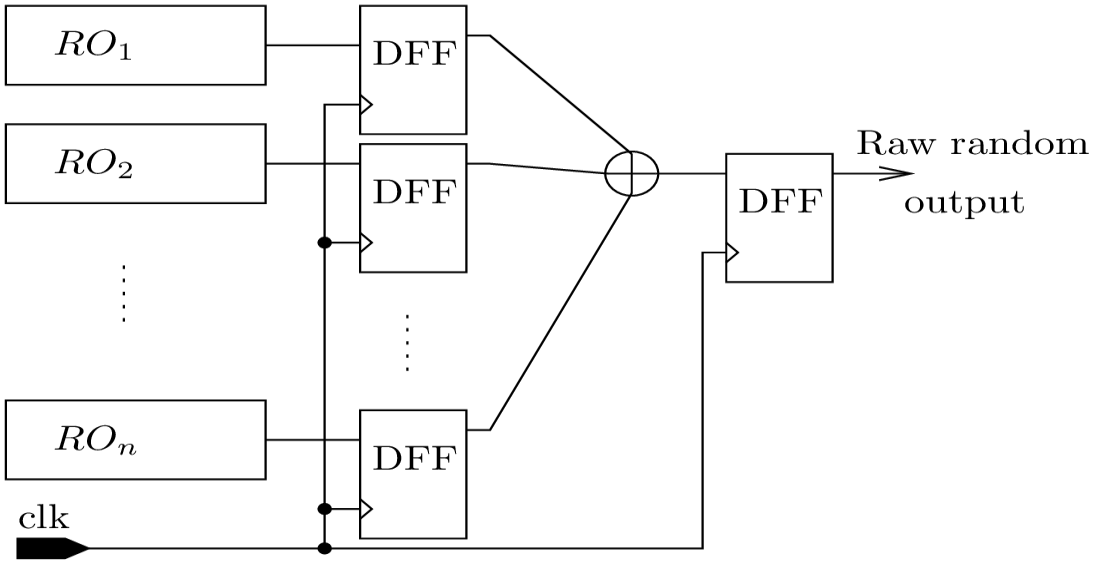
# Two-Ring oscillator TRNG




- There is a race (which depends on the jitter) between the rising edges:
  - ✓Sj = 1 if rising_edge(C0) is ahead of rising_edge(C1), else 0
- RNG = LSB of the number of rising_edge(C1) between two rising_edge(C0)

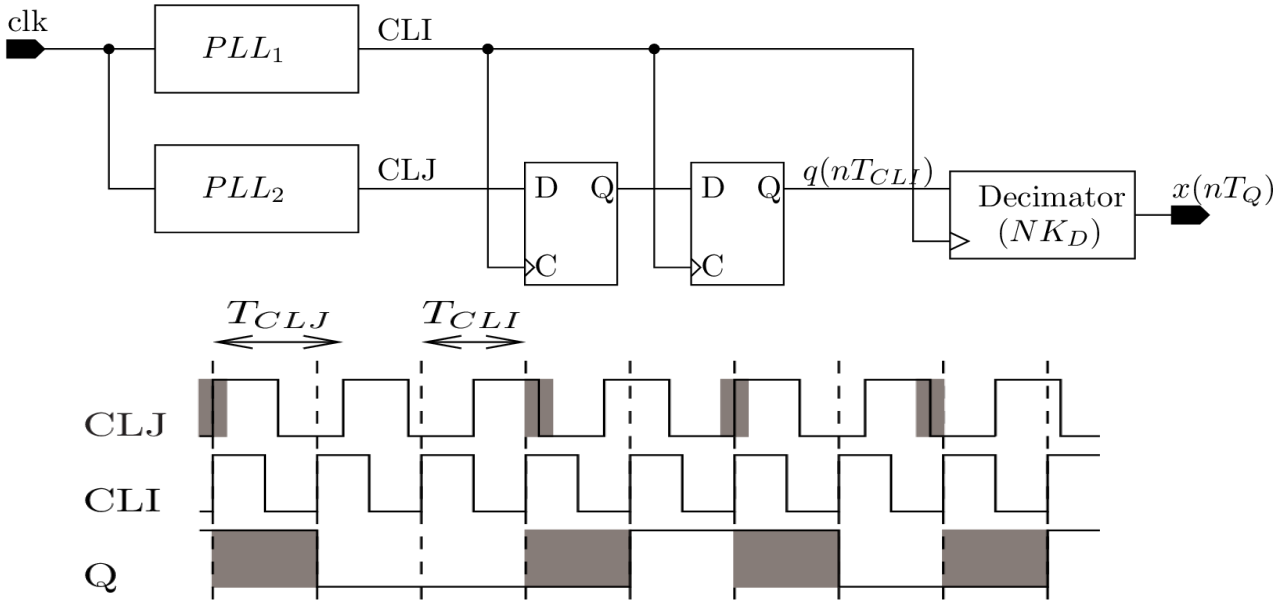
Paul Kohlbrenner and Kris Gaj. An embedded true random number generator for FPGAs. In Proceedings of the 2004 ACM/SIGDA 12th international symposium on Field programmable gate arrays, 2004.

# Multiple ROs TRNG



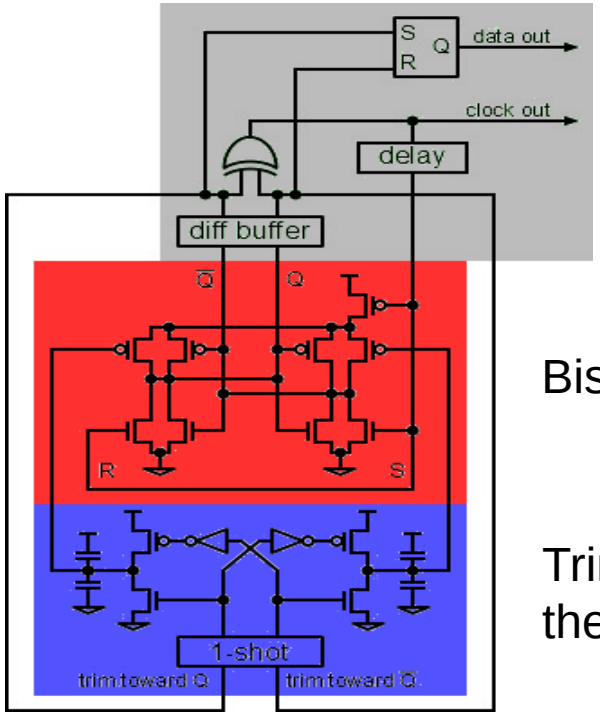

Knut Wold and Chik How Tan. Analysis and enhancement of random number generator in fpga based on oscillator rings. In ReConFig, pages 385–390, 2008.

# PLL-based TRNG




Viktor Fischer, Milos Drutarovský, Martin Simka, and Nathalie Bochard. High performance true random number generator in altera stratix fplds. In Field Programmable Logic and Application, volume 3203 of Lecture Notes in Computer Science, pages 555–564. Springer 2004.

# Intel Ivy bridge TRNG



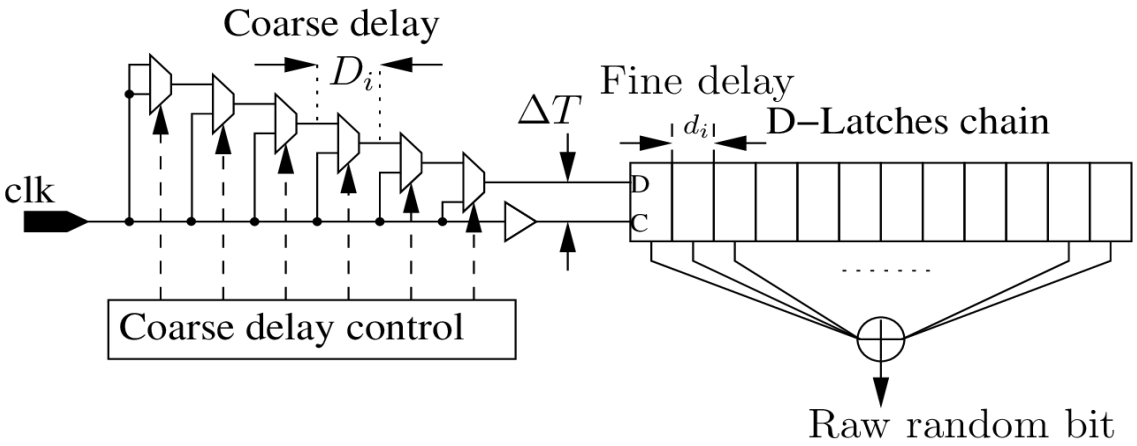

Bistable cell

Trimming circuit to keep the metastable state

Mike Hamburg, Paul Kocher, and Mark E. Marson. Analysis of intel's ivy bridge digital random number generator, March, 12 2012.

# Open Loop TRNG

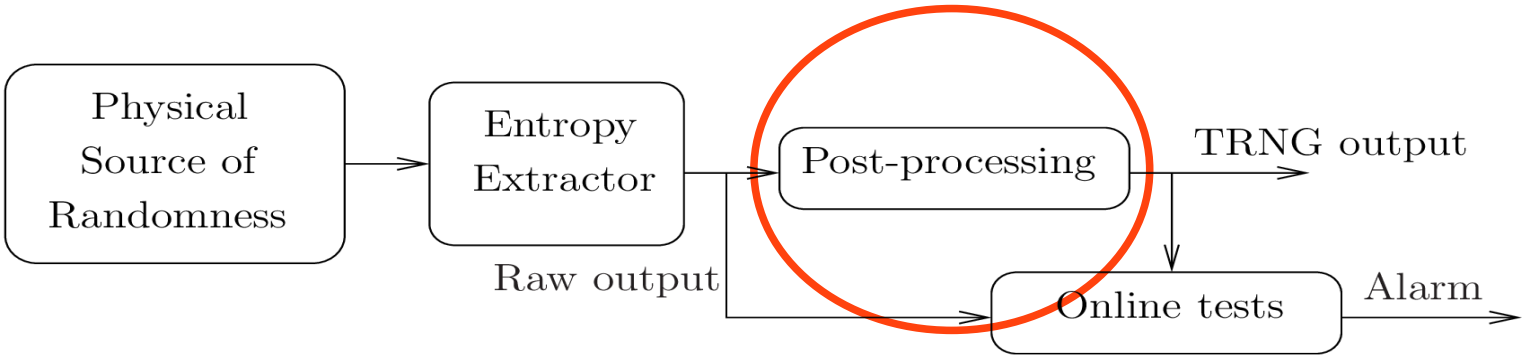
- **A chain of latches with D = C**




Danger, J. L., Guilley, S., & Hoogvorst, P. (2009). High speed true random number generator based on open loop structures in FPGAs. *Microelectronics journal*, *40*(11), 1650-1656.

# Postprocessing

- **Mandatory process:**
  - The source of entropy is biased by the environment
  - The extraction block is not efficient enough
  - The samples are correlated



# Post processing for better entropy

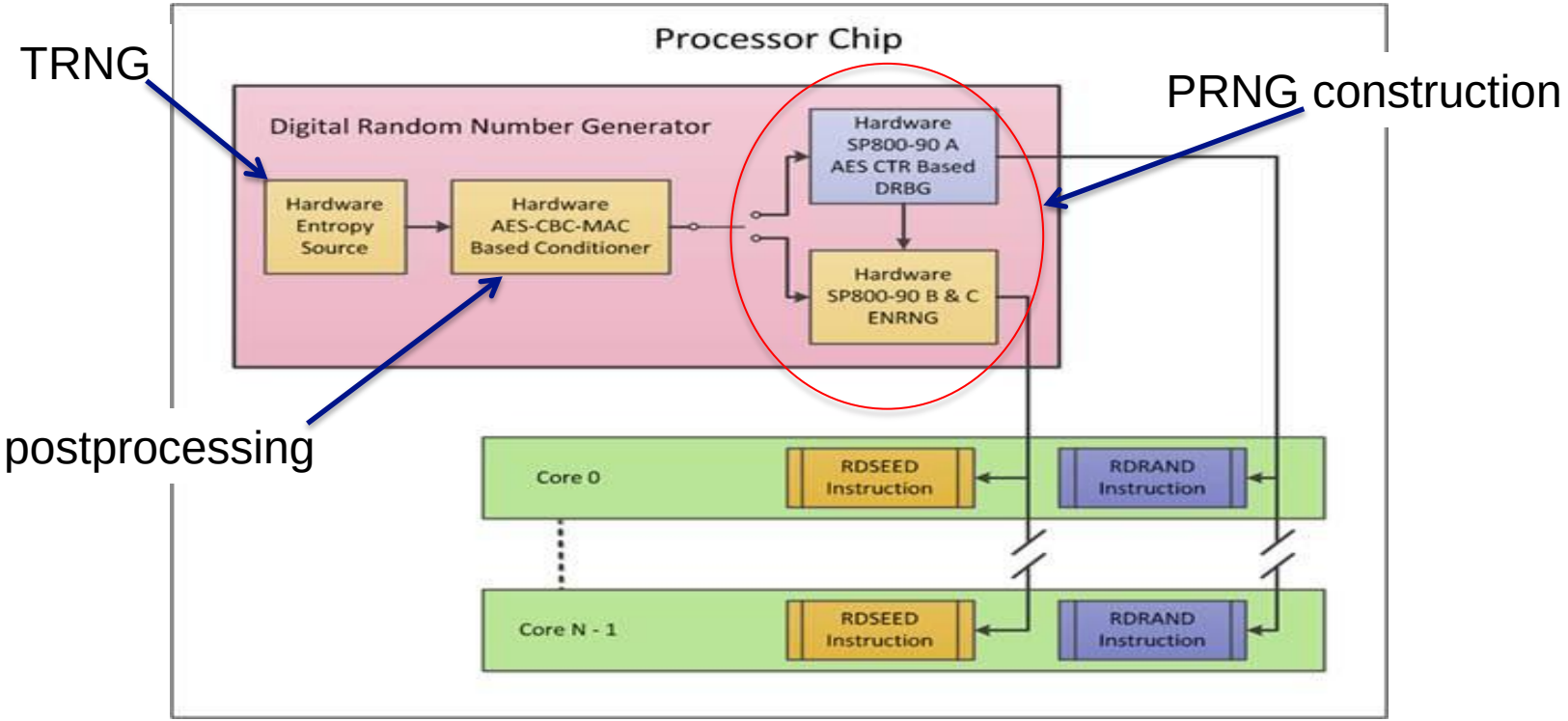
- **Goal: To increase the entropy**
  - by reducing the Bit rate
  - Or increasing data compression
- **Classical methods:**
  - XOR corrector
    - N bits are XORed to get one output bit
    - Constant Bit rate reduction (N)
    - XOR construction proposed by M. Dichtl*
  - Von Neumann corrector
    - 2 identical bits: freeze
    - 2 different bits: 1 if "10" , 0 if "01"
    - Variable Bit rate reduction (at least 4)

* Dichtl, M. (2007, January). Bad and good ways of post-processing biased physical random numbers. In Fast Software Encryption (pp. 137-152). Springer Berlin Heidelberg.

# Cryptographic Postprocessing

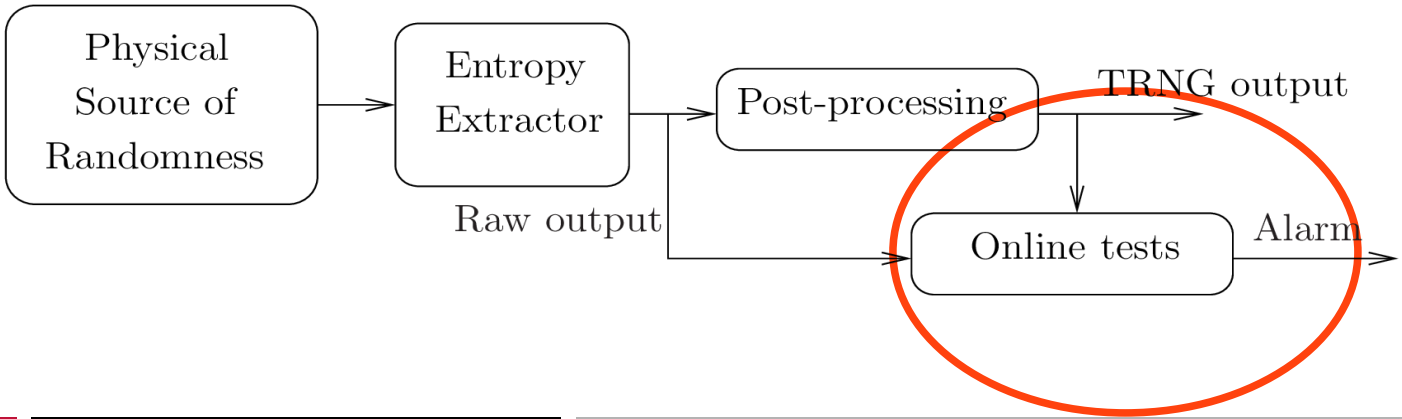
- **Goal: To increase unpredictibility**
  - Exploits non-linearity properties
  - The entropy is the same
- **Hash function**
  - Unpredictability ensures by One-way function
- **Symmetric cryptography cipher**
  - Could be the same as the crypto block.

# Intel TRNG postprocessing



# On-Line tests

- **Embedded tests = health tests**
  - To ensure dynamically that the output bits have good randomness property :
    - Under environmental variations
    - Under attacks



# Example of on-line tests

- **Custom sanity check**

Intel's Ivy bridge

| Bit pattern | Allowable number of occurrences per 256-bit sample |
|---|---|
| 1 | $109 < n < 165$ |
| 01 | $46 < n < 84$ |
| 010 | $8 < n < 58$ |
| 0110 | $2 < n < 35$ |
| 101 | $8 < n < 58$ |
| 1001 | $2 < n < 35$ |

- **Use off-line statistical tests (described in next section)**
  - Can be costly => partial tests
- **2nd order effect:**
  - the noise generated by the test impacts the TRNG which passes the test more easily

# Outline

- Introduction to TRNG
- Architectures
- Randomness
- Security
- Conclusion

# Security evaluation requirements

- **In theory:**
  - Randomness: Entropy
    - Shannon
    - Min-Entropy
  - Unpredictability: Conditionnal Entropy
    - Stochastic process
      - H( state_{i} | state_{i-1} ) ?
- **In practice**
  - Assesment Methods
    - Statistical tests
    - Build a stochastic model

$$H(x) = -\sum p(x) \log_2 (p(x))$$

$$H(x) = -\log_2 \max(p_i(x))$$

*x is a bit vector*

# Statistical tests

- **Off-line tests**
  - FIPS140-2
  - NIST SP800-22
  - DIEHARD
  - AIS31
- **On-line tests**
  - To detect in real time
    - failures abnormal behaviour
    - Health tests
  - partial statistical tests

# FIPS140-2 tests

- **4 tests applied on 20000-bit sequences**
- **Can be embedded in the circuit:**
  - Monobit (pr(bit=0))
  - Poker : uniform distribution of 4-bit groups
  - Runs : check the number of run sequences of ‘0’ and ‘1’ of length between 1 and 6
  - Long runs : no run of ‘0’ or ‘1’ with a length equal or greater than 26 should occur

NIST FIPS (Federal Information Processing Standards). Security Requirements for Cryptographic Modules publication 140-2, May 25 2001. http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf.

# NIST test suite SP800-22

- **List of 15 demanding tests, some requiring 1Mbit of data or up to 1Gbit**
- **Applies to random number generators for cryptographic applications**
  - Usually unfit for testing raw physical entropy sources
  - Fit for testing post-processed or “ready-to-use” outputs

Andrew Rukhin, Juan Soto, James Nechvatal, Miles Smid, Elaine Barker, Stefan Leigh, Mark Levenson, Mark Vangel, David Banks, Alan Heckert, James Dray, and San Vo. A statistical test suite for the validation of random number generators and pseudo random number generators for cryptographic applications, april 2010.

# NIST test suite SP800-22

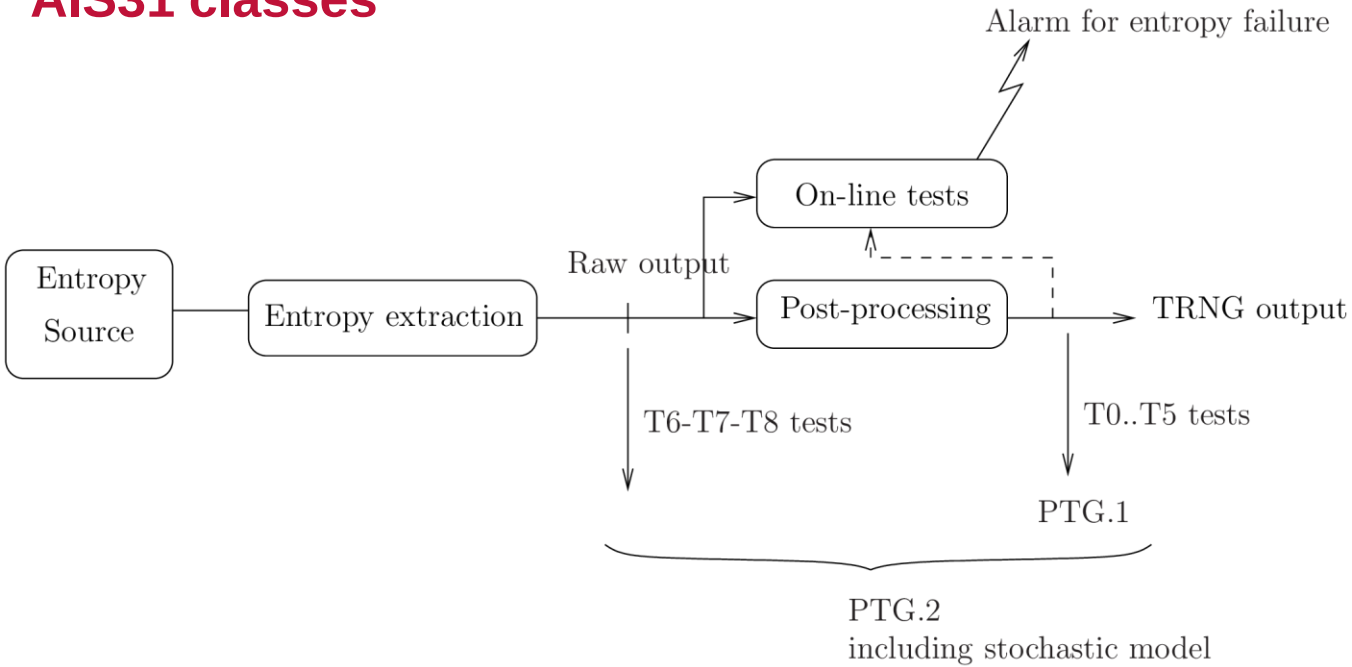
- **NIST test suite (SP800-22)**
  - Among them:
    - Similar to what was first included in FIPS140-2 +
    - Random Binary Matrix Rank : check for linear dependency among fixed substrings
    - Check for periodicity (DFT, template matching tests (2 variants))
    - Compression test (Maurer's "universal statistical" test, similar to a Lempel-Ziv compression, LFSR)
    - Random walk (Cumulative sum, random excursion tests (2 variants))

# BSI AIS31

Bundesamt
für Sicherheit in der
Informationstechnik

- **Strongly common criteria oriented**
- **New approach: testing TRNG output as well as the randomness source**
  - 9 statistical tests
  - Recommends using on-line tests,
    - Entropy source failure tests
    - Statistical defect tests on the raw random sequence
    - Statistical defect tests on the post-processed sequence
  - Defines TRNG classes (PTG.x)

# AIS31 classes




- PTG.1 represents the lower-end class and PTG.2 and PTG.3 the higher-end classes
- PTG.2 adds minimum entropy requirements for post-processed bits and introduces raw data online tests and entropy source stochastic model
- PTG.3 requires cryptographic post-processing (class DRG.3 or DRG.4)

# AIS31 tests

- **Test T0**
  - disjointness test
    - $2^{16}$ 48-bit blocks must be different
    - Can be performed on line
    - Rejection probability $10^{-17}$
- **Tests T1-T4**
  - Idem FIPS140-2 with Rejection probability $10^{-6}$
- **Test T5:autocorrelation test**
    - Performed on 10000 bits Between the sequence of 5000 bits and the 5000 bits shifted sequence
- **Tests T6-8**
  - Dedicated to raw sequences:
    - Multinomial distribution test
    - Homogeneity test
    - Coron's entropy test

# AIS31 Randomness Model

- **AIS-31 Stochastic process**
  - The conditionnal entropy define the dependence between the last state Rn and the previous states

$$H(R_n | R_{1,} \dots R_{n-1}) = -\sum p(R_n | R_{1,} \dots R_{n-1}) \log_2 p(R_n | R_{1,} \dots R_{n-1})$$

  - The process should be ideally stationnary

# NIST tests SP-800 90A, 90B and 90C

- **SP-800 90A considers DRNG**
- **SP-800 90B considers entropy source design and validation**
  - SP-800 90B defines requirements similar to BSI's AIS31
  - Two categories defined:
    - Entropy source
    - Full entropy source
  - Minimal online health tests defined
    - Lightweight runs detection test (repetition count test)
    - Lightweight uniformity test (adaptive proportion test)
- **SP-800 90C considers design of hybrid RNGs using SP-800 90B + SP-800 90A**

# ISO standard

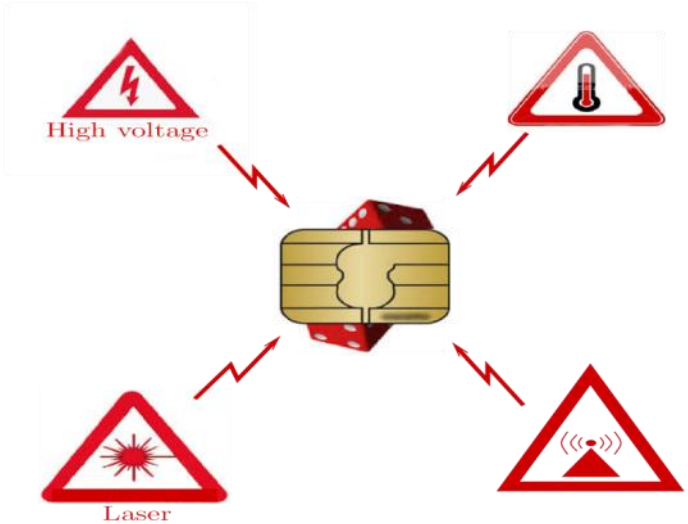
- **ISO /IEC 20543**
  - Test and analysis methods for random bit generators
    - Define RNG classes (example: deterministic, non-deterministic, hybrid) within ISO/IEC 19790 and 15408
    - Define evaluation methods for the various RNG classes

# Outline

- Introduction to TRNG
- Architectures
- Randomness
- Security
- Conclusion

# TRNG threats

- **Malevolent Actions**
  - To decrease the entropy by introducing biases
  - To bypass on line tests
  - To lock the TRNG
  - To locate the TRNG
    - To attack it afterward

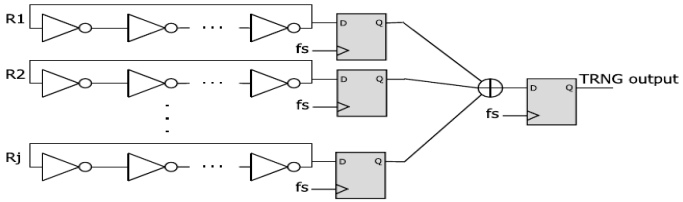


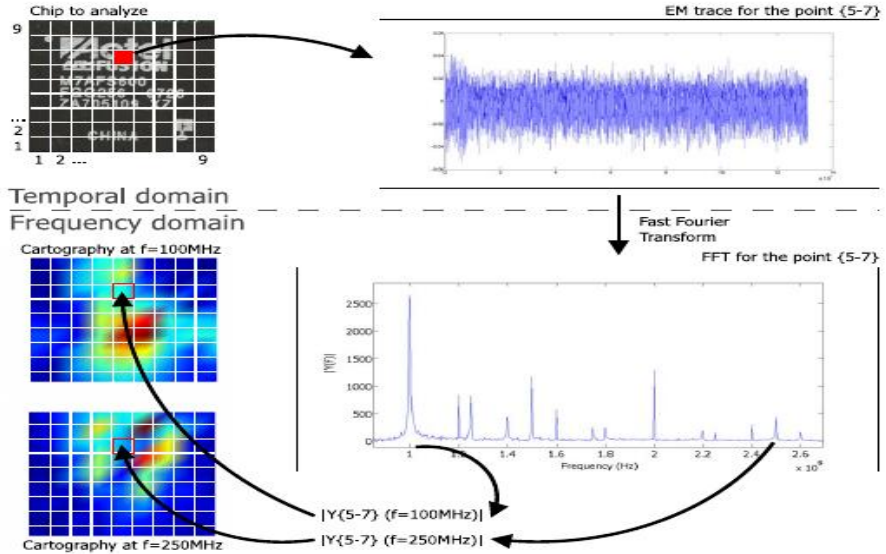
# TRNG attack types

- **Passive attacks**
  - Observation of the TRNG activity
    - Frequency  of RO TRNG
    - Register loads
  - Allows the attacker **to locate** the TRNG
- **Reverse Engineering**
  - Allows the attacker **to locate** the TRNG
- **Active attacks**
  - **Bias** generated by fault injection
  - **Force** the TRNG output or the on-line test alarm
- **Trojans**
  - Bias generated at design or manufacturing stage

# Passive attacks

- **Observing the EM field in RO TRNG**
  - Allows to locate the RO

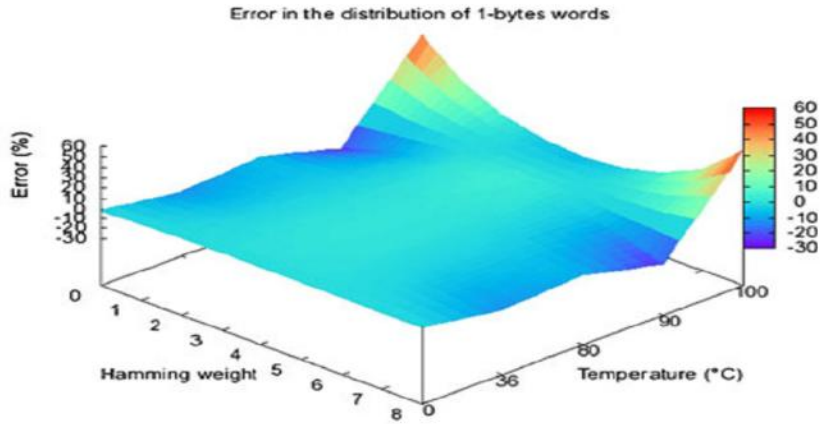



Bayon, P., Bossuet, L., Aubert, A., & Fischer, V. (2013, May). Electromagnetic analysis on ring oscillator-based true random number generators. In Circuits and Systems (ISCAS),

# Active attacks on RO TRNG

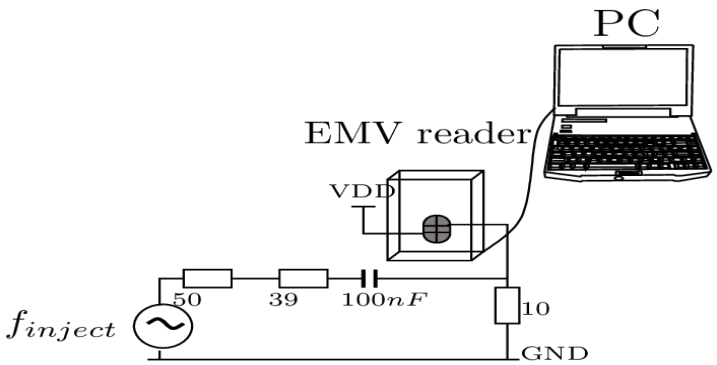
- **Different sources:**
  - T°C  and laser: very sensitive
  - Xray and Radioactivity: not sensitive

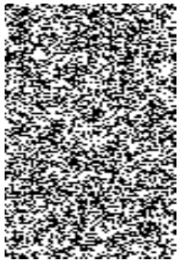



Soucarros, M., Clédière, J., Dumas, C., & Elbaz-Vincent, P. (2013). Fault analysis and evaluation of a true random number generator embedded in a processor. Journal of Electronic Testing, 29(3), 367-381.

# Active attacks on RO TRNG

- **Frequency coupling by the Power Supply**
  - Performed successfully on logic gates, microcontroller and Smartcard (EMV)





No injection

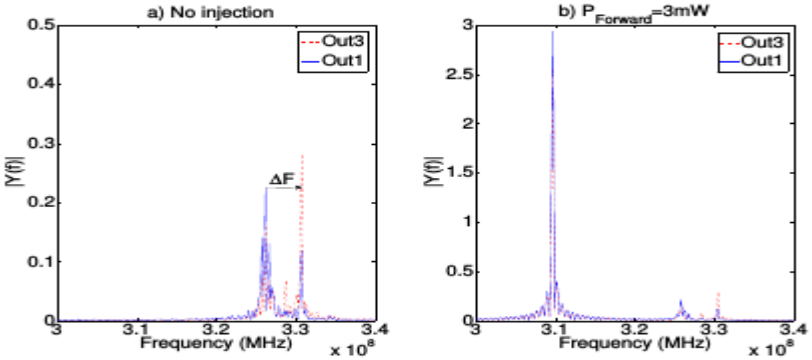

1.822880 MHz injection

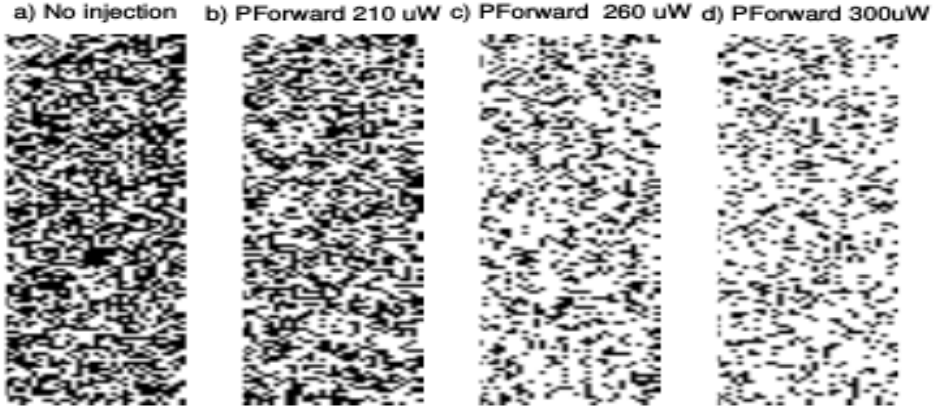

1.929629 MHz injection

Markettos, A. T., & Moore, S. W. (2009). The frequency injection attack on ring-oscillator-based true random number generators. In Cryptographic Hardware and Embedded Systems-CHES 2009 (pp. 317-331). Springer Berlin Heidelberg.

# Active attacks on RO TRNG

- **Frequency coupling by EM field**

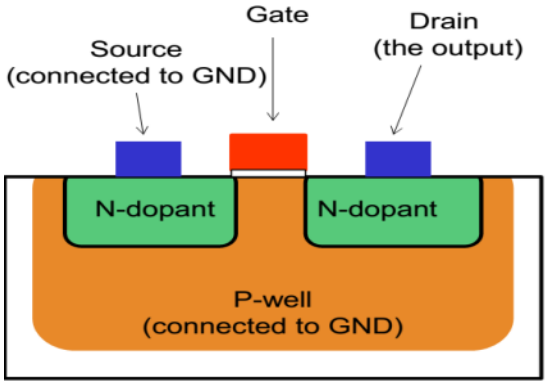


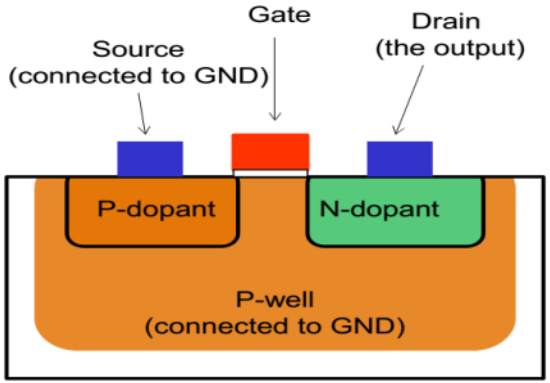

Bayon, P., Bossuet, L., Aubert, A., Fischer, V., Poucheret, F., Robisson, B., & Maurine, P. (2012). Contactless electromagnetic active attack on ring oscillator based true random number generator. In Constructive Side-Channel Analysis and Secure Design (pp. 151-166). Springer Berlin Heidelberg.

# Attack by Hardware Trojan Horse

- **The Dopant of transitors is changed**
  - Extremely difficult to detect optically
  - Can drastically reduced the TRNG entropy




NMOS transistor

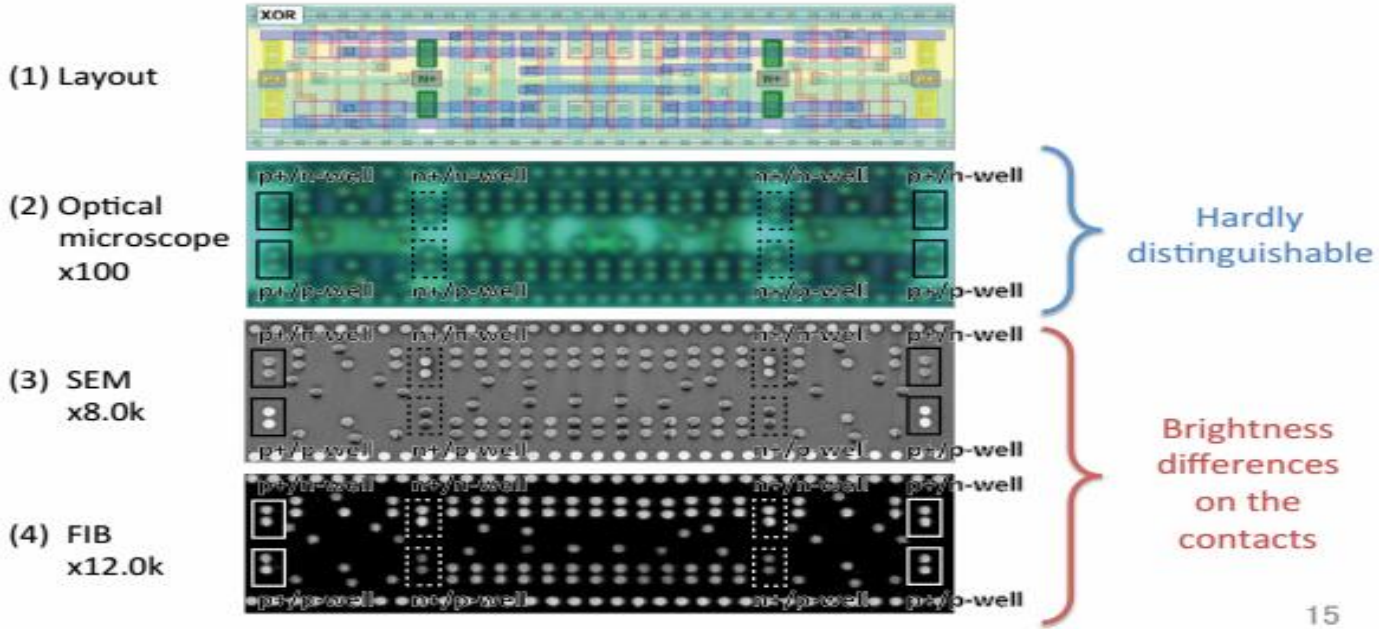



Dopant Change => NMOS transistor with floating output

Becker, G. T., Regazzoni, F., Paar, C., & Burleson, W. P. (2013). Stealthy dopant-level hardware trojans. In *Cryptographic Hardware and Embedded Systems-CHES 2013* (pp. 197-214). Springer Berlin Heidelberg.

# Detection of Stealthy Trojan

- Stealthy dopant-level circuits are measurable




Sugawara, T., Suzuki, D., Fujii, R., Tawa, S., Hori, R., Shiozaki, M., & Fujino, T. (2014). Reversing Stealthy Dopant-Level Circuits. In *Cryptographic Hardware and Embedded Systems–CHES 2014* (pp. 112-126). Springer Berlin Heidelberg.

# Outline

- Introduction to TRNG
- Architectures
- Randomness
- Security
- Conclusion

# Conclusion 1/2

- **Necessity to assess Entropy and Unpredictability**
  - Statistical tests
  - Model
- **Necessity to embed Online test**
  - To detect abnormal behaviour
    - Environment
    - Attacks
  - Future ISO standards in preparation
- **Designing a TRNG is still a challenge**
  - Security
    - To pass statistical tests
    - To build a model
    - To be robust against attacks
  - Design challenges

# Conclusion 2/2

- **Design challenges**
  - Type of randomness extraction
  - Type of Postprocessing
  - Complexity
  - Design automation
  - Bit rate
  - Testability (statistical and integrity tests)
  - Power consumption

**=> A test chip is recommended !**