SIEMENS

Corporate Technology

FPGA Implementations of Physical Random Number Generators Using Logic Gates Only

Markus Dichtl

CT IC 3

Why do we need random numbers?

Many cryptographic protocols use random elements.

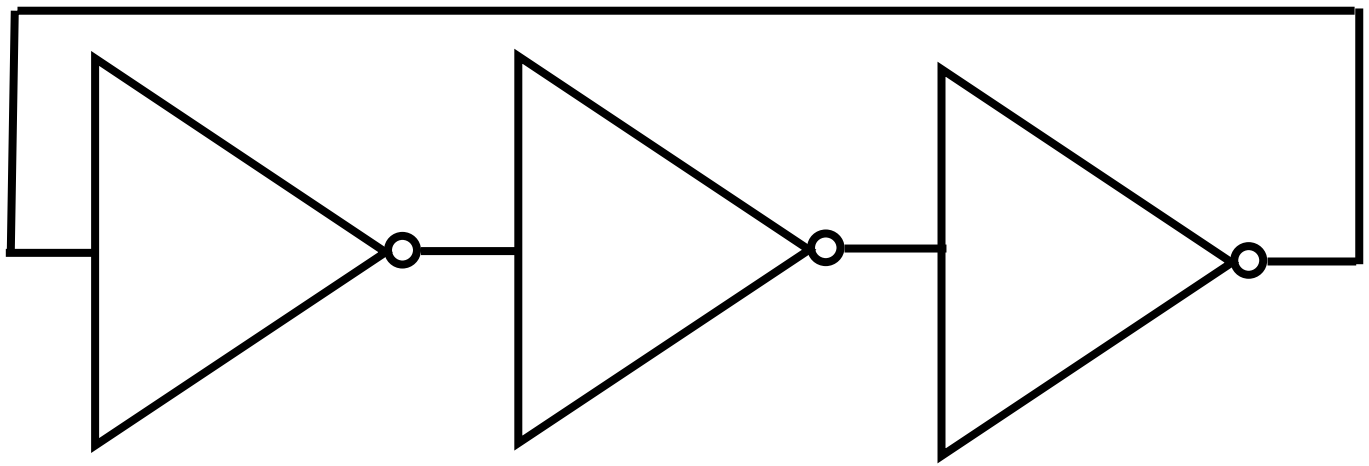
Examples:

- Key generation
- Nonces without memory
- ElGamal-procedures
- Randomized implementations against side channel attacks

Experimental environment

All my experiments used Xilinx Spartan 3 FPGAs

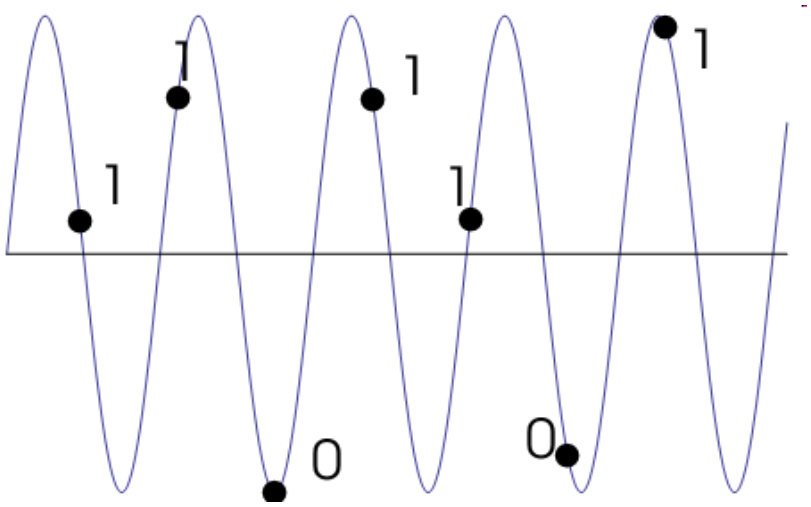
Looking for the simplest digital random number generator



Pseudorandomness from Sampling ROs

Many RO-based TRNG design are really pseudo random number generators, as the rate of jitter accumulation is overestimated.

Pseudorandomeness from sampling a deterministic oscillation:



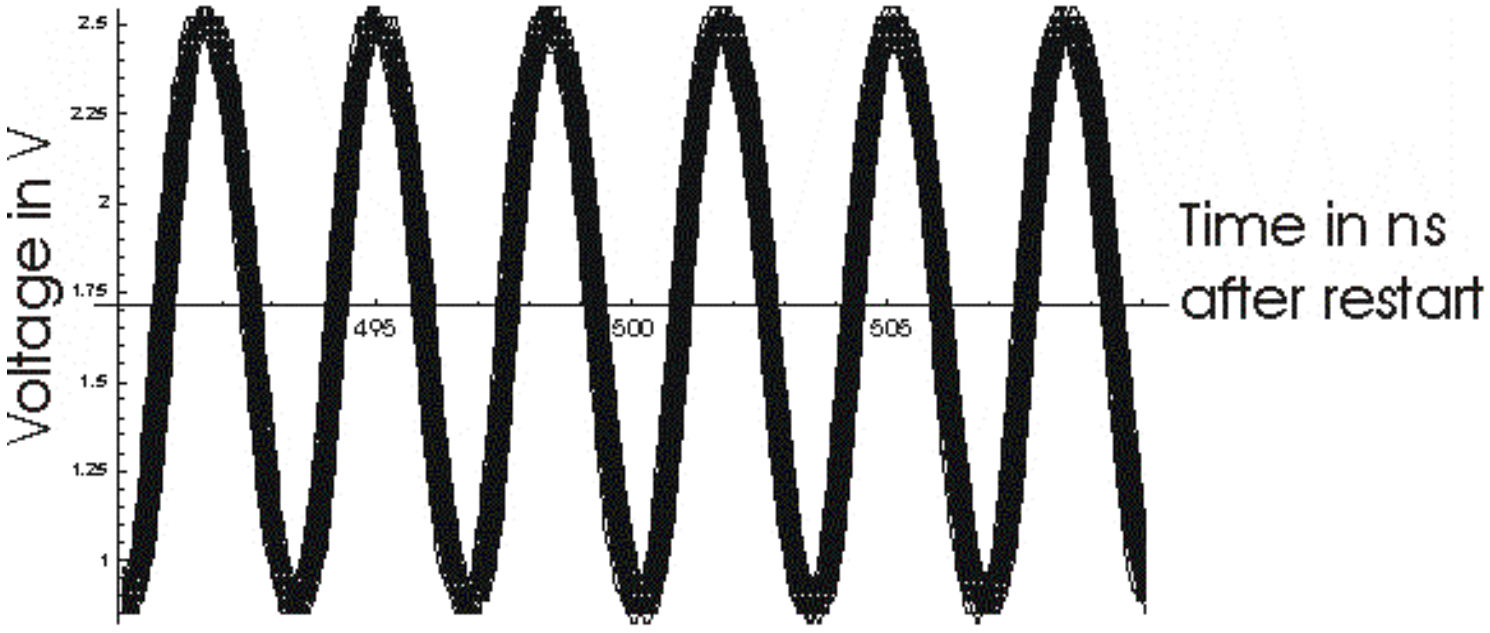
How to Distinguish True and Pseudo Randomness

Statistical tests can not distinguish good pseudo randomness from true randomness

Repeated **restarting** the generator from the same state helps to distinguish:

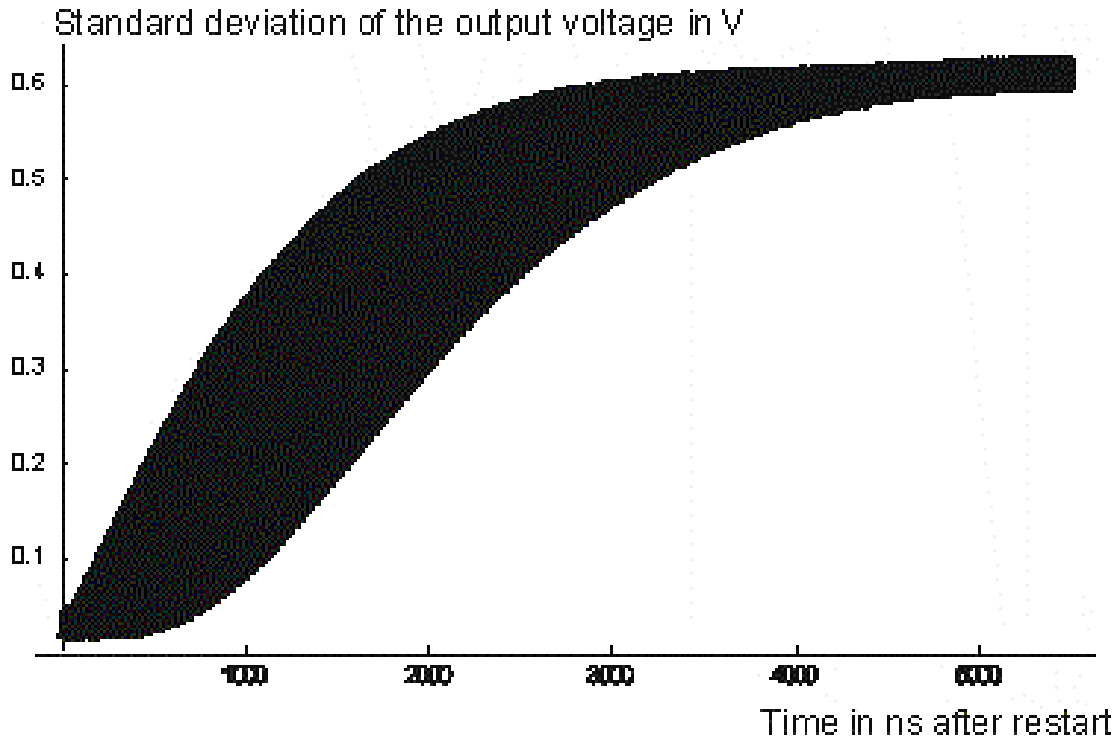
- Pseudorandomness leads to the same behaviour for each restart
- True randomness shows varied behaviour despite identical starting conditions

Restarting a Ring Oscillator I



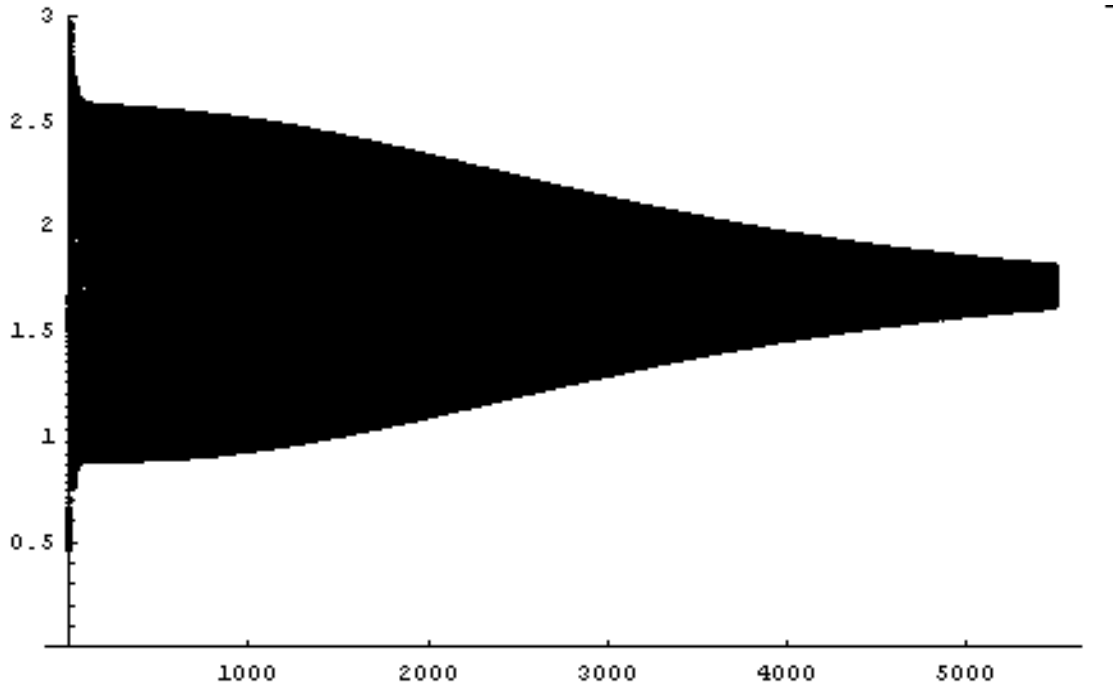
Even after 148 oscillation periods, 100 traces of restarts of a 296 MHz RO only a small amout of jitter has accumulated

Restarting a Ring Oscillator II



The figure shows the standard deviation of the output voltage of 1000 restarts of a 296 MHz RO.

Restarting a Ring Oscillator III



The figure shows the mean of the output voltage of 1000 restarts of a 296 MHz RO.

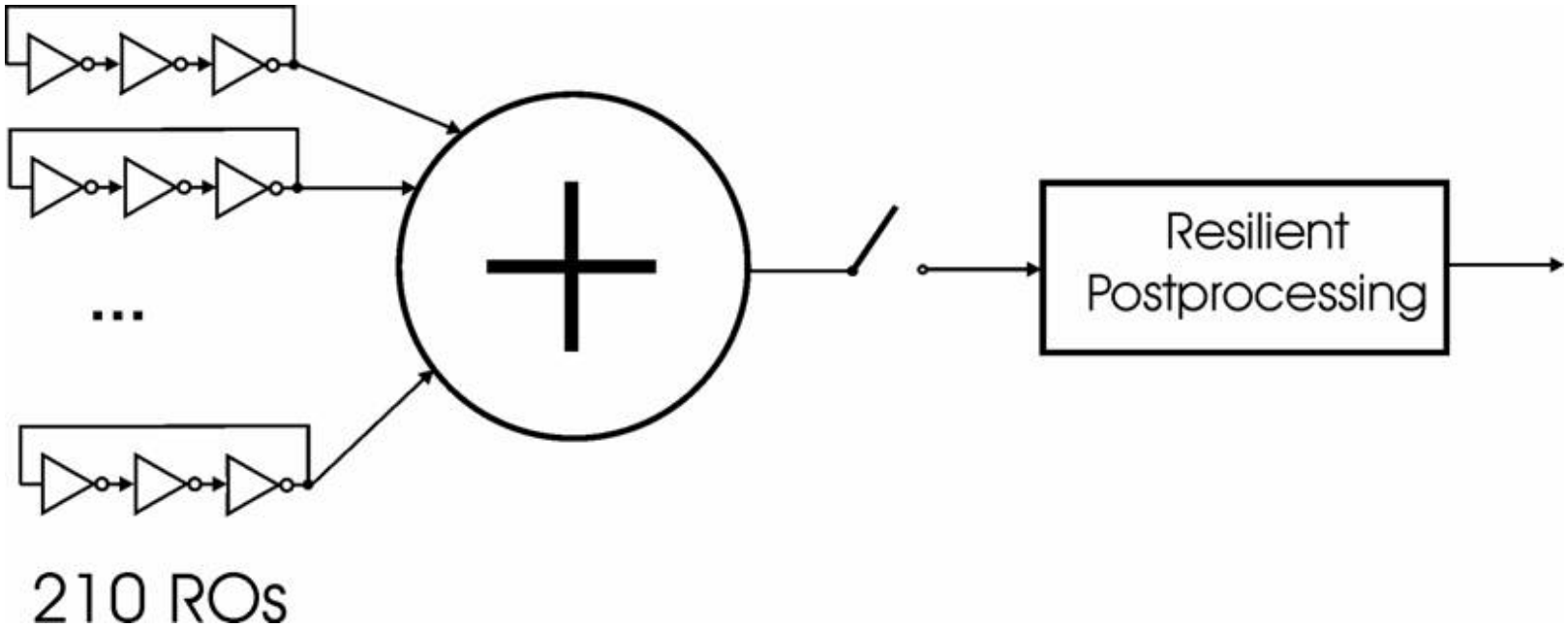
Do many ROs achieve more?

Idea: Use more ROs to get more entropy

Sunar, Martin, Stinson: A Provably Secure True Random Number Generator with Built-in Tolerance to Active Attacks, IEEE Trans. Computers, vol. 56(1), pp. 109-119, Jan. 2007

Schellekens, Preneel, Verbauwhede: FPGA Vendor Agnostic True Random Number Generator, FPL 2006, August 2006

The „provably secure“ TRNG design (Leuven version)



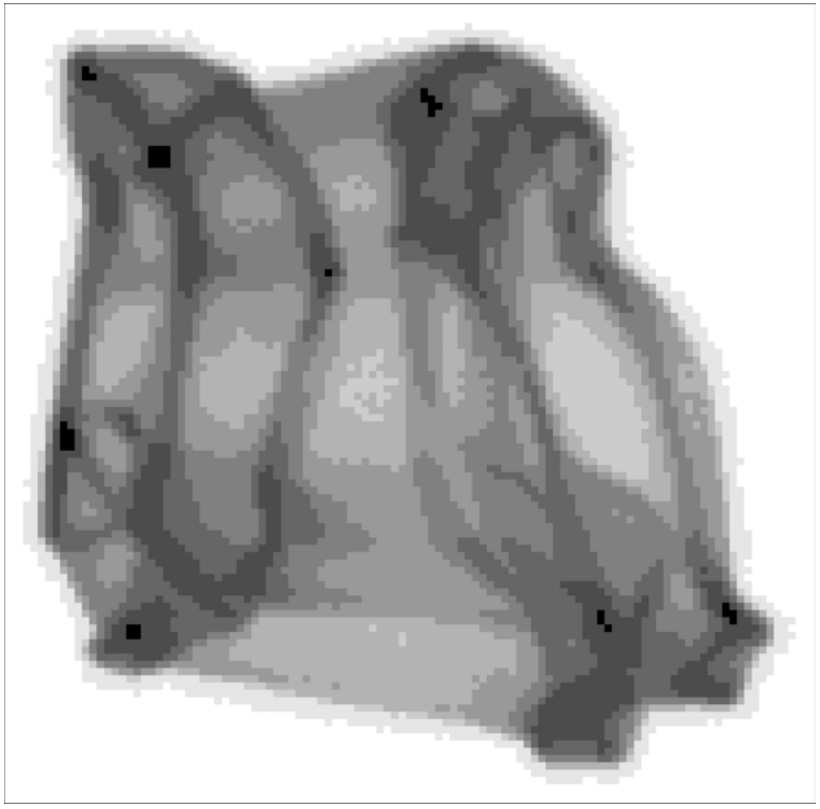
First Reason why the Security Proof Fails

The „security proof“ is based on a very **unrealistic statistical model of jitter**:
It is assumed that each RO has a built in perfect clock, and that jitter occurs only around the edges of the imaginary clock. The model does not allow the accumulation of jitter over time.

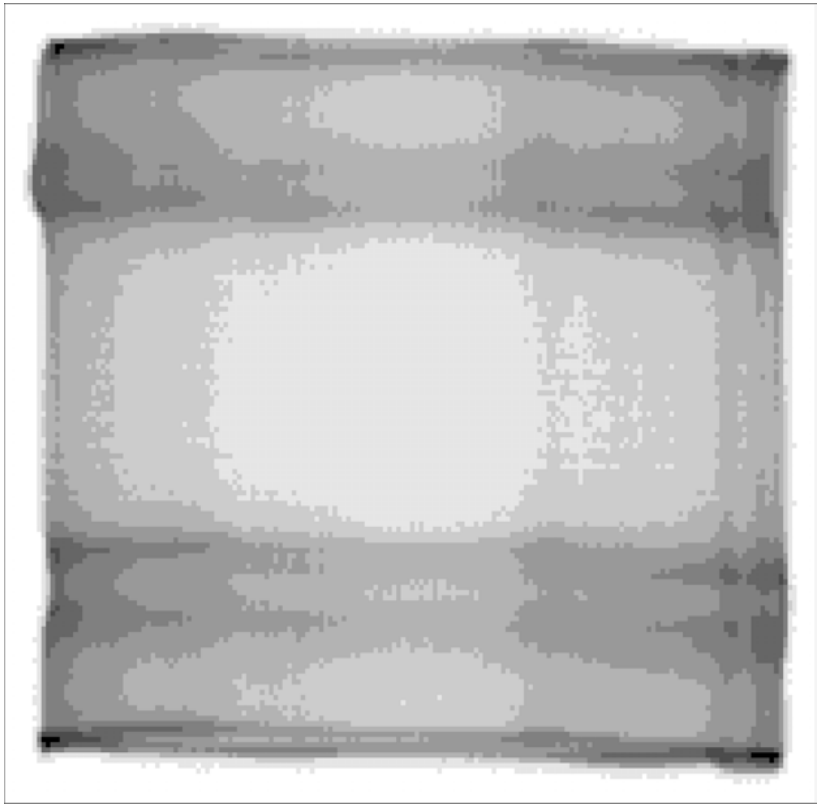
Second Reason why the Security Proof Fails

In the security proof, all the ROs are assumed to be statistically independent.
In reality, ROs implemented on the same chip interact strongly.

Interaction of Two ROs on the same FPGA Chip



Non-Interaction of Two ROs on Two FPGA Boards



Third Reason why the Security Proof Fails

The XOR of many high frequency oscillations results in an extremely high frequency signal impossible to compute.

In the Leuven design, the output of the XOR would have to make **139 billions of transitions per second**, which is clearly impossible with current technology.

Fourth Reason why the Security Proof Fails

Even if the extremely high speed XOR-signal could be computed, it could not be sampled, as the sampling flip-flop has to respect setup and hold times.
The signal has to be constant for some time in order to be sampled reliably.

The Leuven FPGA has to sample a signal with on average 23.8 transitions during the 170 ps the input has to be constant.

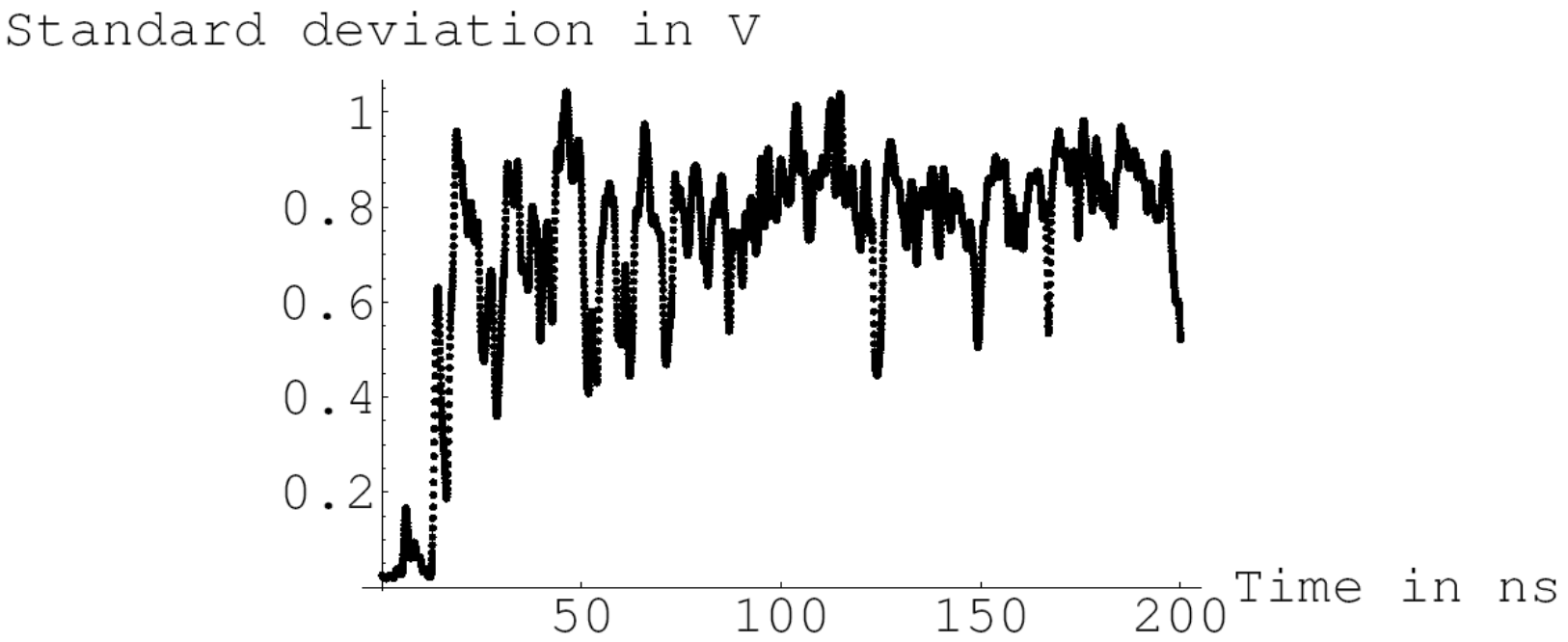
Is it secure?

The design is definitely not “provably secure”, but is it secure?

Statistical tests do not help, as they can not distinguish between true randomness and pseudorandomness

The completely deterministic XOR of only 16 oscillators with slightly different frequencies is so complex that it passes the Diehard test suite.

Restarting the XOR of 114 ROs



The results are surprisingly good, but sampling results in quite biased output

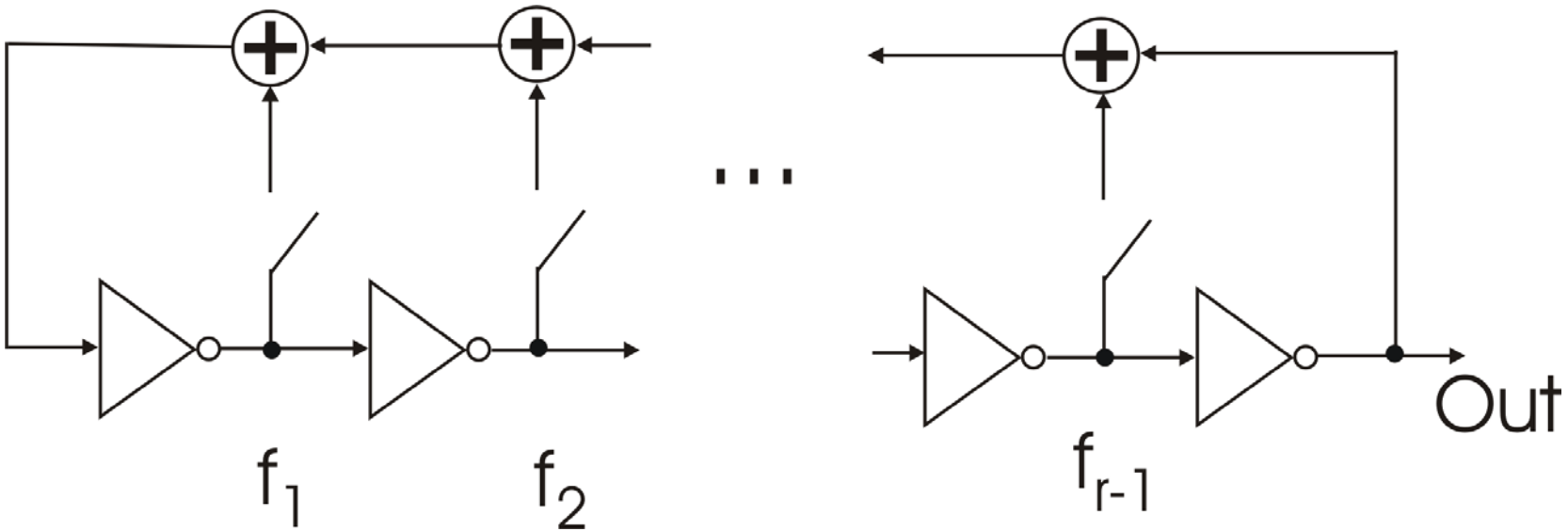
Better Digital TRNGs

Jovan Golić (Telecom Italia) invented two strongly improved variants of ROs,
Fibonacci ring oscillators (FIRO) and
Galois ring oscillators (GARO)

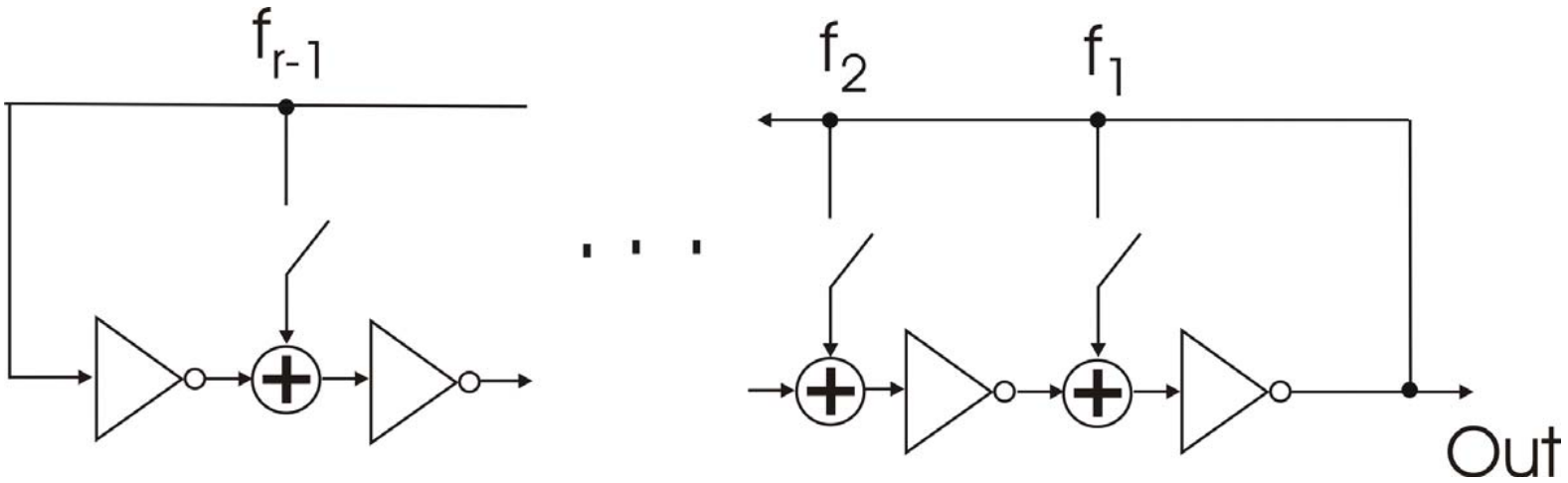
The designs show similarities with Fibonacci and Galois LFSRs, albeit the registers are replaced with inverters

J. Dj. Golić , “New Methods for Digital Generation and Postprocessing of Random Data,” IEEE Trans. Computers, vol. 55(10), pp. 1217-1229, Oct. 2006

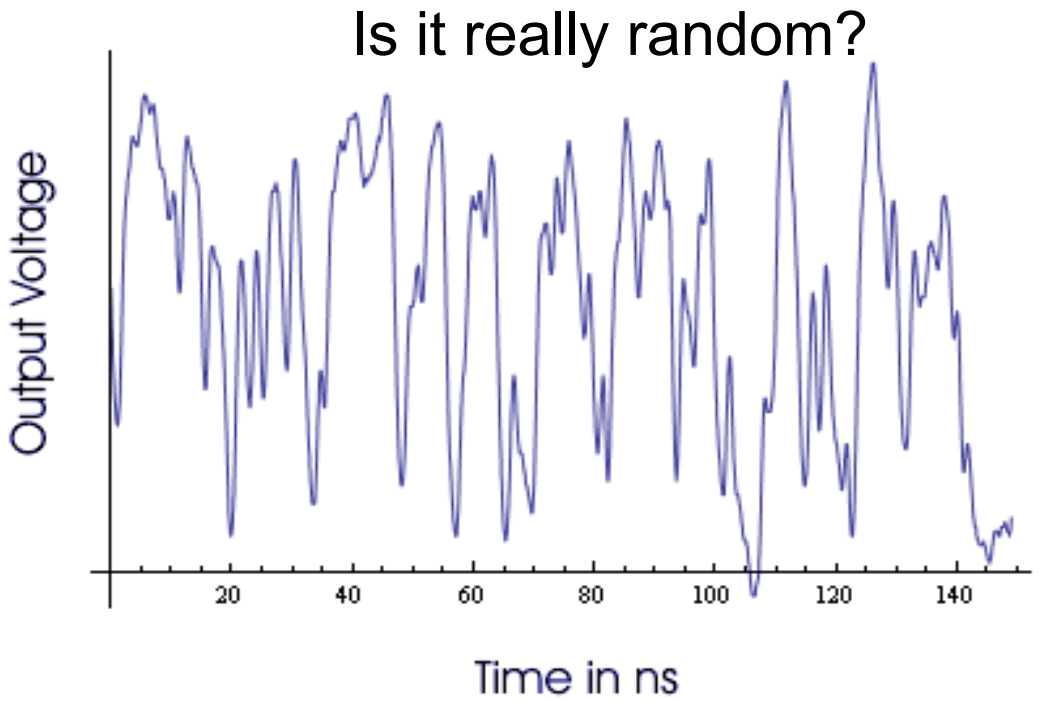
FIRO



GARO

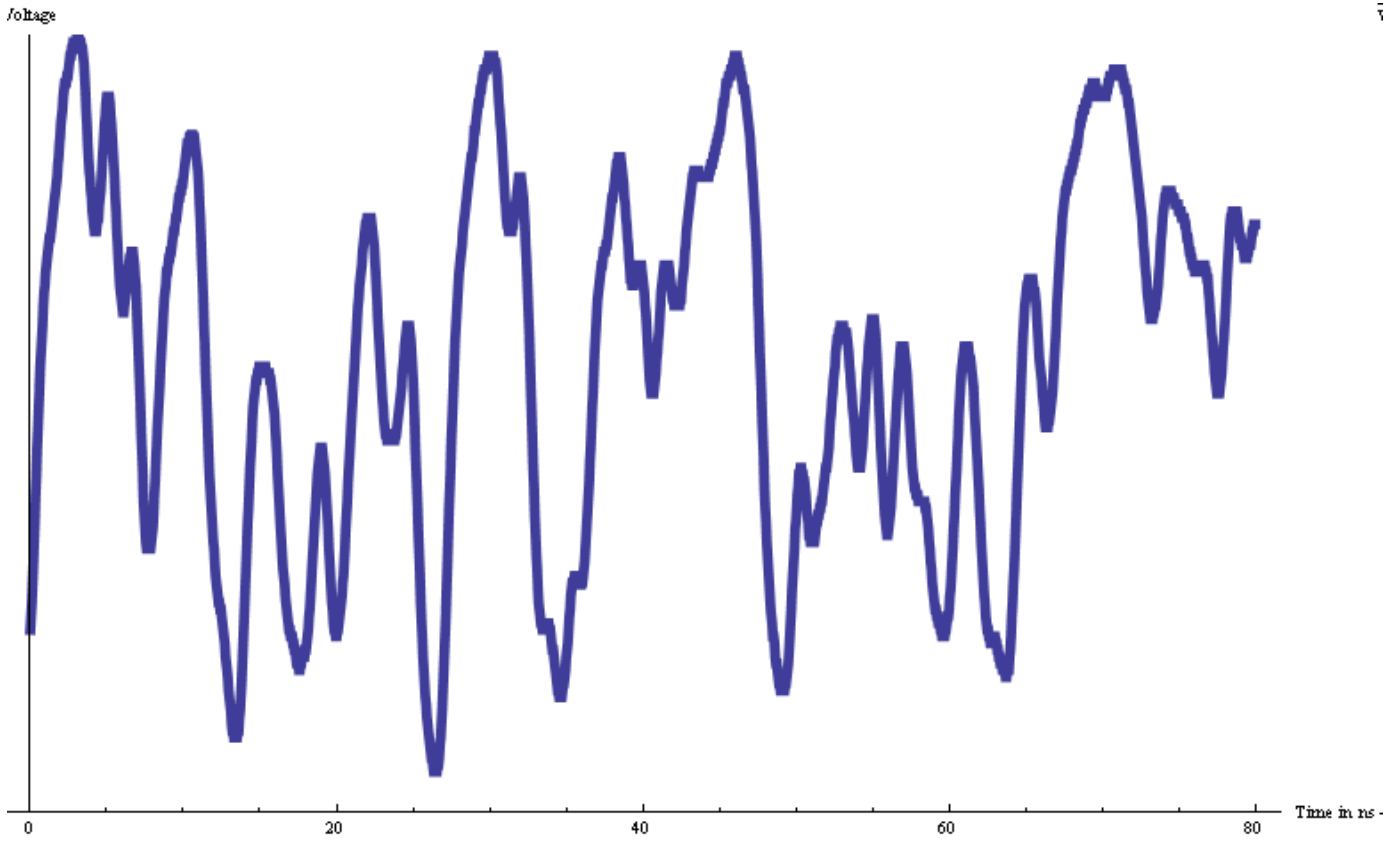


Example of FIRO Output

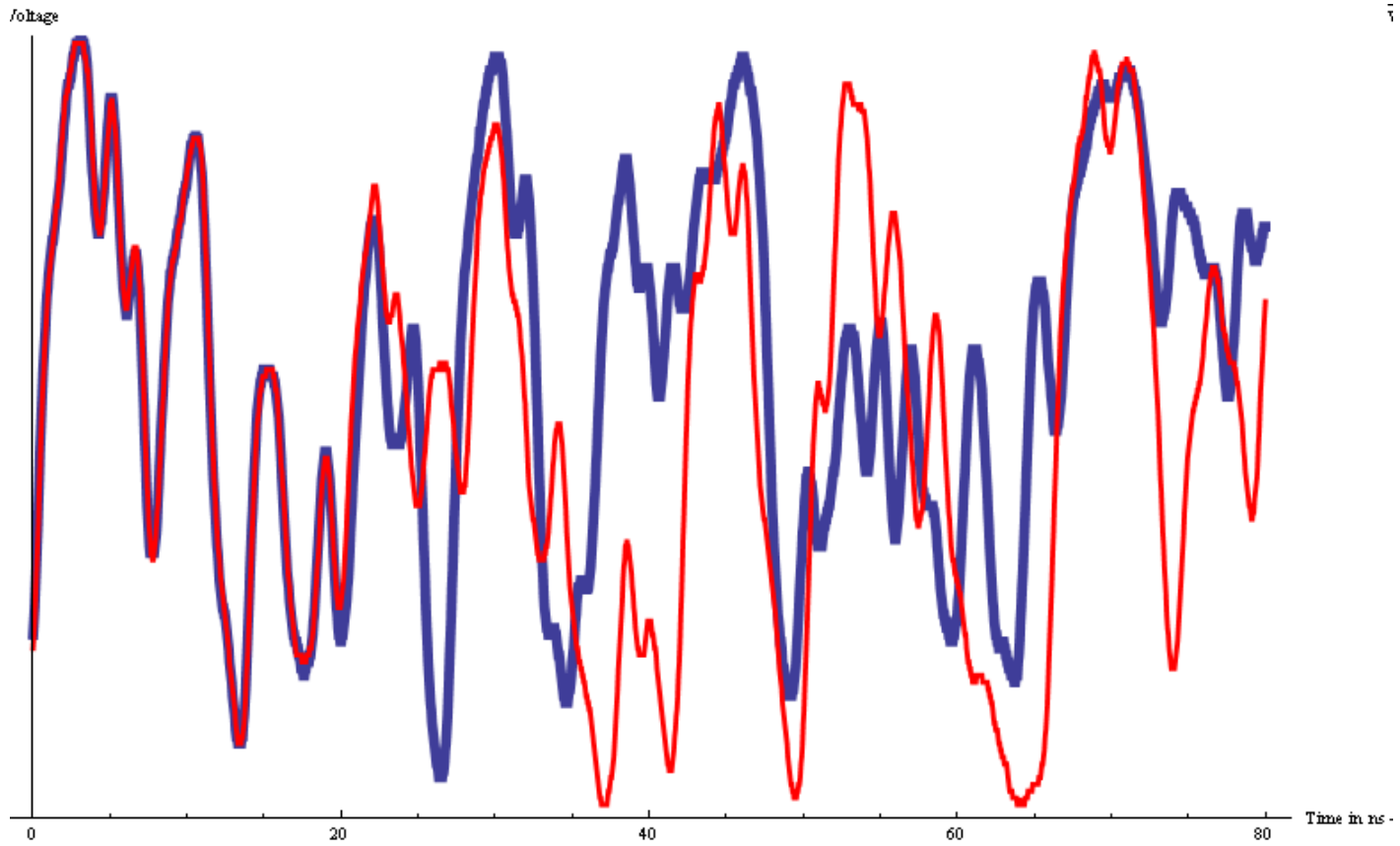


(Feedback polynomial $x^{15}+x^{14}+x^7+x^6+x^5+x^4+x^2+1$)

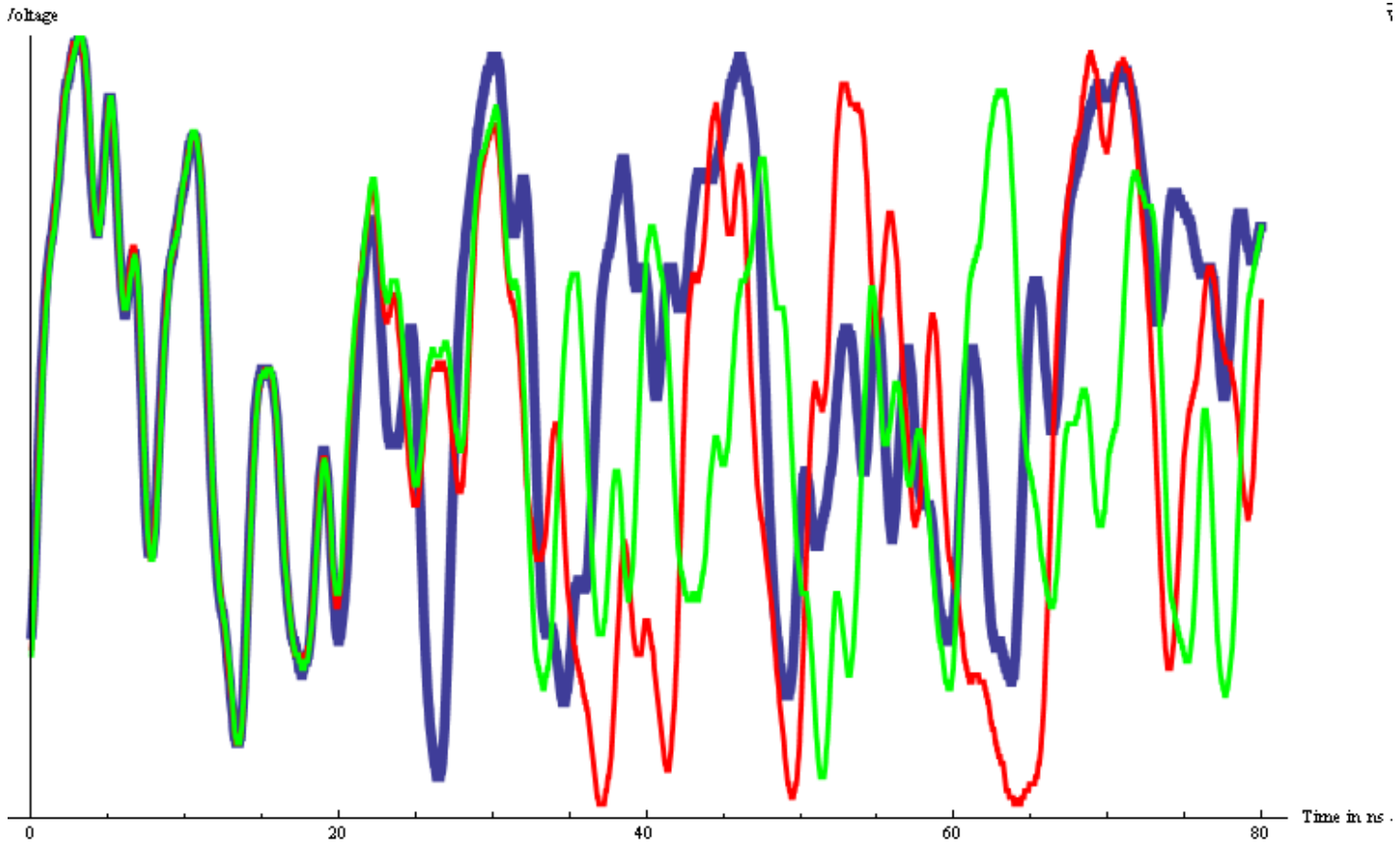
FIRO Restarts from Identical States (I)



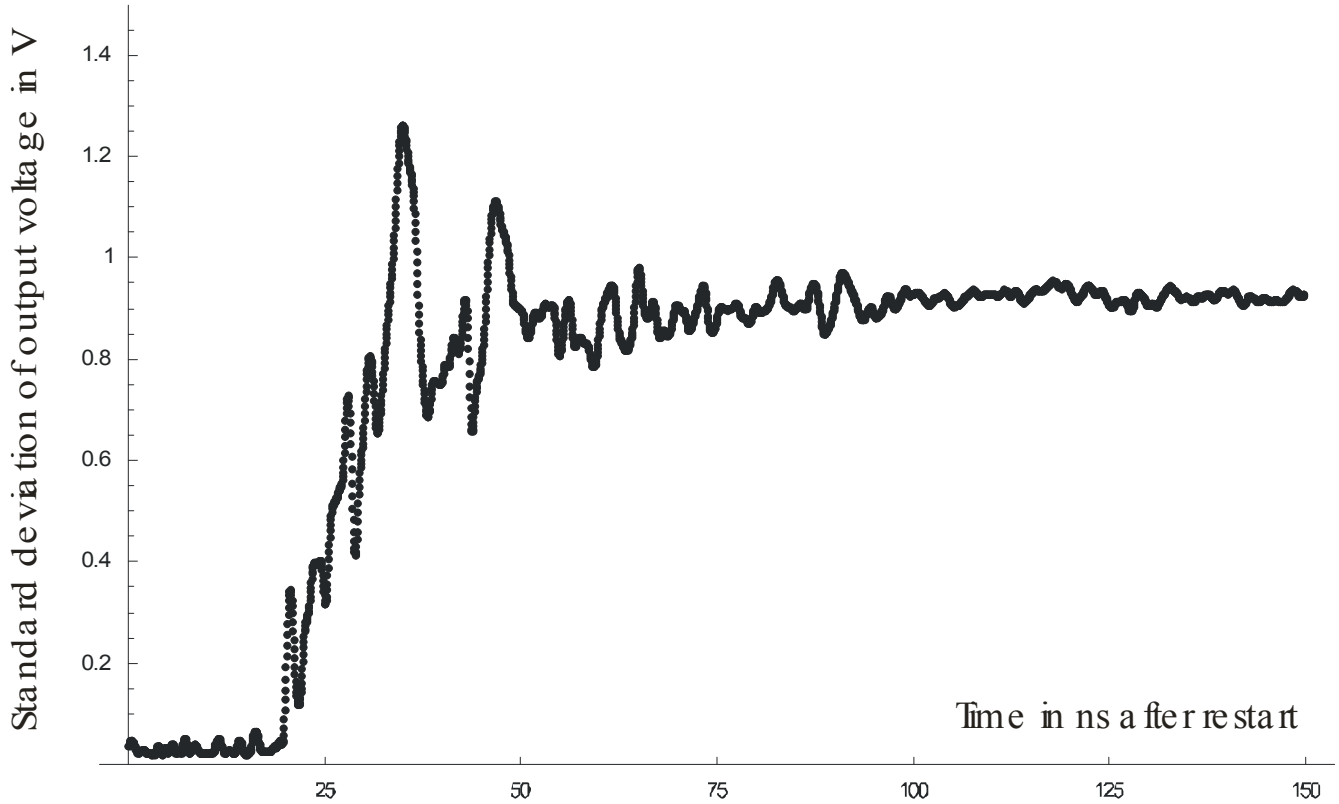
FIRO Restarts from Identical States (II)



FIRO Restarts from Identical States (III)



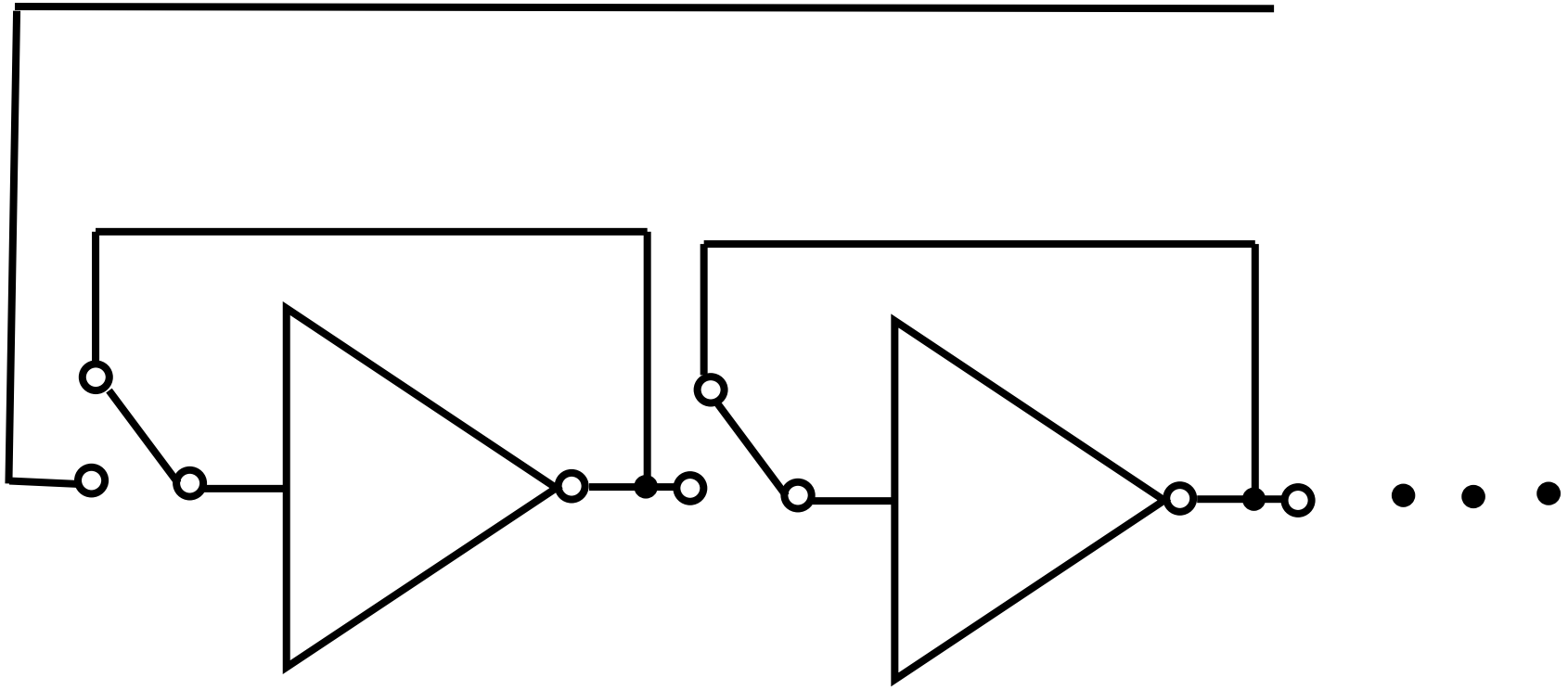
Standard Deviation of 1000 FIRO Restarts



A Cute New Idea

Ihor Vasyltsov, Eduard Hambardzumyan, Young-Sik Kim, Bohdan Karpinskyy (Samsung)
suggest in „**Fast Digital TRNG Based on Metastable Ring Oscillator**“ (preprint, accepted for CHES 2008) a cute new design of a stateless RO variant called meta-RO, which starts for the generation of each bit from the inverter state where the input and output voltage are identical.

A Meta-RO



A Cute New Idea which Does Not Work So Well on FPGAs

Problem: An inverter with feedback from output to input starts to oscillate with a frequency of about 680 MHz on Spartan 3. It seems to be quite tricky to get a stable state on Virtex 2.

Conclusion

- ROs are a reliable, albeit slow source of randomness on FPGAs
- the „provably secure“ design is definitely not provably secure, but may be secure anyway for unclear reasons
- FIROs and GAROS are the way to go
- Meta-ROs are a nice idea, but not the design of choice for FPGAs