Surname	Name	ID

Contents

1 Introduction

TRNG based on an **entropy source** ([2]).

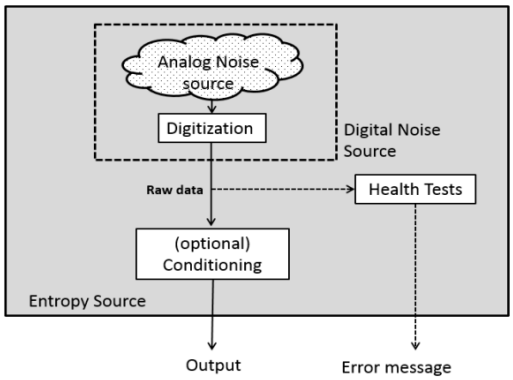

Figure 1: Entropy Source model

- **Noise Source** : the component that provides the actual entropy; it can be analog (digitization required) or digital → output: **raw data**. Noise sources can be **physical** (= dedicated hardware) or non-physical (= system data, such as output of Application Programming Interface (API) functions, RAM data or system time or human input).
- **Conditioning** : optional component, useful for **increasing the entropy rate** of output bits
- **Health Tests** : to detect **failure conditions**. Three categories: start-up tests, continuous tests (primarily on the noise source) and on-demand tests.

2 Noise Source

Most common noise sources:

- **thermal noise** → due to its small amplitude, high-gain and transimpedance amplifiers required + voltage comparators for digitization → **expensive, poor robustness**
- **chaos theory of nonlinear systems** → **complex structures and a high cost**
- **metastability** → Programmable Delay Lines (PDLs) used to carefully plan the arrival of the signal to FFs, hence forcing the violation of setup or hold time

- **jitter** → random bit given by FF sampling in presence of phase jitter. TRNG can be based on **jitter accumulation** (= increasing the number of transition events) or **jitter quantization** (= optimization of single propagation event).

Metastability and jitter events are often exploited together.

3 Possible implementations

3.1 Ring Oscillator approach - [7], [6], [5]

A ring oscillator is composed of an odd number of inverters or delay elements connected in ring configuration. The basic idea is to exploit the oscillations caused by power supply variations, cross talks, semiconductor noise, temperature variations, and propagation delays. The period of these oscillations vary from cycle to cycle causing jitter of the rising and falling edges. A DFF can be used for sampling the outhput of a high frequency oscillator, hence generating a stream of truly random bits. The digital value of the oscillators output changes with a period of approximately 2DL (D = delay of a single inverter, L = number of inverters in an oscillator).
A XOR tree after the ring oscillator or the sampling DFF can be used as conditioning circuit.

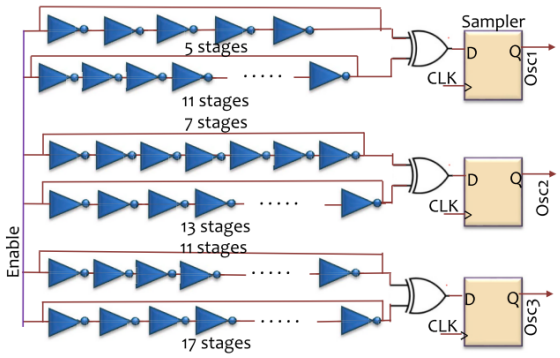

Figure 2: Ring Oscillator generic approach

In a FPGA a LUT approach can be used to create a programmable delay inverter!

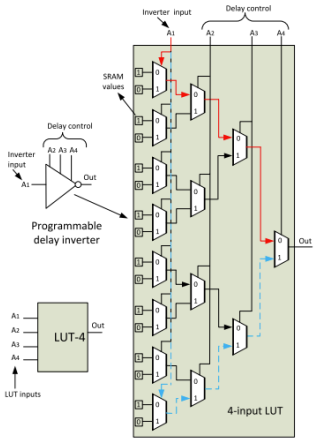

Figure 3: Programmable delay line using a 4-input LUT

3.1.1 Transition Effect Ring Oscillator (TERO) - [7]

Key idea : exploit the oscillatory metastability of latches.
CTRL signal transition → oscillations in feedback loop until latch in steady state.
Random variable : number of oscillations for each transition of CTRL input, due to intrinsic noise in semiconductors.
Output is 0 or 1 depending on even/odd oscillations.

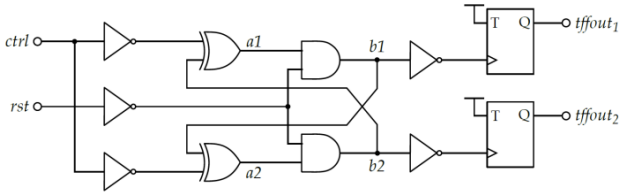

Figure 4: Transition Effect Ring Oscillator

Main problem: low throughput (Solution: asymmetry in branches but consequent reduction of standard deviation of random variable)

3.1.2 Metastable Ring Oscillator (Meta-RO) - [7]

Key idea: metastability of inverters.
mux_sel = 0 → metastable state (the metastable voltage is perturbed by low amplitude noise)
mux_sel = 1 → low amplitude noise is amplified by inverters; the ring starts in a random state; once the oscillator stabilizes to full-logic levels → sampling; delay line to accurately control the sampling instant

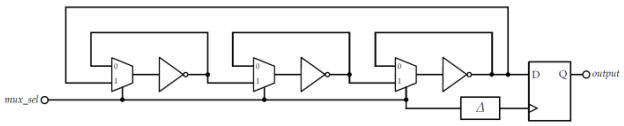

Figure 5: Metastable Ring Oscillator

3.1.3 Fibonacci and Galois Ring Oscillators (FiRO, GaRO, FiGaRO) - [7]

Key idea: DFF in Fibonacci and Galois LFSR configurations replaced by inverters to obtain an asynchronous circuit → randomness due to inverters delay depending on temperature and supply voltage. Additional randomness can be added in the sampling phase (metastability due to setup and hold violations).
Systems described by the characteristic polynomial:

$$P(x) = \sum_{i=0}^{r} f_i x^i \qquad with \quad f_0 = f_r = 1 \tag{1}$$

In both configurations, conditions must be satisfied to avoid fixed points, hence ensuring continuous oscillations. For FiRO, the conditions are:

$$\begin{cases} P(x) = (x+1)h(x) \\ h(1) = 1 \end{cases} \tag{2}$$

with $h(x)$ quotient polynomial of feedback polynomial $P(x)$.

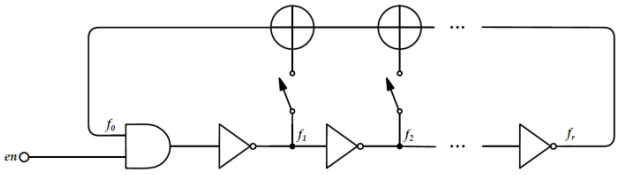

Figure 6: Fibonacci Ring Oscillator

For GaRO:

$$\begin{cases} P(1) = 0 \\ r \; is \; odd \end{cases} \tag{3}$$

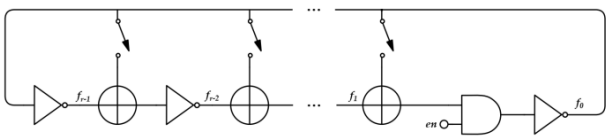

Figure 7: Galois Ring Oscillator

To further increase randomness and reliability, a mixed structure of both FiROs and GaROs can be used. The higher the number of parallel RO stages, the higher the throughput.

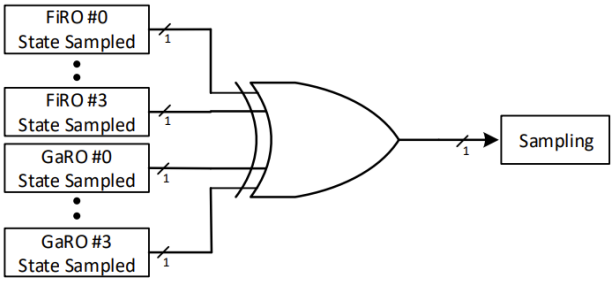

Figure 8: Fibonacci-Galois Ring Oscillator

An architecture and performance comparison of the previous five described circuits is reported in the following tables.

Architecture	**Physical Phenomena Generating Entropy**	**Preliminary Observation**
Transition Effect Ring Oscillator (TERO)	Latches oscillatory metastability	Small bandwidth, large dependence on placing
Metastable Ring Oscillators (Meta-RO)	Analogue metastability of inverter gates	PLL required, dependence on placing
Fibonacci Ring Oscillator (FiRO)	Jitter and Metastability	Good independence from placing
Galois Ring Oscillator (GaRO)	Jitter and Metastability	Good independence from placing
Fibonacci-Galois Ring Oscillator (FiGaRO)	Jitter and Metastability	Independence from placing, higher entropy and robustness respect to single Fibonacci and Galois Oscillator

Figure 9: Architecture comparison of different RO-based TRNG

Table 5. Comparison among the proposed TRNG with other TRNG implementations on FPGA.

	TRNG Type	Platform	LUTs	Registers	Bit Rate (Mbps)	Entropy
This work (4 Stages, 2FiRO-2GaRO)	FiGaRO	Intel Stratix IV	288	190	400	0.995
[31]	ES	Xilinx Spartan 6	10	5	1.15	0.997 (Shannon Entropy)
[32]	RO	Xilinx Virtex 2	–	–	2.5	0.97
[33]	RO—PDLs	Xilinx Spartan-3A	528	177	6	0.9993
[34]	STRs	Xilinx Virtex 6	32	48	4	–
[27]	TERO	Xilinx Artix 7	40	29	1.91	0.9993 (Shannon Entropy)
[35]	STRs	Xilinx Virtex 6	56	19	100	–
[36]	GaRO	Xilinx Artix 7	50	79	280	0.998 (Shannon Entropy)

Figure 10: Performances comparison of different types of TRNG

3.2 DCM approach (FPGA only) - [4]

Digital Clock Managers (hardware primitives) are used to produce one or more output clock signals with a programmable frequency to be sampled.

Main problem : if Dynamic Partial Reconfiguration port (DRP) of DCMs is used to tune the frequency of signals driving the data and the clock input of a FF → slow random bit production rate (hundreds of clock cycles elapse between two consecutive samplings).

Solution : dynamic phase shifting (DPS) → no tuning of the frequency of the output clocks but their phase compensate the difference in routing delays of the signals arriving at the FFs.

The phase shift resolution achieved by the DCM could be not as fine as required → a carry chain primitive and four FF replicas for finer phase shift resolution control at the destination.

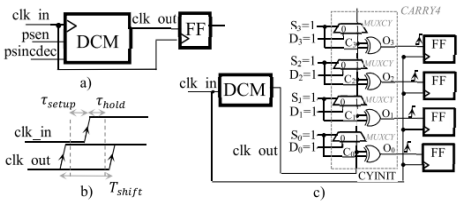

Figure 11: DCM approach

Conditioning required to pass the NIST tests and achieve sufficient entropy.

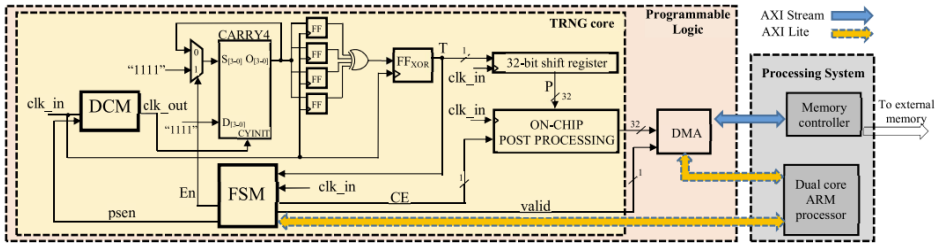

Figure 12: DCM-based TRNG

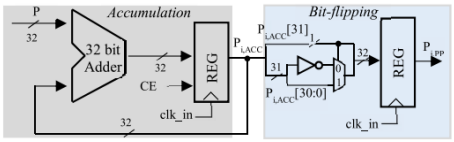

Figure 13: DCM-based TRNG - post-processing block

- **XOR gate** : to merge the outputs of the four FF;
- **FSM** : impose phase shifting transaction in DCM until signal T = 1, i.e. at least one FF in metastability;
- **Post-processing block** : accumulator + 'bit flipping' machine

Performances results:

TABLE II
COMPARISON RESULTS

	Area			Thr.	O.F.	Thr.	Prop.	Bit entr.	Platform
	Lut	FF	Slice	[10^6 bps]	[MHz]	Slice · OF		(AIS T8)	
[2]	128	-	32	2	16.4	3.81e-3	90%	-	Virtex 5
[5]	528	177	270	6	24	9.26e-4	98.7%	0.9993	Spartan 3A
[15]	18	17	-	0.209	100	-	7.5%	-	Virtex 5
[14]	26	19	14	0.419	75.8	3.95e-4	80%	-	Virtex 5
[6]	32	48	-	4	1	-	97.5%	0.9999	Virtex 5
[7]	56	19	-	100	400	-	98%	-	Virtex 6
[11]	-	-	-	100	400	-	-	0.9900	Virtex 5
[13]	24	2	-	290	290	-	70%	-	Virtex 6
[4]	4	3	1	0.8	50	1.6e-2	97%	0.9998	Spartan 6
[12]	32	55	33	12.5	12.5	3.03e-2	-	0.9995	
[8]	866	-	-	7.3	80	-	98.1%	-	Xilinx
This work	38	121	38	300	300	2.63e-2	98.1%	0.9986	Zynq-7000
				100	100	2.63e-2	98%	0.9994	

TABLE III
POWER COMPARISON

Technique	O.F. [MHz]	Power [mW]	Power/Thr. [nJ/bit]
[14]	75.8	238	568
[12]	12.5	9.5	0.76
This work	300	119	0.4

Figure 14: DCM - DPS performances (highlighted)

3.3 PLL approach - [3]

A high-frequecy clock ($clk1$ in figure 15) is generated by a PLL and sampled by a DFF working with a lower frequency clock ($clk0$).
The PLL is characterized by a divison factor K_D and a multiplication factor K_M. Parameters:

- $f_{out} = \frac{K_M}{K_D} f_0$;
- Bit rate R = $\frac{f_0}{K_D}$;
- Sensitivity to jitter S = $\Delta^{-1} = \frac{K_D}{T_{out}}$

Ideal case: no jitter $\rightarrow$ periodic sampler output with period $T_P = K_D T_0$. Solution: synchronous counter to restart the decimator after each period T_P. The decimator is used for XOR-ing K_D samples and obtain 1-bit output.
Real case: jitter due to local noises $\rightarrow$ **random variable** : period of clk1.

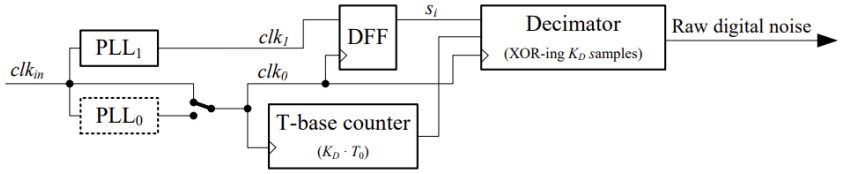

Figure 15: PLL-based TRNG

By reconstructing the period of the sampled signal $clk1$ thanks to the coherent sampling principle, it can be observed that only few samples will have a random value. The entropy rate will depend on the initial phase ϕ_0, the time resolution Δ and the standard deviation σ of the random variable $clk1$. This is why the decimator is needed. Higher K_D means lower time resolution, higher sensitivity to jitter and higher entropy rate. Problem : **Tradeoff entropy rate - output bitrate** $\rightarrow$ solution: since $S = K_M f_0$, increase the entropy rate by increasing K_M and/or f_0. An additional PLL (PLL0 in figure 15) can be used to obtain suitable K_M and K_D factors ($K_M = K_{M1} K_{D0}$, $K_D = K_{D1} K_{M0}$).
Algorithms exist to find suitable values for K_M and K_D ([1]).

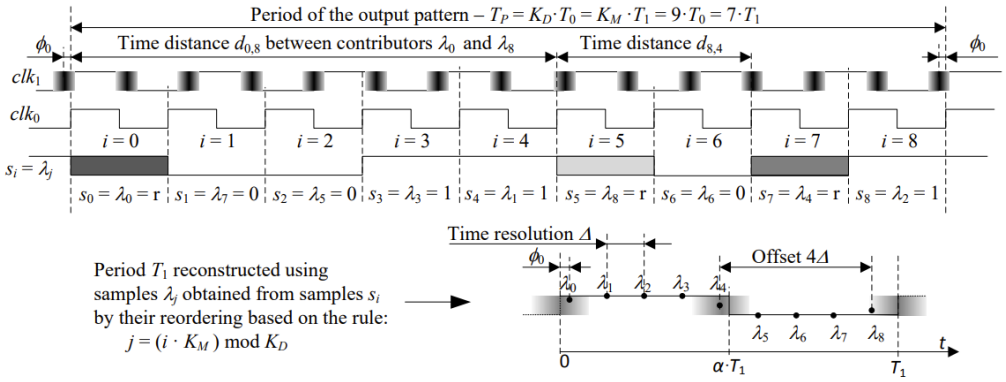

Figure 16: Reconstruction of one sampled clock period T1 by reordering successive K_D samples, ϕ_0 is the initial relative phase shift between clk1 and clk0

The basic architecture can be improved (figure 17) with a series of modifications:

- **More PLL outputs** to increase entropy;
- second DFF to solve possible metastability;
- 1-bit decimator replaced by an **m-bit time-to-digital converter** (TDC) → useful for testing;
- the least significant converter bit cnt(0) is used as the raw random signal, which is temporarily saved in a FIFO memory, whose depth depends on the latency of the total failure test;

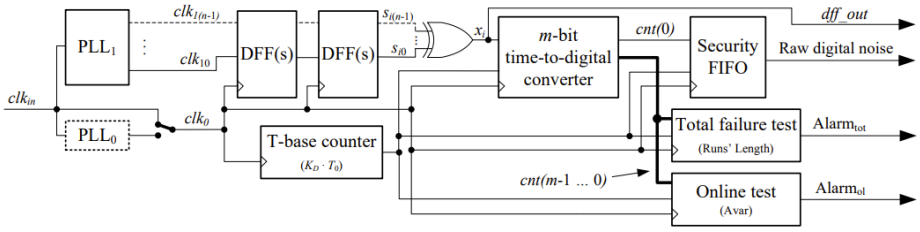

Figure 17: Enhanced PLL-based TRNG

References

[1] Elie Noumon Allini, Oto Petura, Viktor Fischer, and Florent Bernard. Optimization of the pll configuration in a pll-based trng design. In *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1265–1270, 2018.

[2] Kerry McKay Allen Roginsky Meltem Sönmez Turan Elaine Barker, John Kelsey. Nist sp 800-90b: Recommendation for the entropy sources used for random bit generation. 2022.

[3] Viktor Fischer, Florent Bernard, Nathalie Bochard, Quentin Dallison, and Maciej Skórski. Enhancing quality and security of the pll-trng. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2023(4):211–237, Aug 2023.

[4] Fabio Frustaci, Fanny Spagnolo, Stefania Perri, and Pasquale Corsonello. A high-speed fpga-based true random number generator using metastability with clock managers. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 70(2):756–760, 2023.

[5] Annapurna Kamadi and Zia Abbas. Implementation of trng with sha-3 for hardware security. *Microelectronics Journal*, 123:105410, 2022.

[6] N. Nalla Anandakumar, Somitra Kumar Sanadhya, and Mohammad S. Hashmi. Fpga-based true random number generation using programmable delays in oscillator-rings. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 67(3):570–574, 2020.

[7] Pietro Nannipieri, Stefano Di Matteo, Luca Baldanzi, Luca Crocetti, Jacopo Belli, Luca Fanucci, and Sergio Saponara. True random number generator based on fibonacci-galois ring oscillators for fpga. *Applied Sciences*, 11(8):3330, Apr 2021.