HECTOR

D2.1
Report on Selected TRNG and PUF Principles

Project number:	ICT-644052
Project acronym:	**HECTOR**
Project title:	Hardware Enabled Crypto and Randomness
Project Start Date:	1 March, 2015
Duration:	36 months
Programme:	H2020-ICT-2014-1
Deliverable Type:	Report (R)
Reference Number:	ICT-644052-D2.1
Workpackage:	WP 2
Due Date:	29 February, 2016 - M12
Actual Submission Date:	29 February, 2016
Responsible Organisation:	UJM
Editor:	Viktor Fischer
Dissemination Level:	Public
Revision:	1.01
Abstract:	This report represents the final version of Deliverable 2.1 of the HECTOR work package WP2. It is a result of discussions and work on Task 2.1 of all HECTOR partners involved in WP2. The aim of the Deliverable 2.1 is to select principles of random number generators (RNGs) and physical unclonable functions (PUFs) that fulfill strict technology, design and security criteria. For example, the selected RNGs must be suitable for implementation in logic devices according to the German AIS20/31 standard. Correspondingly, the selected PUFs must be suitable for applying similar security approach. A standard PUF evaluation approach does not exist, yet, but it should be proposed in the framework of the project. Selected RNGs and PUFs should be then thoroughly evaluated from the point of view of security and the most suitable principles should be implemented in logic devices, such as Field Programmable Logic Arrays (FPGAs) and Application Specific Integrated Circuits (ASICs) during the next phases of the project.
Keywords:	random number generators, physical unclonable functions, entropy, sources of randomness, stochastic models, statistical testing

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 644052.

Editor

Viktor Fischer (UJM)

Contributors (ordered according to beneficiary numbers)

Martin DEUTSCHMANN, Sandra LATTACHER (TEC)
Jeroen DELVAUX, Vladimir ROŽIĆ, Bohan YANG, Dave SINGELÉE (KUL)
Lilian BOSSUET, Viktor FISCHER, Ugo MUREDDU, Oto PEŤURA (UJM)
Alexandre ANZALA YAMAJAKO (TCS)
Bernard KASSER (STR)
Gerard BATTUM (BRT)

Executive Summary

Random Number Generators (RNGs) and Physical Unclonable Functions (PUFs) are basic cryptographic primitives that can be used as generators of streams of random bits or random numbers (vectors of bits). The generated random numbers appear either during device manufacturing or its deployment: PUFs use randomness coming from the uncertainty appearing only during the device production process and RNGs use dynamic sources of randomness during device operation. Although the principles and characteristics of these two primitives are very different, they have one property in common: the physical security relies on the underlying hardware, topology and exploited technology, which means that they cannot be standardized, because the physical characteristics change technology by technology, device by device. Therefore, instead of proposing standard principles, which in fact cannot exist because of differences in microelectronic technologies, we propose a standard design approach ensuring high physical security.

The aim of this deliverable of the HECTOR project is to preselect principles of RNGs and PUFs that fulfill strict technology, design, quality and security criteria, which are required for generators of confidential keys. Many TRNG and PUF principles have been published up to now, but also many of them are clearly not suitable for the stringent security-aware design. While the need of the pre-selection of suitable TRNG and PUF designs was evident from the beginning of the project, the selection criteria were not clearly defined. Therefore, the requirements on key generators and the selection criteria were first discussed and adopted by all partners involved in WP2.

Since the sources of randomness in TRNGs and PUFs are different, the requirements on both primitives differ too and therefore, they needed to be analyzed separately. During the pre-selection process, HECTOR partners analyzed, evaluated and discussed the best known candidates from the point of view of their suitability for the adopted approach. The aim of this process was to reduce the number of generators that will be further analyzed and implemented in hardware. Despite the efforts made during the TRNG and PUF selection phase, the partners agreed that some new principles could appear in the near future and therefore they decided to leave the list of preselected principles open – any new candidates can be designed, studied and implemented all along the HECTOR project.

This deliverable is organized as follows. After the main topic of the report is introduced in Chapter 1, the theoretical and practical background of the design of RNGs and PUFs is described in Chapter 2. In Chapter 3, requirements specified by industrial partners and other requirements coming from common practical needs in cryptographic applications are discussed in detail. At the end of Chapter 3, main selection criteria for both RNGs and PUFs are presented. The list of these criteria is a fruit of long lasting discussions between partners involved in WP2. Chapter 4 presents selected RNG principles and evaluates each generator regarding criteria, which were defined in Chapter 3. Similarly, Chapter 5 deals with selection of PUF principles and their evaluation depending on given criteria. In Chapter 6, we present results of implementation of pre-selected TRNG and PUF cores in FPGAs. Presented results are discussed in detail at the end of this chapter. Finally, Chapter 7 concludes this deliverable.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Random numbers are crucial in cryptography: they are used as confidential keys, padding data, initialization vectors, nonces in challenge-response protocols, but also as random masks in countermeasures against side channel attacks. Historically, they are generated by random number generators (RNGs). RNGs are cryptographic primitives, which generate a sequence of bits or symbols (e.g. groups of bits) that do not feature any pattern. The generated bits or symbols must be independent and uniformly distributed. Recently, a new type of physical random number generators has appeared - the physical unclonable functions (PUFs). PUFs can be used for hardware authentication in a challenge-response protocol, but also as generators of device-specific confidential keys.

The security of cryptographic systems is mainly linked to the protection of confidential keys. In high end information security systems, when used in an uncontrolled environment, cryptographic keys and random masks should never be generated outside the system and they should never leave the system in clear. Consequently, if the security system is implemented in a single chip (cryptographic system-on-chip), the keys should be generated inside the same chip, i.e. the logic device. However, logic devices are aimed at implementation of deterministic logic systems and not at realization of random number generation relying on analog physical phenomena. The implementation of random number generators and PUFs in logic devices such as field programmable logic arrays (FPGAs) and digital application specific integrated circuits (ASICs) is therefore a major challenge.

The design of electronic devices is evaluated by estimating their highest achievable speed (characterized by the maximum clock frequency, latency and delay of the electronic block), cost (expressed in area, e.g. in number of gates or logic elements), and power/energy consumption. In cryptography, these basic criteria are completed by security characteristics, such as robustness against attacks, and tamper resistance, which take priority.

An ideal device would be able to operate at maximum speed, lowest cost, having the lowest power/energy consumption, and reaching the inviolable security. Clearly, such a device does not exist and several compromises must be made during the design. The level of acceptable compromises depends on the application.

The HECTOR project does not target some special application. Consequently, without knowing the final use of the TRNG or PUF, the interval of acceptable compromises cannot be delimited. Since the ideal solution does not exist, the strategy adopted in the HECTOR project is to select several TRNG and PUF principles giving the best results and fulfilling the largest set of constraints. The probability that the required characteristics of the TRNG or PUF will be realizable using one of preselected principles is then expected to be the highest.

As stated before, security has the highest priority in cryptographic applications. In implementations of algorithmic cryptographic functions, the security of the algorithm is ensured by the thorough reviewing process, during which worldwide experts evaluate publicly security parameters and robustness of the given algorithm. This security is then guaranteed at the algorithmic level by the security standards, which define approved algorithms and their parameters, versions, etc.

However, the physical security, which becomes vital in side channel attacks, random number generators, and physical unclonable functions depends considerably on the underlying hardware, topology, exploited technology, etc. Consequently, cryptographic primitives relying on physical security cannot be standardized, because the physical characteristics change technology by technology, device by device. In order to remedy this potential security weakness, instead of proposing a standard hardware cryptographic block, the HECTOR project targets to propose a standard design approach ensuring high physical security.

Chapter 2

Theoretical and Practical Background on RNGs and PUFs

RNGs and PUFs may be used to generate random numbers depending on random physical processes that act upon the device either during its manufacturing or during its operation. In this chapter, we will discuss the theoretical and practical background concerning these two types of cryptographic hardware primitives. Since their principles and characteristics vary considerably, they will be presented in two separate sections.

2.1 Random Number Generators

Random number generators are physical functions generating a sequence of bits or symbols (e.g. groups of bits – numbers) that are independent, uniformly distributed and do not feature any pattern. RNGs have many applications in modern technologies. They are widely used in cryptography, but also in Monte Carlo simulations of complex systems, as noise generators in telecommunication systems, in games, slot machines, etc.

Cryptographic applications have strong security requirements and RNGs included in them must be cryptographically secure. This means that they must generate random numbers that have good statistical quality and the generated sequences must not be predictable or manipulable.

For this reason, RNGs which are targeted for data security applications must be designed and analyzed very carefully. The next sections help in understanding the secure RNG design and in establishing main RNG evaluation criteria.

2.1.1 RNG Design

Practical RNGs are grouped into two main categories: random number generators, also called true random number generators (TRNGs), and pseudo-random number generators (PRGNs), also called deterministic random number generators (DRNGs). Each category has different characteristics, advantages and disadvantages. Therefore, most of the RNGs used in practice are hybrid. Depending on the level of compromise in the choice of parameters, we distinguish between hybrid TRNGs and hybrid DRNGs.

DRNGs use deterministic, mostly cryptographic algorithms for generating numbers that are undisguishable from perfectly random numbers. Algorithms are selected in such a way that the

resulting generator is very fast and gives perfect statistical results (e.g. the generated numbers are uniformly distributed). However, it is clear that compared to an ideal random number generator, the requirement of independence of output numbers cannot be fulfilled in the case of deterministic generators.

On the other hand, generators of truly random numbers rely on some random process that cannot be controlled. Depending on the source of randomness, we recognize physical (PTRNGs) and non-physical (NPTRNG) random number generators. PTRNGs extract randomness from some physical (in electronic devices mostly electric) phenomena, such as thermal noise, metastability, metastable oscillations, chaotic behavior in electronic devices, random initialization of bi-stable circuits, etc. NPTRNGs extract randomness from unpredictable human-machine interactions, such as electronic mouse movements, frequency of keystrokes, etc.

Hybrid random number generators (HRNGs) are usually composed of a DRNG preceded by a TRNG and they take advantage of both building blocks: statistical quality and speed of DRNGs and unpredictability of TRNGs. Characteristics of the hybrid TRNG are essentially determined by those of the first true random number generator block, while the second deterministic block should ensure computational complexity in the case of entropy failure in the first block. On the other hand, characteristics of the hybrid DRNG are determined by the second deterministic block, which is seeded regularly by the source of entropy situated in the first block.

When evaluating different RNG principles, security is the main criteria. Security in RNG design is related to the statistical quality and unpredictability of the RNG output. Output of a good generator aimed at cryptographic applications must have statistical properties that are indistinguishable from those of an ideal RNG. This was the main classical RNG evaluation criterion (see Fig. 2.1). Besides evaluating statistical parameters, new approaches require evaluation of unpredictability. Unpredictability means that, knowing the current generator's output (or internal state in the case of DRNG), no preceding and following outputs can be guessed with non negligible probability.

Unpredictability of DRNGs is related to the computational complexity of the underlying algorithm, the length of the period, and the way the DRNG is initialized (entropy of the seed). Unpredictability of TRNGs comes from the physical source of randomness and it is related to the entropy rate in generated numbers: entropy rate per output bit equal to one guarantees that the generator output cannot be predicted.

The TRNG is typically composed of a digital noise source, which exploits some physical source of randomness, and an optional entropy conditioning block (see Fig. 2.1). The source of randomness, the digitization mechanism, and the entropy harvesting principle are very dependent on the selected technology and therefore a standard or even recommended TRNG does not exist. Depending on characteristics of the source of randomness and quality of the digital noise, designers select the entropy conditioning (also called post-processing) method aimed at enhancing statistical properties of generated numbers.

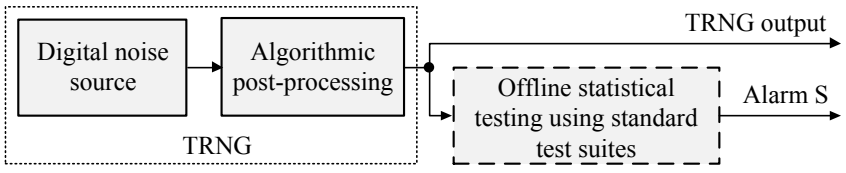

Figure 2.1: Classical TRNG design and evaluation approach

Historically, during the design and the security evaluation and certification process, the TRNG principle and its implementation were evaluated statistically: generated numbers were tested using standard test suites such as FIPS 140-1 [25], NIST SP 800-22 [59], DIEHARD [49], and DIEHARDER [12]. The generator was not certified for the practical use if the statistical tests did not succeed (i.e., if Alarm S from Fig. 2.1, was triggered).

This approach is not suitable for modern data security systems for several reasons: 1) the post-processing can mask considerable weaknesses of the source of randomness; 2) generic statistical tests can evaluate only the statistical quality of generated numbers and not their entropy (which should guarantee unpredictability as the main security objective); 3) high-end standard statistical tests are complex and thus expensive and slow, needing huge data sets. Consequently, they are only executed occasionally or on demand and only on selected sets of data of limited size.

German Federal Office for Information Security (BSI) recently proposed an evaluation methodology for physical random number generators (AIS 31) [39], which should help designers to better consider security aspects in their design and which should help evaluators of generators during the evaluation process. The AIS 31-compliant design and evaluation approach is depicted in Fig. 2.2.

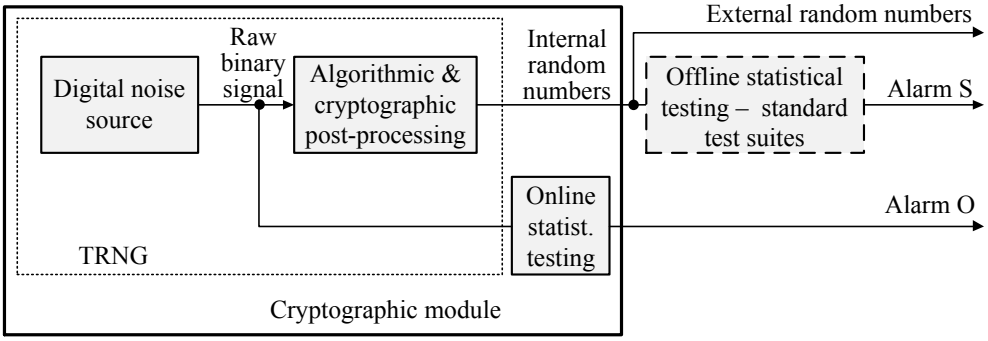

Figure 2.2: AIS 31 compliant TRNG design and evaluation approach

The US government is currently working on a similar standard – NIST SP 800-90B [4]. Compared to the classical approach and similarly to the NIST SP 800-90B approach, the AIS 31:

- Redefines the role of an optional algorithmic post-processing – it should correct some occasional small statistical imperfections of the raw binary signal (digital noise) and eventually increase entropy per bit (usually by some data compression method).
- Adds mandatory cryptographic post-processing to ensure unpredictability of generated numbers in forward and/or backward direction during a permanent or temporary failure of the entropy source for the highest security levels.
- Defines an online testing strategy by permanently executing a simple and fast total failure test and by executing at initialization and on demand an online test suite aimed at testing raw binary signal (preferably) or internal random numbers.
- Requires either the construction of a statistical model aimed at entropy estimation and

management or proof of robustness of the entropy source against variations of environmental conditions.

The entropy rate estimated from the model for standard operational conditions can vary in time. Several threats are related to the RNG design:

- Hardware related threats: Failure in generating good random numbers can be caused by component aging, variations in the manufacturing process, and unstable and/or manipulable entropy sources.
- Data leakage: An attacker might use data leaked from the random number generation process to compromise security of the whole system.
- Temporary failure in provisioning random output: Some design can output weak numbers before enough entropy is accumulated.

It is therefore important to test online the correct operation of the generator using some dedicated tests. As specified in AIS 31, these tests must be based on the stochastic model of the generator, to quickly detect generator-specific randomness failures (see Fig. 2.2). In order to further speed-up reactions of statistical tests, it is preferable to test the quality of the randomness as close to the source of randomness as possible.

Recently, a new enhanced AIS 31 compliant approach has been proposed in [26]. This extended security approach is depicted in Fig. 2.3. Compared to the AIS31 technique, the new

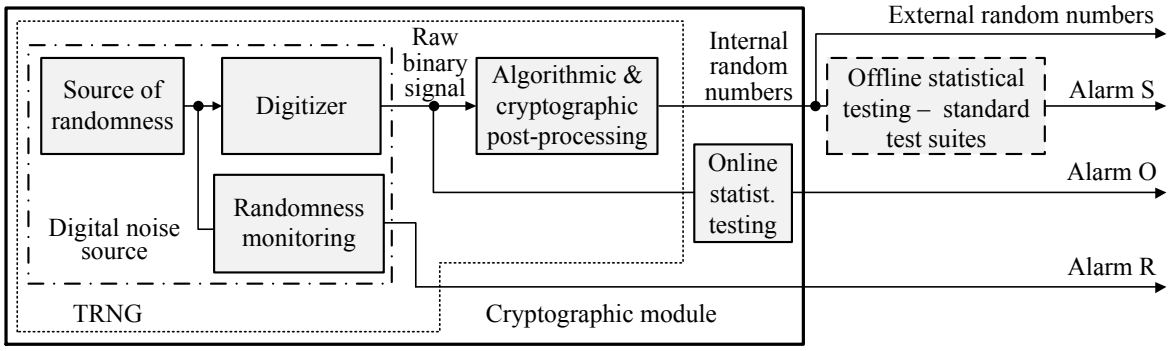

Figure 2.3: Extended security TRNG design and evaluation approach

extended security approach adds evaluation of the source of randomness inside the source of digital noise (before the digitization). This extension of the AIS 31 approach brings several benefits:

- The entropy source testing represents a dedicated statistical test that is better suited to the generator's properties. It can thus be simpler, faster, and, at the same time, more efficient than general purpose statistical tests, since it can better detect generator-specific weaknesses.
- The entropy source testing can be realized as an embedded measurement of a random physical parameter (e.g. thermal noise or phase jitter of a clock signal). The measured value can then be used as an input parameter of the stochastic model of the source of entropy and can serve for entropy estimation.

- The parameterized stochastic model of the source of entropy together with the description of the entropy extraction and algorithmic post-processing algorithms can be used to build a stochastic model of the complete TRNG, which can serve for precise entropy management ensuring the highest security level while maintaining the maximum obtainable bit rate.

2.1.2 Sources of Randomness in Logic Devices

Most cryptographic modules implement more or less complex algorithmic functions and protocols. Therefore, they are implemented in logic devices, such as microprocessors, ASICs, and FPGAs. However, RNGs and PUFs use some physical, mostly analog process as a source of randomness. Unfortunately, implementing analog electronic blocks in integrated circuits needs a mixed analog/digital technology, which is much more complex and expensive to exploit. Therefore, one of HECTOR's objectives is to exploit only a limited set of sources of randomness, which are available in logic devices and which do not need any analog electronic component or block.

The following set of sources of randomness is exploited the most frequently in practice:

- jitter in clock signals generated inside the logic devices,
- metastability of flip-flops and latches,
- oscillatory metastability in oscillating rings,
- initialization of flip-flops and memory elements to a random binary value.

In the next few paragraphs, we will discuss briefly these available sources of randomness and their suitability for the RNG design approach adopted in the HECTOR project.

Jitter of the Clock Signal as a Source of Randomness

Oscillator based TRNGs in logic devices use the fact that the analog random noise signals are transformed into the variation of the phase of the generated clock signal over time in the oscillator circuitry. In the frequency domain, this variation is observed as a phase noise and in the time domain as a phase jitter.

In the great majority of cases, the clock jitter has two components: *random jitter*, which is caused by some non-deterministic phenomena like thermal or flicker noise, and *deterministic jitter*, which is caused by a deterministic process.

As the name of the random jitter suggests, its behavior is random and the statistical tools (e.g. mean value, variance, standard deviation) are therefore often used to quantify it. In most cases, it obeys the central limit theorem and has a Gaussian probability distribution function (PDF).

The deterministic jitter is typically caused by variations in the power supply (e.g. from switching power supplies or other periodical signal sources), by cross-talks, by the electromagnetic interference (EMI), by a simultaneous switching of data signals and other regularly occurring interference signals.

The main confusion when one is confronted with jitter measurement and quantification is that some jitter metrics apply only to random jitter and not to deterministic jitter, whereas deterministic jitter is almost always present.

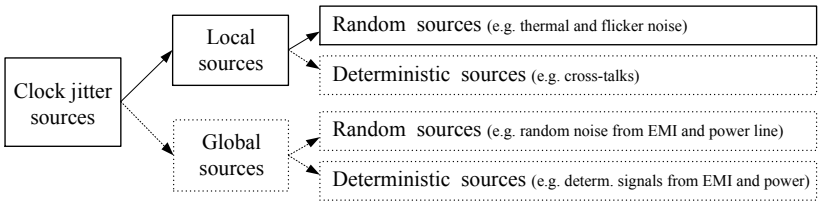

Figure 2.4: Sources of the clock jitter in logic devices, the sources in the dashed boxes should be avoided as they can be used to attack generators

Clock jitter can be caused by local and global sources, as shown in Fig. 2.4. The global jitter sources, which can be random (e.g. the noise of the power supply) or deterministic (e.g. external electromagnetic emanations), affect all transistors in the device in the same way and are thus easier to reduce. The local jitter sources, also random (e.g. thermal noise or flicker noise) or deterministic (e.g. data dependent cross-talks), affect all the transistors differently and are thus more difficult to eliminate.

Global jitter sources are very dangerous, because they are easy to manipulate. Fortunately, the impact of the global jitter sources on the generated random numbers can be reduced using the fact that identical blocks will be affected in the same way. Therefore, by using a differential TRNG design, e.g. by exploiting pairs of identical clock generators and differentiating their output, the effect of global jitter sources can be significantly reduced [27].

The impact of the local deterministic sources on the generated numbers can be reduced only in carefully designed generators. The oscillators used as sources of jittery clock signals must be physically isolated from the rest of the device as well as from each other, the interconnection wires must be as thin and as short as possible to avoid capacitive and inductive coupling. Unfortunately, this is very difficult to achieve in FPGAs and sometimes also in ASICs.

Metastability of Flip-Flops and Latches as a Source of Randomness

Metastability can be defined as the ability of an unstable equilibrium electronic state to persist for an indefinite period in a digital system. The metastable events are very dangerous, since they can cause the whole system to stall in an unknown state for a random time period.

The metastability of a flip-flop or latch can be caused by sampling the input signal exactly during its rising or falling edge. Entering a metastable state causes two random phenomena to occur: 1) the duration of the metastable state (the settling time) is random; 2) the stable state, to which the metastable state resolves, has a random value. For the risk-free operation of the logic device, the metastable states must be avoided, i.e. the input signal must be sampled outside the setup and hold time interval defined by the vendor for each device family or after the settling time.

RNGs claiming to exploit metastability, force the sampling to act inside the setup and hold interval. Although in that case the metastable events are inevitable, they can be very rare and therefore, they are not suitable as a fast source of randomness. Although some papers claiming to use metastability obtain the output bit rate of several Mbits/s, they rather use the noise, which appears during the sampling operation and which determines the state to which the flip-flop recovers after the settling time.

Oscillatory Metastability

Oscillatory metastability can be defined as the ability of a bi-stable circuit (e.g. an RS flip-flop) to oscillate for an indefinite period. It was studied by Reyneri *et al.* in [57]. The oscillatory metastability is very suitable as the source of randomness for two reasons: the oscillating circuits can be very small and the random event (randomly lasting oscillations) can be easily transformed to random numbers by a simple counter of oscillations.

Initialization of Flip-Flops or Memory Elements to a Random Value

Flip-flops and memory elements based on flip-flops (volatile memory elements) can be initialized systematically to one, to zero, or to some random value. Proportion of this three initialization states depends on the technology and the design of the flip-flop. This kind of source of randomness is not suitable for FPGAs, because all available flip-flops are initialized to a known state, but it can be in some circumstances (if the flip-flop is designed so that the random initialization state dominates and the obtained values are not biased) exploited in ASICs. In order to obtain high output bit rate, the flip-flop can be initialized periodically, as it is made in the latest Intel RNG [66].

Single-Event Ring Oscillators as Sources of Jittery Clocks

A single-event ring oscillator is a loop built of logic gates containing an odd number of inverting gates, mostly inverters, and any number of non-inverting gates. Two kinds of ring oscillators are used as sources of randomness (see Fig. 2.5): the conventional free running ring oscillator in the left panel is composed of an odd number of inverters; the ring oscillator with a control input (in the right panel) is composed of a NAND gate and a sufficient number of non-inverting logic gates or buffers used as delay elements.

When the ring is oscillating (oscillations in the controlled ring from the right panel in Fig. 2.5 must be enabled), the output of each inverting gate (the inverter on the left or the NAND gate output on the right side) toggles periodically between two logic states. The main feature of the single-event ring oscillator is that only one event (i.e. a rising or falling edge of the generated periodic signal) is present in the ring at any given time.

The period of a single-event ring oscillator is equal to the time, which the event needs to pass two times across the ring (first as the rising and then as the falling edge of the generated signal or vice versa).

In the simplified noise-free model with ideal interconnections (noise-free, zero impedance connections) and constant delay of all delay elements, the period of the generated signal is

$$T = 2 \cdot \sum_{i=1}^{n} d_i,$$

where d_i is the delay of the i-th element of the ring and n is the number of elements.

In the real world, elements of the ring do not have constant delays, their delays vary individually depending on local and global jitter sources causing the clock jitter.

Today, conventional ring oscillators are the most commonly used sources of randomness. Their success is certainly due to the simplicity of their implementation in logic devices. Their main disadvantage is the strong frequency dependence on temperature and on power supply voltage, which makes them vulnerable to external manipulations. Moreover, it was shown

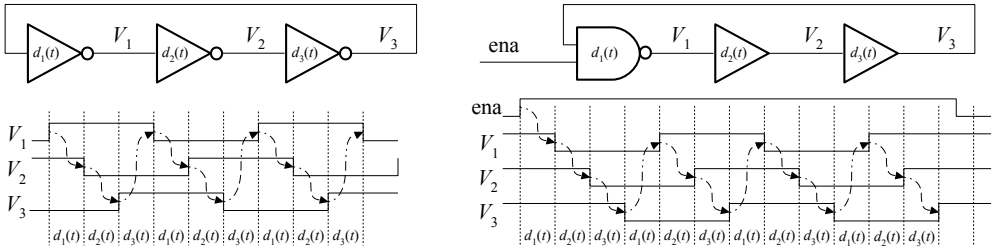

Figure 2.5: Free running, single-event ring oscillator (left panel) and single-event ring oscillator with a control input (right panel): in both cases, only one event crosses the ring at any time

in [9] that in some circumstances, ring oscillators can lock to each other due to cross-talks. This mutual locking can be fatal for the entropy harvesting mechanism. Jitter sources in ring oscillators are well studied and modeled, but because of shrinking submicron technologies, the models have to continuously evolve.

Multi-Event Ring Oscillators with Signal Collisions

A multi-event ring oscillator with collisions is a loop built of logic gates, which contains an even number of inverting gates and any number of non-inverting gates. Because of the even number of inverting gates, the oscillator must be restarted regularly – the multiple events created after each restart circulate inside the loop until a collision occurs, during which the edge which moves faster reaches the slower one. The difference in speed of circulating events is caused by differences in delays in loop branches between inverters and by analog phenomena in the inverter circuitry. The circulating events create temporary oscillations which disappear after the last collision.

The most common configuration of the multi-event ring oscillator with collisions is the transition effect ring oscillator (TERO), which features the so-called oscillatory metastability, in which oscillations last randomly, theoretically sometimes infinitely [57].

The TERO is a loop composed of an even number of inverters and a couple of gates which are used to restart temporary oscillations (e.g. two NAND, two NOR, or two XOR gates). A typical TERO configuration is presented in the left panel in Fig. 2.6: it is composed of two NAND gates and two inverter chains, each inverter chain containing an even number of inverters. The TERO can be seen as an RS latch with two inputs controlled by the same signal V_{ctr} and two outputs V_{out1} and V_{out2}.

Following the rising edge of the V_{ctr} input, the outputs V_{out1} and V_{out2} start to oscillate. The oscillations have a constant mean frequency, but their duty cycle varies over time: it changes monotonously and after a certain number of oscillations, it reaches the rate of either 0% or 100%. At this point, outputs V_{out1} and V_{out2} stop oscillating and remain stable at two opposite logic values. The right panel in Fig. 2.6 presents traces of the V_{ctr} input and V_{out1} output signals captured from the oscilloscope.

The three zooms presented in this panel reveal the varying duty cycle: immediately after the rising edge of the V_{ctr} signal, it is close to 50%, after which it decreases (as the faster edge catches up with the slower one) until it reaches 0%. Consequently, the signal V_{out1} stabilizes at logic level 0. Of course, as far as the duty cycle is concerned, the signal V_{out2} behaves in the opposite way and stabilizes at logic level 1.

The number of oscillations before the outputs stabilize is not constant but variable because

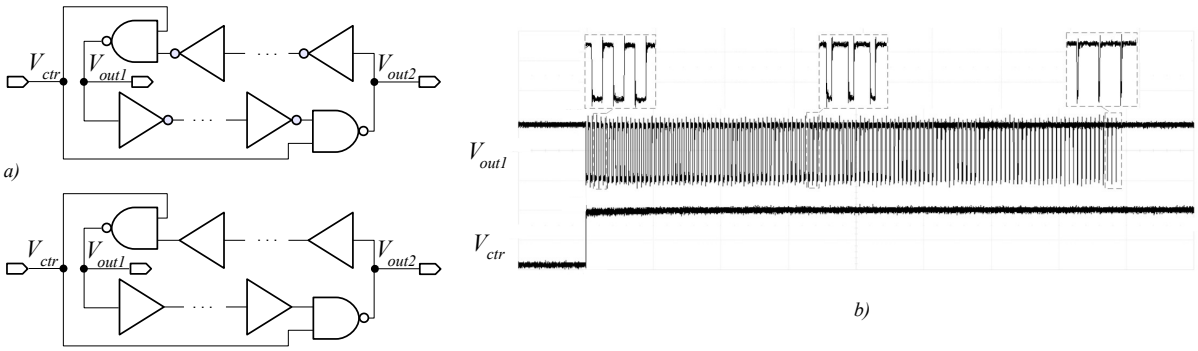

Figure 2.6: Circuit diagram of typical TERO structures based on inverters and buffers (a) and their input/output waveforms (b)

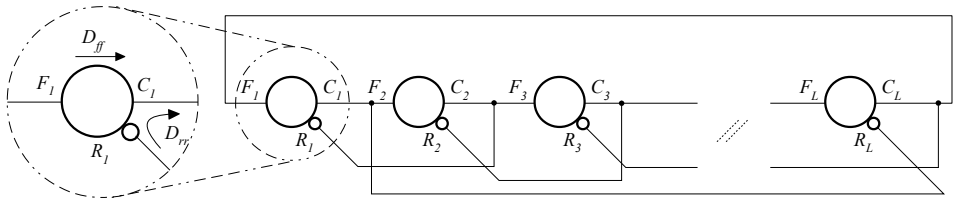

Figure 2.7: Architecture of a self-timed ring

it is affected by the electronic noises which disturb the normal behavior of transistors in the TERO structure, which in turn dynamically alter the delays of individual delay elements.

Multi-Event Ring Oscillators without Signal Collisions

An example of the multi-event ring oscillator without signal collisions is the self-timed ring (STR). STRs are oscillators in which several events (electrical transitions) can propagate evenly-spaced in time thanks to analog mechanisms, which act in each STR stage. The STR structure is depicted in Fig. 2.7. It corresponds to the control circuit of an asynchronous micropipeline as proposed by I. E. Sutherland in [65], which has been closed to form a ring.

The STR is composed of L stages, each consisting of a Muller gate and an inverter. F_i is the forward input of the i-th stage, associated with a forward static delay D_{ff}, R_i is the reverse input of the same stage, associated with a reverse static delay D_{rr}, and C_i is the output of the stage. As is clear from the truth table of one self-timed ring stage presented in Table 2.1, the forward input value is written to the stage output if the forward and reverse input values are different. Otherwise, the previous output is maintained.

The STR stages communicate using the two-phase handshake protocol described in [65]. Each request and acknowledgment signifies the transfer of an event between interconnected stages. In contrast to inverter ring oscillators, several events can propagate without colliding thanks to this handshake protocol, which enables precise, built-in frequency and phase control of the internal clock signals by setting up the appropriate number of propagating events when the ring is initialized.

F_i	R_i	C_i
0	0	unchanged
0	1	0
1	0	1
1	1	unchanged

Table 2.1: Truth table of a stage of the self-timed ring

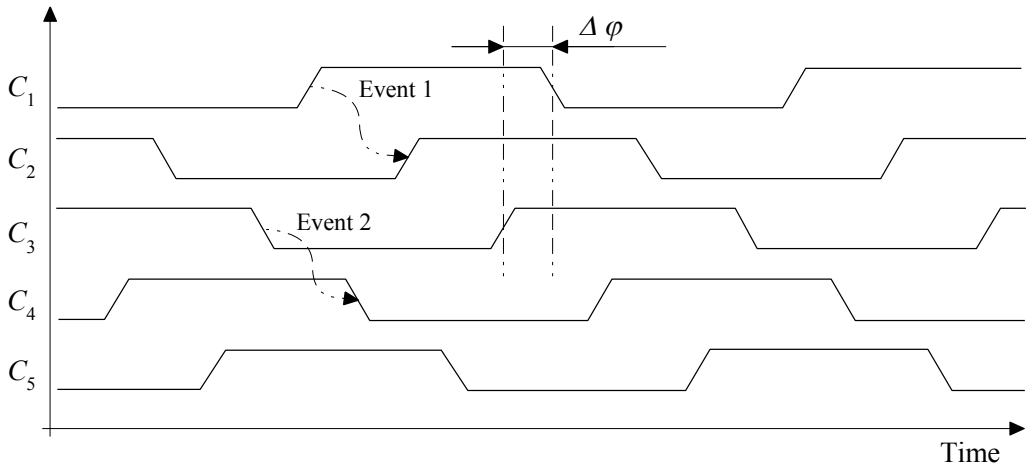

Figure 2.8: Illustration of the evenly-spaced propagation of two events in a 5-stage STR

In practice, the ring is initialized with N events which start propagating during a transient state. Independently of their initial positions and thanks to two analog mechanisms inferred in the ring (the Charlie and the drafting effects), they end up in a steady state in which they arrange themselves in one of two ways: they either form a cluster which propagates in the ring (burst oscillation mode), or they spread out around the ring and propagate with a constant temporal spacing (evenly-spaced oscillation mode). Both these oscillation modes are stable and depend on the static parameters of the ring (mainly the ring occupancy $N/(L-N)$ with respect to the ratio of static delays D_{ff}/D_{rr}). Fig. 2.8 shows the waveforms of the evenly-spaced propagation of two events in a 5-stage STR.

Charlie diagrams were first used by Ebergen *et al.* to predict the timing behavior of a Muller gate as a function of the separation time between the input events which drive the gate [19]. Winstanley *et al.* carried on the study by introducing another analog effect which affects the timings of events in STR stages – the drafting effect [73].

The Charlie effect determines the impact of the separation time between input events on a Muller gate delay: the closer the arrival times of events at inputs, the longer the gate propagation delay. The drafting effect describes the impact of the elapsed time from the last output commutation on the stage propagation delay: the shorter this time, the shorter the stage propagation delay.

In [22], S. Fairbanks *et al.* showed that the Charlie effect promotes the evenly-spaced propagation mode while the drafting effect promotes the burst propagation mode. The Charlie effect causes two close events to push away from each other due to the increased delay experienced by a ring stage when driven by two events separated with a short time lapse.

If several events are constrained in a short ring structure, the Charlie effect is retroactive: events keep pushing away from each other until they are evenly spread out across the ring,

which causes the evenly-spaced propagation mode of the STR.

On the other hand, the drafting effect causes two events to approach each other because of the reduced propagation delay of a ring stage when it switches faster, thereby promoting the burst propagation mode of the STR.

In practice, the evenly-spaced mode is obtained for a range of events centered around N_0 which fulfills the following relation [31]:

$$\frac{N_0}{L - N_0} = \frac{D_{ff}}{D_{rr}} \tag{2.1}$$

The burst oscillation mode is obtained for corner values of the number of events. The stronger the Charlie effect, the longer the interval between events when the ring achieves the evenly-spaced propagation mode. For example, a 64-stage self-timed ring in Altera Cyclone III with $\frac{D_{ff}}{D_{rr}} \simeq 1$ in [13] used the evenly-spaced mode for N varying between 22 and 42, while the same 64-stage configuration in Xilinx Virtex 5 achieved the evenly-spaced mode for N between 28 and 38. This suggests the Charlie effect is stronger in the Cyclone III STR implementation.

In contrast to inverter ring oscillators, the frequency of an STR in the evenly-spaced regime is a function of its occupancy and not of the number of its stages. The frequency of oscillations increases with number of events N, then starts to drop when the number of free stages is less than the number of events to be processed with respect to the asymmetry of the ring stages. The maximum frequency is reached when the number of events is equal to N_0 from Eq. (2.1).

Conversely to inverter ring oscillators, STRs can reach phase resolutions which are fractions of the propagation delay of a single logic gate. Fig. 2.8 shows how a phase resolution below to the propagation delay of a single ring stage can be obtained using the STR. The propagation of an event in an STR causes a 180 degree phase shift of the oscillating signal. If N events are confined in L stages and spread evenly around the ring, the phase shift between two stages separated by n stages is [22]:

$$\varphi_n = n \times \frac{N}{L} \times 180 \tag{2.2}$$

Therefore, if the number of events is a factor of the number of stages, some stages may exhibit the same absolute phase. But if the number of events and the number of stages are co-prime, the STR exhibits as many different equidistant phases as the number of stages. In this case, if T is the oscillation period, the phase resolution can be expressed as follows [13]:

$$\triangle\varphi = \frac{T}{2L} \tag{2.3}$$

On the other hand, the oscillation period of an STR is a function of its occupancy rather than of the number of its stages. This means that it is possible to increase the number of ring stages (L) while keeping a constant frequency. Consequently, the phase resolution of an STR can theoretically be set as finely as needed. Elissati *et al.* demonstrated the efficiency of the method in [20] by implementing several designs and obtained phase resolutions in the order of picoseconds.

2.1.3 Randomness Extraction

Entropy harvesting methods are used to obtain digital noise from the clock signals. In general, two principles can be used: *sampling* of a jittery clock signal and *counting* the edges of the

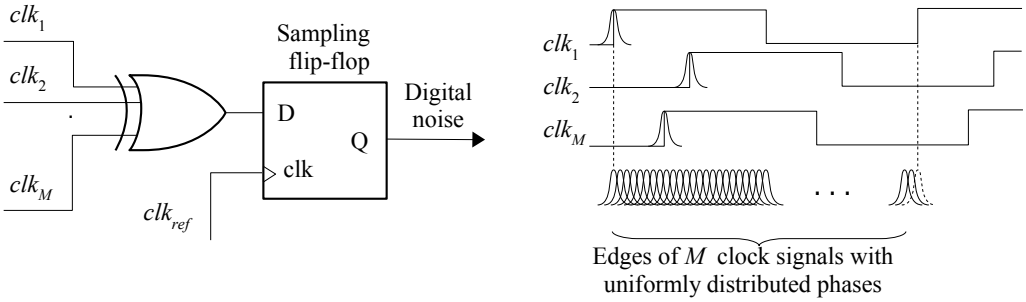

Figure 2.9: Entropy harvesting by sampling M jittery clock signals

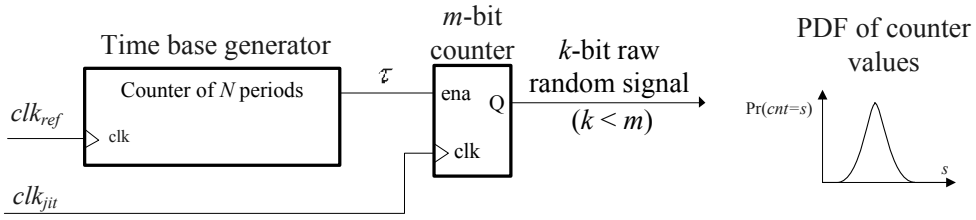

Figure 2.10: Entropy harvesting by counting the periods of the jittery clock signal

jittery clock signal during a fixed time interval. Although the two principles may seem similar, their stochastic models can considerably differ.

The principle of the entropy harvesting method based on sampling of jittery clock signals is presented in Fig. 2.9. It uses M jittery clock signals as M sources of randomness. The clock signals are added modulo 2 in a XOR gate before being sampled at the reference clock in a D flip-flop. Although the phases of the clock signals are uniformly distributed, the frequency of the reference clock signal and hence the output bit rate of the generator can be very high comparing to the agility (spectrum) of the jitter. Another practical limit of the output bit rate is the bandwidth of the XOR gate (many high-frequency signals have to be XOR-ed). However, this problem can be solved by synchronizing the clock signals generated in rings on the reference clock in additional D flip-flops connected between rings and the XOR gate (not depicted in Fig. 2.9).

The entropy harvesting method based on counting the jittery clock periods is presented in Fig. 2.10. This method uses two clock signals: a reference clock signal and a jittery clock signal. In practice, both signals include some jitter, but if they come from independent generators, the jitter of the reference clock can be added to the jitter of the jittery clock signal and the reference clock can then be assumed to be jitter free. The time base generator generates the time interval τ which enables the m-bit counter. Because of period instability of the jittery clock signal, the counter values vary after each measurement interval τ as depicted in the right panel in Fig. 2.10.

2.1.4 Evaluation and Testing of RNGs

RNGs must be thoroughly evaluated from the point of view of security during their design, but also during the certification procedure according to AIS 20/31 [39], or NIST SP 800-90 [4].

Evaluation of the Security and Entropy Estimation

Security in random number generation is related to the unpredictability of the generated numbers. The unpredictability is characterized by the entropy, which is a measure of the guesswork of the attacker. In order to guarantee unpredictability, the entropy rate of an n-bit random vector should be as close as possible to n.

Several definitions of entropy exist. The most general entropy definition is that of Rényi [56]. The Rényi entropy is defined as follows:

$$H_\alpha(X) = \frac{1}{1-\alpha} \log_2 \left(\sum_{i=1}^{n} (p_i)^\alpha \right), \tag{2.4}$$

where $\alpha \geq 0$, $\alpha \neq 1$ and $p_i = \Pr[X = x_i]$. In general, the Rényi entropy is difficult to estimate in practice. Instead, two special cases of the Rényi entropy are used: the min-entropy and the Shannon entropy.

The *min-entropy* is the most conservative entropy measure. It is based on the fact that the Rényi entropy is monotonously decreasing with α. Consequently, it reaches the minimum value (the min-entropy) for $\alpha \to \infty$. The min-entropy is thus defined as follows:

$$H_\infty(X) = \inf_{i=1..n} (-\log_2(p_i)) = -\log_2 \sup_{i=1..n} p_i. \tag{2.5}$$

The conservativeness of the min-entropy can be useful for guaranteeing security, however, its definition as presented in Eq. (2.5) is valid only for independent variables. Therefore, when using min-entropy as the entropy measure, the designer must thoroughly evaluate independence of the generated values.

The most common entropy definition is the *Shannon entropy*. It can be obtained from the Rényi entropy for $\alpha \to 1$:

$$H(X) = H_1(X) = -\sum_{i=1}^{n} p_i \log_2 p_i \tag{2.6}$$

Similarly to min-entropy, Eq. (2.6) is valid directly only if the generated random numbers are independent. If the generated numbers are somehow mutually dependent, the so-called conditional entropy based on the Shanon entropy definition can be used [39].

The probability of an n-bit vector with a nearly uniform distribution should be close to $\Pr(X_1 = x_1, X_2 = x_2, ..., X_n = x_n) = 1/2^n$. As stated before, the entropy of this vector is close to n. Consequently, the entropy per bit of a TRNG should be close to 1 (according to AIS31, for internal random numbers, $H(X) > 0.997$).

A high entropy rate guarantees that the preceding or succeeding bits cannot be guessed with a probability different from 0.5. Random variables, which are independent and uniformly distributed, feature the so-called *full entropy* (ideal RNG outputs full entropy numbers).

Unfortunately, the entropy is the property of random variables and not that of observed realizations. Consequently, the entropy cannot be directly measured. In a mathematically stringent approach, the entropy should be estimated using the stochastic model of the generator.

Two different approaches are applied in Europe and in USA for entropy estimation. The European AIS 20/31 [39] requires designer to construct a stochastic model and to use it for entropy estimation. On the contrary, the draft of the American standard NIST SP 800-90B uses a set of simple statistical tests to estimate the min-entropy. This second approach can be security critical, especially if the generated numbers are not independent (e.g. they include some pseudo-randomness).

Stochastic Modeling of Random Number Generators

The stochastic model specifies the family of probability distributions that contains all possible distributions of the generated random numbers. The stochastic model can include or not the post-processing function. The main objective of the stochastic model is to characterize probability that an output bit is equal to one ($\Pr(X = 1)$), and/or probability that an n-bit vector features some pattern ($\Pr(X_1 = x_1, X_2 = x_2, ..., X_n = x_n)$) and from them the entropy (if the variables are independent and identically distributed – IID) or conditional entropy (if the variables are not IID) per output number.

When constructing or verifying the stochastic model, the designer and the evaluator must:

- observe if the generated numbers are stationary, i.e. if their statistical parameters do not change in time,
- observe the distribution of random numbers in different extreme and corner conditions in order to verify that the generator is robust to variations of operational conditions if they vary in acceptable ranges,
- observe if the random variables are IID or not,
- characterize (or verify) the distribution of generated random numbers – the stochastic model.

The stochastic model has double utility in exploitation of RNGs: 1) it serves for characterizing unpredictability of the generated numbers (the entropy); 2) it serves as a basis for construction of RNG-specific statistical tests (i.e. dedicated tests).

If the generated random numbers are IID, the min-entropy or Shannon entropy can be used to characterize unpredictability. If the generated numbers are not IID, the conditional entropy should be used instead.

Evaluation of the Statistical Quality of Generated Numbers

During the RNG design, during the security certification process, and also when the generator is in operation, the RNGs must be thoroughly tested using statistical tests. The statistical tests are mathematical tools aimed at evaluation of the statistical quality of generated numbers.

Following the common strategy in statistical testing, different statistical features of an *ideal RNG* are evaluated and then compared with output values of a real RNG (i.e. design under test – DUT) for verification of the so-called *null hypothesis* – H_0. The null hypothesis is the assumption that the generator is indeed random, i.e. that it behaves as an ideal RNG.

The number of more or less complex statistical features, which could be evaluated, is theoretically unlimited. For practical reasons, usually only up to 16 features are tested. The most common general-purpose statistical test suites (also called the black box tests) are FIPS 140-1 [25], NIST SP 800-22 [59], DIEHARD [49], and DIEHARDER [12].

Recall that the statistical testing of the generator is necessary, but not sufficient for the thorough security evaluation – it cannot substitute the cryptanalysis in the case of DRNGs and the estimation of the entropy rate in the case of TRNGs. The statistical testing is performed in two phases of the RNG exploitation: offline and online.

The RNG is tested *offline* during the design and security evaluation/certification process (by developers and testers). This kind of testing is performed using testing procedures required by

security standards (i.e. AIS 20/31 [39] or NIST SP 800-90B [4]) and optionally also using the general purpose (black box) statistical tests.

The AIS20/31 testing strategy requires two groups of tests to be executed in two testing procedures: Procedure A and Procedure B. In Procedure A, tests T0 to T5 are applied on generated random numbers (so-called internal random numbers). In Procedure B, tests T6 to T8 are applied either on raw random numbers (if they are available) or on internal random numbers.

The NIST SP 800-90B uses two groups of tests, too. The aim of the first group is to verify that the generated numbers are IID and in the positive case to estimate min-entropy using Eq. 2.5. If the generated numbers are not IID, the second group of tests is used for entropy estimation. The most conservative entropy estimation is then taken as a result of testing.

According to recent security standards [39], [4], the RNG must also be tested *online*, to detect dynamically intolerable changes in quality of generated numbers. The online testing is performed using embedded, RNG-specific, statistical tests based on a suitable stochastic model. Three kinds of dedicated online tests must be performed:

- Continuous test(s),
- Online test(s),
- Startup test(s).

The coarse *continuous tests* must be able to rapidly detect important deviations from the normal RNG operation, e.g. to detect the total entropy failure. If the test fails, the generator must stop the generation of random numbers immediately.

The continuous test must have low probability of the false alarm, because each alert causes the full restart of the generator, including the time consuming startup tests. Consequently, in the worst case, i.e. when the alarm is triggered too often, the generator could be permanently disabled. The speed and the reliability of the continuous tests can be significantly increased if the tests were based on an appropriate stochastic model – the test could evaluate the most important statistical features of the generator.

More precise *online tests* are aimed at detection of subtle entropy failures. The level of failures that can be accepted is determined by the post-processing mechanism (see the following subsection). Due to the higher precision of online tests, their execution time is much higher than that of the continuous tests.

Consequently, the time interval between the entropy failure and the alarm is also much longer. During this time interval, the correct operation of the generator must be ensured by some backup solution: the generator must behave as a pseudo-random number generator or some sufficient entropy amount must be accumulated and delivered from an internal buffer. Of course, in order to increase the speed and the precision of the online tests, they should be based on the stochastic model of the generator, too.

The *startup tests* are launched at each initialization of the generator and after each security alert (e.g. because of alarms coming from the continuous tests or from the online tests). During startup tests, first, the operation of the continuous tests and of the online tests must be tested (the tests themselves must be tested), then, they must be executed at least once, and finally, the algorithmic part of the generator (e.g. the randomness harvesting mechanism and the post-processing block) must be tested, too.

2.1.5 Post-Processing of the Raw Random Numbers

As explained in Section 2.1.1, the post-processing of the raw random numbers plays a double role in the RNG design: 1) the algorithmic post-processing block corrects some occasional small statistical imperfections of the raw random numbers and eventually increase entropy per bit; 2) the cryptographic post-processing function ensures temporarily the unpredictability of generated numbers in forward and/or backward direction, before the online tests detected some previously appeared failure of the entropy source.

Algorithmic Post-Processing

The role of the entropy extractor as a general kind of the algorithmic post-processing function is to transform generated numbers featuring an imperfect distribution to numbers featuring (nearly) uniform distribution. The closeness of the distribution of generated numbers to that of an ideal one (i.e. to the uniform distribution) can be evaluated using a *statistical distance* defined as:

$$\Delta(X, Y) = \max_{T \subset S} |\Pr[X \in T] - \Pr[Y \in T]|, \tag{2.7}$$

where X and Y are random variables having different probability distributions. Then, we say that X and Y are ε-*close* if $\Delta(X, Y) \leq \varepsilon$.

The deterministic entropy extractor (called (k, ε)-*extractor*) takes an n-bit sample from a weak random source as an input and gives an m-bit output ($n > m$) that is statistically ε-close to the uniform distribution U_m. In other words, the m-bit vector present at the output of the post-processing has the min-entropy k (i.e. m is close to k).

The entropy extractor is usually realized by some data compression method. The compression factor can be determined from the known distribution of the raw random numbers at the input of the post-processing function and the required ε-closeness to the uniform distribution. The value of ε can be computed from the required lower bound of the min-entropy and Equation (2.5).

Cryptographic Post-Processing

As explained before, the cryptographic post-processing ensures unpredictability of the generator output, when the source of entropy fails. The cryptographic post-processing block should use an approved cryptographic algorithm depending on the required security level (direction of unpredictability) and according to rules defined for cryptographically secure pseudo-random number generators [39].

2.1.6 Conclusion

It is clear that unlike algorithmic cryptographic primitives, TRNG principles cannot be standardized. However, for practical reasons, it could be very useful if several principles that are compliant with the new enhanced security approach could be pre-selected and some convenient stochastic models and embedded tests developed for them. Moreover, according to entropy estimation, a suitable post-processing method should be proposed. The role of the TRNG designer would then be simple:

- Depending on performance and security requirements, implement one generator principle in selected technology.

- Evaluate the amount of randomness in the given technology using existing embedded tests of entropy (note that pre-selected TRNG principles must have suitable embedded tests/randomness measurement available).
- Using the measured parameter, estimate the entropy per bit of the raw binary signal using an existing stochastic model of the source of randomness and select suitable entropy extraction and algorithmic post-processing methods (from a set of options).
- Depending on the stochastic model of the source of randomness, entropy extraction and postprocessing algorithm selection, estimate the entropy per bit (pattern) at the generators output.
- Evaluate the generator using the AIS 31 methodology and known entropy.

2.2 Physical Unclonable Functions

For the design of digital electronic systems, two possible techniques exist that enable the traceability of integrated circuits using intrinsic identification: fingerprinting and watermarking. Fingerprinting is the measurement of a physical or behavioral characteristic of an integrated circuit. This characteristic allows the designer to identify each integrated circuit individually. Watermarking is a technique of steganography which proves the ownership of an integrated circuit (or an IP block) by checking for the presence of hidden information, called a watermark.

A new approach to fingerprinting electronic devices emerged recently and has attracted wide attention in the last few years. The new cryptographic primitive aims to physically identify hardware systems, instead of giving them an explicitly programmed digital identity. The concept of physical unclonable functions (PUFs) was first introduced by Pappu in [54].

2.2.1 PUF Design

Silicon PUFs can extract unique secret keys from the physical characteristics of the device using a challenge and response procedure based on a physical interaction that is extremely hard or even impossible to reproduce. Entropy is derived from a physical random variable such as the mismatch between transistor attributes (length, width, oxide thickness, etc.) caused by variability of the manufacturing process (MPV).

The basic principle is that MPV is neither controllable (it is not predictable) nor reproducible, but can be measured. Ideally, when a PUF is challenged, its response is unique (each device has a unique, non-reproducible response based on its unique physical characteristics), random (it is uniformly distributed and cannot be predicted), steady (each device always gives the same response to a given challenge) and in some cases tamper resistant (probing the PUF changes its physical behavior and hence the response obtained). Figure Fig. 2.11 is an illustration of the behavior of a silicon PUF.

Many silicon based PUF architectures exist, but two main approaches are used to extract entropy from MPV in digital devices: methods based on the measurement or comparison of timing, and methods that exploit the resolution from a metastable state.

The arbiter PUF [42] relies on the race between two events (electrical transitions) in two identical delay lines. The ring oscillator based PUF [63] (RO-PUF) leverages the frequency mismatch between several identically designed ring oscillators (ROs). SRAM-PUFs [36] and

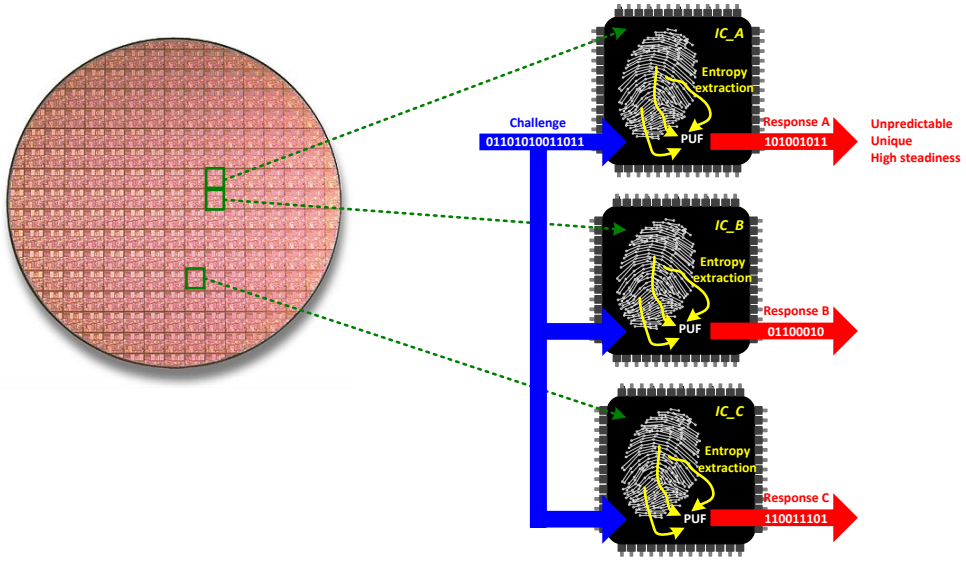

Figure 2.11: Illustration of the behavior of PUF: three chips from the same wafer embed the same PUF, they are packaged in three integrated circuits, and after receiving the same challenge, each PUF provides a unique, unpredictable, and steady response because of variations in the CMOS production process

butterfly PUFs [41] rely on the settling state of cross-coupled elements: at the initialization of a SRAM, most cells' outputs are biased toward 1 or 0 depending on the MPV.

2.2.2 Statistical Evaluation of PUFs

Concerning statistical evaluation of PUFs, comparing to TRNG evaluation, the situation is much more complex. While TRNGs can generate huge amount of data, the evaluation of PUFs needs huge amount of PUF implementations (devices). Clearly, this is practically never the case. Practical testing is thus often replaced by simulations.

The statistical evaluation of a PUF must consider at least three factors: inter-device variation, intra-device variation obtained at nominal and corner temperatures and power supply voltages, and randomness of the response. They are defined as follows:

- *Inter-device variation* (also called *uniqueness* [48], [37]) shows to what extent the responses generated by the PUF in different devices are unique.
- *Intra-device variation* (also called *reliability* [48], *robustness* or *steadiness* [37]) quantifies changes at the output of a PUF function over many measurements with environmental changes (i.e. temperature variation, VDD variation).
- *Randomness* is evaluated by statistically testing the PUF response (often only the bias of the response is evaluated).

In the following, let us consider a set of N devices denoted $(d_i)_{1\leq i\leq N}$, and let n-bit responses be extracted L times from each device. We denote $r_{i_j}(p)$ (with $1 \leq i \leq N$ and $1 \leq j \leq L$) the j-th response of the device d_i to the challenge c_p.

Inter-Device Variation

For a given challenge c_p, the responses from two devices d_i and d_j must differ with high probability ($r_{i_l}(p) \neq r_{j_m}(p)$ with $1 \leq l, m \leq L$). One common indicator of this characteristic is the inter-chip Hamming distance, which is computed using the following equation:

$$EC = \frac{1}{N(N-1)L} \sum_{i=1}^{N} \sum_{k=1, k\neq i}^{N} \sum_{j=1}^{L} \frac{HD(r_{i_j}(p), \bar{r_k}(p))}{n} \times 100\% \tag{2.8}$$

where $r_{i_j}(p)$ is the j-th response obtained from the device d_i, $\bar{r_k}(p)$ is the mean value of L responses obtained from device d_k, n is the size of the response vectors, and HD is the Hamming distance between two vectors.

The optimal value of this indicator is 50%. In the following, we refer to this indicator simply as to uniqueness. In practical PUF applications, the uniqueness determines the size of identifiers needed to distinguish between certain number of devices.

Intra-Device Variation

For a given challenge c_p, which is repeated several times, all the responses of device d_i should be the same (i.e. response $r_{i_j}(p)$ should not vary over time). A common indicator used to characterize the steadiness (also called reliability or robustness) is the intra-chip Hamming distance. For device d_i operating at temperature T and power supply voltage V, it is computed using the following equation:

$$IC_i(T, V) = \frac{1}{L} \sum_{j=1}^{L} \frac{HD(r_{i_j}(p), r_{ref}(p))}{n} \times 100\%, \tag{2.9}$$

where $r_{ref}(p)$ is a reference response (associated with the challenge c_p and the device d_i) obtained at nominal voltage V_n and temperature T_n.

The optimal value of this indicator is 0%. In the following, we refer to this indicator simply as to steadiness. Low steadiness values mean that the PUF is reliable. In practice, the PUF steadiness determines the cost of error correction needed to obtain reliable identifiers and/or the voltage and temperature ranges for the proper PUF operation.

Randomness

For a given challenge c_p, the responses $r_{i_j}(p)$ (with $1 \leq j \leq L$) should be unpredictable and uniformly distributed. In practice, the randomness determines the vulnerability of the PUF to brute force and modeling attacks. Evaluating randomness of PUFs can be challenging, because large amount of data (and therefore large number of test chips) is required. However, some statistical tests issued from the NIST SP 800-22 test suite [59] can be adapted to test small amount of data, although these tests will consequently have a low confidence level. The main objective of this evaluation is to rapidly eliminate responses which are not random. On the other hand, the designer wants to evaluate randomness, which comes exclusively from the MPV. He can for example merge sufficient number of PUF responses to obtain a longer sequence, which can then be tested using six statistical tests (depending on parameters, which define the minimal length of the input [59]):

- T1 is the frequency (monobit) test, which evaluates the bias of the sequence,

- T2 is the frequency test within a block (for 2-bit, 3-bit and 4-bit blocks),
- T3 is the cumulative sums test,
- T4 is the runs tests (which evaluates the distribution of runs of ones),
- T5 is the longest run test (the longest run of ones is searched),
- T6 is the approximate entropy test which evaluates the distribution of M-bit overlapping blocks in the sequences (with $M = 2, M = 3$ and $M = 4$).

In next steps, the designer can estimate the overall bias of the responses using the Shannon entropy. Here again, responses have to be filtered to remove the noise and to estimate entropy coming from the MPV and not from the random noise. The Shannon entropy H is computed as follows:

$$\bar{H} = \frac{1}{n \times N} \sum_{k=1}^{n} \sum_{i=1}^{N} -p_{k,i} log_2(p_{k,i}) - (1 - p_{k,i}) log_2(1 - p_{k,i}) \tag{2.10}$$

where $p_{k,i}$ is the probability that $r_k\bar{(p)} = \frac{1}{L} \sum_{j=1}^{L} r_{k,j}(p)$ is equal to 1. The optimal value of this indicator is 1.

Note that most of previous works evaluate the randomness only from the bias. This approach is clearly not sufficient, since it does not take into account correlations between the generated bits.

2.2.3 Post-Processing of the PUF Output

Since PUFs are physical noisy functions, mechanisms have to be put in place to cope with this noise. Post-processing by helper data algorithms is indispensable to meet stringent requirements for cryptographic keys in security-critical products: reproducibility, high-entropy and output control requirements.

1. **Reproducibility** – this is always required. It is common practice to define a maximum failure rate for the key reconstruction phase, e.g. $P_{FAIL} \leq 10^{-6}$. This is coupled to a certain manufacturing yield: PUF noisiness and hence also P_{FAIL} shows a spread among devices.
2. **Uniformity of the distribution** – keys are assumed to have maximum entropy. For some applications, entropy loss can be forgiven, given the use of a longer key. E.g. for the computation of a HMAC, the application penalty might be low.
3. **Output control requirements** – represented by three requirements:
   (a) **PUF independence** – one might wish to program a key which does not depend on the PUF. This requirement does not necessarily imply a full control over the key bits: the helper data algorithm might perform additional hashing for instance. Consider for example a symmetric key communication between two PUF devices with the same helper data algorithm, or the replacement of a malfunctioning device with preservation of the key.
   (b) **Mathematical restrictions** – the key might have to satisfy certain mathematical properties. Consider for example the primality test of RSA.

(c) **Controllability** – one might need the ability to program any given key in the device. Consider for example a symmetric key communication with a legacy device. This requirement supersedes the two previous requirements.

A practical helper data algorithm is an assembly of components rather than a single building block. One can typically distinguish three consecutive steps, as represented in Figure 2.12. First, one applies bit selection, to discard the least reliable bits. This alleviates the burden of the second step: error-correction. An interaction of former steps results in a reasonable failure rate P_{FAIL}, but the outgoing bits have non-maximum entropy. The third step performs data compression to increase entropy.

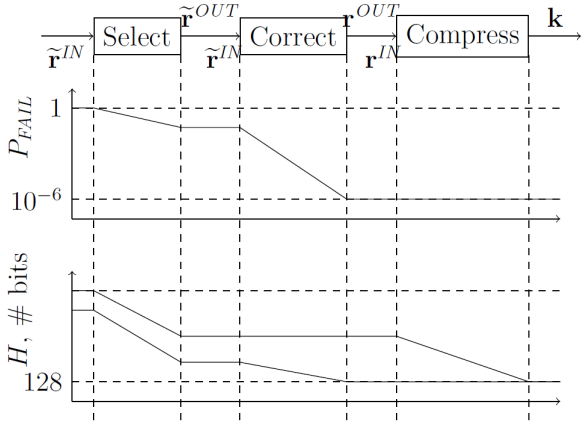

Figure 2.12: Consecutive steps of a helper data algorithm. Typical profiles for failure rate P_{FAIL}, the number of bits and their total entropy.

Bit selection

The idea of bit selection schemes is to discard the least reliable bits already beforehand. Statistical investigations on PUFs have revealed, that only a few *bad bits* cause a large number of bit errors [60, 38]. This leads to the assumption that there is a lot to gain: if these bits were removed beforehand, the total bit error probability could be significantly reduced. As a consequence, one could use more efficient and lightweight error-correcting schemes. Three bit selection approaches can be identified:

Global Thresholding
Imposing a global threshold for the bit error rate is the most intuitive idea. This has first been proposed in [61], with employment later-on in [63, 3, 34, 35, 8]. As discussed in [34], the scheme is highly vulnerable to helper data manipulation.

Local Thresholding Using 1-out-of-n Bits
A local equivalent of global thresholding has been proposed in [63]. Responses are subdivided in non-overlapping sections of length n. For each section, only the most reliable bit is retained.

Local Thresholding Using Index-Based Syndrome
A variation on 1-out-of-n selection has been proposed in [74], with employment in [76] later-on. The index-based syndrome (IBS) scheme is able to satisfy the helper data algorithm requirement 3a. The most reliable 0 or 1 within each segment is selected, with equal probability.

Error-Correction Schemes

Error-correction, also referred to as *information reconciliation*, ensures the key to be reproducible. Various approaches are applied within the context of PUFs:

Temporal Majority Voting
Majority voting can be performed during reconstruction [3]. However, in contrast to enrolment, additional IC hardware (counters) is required. Small bit error rates can be successfully suppressed, but large rates stay large. Therefore the method is never sufficient by itself: further error-correction or prior bit-selection is advised.

Exhaustive Search
The IC could perform an exhaustive search for the error pattern [53]. However, this might consume too much resources to be practical. The overhead also includes a secure check for correctness, as a stopping criterion.

Secure Sketch – Code-Offset and Syndrome Construction
Secure sketches, as defined in [18, 17], are the workhorse of most helper data algorithms (HDAs). They allow for the construction of a fuzzy extractor. Despite the rather generic definition, two constructions dominate the implementation landscape. Both the code-offset and syndrome construction employ a binary $[n, k, t]$ block code C, with t the error-correcting capability. A variant of the code-offset construction which allows fulfilling the helper data algorithm requirement 3.a is [68]. Instead of outputting the error-corrected response, one could also feed it into the entropy compression hash. Or alternatively m, the message corresponding to the codeword, as it would require less compression. The syndrome construction requires a linear block code, as it employs the parity check matrix H. Successful reconstruction is guaranteed for both constructions, given Hamming weight $e \leq t$.

The code-offset construction is employed in [10, 46, 45, 68]. The syndrome construction is employed in e.g. [47, 62, 69]. BCH codes in particular are very popular: they are implemented in [47, 76, 62, 75, 69]. Also repetition codes are rather popular, due to their ease of implementation [10, 45]. The latter is fundamentally different from temporal majority voting, although there is some similarity for the number of errors they can correct. Other codes have been implemented as well, such as Reed-Muller [10, 45, 68] and Golay [10, 68].

Codes in Parallel
Very often, it is not feasible to process all response bits with a single code, due to the decoding complexity [10, 47, 45]. This is resolved by subdividing the PUF response r in x non-overlapping sections of length n, processed independently by a smaller code. To save area and power, sections are decoded one-by-one.

Concatenated Codes
Concatenated codes, in the context of PUFs, have first been proposed in [10]. They have also been adopted in e.g. [47]. They are particularly useful when the average bit error rate is high. Consider the concatenation of two block codes: $[n_2, k_2, t_2] \circ [n_1, k_1, t_1]$, with n_1 an integer multiple of k_2. Code C_2 should be able to correct many errors (a high ratio t_2/n_2). Repetition codes in particular are very popular. Code C_1 only needs to correct a few errors (a low ratio t_1/n_1), but therefore a high k_1 to maintain entropy.

Soft-Decision Decoding
Soft-decision decoding, in the context of PUFs, has been introduced conceptually in [46], with a subsequent implementation in [45]. The error-correcting capabilities improve with respect to

traditional hard-decision decoding. Less PUF bits are required for the same failure rate and key length, taking leakage into account. The original proposal comprehends collaboration with the code-offset method, hereby exposing all bit error rates as public helper data. Note that this could facilitate modeling attacks, assuming the use of a strong PUF.

Unfortunately, the decoding effort increases significantly. *Soft-Decision Maximum-Likelihood* (SDML) decoding offers the best performance, by iterating over all codewords and computing the likelihood each time. The computational complexity is hence exponential with the code dimension k. There are faster procedures for some codes at the cost of reduced performance. Consider *Generalized Multiple Concatenated* (GMC) codes: one exploits for example the fact that a large Reed-Muller code can be split recursively in two smaller Reed-Muller codes.

Convolutional Codes
The popularity of block codes does not exclude other possibilities. The use of convolutional codes has been proposed in [34, 33]. They implement a hard-decision Viterbi algorithm.

Substring Matching
An alternative for error-correcting codes has been proposed in [55]. The basic idea works as follows. Consider a lengthy string of PUF bits r. During enrollment, a shorter substring r_{SUB} is selected at random, possibly considering r to be circular, and exposed as public helper data. Former selection procedure is repeated for several strings: substring indices are combined to obtain a key k of sufficient length. During reconstruction, substrings are shifted along newly generated strings $\widetilde{r}$. The correct indices are retrieved via their low Hamming distance. The scheme is able to fulfill helper data algorithm requirement 3a and possibly also requirements 3b and 3c if substring indices are simply concatenated to form the key.

Data Compression

The entropy of the PUF response r is non-maximum because of two reasons. First, the correlations and bias of the PUF. Second, additional entropy loss by the helper data algorithm, referred to as leakage. Indeed, helper data unavoidably leaks information about the PUF response. A compression step ensures the key k to be nearly uniform. This step is also referred to as privacy amplification. The total amount of entropy is preserved, but the bit-average increases by having more input than output bits. A hash function is the well-established solution. One computes $k = Hash(r)$.

2.2.4 Discussion

The quality of the statistical evaluation of a PUF is significantly limited by the number of available devices: several million devices would be needed for reliable statistical evaluation. This means that no work published on PUFs provides a sufficiently large statistical evaluation of PUFs, so designers cannot take results in the literature seriously. This could have disastrous security and economic consequence in the future.

Moreover efforts towards standardization of PUFs are just starting. The reason is similar as for TRNGs: PUFs rely not only on the principle, but also on its physical implementation in selected technology. Moreover, no standard methodology for evaluation of PUFs (like AIS 31 for TRNGs) was proposed up to now.

Another important problem of PUF evaluation is aging. Usually, designers use aging models that were developed in a very different context and very often (if not always!) aging simulation

results differ seriously from practical experience.

It is however already an important step forward, that PUF technology has been taken up in the discussions about an ISO standard – ISO/IEC NP 20897: "Security Requirements and Test Methods for Physically Unclonable Functions for Generating Non-Stored Security Parameters." During the ISO meeting in Malaysia on 4th - 8th May 2015, at least five countries cast a positive vote (ISO stage: 10.20) to initiate officially a so-called Working Draft (WD) process. The project is now in the *approved*-stage (10.99) since 2015/09 and is now waiting to be registered in TC/SC work programme (20.00). The scope of the ISO working draft document will include the applications descriptions (weak vs. strong PUFs), the reliability testing, security measurements and security levels. The first Committee Draft is supposed to be available by 2016/10 and should be published and adopted by 2017/10. The main editors of the documents are Sylvain Guilley (Télécom Paris Tech, France) and Soshi Hamaguchi (Cosmos, Japan).

2.2.5 Conclusion

It is clear that unlike algorithmic cryptographic primitives, and similarly to TRNGs, PUF principles cannot be standardized. However, it could be very useful if several principles that are easier to characterize could be pre-selected and some convenient stochastic models and physical characterization of the source of randomness developed for them. The role of the designer would then be as follows:

1. Depending on performance and security requirements, implement one PUF principle in selected technology.

2. Evaluate the amount of randomness in the given technology by quantification of the physical source of entropy such as the CMOS process variations.

3. Using the measured parameter, estimate the entropy per bit of the raw binary response using existing stochastic model of the PUF and select suitable entropy extraction and algorithmic post-processing methods.

4. Depending on the stochastic model of the source of randomness, entropy extraction and post-processing algorithm selection, estimate the entropy per bit (pattern) at the PUF output.

Chapter 3

HECTOR Requirements on Key Generators and Selection Criteria

As explained in the previous chapters, the HECTOR project considers TRNGs and PUFs mainly as generators of confidential keys, for which the security requirements are the most restrictive. In the first phase of the project, the industrial partners discussed and specified requirements for TRNGs and PUF that should be selected and evaluated. Next, all HECTOR partners discussed and specified selection criteria, which should be considered in the selection process.

3.1 HECTOR Requirements on TRNGs and PUFs

Although the applications of both RNGs and PUFs in cryptography and data security can be similar, their principles are very different. We recall that the first one uses dynamic sources of randomness during device operation and the second one uses randomness coming from the uncertainty appearing only during the device production process. Consequently, the HECTOR requirements on TRNGs and PUFs are different and will be analyzed separately.

3.1.1 Requirements on TRNGs

Since one of the most important requirements claimed by HECTOR industrial partners is the evaluability and thus certifiability of the selected TRNG according to AIS 20/31, the HECTOR project will deal with the TRNGs, which:

- are based on a physical principle for which a comprehensible stochastic model is feasible,
- have an internally quantifiable physical source of randomness that represents one of the main input parameters of the stochastic model,
- have the raw random signal (digital noise) available for online and offline testing,
- could be tested using fast and efficient dedicated statistical tests based on the stochastic model (total failure test, online test and startup test) in order to detect entropy loss due to some attack or variation of operational conditions and aging.

Other priority requirements on TRNGs specified by industrial partners in WP1 are compatibility with industrialization and manufacturing constraints, portability, and in the case of ASICs compatibility with digital design constraints.

3.1.2 Requirements on PUFs

According to HECTOR industrial partners, the selected PUF must fulfill following requirements:

- must have sufficient uniqueness, steadiness and randomness in corner conditions,
- must be based on a comprehensible physical principle, the feasibility of a stochastic model for the selected PUF principle would be a clear advantage,
- should be robust against environmental fluctuations, aging and attacks.

As in the case of TRNGs, other priority requirements on PUFs specified by industrial partners in WP1 are compatibility with industrialization and manufacturing constraints and portability.

3.2 Selection Criteria for TRNGs

3.2.1 Technology Criteria

Feasibility in FPGA and ASICs
Some generator principles can exploit logic elements or need interconnection resources, which are not available in certain technologies, either in FPGA technologies or in ASIC technologies or in both. The generator, which will be finally selected for implementation in FPGAs, should be feasible in all available families and if possible, also in future FPGA families. Similarly, the generator selected for implementation in ASIC should be feasible independently from the technology.

Repeatability and Portability
Repeatability of the design means that the design of the generator can be copied from one device to another, while giving the results having the same statistical properties. In other words, outputs of two identical devices will give the same statistical results. In practical industrial applications, the repeatability is a must. Repeatability is also tightly linked with industrialization requirements and constraints since repeatability should be achievable without individual trimming, and shouldn't impact production yields.

Portability means that one TRNG design can be easily ported from one technology to another. Nevertheless, some manual intervention concerning placement and routing during the design phase is acceptable.

In case of HECTOR TRNGs for ASICs, the portability criteria translate into easy integration into digital ASIC design flows. For design reuse and complexity management purposes, digital ASICs products are never full custom designs: they are designed using standard digital design flows, EDA tools and pre-qualified digital IPs. These IPs are synthesized using standard and pre-qualified digital standard cell libraries. Some critical or analog IPs can also be sourced as hard macros. Designing, characterizing and qualifying hard macro IPs is a heavy process, that needs to be repeated for every target silicon technology. An important portability criteria

for HECTOR ASIC security IPs is the possibility to synthesize them using standard digital libraries using standard digital design flows and not requiring the design of technology-specific hard macros. In case the TRNG principle requires tight control over topology or timings, it should be possible to impose such constraints through tools available in standard digital design flows.

Scalability
The scalability represents the possibility to easily adapt (for example by the use of some parameter) the design of the generator to the application needs. This concerns especially the area, the bit rate and eventually the power consumption, too.

3.2.2 Design Criteria

Output Data Rate
The output data rate depends usually on the number of bits per sample and on the sampling frequency. Besides the value of the speed, another important feature is its regularity: the data rate of some generators can vary in time.

Cost – Area and Patents Pending
The cost of the generator is expressed in number of logic resources needed (equivalent logic gates in ASIC or logic cells in FPGAs). Another important factor, which determines the cost of the solution are license fees because of patent protection.

Power and Energy Consumption
Power consumption is determined by the current delivered by the power supply at given voltage at given instant. Energy consumption can be obtained from the power consumption by its integration over time. Energy consumption of a fast generator can be reduced by stopping the generator when random data are not needed.

3.2.3 Security Criteria

Evaluability According to AIS31
Before products will be applied in real-life, their design and performance has to be evaluated. This is usually done by independent security laboratories. These labs will verify the performance according to the claimed security class. AIS31 provides methods that allow objective testing of the RNG output. Besides this performance criterion, the evaluation lab will also verify the design of the RNG. This should also match the claimed criteria. Depending on the selected class, the stochastic model should provide clues on the source of randomness, its performance and on dependencies on implementation choices and environmental conditions.

Security evaluations can be costly and time-consuming. It is therefore important for the industry to reduce the time and effort related to conformance with the security standard and security testing. To achieve such goal, designers are encouraged to make RNG designs not 'overly complicated'. A design, which is made following the guidelines of the standard and which is simple to understand, is usually also easy to describe (stochastic model) and to evaluate. This increases the 'evaluability' of the design, which finally reflects on a product with higher assurance that the product meets its claims, while offering lower costs and faster time-to-market.

The evaluability of the generator is related to its complexity and possibility to specify the true source of randomness, which can be quantified (measured) inside the device. In addition,

a class has to be specified for the RNG, which depends on the final use-case of the product. In general, this may be a class specified by AIS31. Depending on the claimed class, different implementations are allowed. The differences between the classes come from the type of the generator and from the extent to which the RNG output quality is internally tested. Different strategies should be applicable to test generated raw random numbers and internal random numbers (the output of the post-processing block). The raw random numbers must be available (in the test mode) outside the generator. The entropy harvesting mechanism (e.g. the sampler) and the post-processing function must be testable. The embedded tests themselves must be testable, too.

Availability or Feasibility of the Model
In the best case, the stochastic model for the given generator already exists. If it is not the case, it is important that the model is at least feasible. This means in particular, that the generator should not mix in some inseparable manner the true randomness and the pseudo-randomness, since the pseudo-randomness could prevent the possibility to characterize the true randomness.

The stochastic model should be robust, i.e. independent from variations of parameters other than sources of randomness.

Testability
The testability of the generator means that internal signals should be available for testing. The full testability means that without the presence of randomness, these internal signals would have some constant value (zero or one), which is easy to detect by the total entropy failure test.

Robustness
The generator should be as robust as possible to variation of environmental conditions (such as temperature and power supply voltage) and to active attacks (for example by electromagnetic induction).

3.3 Selection Criteria for PUFs

3.3.1 Technology Criteria

Feasibility in FPGA and ASICs
Some PUF principles could make use of logic elements or need interconnection resources that are not available in certain technologies, either in FPGAs or in ASICs or in both. The PUF that will be finally selected for implementation in FPGAs, should be feasible in all available families and if possible, also in future FPGA families. Similarly, the PUF selected for implementation in ASIC should be feasible in ASIC independently from the technology.

Repeatability and Portability
Repeatability of the design means that the design of the PUF can be copied from one device to another, while given results have the same statistical properties. In practical industrial applications, the repeatability is a must.

Portability means that one PUF principle can be easily ported from one technology to another. Nevertheless, some manual intervention concerning placement and routing during the design phase is acceptable.

Scalability
The scalability represents the possibility to easily adapt (for example by the use of some pa-

rameter) the design of the PUF to the application needs. This concerns especially the area, the number of returned bits per challenge and eventually the power consumption, too.

3.3.2 Design Criteria

Number of Returned Bits per Challenge
When queried with a challenge, the PUF will produce a response. The raw bit length of this response determines the number of bits that can be generated per challenge.

Cost – Area and Patents Pending
The cost of the PUF is expressed in the number of logic resources needed (equivalent logic gates in ASIC or logic cells in FPGAs). Another important factor, which determines the cost of the solution are license fees because of patent protection.

Power and Energy Consumption
Power consumption is determined by the current delivered by the power supply at given instant and given voltage. Energy consumption can be obtained from the power consumption by its integration over time. Energy consumption of an efficient PUF can be reduced by stopping the PUF when data are not needed.

3.3.3 Quality Criteria

Uniqueness
The uniqueness of the PUF is sometimes also denoted as the *inter-device variation*. This parameter shows to what extent the responses generated by the PUF in different devices are unique, i.e. whether the response, based on its unique physical characteristics, is non-reproducible on different devices.

Steadiness
The steadiness of the PUF is sometimes also denoted as the *intra-device variation* or *reliability*. This parameter quantifies the changes at the output of a PUF function over many measurements with environmental changes (i.e. temperature variation, VDD variation). The metric for steadiness is the bit error probability.

Randomness (entropy rate)
The response of the PUF should be uniformly distributed and unpredictable. The entropy per response bit of the PUF is defined as the entropy rate.

3.3.4 Security Criteria

Feasibility (availability) of the model
A stochastic model of the PUF describes physical processes, which determine the PUF response. Such a model allows to study the PUF behavior in full detail, including average and worst case bit error probabilities, and is an invaluable tool for designing more efficient and better adapted PUFs and PUF-based systems.

Robustness against attacks
As any other security building block, PUFs could be vulnerable to attacks. Some design choices could facilitate certain categories of attacks: passive attacks like side-channel attacks or active

attacks like fault injection using EMA. In case of strong PUFs, modeling attacks need to be considered as well.

3.4 Conclusions on TRNGs and PUFs Selection Criteria

All the requirements presented in this chapter have different priority depending on the targeted application. For example, for light-weight cryptography applications, the area and possibly power consumption will take priority before the speed and security. Nevertheless, for practical reasons, the repeatability of the design will always be required.

In the following phase of the HECTOR project, some marking (score) of the selected characteristics should be proposed (e.g. from 0 to 10 as the lowest and the highest quality level), and finally, in the process of the final selection, different weights will be attributed to individual characteristics according to the targeted application. The principle, which will obtain the best score will be selected as the best candidate.

Chapter 4

Selected TRNG Principles

According to the HECTOR project objectives, at least one TRNG which is suitable for implementation in ASICs and at least one that is suitable for FPGAs should be preselected. The preselected principles must fulfill all TRNG security criteria described in Chapter 3. Following the HECTOR's planning, the preselected TRNG principles should then be further studied, enhanced and hardened.

Currently, only jittery clock signals and random initialization of the flip-flops (or of the RAM memory bits) seem to be sources of randomness that can be easily exploited in logic devices. Since the jitter of the clock signal is easier to quantify[1], we preselected the following TRNG principles based on jittery clock signals in Task 2.1:

- Elementary Oscillator-based TRNG (ELO-TRNG)
- Coherent Sampling Oscillator-based TRNG (COSO-TRNG)
- Delay Chain-based TRNG (DC-TRNG)
- Transition Effect RO-based TRNG (TERO-TRNG)
- PLL-based TRNG (PLL-TRNG)
- Self-Timed Ring-based TRNG (STR-TRNG)

These TRNGs will next be analyzed from the point of view of the criteria defined in Sec. 3.2.

4.1 Elementary Oscillator-Based TRNG

4.1.1 Principle and Design

The elementary oscillator based TRNG (ELO-TRNG) is a physical TRNG (P-TRNG) similar to the one proposed by Fairfield *et al.* in 1985 [23]. Its structure is very simple (see Fig. 4.1): the ELO-TRNG consists of two free running oscillators, a counter and a D flip-flop as a sampler.

The output of one ring oscillator is periodically sampled by a D flip-flop. The sampling period lasts K periods of the reference clock signal generated by the second ring oscillator.

The ELO-TRNG was thoroughly analyzed by Baudet *et al.* in [5], and the authors proposed a stochastic model which can be used to determine the entropy rate at the ELO-TRNG output.

[1]Note that the quantification of the source of randomness is intended to be used as a basis for the efficient dedicated statistical tests.

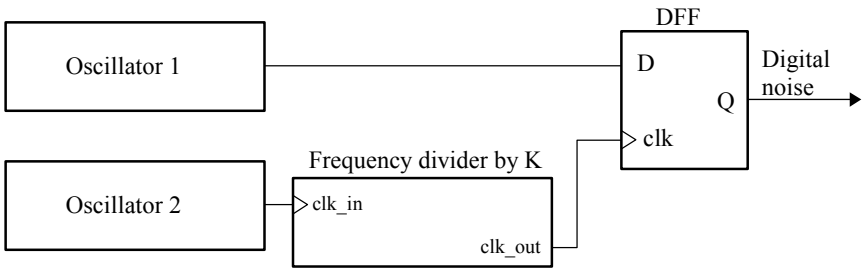

Figure 4.1: Elementary oscillator based TRNG

The model relies on exploitation of thermal noise. It is based on observations and on characterization of the thermal noise in the frequency domain. Therefore, it is easy to understand and simple to use.

The main advantage of the ELO-TRNG is that it is very simple to implement in logic devices. Since it only contains two ring oscillators, their placement and routing is easier and can be done in such a way that the mutual influence of the rings, which creates autocorrelations in the RNG output signal, is reduced to a minimum. It is also easy to detect mutual locking of the two rings, which would have catastrophic consequences for security, as the entropy rate at the output would drop to zero.

The main disadvantage of the ELO-TRNG is its low output bit rate. Depending on the model, the entropy rate at the output of the generator required by security standards comes from the thermal noise. Because the jitter originating from the thermal noise is very small (only few ps), it has to accumulate for a very long time: several hundred thousand periods of the jittery clock signal are necessary [29]. The output bit rate is thus in the order of few kbits per second.

4.1.2 Evaluation According to the Technology Criteria

Feasibility in FPGA and ASICs
The ELO-TRNG is easy to implement in both FPGAs and ASICs. The free-running oscillators can be implemented as ring oscillators based on simple inverters. Depending on the frequency of the reference clock (the output clock of Oscillator 2 in Fig. 4.1), the frequency divider can be implemented as an asynchronous or a synchronous counter. Both are very easy to implement in logic devices using standard gates and logic elements. The same is valid for the D flip-flop used as a sampler.

The main difficulty is related to the correlation between oscillators. It is caused mainly by cross-talks. The correlation between oscillators can be reduced by an appropriate topology of this part of the generator. Of course, the designer using ASIC technology has much more freedom when creating the TRNG topology than the designer working with FPGAs. Namely, long parallel wires, which build up interconnection signals in FPGAs are susceptible to create unwanted cross-talks.

Repeatability and Portability
The ELO-TRNG has perfectly repeatable design, i.e. all the generators having the same topology (structure, placement and routing) will generate numbers having the same statistical quality in all devices having the same hardware (i.e. the same technology of ASIC or the same device in FPGAs).

The ELO-TRNG is potentially portable, however, the size of the oscillators (the frequency) and of the counter (number of bits) must be adapted to the selected technology.

Scalability
As the name of the generator indicates, the elementary oscillator based TRNG has extremely small area requirements. It is therefore (by definition) not scalable. Nevertheless, the output bit rate can be increased by merging several elementary generators into one higher speed generator. However, possible correlations between oscillators implemented in the same device must be taken into account.

4.1.3 Evaluation According to the Design Criteria

Output Bit Rate
The output bit rate of the generator depends on security requirements, namely on required entropy rate. For a smaller accepted entropy rate, the division factor K from Fig. 4.1 can be significantly decreased and the bit rate can become much higher. However, for high entropy rates (e.g. those required by AIS20/31) the bit rate is limited to several kbits per second.

Cost – Area and Patents Pending
The ELO-TRNG itself is extremely small (of course, at the cost of the bit rate) and, as far as we know, no patent is pending.

Power and Energy Consumption
The energy and power consumption of the ELO-TRNG seems to be relatively small, but regarding the bit rate, the energy spent per generation of one bit could be considerable, because long jitter accumulation periods are necessary (division factor K is very high). The energy and power consumption depends on technology and namely on the size of the jitter – higher jitter needs smaller accumulation times.

4.1.4 Evaluation According to the Security Criteria

Evaluability According to AIS31
The design of the ELO-TRNG is extremely simple and therefore, impact of each element of the generator (e.g. the source of randomness, the randomness harvesting mechanism, etc.) can be easily inspected and the stochastic model can be easily validated during the certification process.

Availability or Feasibility of the Model
A comprehensible stochastic model is already available [5]. The main difficulty of the model is related to the fact that it supposes that the randomness comes only from the thermal noise. Therefore, the impact of the flicker noise on the global noise must not be taken into account in estimation of randomness.

The robustness of the model is related to the ability of separate the impact of the thermal noise from that of the flicker noise, since the model is based on the size of the jitter coming from the thermal noise. The locking of the two ring oscillators is another critical point that can decrease the entropy rate.

Testability
The generator is testable using the proposed stochastic model [5]. The tests based on the

embedded measurement of the jitter coming from the thermal noise (the noise that is used as a source of randomness) was recently published [29]. Nevertheless, the cost of the tests is much higher than that of the generator itself. Some efficient and less expensive tests would be useful. The tests must be also able to detect locking of the oscillators.

Robustness
Free running oscillators in general and ring oscillators in particular are known to be very sensitive to variations of operational conditions. The generator uses differential principle: the two oscillators are influenced by the global environmental conditions in the same way and the global changes in both oscillators mutually cancel each other. It is therefore very robust to variations of the operational conditions.

4.2 Coherent Sampling Oscillator-Based TRNG

4.2.1 Principle and Design

The coherent sampling oscillator based TRNG (COSO-TRNG) is a P-TRNG similar to that presented in the previous section (see Fig. 4.2): it uses two oscillators as the sources of jittery clock signals, a D flip-flop and a counter, which counts number of the reference clock periods during one sampling period (the beat signal at the output of the D flip-flop). It was first presented in [40], where the n-bit counter from Fig. 4.2 was replaced by a simple T flip-flop (1-bit counter).

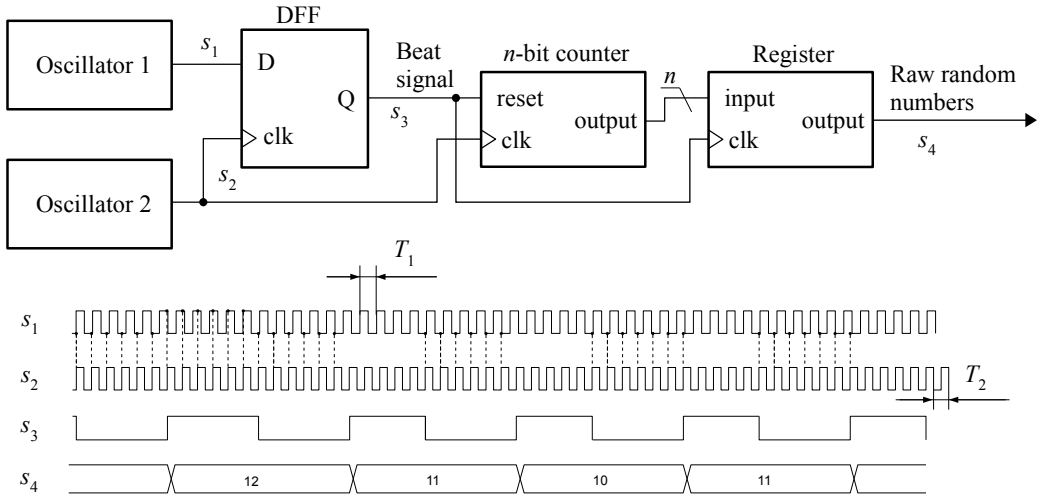

Figure 4.2: Coherent sampling oscillator based TRNG and its waveforms

The main difference between ELO-TRNG and COSO-TRNG is in the way the sampling period is generated. Contrary to the ELO-TRNG, in which the sampling period is derived from K periods of the reference clock, in the COSO-TRNG, the sampling period depends on the absolute value of the difference in periods of the two clocks $\Delta_T = |T_1 - T_2|$. Note that in the coherent sampling, a periodic signal is sampled in such a way that an integer number of its cycles fits into a predefined sampling window. This is exactly what happens in the COSO-TRNG.

Since both s_1 and s_2 are jittery clock signals, their periods, and also the difference in periods vary dynamically in time. Consequently, the sampling period varies too and so does the number

of the reference clock periods. For simplicity, let us assume that the periods T_1 and T_2 remain constant during the measurement interval (one sampling period). The counter value would be as follows:

$$N = \left\lceil \frac{T_1}{\Delta_T} \right\rceil . \tag{4.1}$$

The sensitivity of the generator to the jitter depends on the difference in periods Δ_T. This difference must be smaller than the jitter in order to make N vary in time (otherwise the output of the generator will stall at a constant value). Smaller the Δ_T, higher the entropy in the counter value, but at the same time, smaller the sample rate at the output of the generator.

4.2.2 Evaluation According to the Technology Criteria

Feasibility in FPGA and ASICs
Although the logic structure of the COSO-TRNG is easy to implement in both FPGAs and ASICs (it is practically the same as that of the ELO-TRNG), the main difficulty is in obtaining sufficiently small difference in periods T_1 and T_2 (Δ_T). Recall that the jitter size obtained commonly in free running oscillators is in the order of picoseconds and that by definition, the precision of setting up the difference in periods must be higher. This is not an easy task neither in ASICs nor in FPGAs.

Repeatability and Portability
Our experience shows that even if the difference in periods was sufficiently small in one device, another identical device could have very different value of Δ_T and could output very weak random numbers (if any). Consequently, the COSO-TRNG itself does not seem to have repeatable design. One possible solution would be to set up dynamically the difference in periods.

The COSO-TRNG design is potentially portable, however, the size of the oscillators (the frequency) and of the counter (number of bits) must be adapted to the selected technology and of course the problem related to the repeatability must be solved.

Scalability
Similarly to the ELO-TRNG, the COSO-TRNG has extremely small area requirements. It is therefore (by definition) not scalable. Nevertheless, the output bit rate can be increased by merging several such generators into one higher speed generator if the correlation between different oscillators is taken into account.

4.2.3 Evaluation According to the Design Criteria

Output Bit Rate
The output bit rate of the generator depends on security requirements, namely on required entropy rate. For smaller accepted entropy rates, the difference in periods (Δ_T) can be larger, causing the bit rate to increase (at the cost of decreasing entropy). For high entropy rates (e.g. those required by AIS20/31), the bit rate is limited to several tens of kbits per second. However, because of the principle of the generator (the sampling period depends on the difference in periods, which changes dynamically), its output bit rate is not regular.

Cost – Area and Patents Pending
The COSO-TRNG itself is extremely small (of course, at the cost of the bit rate). It will be necessary to verify if any patent is pending.

Power and Energy Consumption
The energy and power consumption of the COSO-TRNG seems to be relatively small, but regarding the bit rate, the energy spent per generation of one bit could be considerable, because long jitter accumulation periods are necessary (difference in periods must be small). The energy and power consumption depends on technology and namely on the size of the jitter – higher jitter needs smaller Δ_T.

4.2.4 Evaluation According to the Security Criteria

Evaluability According to AIS31
The design of the COSO-TRNG is extremely simple and therefore, the impact of each element of the generator (e.g. the source of randomness, the randomness harvesting mechanism, etc.) can be easily inspected and the stochastic model can be easily validated during the certification process.

Availability or Feasibility of the Model
A stochastic model of the COSO-TRNG has not been published, yet. However, the model of the ELO-TRNG could probably be applied. Because of the coherent sampling principle, it is possible that some simpler model could also be feasible. Both solutions should be inspected during the HECTOR project.

Testability
The generator is testable. Namely, the counter output can be utilized to characterise the jitter. However, the tests must be also able to detect locking of the oscillators. Because of the closeness of oscillator frequencies, the locking of the two oscillators (causing significant entropy failure) is menacing. Fortunately, it can be easily detected.

Robustness
Free running oscillators in general and ring oscillators in particular are known to be very sensitive to variations of operational conditions. However, since the generator uses a differential principle, in which the two oscillators are influenced by the global environmental conditions in the same way, the global changes in both oscillators mutually cancel each other. The COSO-TRNG is therefore very robust to variations in the operational conditions.

4.3 PLL-Based TRNG

4.3.1 Principle and Design

The TRNG based on the jitter, which is introduced by the phase-locked loop (PLL), was first proposed in [28]. The PLL-based TRNG (PLL-TRNG) uses the coherent sampling method for generating random bit stream.

The basic version of the PLL-TRNG is depicted in Fig. 4.3. Thanks to the control loop of the PLL, the frequency of the output clock signal (clk_{jit}) depends on the frequency of the PLL input clock signal (clk_{ref}) in the following way:

$$f_{clk_{jit}} = f_{clk_{ref}} \cdot \frac{K_M}{K_D}, \tag{4.2}$$

where K_M and K_D are multiplication and division factors of the PLL, respectively.

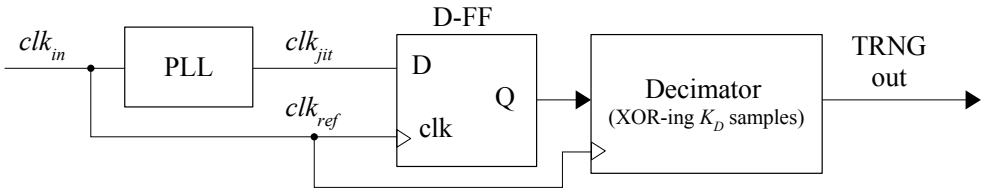

Figure 4.3: Phase-locked loop-based TRNG (PLL-TRNG)

In the PLL-TRNG, the jittery clock signal (clk_{jit}) is sampled in a D flip-flop (D-FF) using the reference clock signal (clk_{ref}) and the output of the flip-flop is decimated in the decimator, in which K_D samples (outputs of the D-FF) are added modulo 2 to form one random bit at the output of the generator.

The source of randomness in the PLL-TRNG is the tracking jitter of the PLL, i.e. the difference in phases of the reference clock and the PLL output clock. Because of the PLL principle, the tracking jitter is bounded and it depends on the jitter of the reference clock and the parameters of the PLL (the jitter of the voltage-controlled oscillator, the bandwidth of the filter and the dumping factor) [21].

Figure 4.4 depicts an example of input/output waveforms of the PLL-TRNG, in which the multiplication factor $K_M = 5$ and the division factor $K_D = 7$. It can be observed that the rising edges of the reference clock signal can be situated in seven positions during one period $T_Q = K_D T_{clk_{ref}} = K_M T_{clk_{jit}}$. In two of them, the rising edges of the signal clk_{ref} appear when the sampled signal is equal to zero (samples 5 and 6 situated in the second half of the sampled clock). At the moment when two rising edges occur, the sampled signal is equal to one (samples 1 and 2) and finally at three other rising edges the sample value is uncertain (samples 0, 3, and 4).

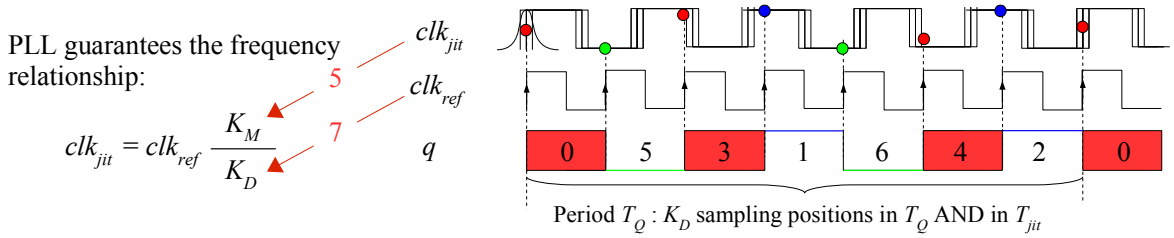

Figure 4.4: Example of the PLL-TRNG input/output waveforms

It was shown in [28] that if the standard deviation of the jitter (σ_{jit}) fulfills the following condition:

$$\sigma_{jit} > \text{MAX}(\Delta T_{min}), \tag{4.3}$$

at least one sample during each T_Q period will have some random value. The term $\text{MAX}(\Delta T_{min})$ in condition (4.3) represents the worst case distance between the rising edges of the clock signal clk_{ref} and rising or falling edges of clk_{jit} during the period T_Q. It is given by

$$\text{MAX}(\Delta T_{min}) = \frac{T_{CLK}}{4K_M}\text{GCD}(2K_M, K_D) = \frac{T_{CLJ}}{4K_D}\text{GCD}(2K_M, K_D), \tag{4.4}$$

where GCD means the Greatest Common Divisor.

As shown in [28], if K_M and K_D are relatively prime and K_D is odd, the TRNG output bit rate is $R = T_Q^{-1} = f_{ref}/K_D$ and the sensitivity to jitter is $S = \Delta^{-1} = K_D/T_{jit}$. The output bit rate and the sensitivity are closely related. Following relationships between parameters of the PLL-TRNG can be observed:

- to increase R and S, f_{ref} should be as high as possible,
- to increase R, K_D should be as small as possible,
- to increase S, K_M should be as large as possible.

In some technologies, condition (4.3) cannot be fulfilled using a single PLL. In this case, two PLLs connected in series or in parallel can be used to increase the bit rate and sensitivity to jitter by increasing the multiplication and division factors (see the top panel and the bottom panel in Fig. 4.5, respectively).

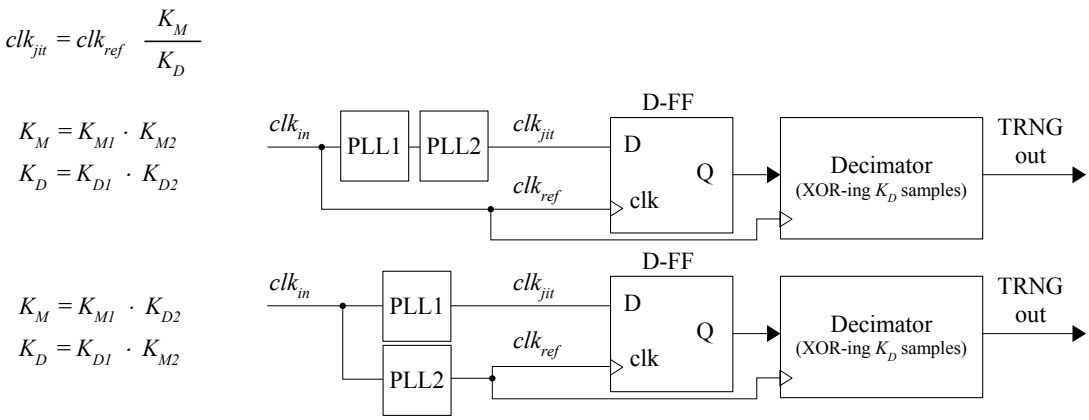

Figure 4.5: PLL-TRNG using two PLLs in series or in parallel

4.3.2 Evaluation According to the Technology Criteria

Feasibility in FPGA and ASICs
The PLL-TRNG is more suitable for implementation in FPGAs, because PLLs are freely available. It is feasible also in ASICs, but it is considered to be expensive because of the use of dedicated PLLs. The disadvantage of the PLL-TRNG implementation in FPGAs is that the PLL characteristics are defined by the FPGA vendor and that they can be modified only in a limited extent (mostly only K_M and K_D, sometimes also filter parameters). On the other hand, its important advantage is that the PLLs have separate power supply wires in FPGAs.

Repeatability and Portability
The portability of the design is limited by the size of the jitter in relationship with the maximum possible values of the multiplication and division factors. The design of the generator (the choice of K_M and K_D and the decimator) must be adapted to these characteristics given by the technology and the PLL structure and topology. The design does not need manual placement and routing. Once designed for the given technology, the generator is freely repeatable.

Scalability
The TRNG design is not scalable – more PLLs would be needed.

4.3.3 Evaluation According to the Design Criteria

Output Bit Rate
The output bit rate depends largely on the tracking jitter that is not known in advance (it depends on the technology and the structure and topology of the PLL). It depends also on the multiplication and division factors in relationship with the jitter. Depending on technology, output bit rates up to 1.5 Mbits/s can be expected.

Cost – Area and Patents Pending
The TRNG area is determined by the PLL. The area of the additional logic is negligible (one D-FF, the counter of K_D periods and an accumulator with the XOR gate). The TRNG is not patented.

Power and Energy Consumption
The energy and power consumption is technology dependent, but it is relatively high because of the PLL (the VCO oscillates at high frequency). In some FPGAs, the PLLs can be stopped, when the TRNG is not in use.

4.3.4 Evaluation According to the Security Criteria

Evaluability According to AIS31
The principle is very simple and clear. The source of randomness is clearly defined and can be easily quantified inside the device. The raw binary signal is available. The TRNG should be easy to evaluate.

Availability or Feasibility of the Model
Simple stochastic model of the PLL-TRNG was published in [7]. However, the model does not take into account the correlation between successive samples. This correlation can be impacted by the dumping factor of the PLL. Extensions of the existing model should be feasible.

Testability
The raw binary signal is available and testable (it does not contain a pseudo-randomness, if the T_Q is set properly). Based on the coherent sampling principle, the tracking jitter, which is used as the source of randomness, can be quantified inside the chip. The embedded jitter measurement can constitute a basis for dedicated online statistical tests, which can be embedded inside the device.

Robustness
Because of the control loop of the PLL, the generator is very robust to environmental changes. The PLL is often physically isolated from the rest of the device, which contributes to the robustness.

4.4 Delay Chain-Based TRNG

4.4.1 Principle and Design

The delay chain based TRNG (DC-TRNG) proposed by Rožić *et. al.* [58] is shown in Figure 4.6. This TRNG uses a free running ring oscillator as the source of entropy. The phase of the ring-oscillator (the position of the signal edge) is affected by the white noise. After

the jitter accumulates for a short time, the position of the edge becomes unpredictable due to the accumulated Gaussian jitter. This position is sampled with high precision using tapped delay lines. These tapped delay lines are made using carry-chain primitives available on most commercial FPGAs. These primitives are designed to be ultra-fast which makes them suitable for high-precision time-to-digital conversion. Data sampled in the delay lines are encoded using XOR gates and a priority encoder to detect the position of the signal edge. The neighboring positions are mapped into different output bit values.

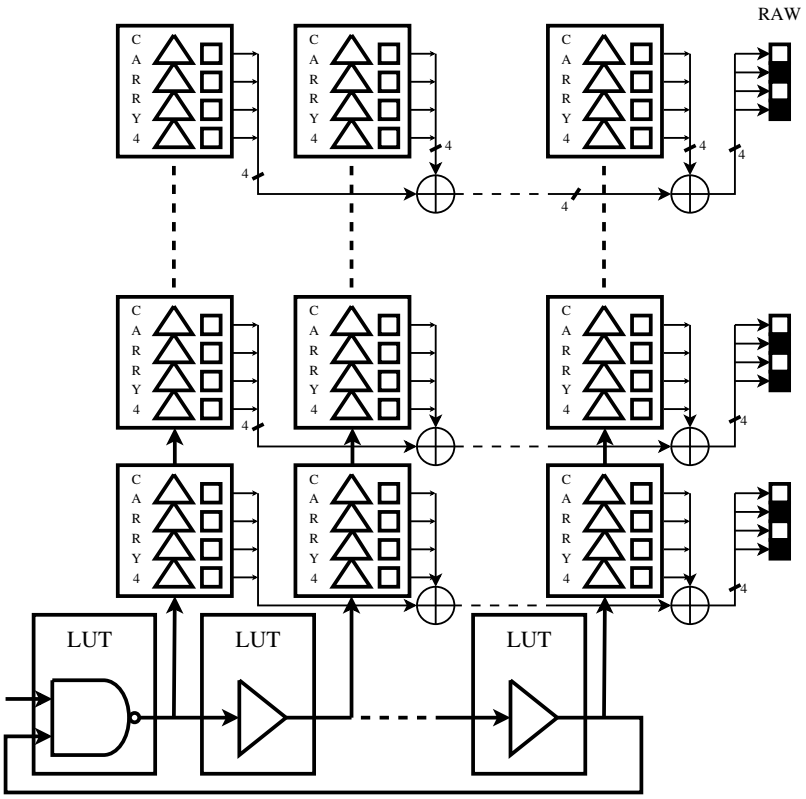

Figure 4.6: True random number generator based on the Delay Chain

There are four main advantages of this TRNG:

1. low cost – compact implementation using less than 70 slices,
2. high throughput of more than 10 Mb/s,
3. high portability between FPGAs,
4. low design expertise required.

The main disadvantage of this TRNG is related to its implementation aspects. Only slices with carry-chain primitives can be utilized to implement the tapped delay lines. For example, on Xilinx Spartan-6, only one half of the slices contain this primitive. However, the FPGA fabric available on Xilinx Zynq-7000 SoC contains carry-chain primitives in every slice.

4.4.2 Evaluation According to the Technology Criteria

Feasibility in FPGA and ASICs
The design is feasible on all FPGAs containing dedicated carry-chain primitives. These primi-

tives are provided by all popular FPGAs vendors. Up to now, this design has only been implemented in FPGAs. However the underlying principle can be used for TRNG implementation in ASIC.

Repeatability and Portability
The design of the DC-TRNG is well repeatable. Practical experiments show that the quality of the output doesn't vary significantly across different devices of the same family.

The design is portable to any FPGA with carry-chain primitives. Only few platform parameters such as the oscillator period, the jitter strength and the propagation delay of the carry-chain need to be determined once for each FPGA family. The architecture and the placement strategy remain the same across different platforms. This design only requires placement constraints to be defined, no routing constraints are required.

Scalability
Design duplication is possible due to its compact implementation and availability of used primitives (LUTs and carry-chains).

4.4.3 Evaluation According to the Design Criteria

Output Bit Rate
The output bit rate depends on the platform and design parameters. For Spartan-6, 2 versions were implemented and the throughput of 1.53 Mb/s and 14.3 Mb/s was achieved after the post-processing using a parity filter. Possibly, better throughput could be achieved by using better post-processing techniques. Better results are also expected on newer platforms.

Cost – Area and Patents Pending
This design has very compact implementation. Two versions of 40 and 67 slices including the entropy source and post-processing were implemented.

Currently, there are no patents pending on this design.

Power and Energy Consumption
Currently, this design is implemented only in FPGAs. So power and energy consumption is not relevant.

4.4.4 Evaluation According to the Security Criteria

Evaluability According to AIS31
This TRNG relies on the accumulated jitter in a free-running ring-oscillator. The process of jitter accumulation in ring-oscillators is well known.

Availability or Feasibility of the Model
The first version of the stochastic model is available in the paper describing the original design [58]. A lower bound of entropy can be computed from the jitter parameters. A more detailed version of the model is in preparation by the KUL partner.

Testability
The raw binary signal of this TRNG is available for testing. A dedicated test based on the online jitter measurements is also in preparation.

Robustness

The design parameters can have margins to improve robustness at the cost of reduced throughput.

4.5 Transition Effect RO-Based TRNG

4.5.1 Principle and Design

The P-TRNG based on the TERO structure (TERO-TRNG) proposed by Varchola and Drutarovsky in 2010 [71] is depicted in Fig. 4.7. In this design, the TERO circuitry generating a random number of temporary oscillations is completed by an n-bit asynchronous counter which counts the rising edges of the oscillating signal. The counter output represents realizations of the random variable, i.e. the number of oscillations in subsequent control periods. Because of the temporary nature of oscillations, TERO must be periodically restarted by the control signal. The control period defines the output bit rate of the generator.

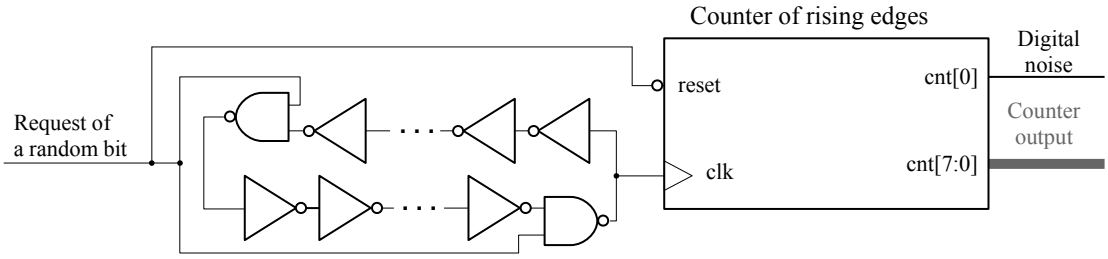

Figure 4.7: True random number generator based on the TERO structure

The random binary sequence is obtained by successively concatenating the least significant bits of the counter, i.e. only one T flip-flop is needed in the counter (the n bit counter is only needed for the characterization of the generator, or alternatively, it can be used to build embedded randomness tests).

The main advantage of the TERO-TRNG is its simple structure (especially if only a T flip flop is used instead of an n bit counter) and its high output bit rate – several megabits per second can be obtained. The main weakness of this design is related to its repeatability.

We stress that the mean number of oscillations is a function of the TERO intern al structure and parameters. Unfortunately, TERO behavior is very sensitive to variations in the production process and several TEROs with identical topology could feature a very different number of oscillations.

As shown by Haddad *et al.* in [30], the entropy rate in the generated bit stream is closely linked to the number of oscillations. In order to accumulate enough entropy, a sufficient number of oscillations defined by the stochastic model must occur after each restart.

Two negative events can occur: 1) because of internal parameters of the TERO circuitry, the oscillator does not oscillate for a sufficiently long time; 2) the TERO is too symmetrical and the oscillations last longer than the control period and consequently, number of oscillation periods is not random. In both cases, the entropy rate at the output of a given TERO-TRNG would be insufficient.

It thus seems that TERO-TRNGs are better suited for ASICs, because their internal parameters can be better adjusted. Although such a TRNG design is globally very attractive, the question if its ASIC implementation can be tuned to be robust against normal manufacturing variations while maintaining enough entropy remains open. Actual implementations in ASIC technology are needed and planned, since FPGA implementations are not sufficient to evaluate this feature of the design.

4.5.2 Evaluation According to the Technology Criteria

Feasibility in FPGA and ASICs
The TERO-TRNG has logic elements that are easy to implement both in ASICs and FPGAs. The main difficulty in this P-TRNG design is related to the necessity to unbalance the two TERO branches in a precise manner: too big asymmetry would cause that the number of oscillation and consequently the entropy would be too small. Perfectly balanced TERO could oscillate for too long time, i.e. oscillations could last longer than the control period (i.e. the time interval between requests of the random bit from Fig. 4.7) and consequently, the counter values would not be random.

Feasibility in FPGA seems problematic. ASIC technology offers finer grain control, however as explained in the previous section making such a design robust against normal manufacturing variations remains an open question. An ASIC implementation is needed, planned and will help assess TERO-TRNG feasibility in ASIC technology.

Repeatability and Portability
Our experience shows that even if the difference in delays of the two TERO branches is set up correctly in one device, another identical device could have very different number of oscillations and consequently, it could output weak random numbers. The above-explained control over critical design parameters is not important only for feasibility, but even more for repeatability. An on-silicon (ASIC) verification is planned and needed.

The TERO-TRNG design is potentially portable, however, the size of the oscillators (the frequency) and of the counter (number of bits) must be adapted to the selected technology and of course, the problem related to the repeatability must be solved.

Scalability
The TERO-TRNG occupies small area and potentially high output bit rate. The mean bit rate can thus be adapted to practical needs by stopping the generator. Because of the small size, several generators can be used in parallel to increase the output bit rate.

4.5.3 Evaluation According to the Design Criteria

Output Bit Rate
The output bit rate is relatively high (considering the small size of the generator). Depending on the technology, the bit rate of several Mbits per second can be reached.

Cost – Area and Patents Pending
The area of the TERO-TRNG is very small – only few gates and a counter are needed. Currently, it is not known if any patent is pending on it.

Power and Energy Consumption
The power consumption is relatively high (the ring is oscillating at high frequency and two

events are propagated in parallel). However, by taking profit from the high bit rate, the energy consumption can be decreased by stopping the generator if random data is not needed.

4.5.4 Evaluation According to the Security Criteria

Evaluability According to AIS31
The principle of the TERO-TRNG is very simple and the source of randomness is well known. The internal signal (output of the counter) can be used to characterize the source of entropy.

Availability or Feasibility of the Model
The comprehensive model of the TERO-TRNG was already published by the HECTOR project partners [30]. The model uses three physical parameters as input. The values of these parameters can be obtained by statistical evaluation of the counter output.

Testability
The generator is easy to test – the parameters of the underlying stochastic model can be computed from the distribution of counter values.

Robustness
It seems that the mean value of the counter can be manipulated externally by changing operational conditions (e.g. temperature and power voltage), but the variation of the counter values (and thus entropy) cannot be easily manipulated.

4.6 Self-Timed Ring-Based TRNG

4.6.1 Principle and Design

As explained in Section 2.1.2, the STR is a multi-event ring oscillator without collision. In the evenly-spaced mode the events are uniformly distributed over half an oscillation period. It will also be recalled that the oscillation period in the steady state of STR is defined by the time interval between two subsequent events traversing the stage and not by the number of stages. A self-timed ring thus provides L jittery signals C_i, $1 \leq i \leq L$, with the same period T and a constant mean phase difference between them.

In contrast to TRNGs using multiple ring oscillators as sources of randomness, in which the uniform distribution of the phases of multiple clock signals originates from the independence of rings [64], the uniformity of distribution of phases in STR is guaranteed by the STR principle, which is based on two analog phenomena – the Charlie and the drafting effect.

Nevertheless, because of the presence of many jittery clock signals with uniformly distributed phases, the entropy harvesting principle can be the same as in [64]. Left panel in Fig. 4.8 depicts the STR-based TRNG published in [15]: the outputs of all the STR stages C_i are first sampled at reference clock frequency to obtain samples s_i, $1 \leq i \leq L$, which are then added modulo 2 using an XOR gate to obtain the digital noise signal

$$\psi = s_1 \oplus s_2 \oplus ... \oplus s_L.$$

The entropy harvesting principle is illustrated in the right panel in Fig. 4.8. Since each signal C_i is sampled at the same frequency (reference clock clk), for any sampling instant t, there exists j such that $|t - t_j| \leq \frac{\triangle\varphi}{2}$, where t_j is the switching time of the signal C_j.

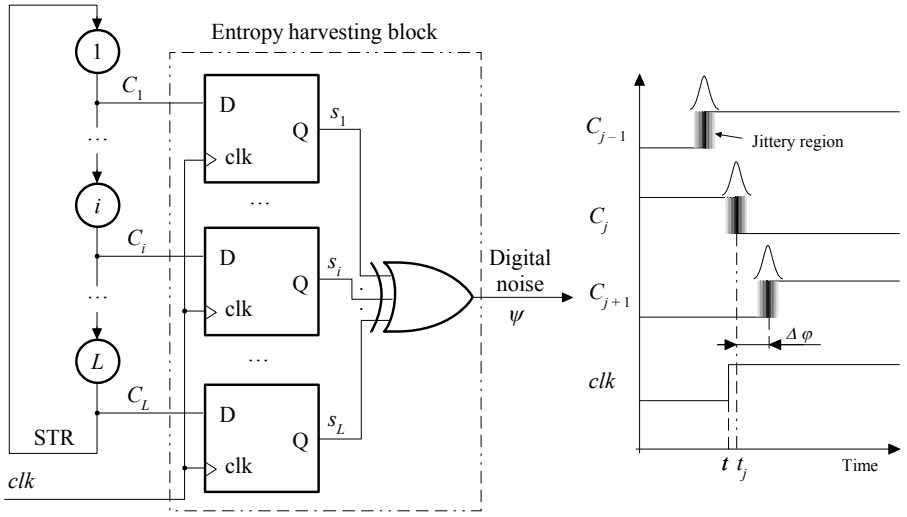

Figure 4.8: STR-TRNG core architecture and working principle

If the jittery region is larger than the phase difference $\triangle\varphi$, the signal C_j is sampled in this region, as shown in the right panel in Fig. 4.8. The resulting sample s_j then has a random value, and hence the output of the XOR gate is also random.

The entropy rate of the output bit ψ is at least equal to the entropy of the sample s_j. The higher the magnitude of the jitter and the lower the phase difference $\triangle\varphi$, the higher the entropy of the sample s_j and at the output of the TRNG.

The main advantage of using of the self-timed ring for the generation of random numbers is that it makes it possible to adjust the mean time elapsed between successive events precisely. This time lapse can be set as short as needed, and can thus be adjusted to the jitter magnitude of a self-timed ring stage.

Another advantage of the STR is that the entropy rate at the output of the generator is practically independent of the sampling clock frequency. It is sufficient that this frequency is smaller than or equal to the generated clock frequency. Last but not least, the locking of the ring clock to the reference clock thus will not have catastrophic consequences.

On the other hand, an attacker could reduce the number of events in the ring by fault injection and thereby reducing the phase resolution. However, this is quite easy to detect and the ring can be restarted with a right number of events.

4.6.2 Evaluation According to the Technology Criteria

Feasibility in FPGA and ASICs
The self-timed rings use logic functions which can be theoretically easily implemented in both FPGAs and ASICs. However, the Muller cells, which are basic construction elements of the STR, are not always available in standard cell libraries. This fact is in conflict with industrial requirements: portability and compatibility with a standard digital design flow. Because of routing restrictions, STRs with many cells used in STR-TRNG are difficult to implement in FPGAs.

Portability and routing restrictions are apparently the biggest disadvantages of the STR TRNG and the price to pay for obtaining a very high output bit rate of several hundred

megabits per second [16].

Repeatability and Portability
The STR-TRNG is repeatable, meaning that two different devices containing the same project will give results with the same statistical properties. However, the design is not directly portable from one technology to another.

Scalability
Thanks to the Charlie and drafting effects, the jitter caused by the local noise sources in a cell does not propagate across the ring as shown in [13]. Consequently, multiple outputs of the rings can be used as sources of randomness. The generator is thus perfectly scalable.

4.6.3 Evaluation According to the Design Criteria

Output Bit Rate
The generator can reach extremely high bit rates of several hundred megabits per second at the expense of high power consumption, since many high frequency events are propagating across the ring.

Cost – Area and Patents Pending
The cost of the generator is relatively high – a high entropy generator needs many Muller cells to properly operate.

Power and Energy Consumption
The power consumption of the generator is considerable. However, because its output bit rate is very high, the generator can be stopped, once the needed amount of random bits has been generated.

4.6.4 Evaluation According to the Security Criteria

Evaluability According to AIS31
The structure of the generator is very simple and regular. It can be easily studied and evaluated.

Availability or Feasibility of the Model
The stochastic model is not available up to now, but should be relatively easy to construct.

Testability
The entropy of the generator can be easily estimated from the known size of the jitter. The main difficulty is therefore related to the existence of an appropriate method of jitter quantification (jitter measurement).

Robustness
Like other free running oscillator, STRs are sensitive to variation of the global operational conditions. This sensitivity can be reduced by exploitations of the differential principles: both jittery clocks and the sampling clock must be generated by the similar STRs. The only security weakness is the possibility to change the mode of oscillations from the evenly-spaced to burst mode by fault injection. The oscillations in the burst mode could significantly reduce the entropy.

4.7 Conclusions on RNG Selection

From the above presented analysis it is clear that an ideal TRNG does not exist. All the presented generators will be first implemented in FPGAs and their parameters thoroughly evaluated. This will provide further inputs and comparison data. Among the above selected principles, at least one will be chosen for implementation in ASIC.

Chapter 5

Selected PUF principles

According to the HECTOR project objectives, at least one PUF which is suitable for implementation in ASICs and at least one that is suitable for FPGAs should be preselected. The preselected principles must fulfill all PUF criteria described in Chapter 3. Following the HECTOR's planning, the preselected PUF principles should then be further studied, enhanced and hardened.

We preselected the following PUF principles:

- RO-based PUF (RO-PUF)
- TERO-based PUF (TERO-PUF)
- SRAM-based PUF (SRAM-PUF)

These PUFs will next be analyzed from the point of view of the criteria defined in Sec. 3.2.

5.1 RO-Based and TERO-Based PUFs

5.1.1 Principle and Design

Ring oscillators (ROs) are digital oscillators consisting of a chain of inverting logic elements connected to form a loop. Traditionally, they are composed of an odd number of inverters (which may include an initialization stage) to ensure steady oscillatory behavior.

Behavior of the transient effect ring oscillators (TEROs) corresponds to a very specific configuration of ROs in which the number of inverters is even. Contrary to classical ROs, TEROs require two initialization stages and have two functioning modes: a transient oscillatory state followed by a stable steady state. Figure 5.1 shows the generic architecture of ROs and TEROs. The oscillatory behavior in both ROs and TEROs is due to the propagation of electrical transitions across the ring structure. When a stage i has the same input and output value, it switches its output after a time D corresponding to the propagation delay of the gate. We refer to this process as an "event" happening at the output of the stage i.

Inside an RO, one inverting cell has always conflictual output/input values due to the odd number of inverters. This cell switches its output after a propagation delay D, which transfers the output/input conflict to the following cell: an event propagates from cell i to cell $i+1$.

Figure 5.2 shows typical output sequences of RO and TERO. For 7-stage ROs, an output state is represented by a vector of 7 bits corresponding (from left to right) to the nodes C_0 to

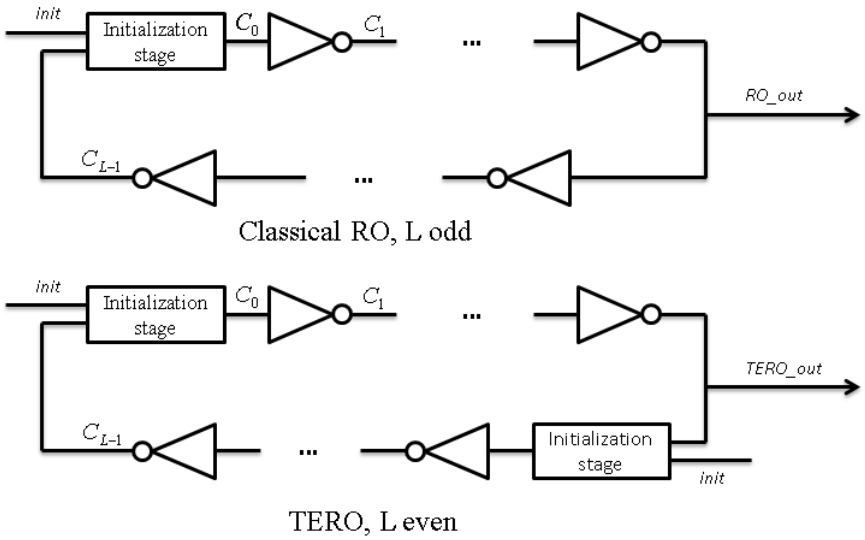

Figure 5.1: Generic architecture of ROs and TEROs

C_6 with respect to the index in Fig. 5.1. Whatever the propagation delays in the ring stages, all 7-stage ROs have the same sequence of states although the time of each state may vary. The oscillatory behavior of the RO corresponds to the constant propagation of one event across the ring structure.

Figure 5.2 represent the sequences of output signals of two 8-stage TEROs, where each stage has a different timing characteristics, from node C_0 to node C_7 according to the index in Fig. 5.1. The goal of the initialization stages in TEROs is to trigger two events that can propagate simultaneously across the ring structure (by forcing the initial state in Fig. 5.2). The transient oscillatory regime in TEROs is due to the propagation of those two events across the structure.

Contrary to ROs, the output sequence in TEROs can vary from one TERO to another depending on specific timing characteristics of each ring stage. For instance, in TERO 1, the propagation delay of stage C_1 is shorter than the propagation delay of stage C_5, whereas in TERO 2, the propagation delay of stage C_1 is longer than the propagation delay of stage C_5. Starting from the same initial state "00101101" (Sequences 1.1 and 2.1), the following state in TERO 1 is "01101101" (Event 1 propagates first) whereas in TERO 2 it is "00101001" (Event 2 propagates first). More generally, the transition from one state to another depends on the propagation delays of the stages in which the two events occur, but also on the relative duration of the previous state (or in other words, the relative position of the two events). When two events happen at adjacent stages (Sequences 1.3, 2.2 and 2.3), the input level of one stage may vary before it has completely switched its output (for example to '1'), the output level starts switching in the opposite level (to '0') before the appropriate voltage level ('1') is detected by the next stage (Sequences 1.3 and 2.3). We refer to this situation as a collision between the two events. When events ultimately collide in a TERO, one of its two final steady states (either "01010101" or "10101010" in the 8-stage TERO) is reached.

In practice, the RO produces a digital clock with a duty cycle that approaches 50%. Its frequency depends on the number of inverters as well as on the propagation delay of each ring stage. The TERO produces a signal with transient oscillations and a variable duty cycle. The oscillation frequency depends on the propagation delays of the ring stages as well as on

their number. The duty cycle corresponds to the "distance" between the two events as they propagate across the ring (*i.e.* the time that elapses between the two events seen at the output of one ring stage). The number of transient oscillations depends on the propagation delay of each stage but also on the capacitive charge and discharge parameters of its output.

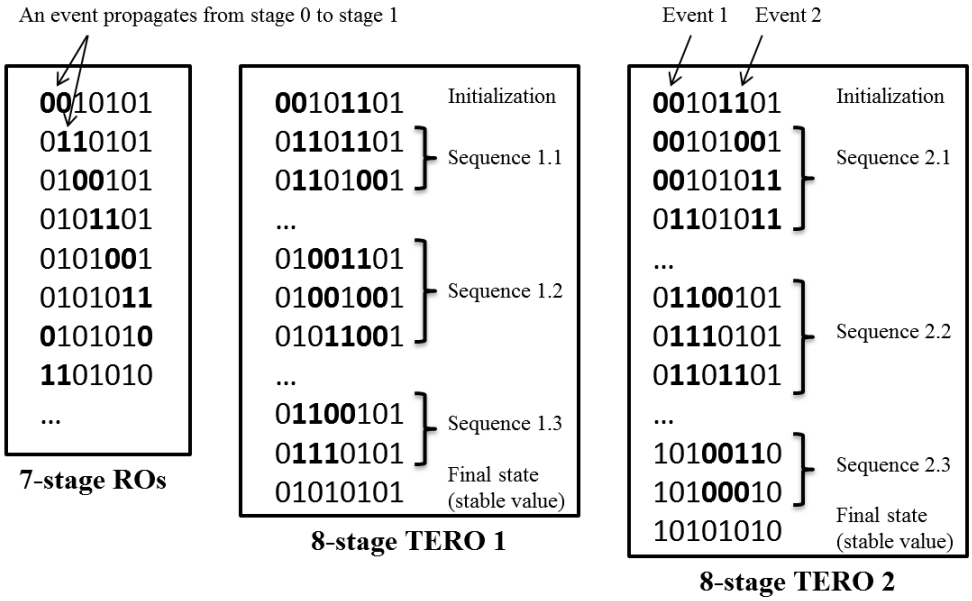

Figure 5.2: Typical output sequences of ROs and TEROs

Based on [63] and [11], Fig. 5.3 proposes a generic architecture for oscillator based PUFs. As originally proposed in [63], the RO-PUF extracts MPV in a digital circuit by comparing the oscillation frequencies of two identically implemented ROs. In Fig. 5.3, the oscillators are ROs whereas the subtractor is a comparator (which can be seen as a 1-bit subtractor). The number of oscillations counted during a fixed time window is compared to provide a one-bit response to the challenge, which consists in selecting two ROs.

In [11], ROs are replaced by TEROs and the comparator is replaced by an n-bit subtractor. The core architecture of a TERO PUF is composed of two blocks of L identical TEROs. When selected and initialized, each TERO presents transient oscillations at its output (contrary to ROs, which present permanent oscillations). The number of transient oscillations varies between identically implemented TEROs due to MPV.

In a TERO-PUF, each challenge, denoted $C(i, j)$ (with $0 \leq i, j \leq L-1$), consists of selecting TERO i from Block A, and TERO j from Block B using multiplexers. The number of available challenges is L^2. An n-bit response is obtained from each challenge by subtracting the number of TEROs' oscillations. The goal of this subtraction is to reduce the influence of environmental global variations since they tend to affect each logic cell in the circuit equally. The L^2 challenges can be divided into k sets of m challenges in order to obtain responses of $m \times n$ bits.

When used in security primitives such as PUFs and TRNGs, TEROs have many advantages over classical ROs due to their transient oscillations. Locking phenomena are theoretically less likely due to the very brief time during which the TERO oscillates. In an RO, the number of oscillations in a fixed time window depends directly on the ring frequency. By acquiring the ring frequencies of a RO-PUF using the EM channel, an adversary can mathematically clone the PUF. In Section 3 and Section 5, we show that the number of transient oscillations in a TERO-PUF does not only depend on its frequency, but also on the signal slopes. Measuring

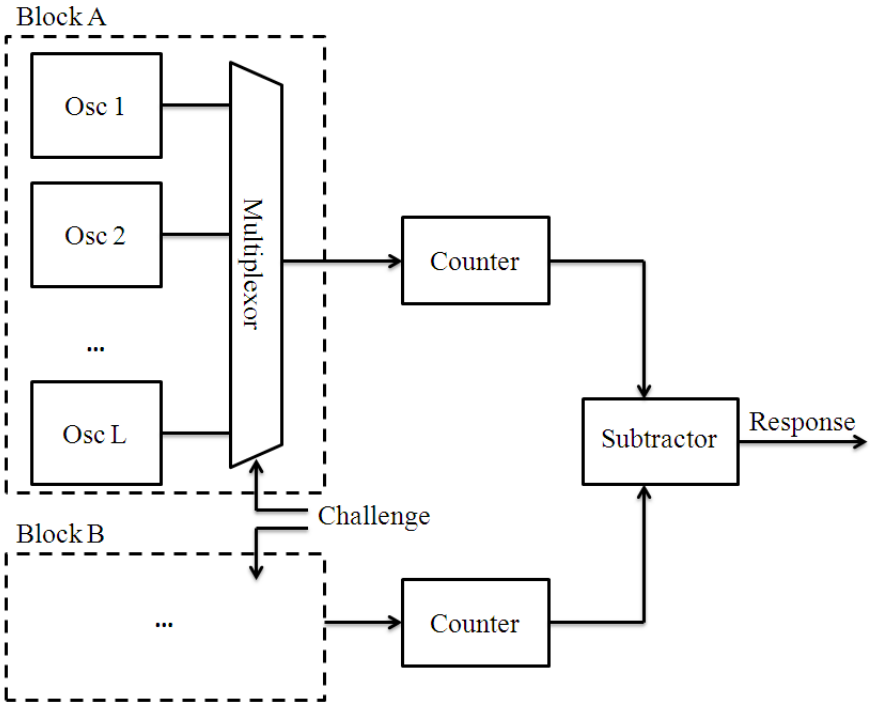

Figure 5.3: Generic core architecture of RO-PUFs and TERO-PUFs

the ring frequencies using the EM channel and obtaining all the required parameters to build a model is obviously more challenging and cannot be achieved only using contactless scanning.

Table 5.1: Comparison of the RO-PUF presented in [63] with the TERO-PUF presented in [11]

	RO-PUF [63]	TERO-PUF [11]
Entropy source	MPV	MPV
Entropy extractor	RO, steady oscillating state	TERO, transient oscillating state followed by a stable steady state
Extraction vector	Oscillation frequency	Duration of the transient oscillations
Bit generation	Frequency comparison	Differential measurement of the number of transient oscillations
Number of response bits per challenge	1 bit in [63], more than one bit possible in theory	Several
Leakages	RO frequencies (EM channel)	Unknown

Table 5.1 compares the RO-PUF, as proposed in [63], with the TERO-PUF. Like the RO-PUF, the TERO-PUF is based on logic elements and adapts very well to FPGA and ASIC design flows and process technologies. The proposed architecture allows a large number of challenge and response pairs (CRPs): L^2 in Fig. 5.3. Responses can be built using one or more bits from the subtractor value. The way the responses are generated significantly affects their statistical quality and the overall size of the design.

5.1.2 Evaluation According to the Technology Criteria

Feasibility in FPGA and ASICs
RO-PUF and TERO-PUF are suitable for implementation in both FPGAs and ASICs. Partner

UJM has available hardware implementations of RO-PUF and TERO-PUF with 350 nm CMOS ASIC (30 devices available), 45 nm CMOS Xilix Spartan 6 SRAM FPGA (30 devices available), 28 nm Altera Cyclone V SRAM FPGA (30 devices available) CBM15. TEC has the RO-PUF implemented in a 65 nm ASIC technology available [44]. Other implementations of RO-PUF are available in the literature, e.g. [48] (90 nm Xilinx Spartan-3E).

Repeatability and Portability
The portability of the design is not limited. The topology of the design remains the same from one technology to another one. Nevertheless, the design needs manual placement and routing for each technology. Once designed for the given technology, the RO-PUF and TERO-PUF are freely repeatable.

Scalability
The design of RO-PUF and TERO-PUF is scalable (but the size of the multiplexers could limit the implementation in ASIC).

5.1.3 Evaluation According to the Design Criteria

Number of Response Bits per Challenge
For the TERO-PUF, by using the difference of the number of oscillations, it is possible to generate from one to three bits after the subtractor to obtain response with the steadiness of up to 10% and the uniqueness of more than 40% in corner conditions for both FPGA and ASIC.

For the RO-PUF, the usual architecture uses only the result of frequency comparison, providing only one bit per challenge. However, it should be possible to use also the difference of the frequencies to generate from one to two bits after the subtractor.

Cost – Area and Patents Pending
The PUF area is determined by the length of the ring. For the same frequency of oscillations, the TERO-PUF uses one AND gate more than the RO-PUF. The area of the additional logic is not negligible: 2 multiplexers, 2 counters, 2 shift-registers (only for TERO-PUF), and a subtractor or comparator is needed.

We underline that for both PUFs, the ASIC implementations require larger oscillators than the FPGA implementations while having similar PUF performance at nominal temperature and voltage conditions. The ASIC implementation of the RO-PUF uses more than two times more basic cells than the TERO-PUF implementation, while having slightly lower PUF performance at nominal temperature (T) and voltage (Vdd) conditions [14]. The RO-PUF and TERO-PUF are apparently not patented.

Power and Energy Consumption
The energy and power consumption is technology dependent, it should be limited if the ROs and the TEROs are not supplied when the PUF is not in use.

5.1.4 Evaluation According to the Quality Criteria

For illustration, we provide below the results of one TERO-PUF and two RO-PUF implementations published in [14] and [48], respectively. To make this comparison relevant, we only compare the TERO-PUF configuration with a one bit response per challenge with RO-PUFs which also generate a one bit response per challenge.

Uniqueness

The uniqueness of the RO-PUF is 47.3% in a 90 nm Xilinx FPGAs [48] and 49.5% in a 65 nm CMOS ASIC [44]. On the other hand, the uniqueness of the TERO-PUF is 48.46% in a 45 nm Xilinx FPGA, 47.62% in a 28 nm Altera FPGA [14], and 49.7% in a 350 nm CMOS ASIC, .

Steadiness
Steadiness of the RO-PUF is 0.9% in a 90 nm Xilinx FPGA [48] and 2.8% in a 65nm CMOS ASIC [44]. Steadiness of TERO-PUF is 1.8% in a 45 nm Xilinx FPGA, 2.63% in a 28 nm Altera FPGA [14], and 0.6% in a 350 nm ASIC, .

Note that for both the RO-PUF and the TERO-PUF, responses can easily be made more unique and more robust to environmental variations by increasing the number of stages in the oscillators, which makes them at the same time less sensitive to the intrinsic noise [14].

Randomness
The TERO-PUF provides 0.97 to 0.99 bits of entropy per response bit in a 350 nm ASIC and 0.85 to 1 bit of entropy per response bits in a 45 nm Xilinx FPGA [14]. Randomness of the RO-PUF implemented in a 60 nm ALTERA FPGA is measured experimentally in [24]. The authors use the *response uniformity* metric close to *bit-aliasing*, while taking into account the independence of bits. Experimental results show response uniformity close to 0.5, but the results depend on the placement of ROs in the device.

5.1.5 Evaluation According to the Security Criteria

Availability or Feasibility of the Model
Currently, stochastic models of the RO-PUF or TERO-PUF are not published/available.

Robustness Against Attacks
The RO-PUF is sensitive to the EM analysis and attacks as shown in [6], [51] and [50]. By using the number of oscillations instead of the oscillation frequency, the TERO-PUF should be less sensitive to this kind of side channel attacks.

5.2 SRAM-PUF

5.2.1 Principle and Design

SRAM PUFs, as proposed by Holcomb et al. [36], are among the most popular and therefore best-studied PUFs in research literature. The original design consists of a default SRAM component, without any modifications to the circuitry. Due to manufacturing variations in the six-transistor (6T) cells, in particular mismatch between the cross-coupled inverters, each cell has a preference to boot either in the zero or one state. So when ramping up the supply voltage of the SRAM, from $0V$ to V_{dd}, a unique fingerprint appears, as illustrated in Figure 5.4. To each cell, we assign a probability p that it initializes in the one state, as can be determined experimentally by booting the SRAM numerous times. We distinguish between stable cells, having $p \approx 0$ or $p \approx 1$, and unstable cells, approaching $p \approx 0.5$. As with any other PUF, helper data algorithms are required in order to obtain a stable and uniformly distributed fingerprint.

The popularity of SRAM PUFs originates from its convenient *off-the-shelf* nature. Inspired by [36], numerous experimental studies have been performed to quantify the PUF characteristics, including intra-distance and inter-distance. For instance, in the European FP7 project UNIQUE, running from 2009 to 2012, SRAM PUFs were cast in $65nm$ CMOS technology. We

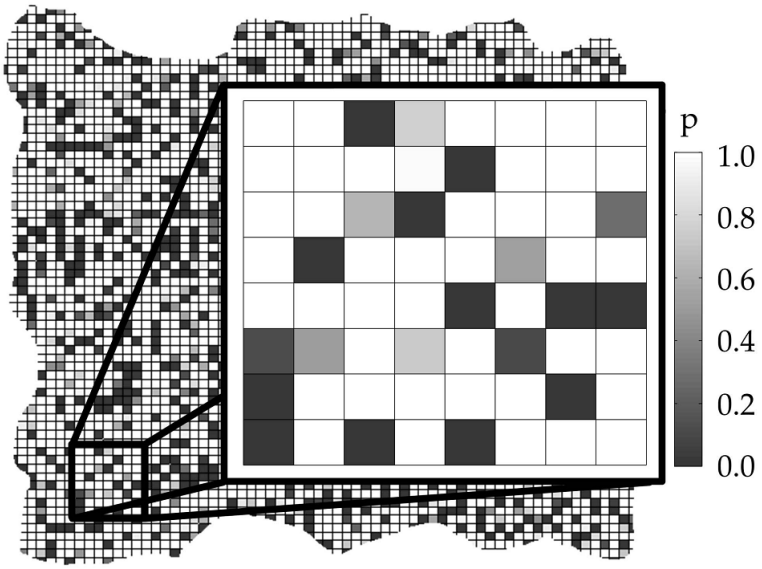

Figure 5.4: An SRAM fingerprint [36]. The grayscale encodes the probability p that a given cell initializes in the one state. White and black cells, corresponding with $p \approx 0$ and $p \approx 1$ respectively are referred to as stable. Gray cells, centered around $p \approx 0.5$, are referred to as unstable.

refer to [38] for the numerical results summarized in tables. Also, we note that SRAM PUFS are commercially available, provided by Intrinsic-ID [2].

Many variations on the original SRAM PUF design have been proposed. A first category of variations applies to the 6T cell. Simply stated, we can plug-in any other bistable memory element. An example thereof is the so-called butterfly PUF [41], targeting FPGA platforms. A second category of variations aims to facilitate the helper data algorithm. It is a common practice to retain the most stable bits only, or stated otherwise, to discard the most unreliable bits. The SRAM circuitry may be adjusted so as to provide rapid thresholding of the aforementioned parameter p, eliminating the need to boot-cycle the SRAM numerous times. This was proposed by Bhargava and Mai in [8], with a prototype manufactured in $65nm$ CMOS. The same approach was also adopted in the European FP7 project HINT [1], running from 2012 to 2015.

5.2.2 Evaluation According to the Technology Criteria

Feasibility in FPGA and ASICs
SRAM PUFs are primarily suitable for ASIC platforms. However, FPGA platforms are not to be excluded. A first example is the aforementioned butterfly PUF [41]. Second, the internal block memories of recent FPGAs might allow for boot-cycling [72]. In addition, the internal SRAM of several commercial microcontrollers might be used as a PUF [70].

Repeatability and Portability
The portability of the design is not limited. The topology of the design remains the same from one technology node to another. Furthermore, SRAMs are typically among the first components that are being mapped to a new technology node. Once designed for the given technology, the SRAM PUF is freely repeatable.

Scalability
SRAMs are scalable. For large data sizes, cell contents are typically distributed over independent memory blocks.

5.2.3 Evaluation According to the Design Criteria

Number of Response Bits per Challenge
The data address is regarded as a challenge while the data itself is regarded as the response. The length of a single response is typically several bytes. The total number of response bits is typically several kbyte or Mbyte.

Cost – Area and Patents Pending
The design comprehends a so-called weak PUF: the total circuit area is roughly proportional to the total number of response bits.

Intrinsic-ID [2] applied for several patents, aiming to enhance the circuitry[1]. However, the core SRAM PUF functionally is not patented, as this design originates from open research [36].

Power and Energy Consumption
The energy and power consumption is technology dependent. It should be limited if the SRAM is not supplied when the PUF is not in use.

5.2.4 Evaluation According to the Quality Criteria

In the following section, essential quality criteria have been evaluated based on real PUFs. The PUF source are 65nm TSMC ASICs, which were developed in the course of the FP7 research project UNIQUE [2].

Randomness
In the PUF context, the Hamming weight gives a first impression of the distribution of 1 and 0 within a response. It is desirable for PUFs to have a fractional Hamming weight of close to 0.5 since this indicates that the PUF response does not show a preference for a certain value. In case of SRAM PUFs, the fractional Hamming weight is always close to the optimum of 0.5 even for different environmental temperatures. This is also reflected by the entropy. SRAM-PUFs provide 0.99 to 1.00 bits of entropy per response bit.

Uniqueness
If the Hamming weight deviates from 0.5, not only the entropy is influenced but also the resemblance between different PUFs will increase, which negatively impacts uniqueness [67, p. 22]. Furthermore, the responses from two different devices should not be correlated and must differ with high probability. The inter-device distance, cf. EC in Section 2.2.2, is used to evaluate the uniqueness of a PUF device. For the evaluated SRAM-PUFs the uniqueness is around 49%.

Steadiness
The bit error probability is the metric for steadiness and corresponds to the intra-device distance, cf. IC in Section 2.2.2, which is defined as the distance between the two responses, resulting from applying the same challenge twice to a PUF. For SRAM-PUFs the steadiness is

[1] http://www.faqs.org/patents/assignee/intrinsic-id-bv/
[2] www.unique.technicon.com

5% to 6% at room temperature, and slightly higher (7%) for extreme temperatures (−40°C, 85°C).

5.2.5 Evaluation According to the Security Criteria

Availability or Feasibility of the Model
Roel Maes [43] presented a probabilistic reliability model, which allows to model the bit rate of SRAM PUFs.

Robustness Against Attacks
As any other security building block, SRAM-PUFs are potentially vulnerable to side-channel attacks. Helfmeier *et al.* [32] described optical emission attacks on SRAM-PUFs, while Oren, Sadeghi and Wachsmann [52] presented remanence decay side-channel attacks on SRAM PUFs.

5.3 Conclusions on PUF Selection

From the above presented analysis it is clear that an ideal PUF does not exist. All the presented physical unclonable functions will be first implemented in FPGAs and their parameters thoroughly evaluated. This will provide further inputs and comparison data. Among the above selected principles, at least one will be chosen for implementation in ASIC.

Chapter 6

Implementation of Selected TRNG and PUF Cores in FPGAs

6.1 Methodology

As explained in previous chapters, the objective of the implementation of selected TRNG and PUF cores in FPGAs was to fairly evaluate their design and behavior when operating at the same conditions. One of the advantages of using FPGAs in this phase of the project was, that thanks to the flexibility of FPGAs, many DUT configurations and topologies could be tested in a relatively short time. On the other hand, it must be taken into account that some obtained results and observations cannot be directly generalized to ASICs.

We recall that our objective was to use the same hardware configuration, and the same FPGA family for all TRNG and PUF cores. For practical reasons (availability of evaluation boards), we decided to use the Xilinx Spartan 6 FPGA family for implementation of all TRNGs and PUFs in this first stage of the project.

Since the HECTOR evaluation board with the Spartan 6 module was not available from the very beginning of the HECTOR project, we adapted existing Evariste hardware/software design tools available at the Hubert Curien Laboratory (UJM partner) to our needs. The modifications of the Evariste system made in the framework of the HECTOR project are described briefly in the next sections.

6.1.1 Data acquisition hardware/software

In the first stage of the HECTOR project, the Evariste acquisition card and associated software was adapted to transfer data streams of up to 4 MB from the Device Under Test (DUT) to the PC via USB bus. The complete acquisition system has three components (see Fig. 6.1): the DUT, the Acquisition card and the PC running the software.

The DUT can be implemented in any hardware device featuring outputs that support low voltage differential signaling (LVDS), e.g. the Evariste FPGA hardware modules. The acquisition card is composed of the Evariste motherboard containing the Cyclone III FPGA module, which includes a 4 MB RAM memory. The DUT is connected to the acquisition card using a simple serial connection and the acquisition board is connected to the PC using the USB bus.

It is clear from Fig. 6.1 that the DUT is expected to send a serial data stream and a data strobe signal via two LVDS links. This is advantageous for several reasons:

1. The data interface in the DUT is very simple and its impact on the operation of the TRNG or PUF core is reduced to a minimum.
2. Because of the use of the LVDS data transfer method, the DUT can be placed relatively far from the acquisition card (e.g. placed in a Faraday cage) and the data transfer is more robust.
3. The use of the memory block guarantees undisrupted data transfers from DUT to PC (note that the USB bus cannot guarantee continuity of data transfers).

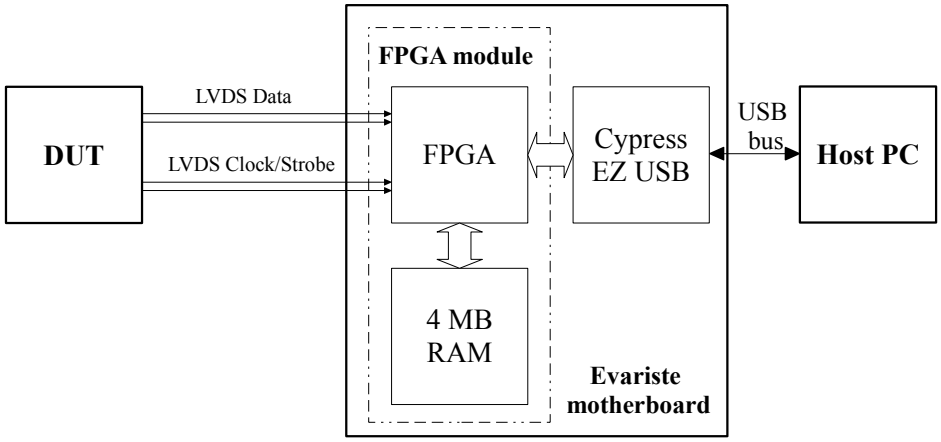

Figure 6.1: Block diagram of the acquisition system based on the Evariste hardware/software tools

The bit rate of the serial data stream can vary between 0 and 16 Mbits/s in the asynchronous data transfer mode and between 6 and 400 Mbits/s in the synchronous data transfer mode. The two basic modes of the data transfer are described in detail in the next sections.

6.1.2 Asynchronous Data Acquisition Mode

In the asynchronous data acquisition mode, the data in the bit-stream are validated by the rising edge of the data strobe signal. The data rate can vary between 0 (or almost zero) and 16 Mbits/s and it can be irregularly distributed in time. The width of the data strobe pulse must be longer than 31.25 ns (see Fig. 6.2).

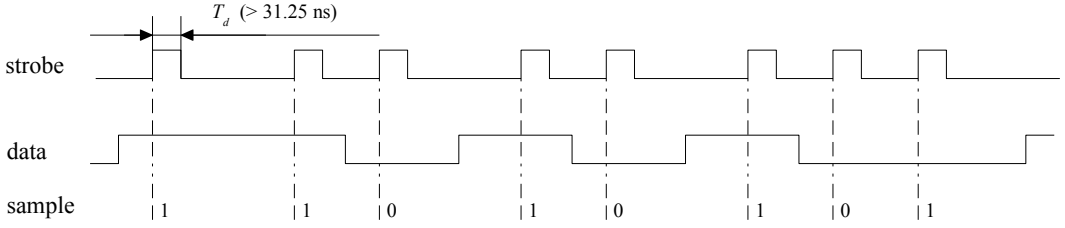

Figure 6.2: Waveforms of the asynchronous data acquisition

Up to 4 MB of data can be saved in the acquisition card memory in real time, so the smallest data rate is limited only by the total acquisition time, which can be very long for small frequencies.

6.1.3 Synchronous Data Acquisition Mode

In the synchronous data acquisition mode, the data arrival to the acquisition card is synchronized by the data clock. Inside the acquisition card, the acquisition clock signal is synchronized to this input data clock using a phase-locked loop (PLL). The data clock signal coming from the DUT can have any frequency between 6 MHz and 400 MHz and it has to be provided continuously (the internal logic runs using this clock signal). Data bits arriving to the acquisition card are registered on the rising edges of this clock signal (see Fig. 6.3).

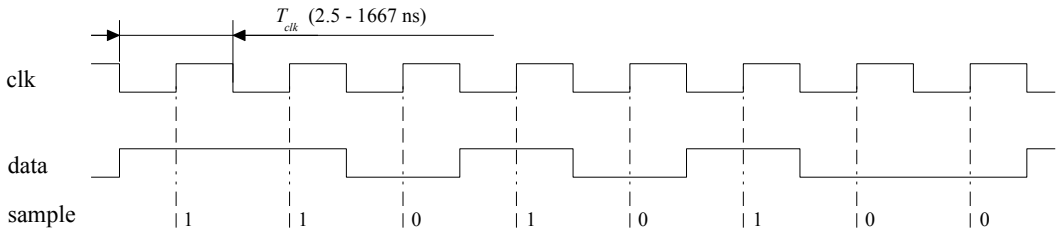

Figure 6.3: Waveforms of the synchronous data acquisition

6.2 Strategy of TRNG and PUF Implementation and Evaluation

We repeat that our aim was to fairly evaluate several aspects of the TRNG and PUF designs: their feasibility in FPGAs and difficulties related to their implementation in reconfigurable devices, their cost in term of combinatorial and sequential elements such as LUTs and flip-flops used, their power and/or energy consumption, the entropy rate and bit rate at the output in case of TRNGs and the number of bits per challenge in case of PUFs.

Concerning the topology of individual TRNGs and PUFs, our objective was to reduce internal complexity of the modules, while maintaining the data interface described in the previous section. In order to reduce the vulnerability of the generators and PUFs to external manipulations, we avoided to use external clocks – all clock signals are generated inside the evaluated blocks, for example using a ring oscillator with a convenient topology.

The architectures of logic elements (or logic cells) in various FPGA families are slightly different. Nevertheless, all of them are currently divided into two parts: LUTs and registers. In our area comparisons, we distinguish between the use of LUTs (purely combinatorial cells), registers (purely register cells) and the combined use of LUTs and registers (combined combinatorial and sequential cells). These three cases can be usually recognized in all implementations in all families.

One of the parameters used for design evaluation is the power consumption. The power consumption of TRNGs and PUFs is relatively small and it is mostly comparable to, or even

smaller than, the standby power consumption of an empty device. For this reason, we first implemented a reference design, in which an input static signal just crossed the device and no logic was implemented inside. The Spartan 6 device with this small reference project consumed 4.7 mW. This power is then subtracted from the total power consumptions measured in all experiments. The tables presented in the following sections give thus the net power consumption of the tested designs.

For a coarse evaluation of the statistical quality of the generated signals, we use the NIST SP 800-90B test suite, because it makes possible an easy evaluating the independence of random variables and evaluation of the entropy rate.

6.3 Study of the ELO-TRNG Core Implemented in FPGA

6.3.1 TRNG Design

The block diagram of the elementary oscillator based TRNG (ELO-TRNG) core implemented in the selected Spartan 6 FPGA is depicted in Fig. 6.4. The TRNG core uses two identical ring oscillators (RO1 and RO2) as sources of randomness. Thanks to the differential principle, the impact of the global sources of randomness, which can be easily manipulated, is significantly reduced. The two identical rings were composed of two and five buffers ($N = 2$ and $N = 5$) in two different TRNG configurations, which were tested.

The time base (the sampling period) was generated using a 20-bit synchronous counter, which divided the *RO2* output clock frequency by K. The sampling period (and thus the parameter K) was set up depending on the clock frequency and the size of the jitter, which was measured at an LVDS output of the FPGA using a high bandwidth differential probe of the oscilloscope. The parameter K was computed according to Equation (2) in [29]. The period jitter of the clock signal generated in both ring oscillators was between 4.8 and 5.3 ps for the clock periods of about 5 ns and between 5.5 and 6 ps for the clock periods of about 10 ns and the total period jitter exploited in the TRNG was about 7 ps and 8 ps, respectively.

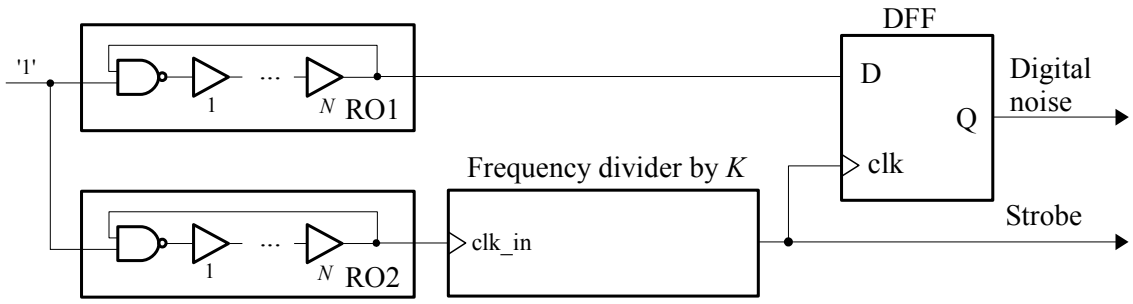

Figure 6.4: Architecture of the ELO-TRNG core as implemented in Spartan 6 FPGA

The two ring oscillators were placed and routed manually in order to maintain repeatability of the design in future TRNG implementations, since we plan to study the impact of the activity of the surrounding logic (such as a USB data interface or a cipher). Although both rings were placed in the close vicinity, apparently, they did not lock.

6.3.2 Implementation Results

Results of implementation of the two configurations of the ELO-TRNG core in the Xilinx Spartan 6 FPGA family are presented in Table 6.1.

Table 6.1: Results of implementation of the ELO-TRNG in the Xilinx Spartan 6 FPGA family

Configuration	**Frequency** [MHz]	**K** [$\cdot 10^3$]	**Bit rate** [Mbits/s]	**Area** (LUT/Reg/L&R)	**Power cons.** [mW]	**Entropy** per bit
1 NAND + 2 Buf.	$\simeq$ 200	51.766	0.0034	23/0/18	1.38	0.896
1 NAND + 5 Buf.	$\simeq$ 100	122.432	0.0008	31/0/19	0.96	0.931

6.3.3 Discussion

Several facts concerning the ELO-TRNG core can be derived from implementation results:

- The area of the TRNG is relatively small at the expense of a low output bit rate.
- The design is simple and it is thus simple to analyze and to model.
- The rings should have the same topology in order to compensate the impact of the global jitter sources on the generator. This requires manual placement (and eventually routing) of rings, which is a relatively simple operation.
- Higher clock frequency increases the output bit rate at the cost of higher power consumption.
- Since at each moment, there is only one event (rising or falling edge), which propagates in the ring, the power consumption of the ring does not depend on the number of delay elements and it is almost constant in all rings. Therefore, the higher power consumption of the TRNG is not caused by a higher number of delay elements in the rings, but rather by the frequency of the clock signal used in the counter.

6.4 Study of the COSO-TRNG Core Implemented in FPGA

6.4.1 TRNG design

The block diagram of the coherent sampling oscillator based TRNG (COSO-TRNG) core implemented in the selected Spartan 6 FPGA is depicted in Fig. 6.5. The two oscillators have the same number of elements ($N = 2$) and their topology (placement of the delay elements) is the same. The period jitter of both clock signals, which was measured using the LVDS outputs of the device and a differential oscilloscope probe, was about 4.8 ps with a period of about 5 ns. The total jitter exploited in the TRNG (when assuming independence of the two clock generators) was therefore about 6.8 ps. This value is needed to know the maximum acceptable difference between the two clock periods.

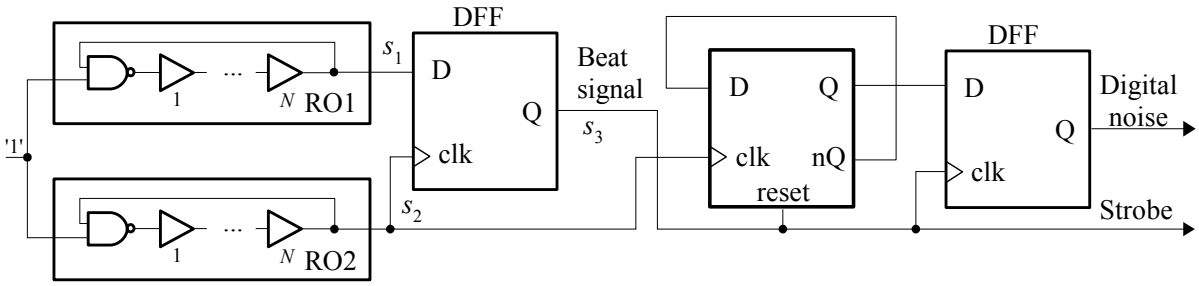

Figure 6.5: Architecture of the COSO-TRNG core as implemented in Spartan 6 FPGA

The clock signal s_1 was sampled in a D flip-flop on rising edges of the clock signal s_2. The resulting beat signal was then used to flip the T flip-flop (a 1-bit counter).

The main difficulty in designing the COSO-TRNG is the need to precisely set up the periods of the two rings – the difference in the output periods must be small enough, i.e. comparable in size to the period jitter. Unfortunately, even the rings that have exactly the same topology can generate clock signals with periods, which differ too much because of dispersion of electrical parameters inside the device.

In order to obtain two periods that are sufficiently close, we placed one oscillator at a stable position inside FPGA and the other ring was moved automatically (using a script written in the TCL language) to different places in the device until the period difference between the two generated clock signals was sufficiently small. Once a convenient difference in periods was obtained, the placement and routing constraints of the two oscillators were saved to prevent any changes caused by future project recompilations.

The process of finding the right solution can be further optimized, however, the results will remain device dependent – this optimization process must be repeated individually for all devices. This makes the practical use of the generator questionable, if some other automatic way of setting up the clock frequencies will not be found.

6.4.2 Implementation Results

Results of implementation of the COSO-TRNG core in the Xilinx Spartan 6 FPGA family are presented in Table 6.2.

Table 6.2: Results of implementation of the COSO-TRNG in the Xilinx Spartan 6 FPGA family

Configuration	Fr. RO1 [MHz]	Fr. RO2 [MHz]	Bit rate [Mbits/s]	Area (LUT/Reg/L&R)	Power cons. [mW]	Entropy per bit
1 NAND + 2 Buf.	$\simeq$ 204.7	$\simeq$ 204.24	1.32	5/1/2	0.8	0.9597

6.4.3 Discussion

Several facts concerning the COSO-TRNG core can be derived from implementation results:

- The area of the TRNG is very small, while the output bit rate can be relatively high.

- The architecture of the TRNG is simple, but the design needs placement of at least one ring on a per-device basis. Once the difference in periods is sufficiently small, the design can be easily modeled.
- Since the rings have the same topology (this requirement is a must for reducing the period difference), the impact of the global jitter sources on the generator is significantly reduced.
- Higher clock frequency increases the output bit rate, while the power consumption does not change significantly.

6.5 Study of the PLL-TRNG Core Implemented in FPGA

6.5.1 TRNG design

The block diagram of the PLL-TRNG core implemented in the Spartan 6 FPGA is depicted in Fig. 6.6. The Spartan 6 device used in the Evariste modules has only two PLLs available. Moreover, the device-specific PLL routing constraints make the design of the PLL-TRNG less straightforward than in other FPGA families (e.g. Altera or Microsemi).

The main difficulty in designing the PLL-TRNG in Spartan 6 family is related to the fact that the outputs of the PLL block are routed via dedicated clock wires into different clock regions. However, because of the TRNG principle (one clock signal must be sampled in a D flip-flop using another clock signal), both clock signals generated in two different PLLs must appear in the same clock region.

The easiest solution currently available is to use a non-dedicated clock path constraint that forces the router to route the PLL output signal via general purpose interconnection wires (not the dedicated clock paths). We ensured that only the clk_{jit} (i.e. the sampled clock) signal was routed this way and the clk_{ref} (the sampling clock) was routed via a standard dedicated clock path. We are aware that this kind of routing could be more sensitive to ambient noises and it could thus bring more unwanted (manipulable) jitter to the clock signal.

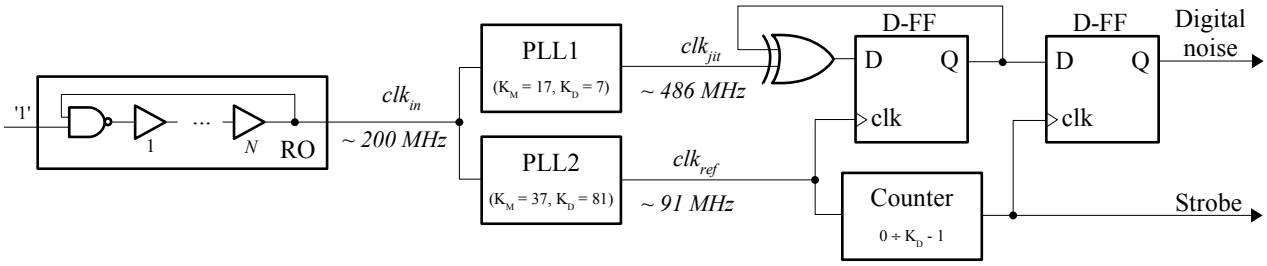

Figure 6.6: Architecture of the PLL-TRNG core as implemented in Spartan 6 FPGA

Fortunately, PLL blocks in Altera and Microsemi FPGAs do not have this kind of constraint and we can therefore expect that the PLL-TRNG will be easier to implement in these FPGAs.

6.5.2 Implementation Results

Results of implementation of the PLL-TRNG core in the Xilinx Spartan 6 FPGA family are presented in Table 6.3.

Table 6.3: Results of implementation of the PLL-TRNG in the Xilinx Spartan 6 FPGA family

Config.	RO fr.	PLL1	PLL2	Final	Bit rate	Area	Power con.	Entropy
	[MHz]	$[K_M/K_D]$	$[K_M/K_D]$	$[K_M/K_D]$	[Mbits/s]	(L/R/L&R)	[mW]	per bit
Two PLLs	$\simeq 200$	17/7	37/81	1377/259	0.35	15/0/13	10.5	0.860

6.5.3 Discussion

Several facts concerning the PLL-TRNG core can be derived from implementation results:

- The PLLs are expensive hardware blocks, since they occupy huge silicon area. However, in the context of FPGAs, they are available 'for free'. However, on the other side, once used in TRNG, they will not be available for the rest of the design. This disadvantage can be reduced by sharing at least one PLL by the TRNG and the application design.
- Besides the area occupied by PLLs, the additional area of the TRNG is very small – it is occupied essentially by the ring oscillator serving as a source of the clock signal.
- Because of the use of voltage controlled oscillators (VCO) oscillating at high frequencies, the power consumption of the TRNG is relatively high.
- The generator does not need manual placement and routing.
- The source of randomness is perfectly isolated from the rest of the device.

6.6 Study of the DC-TRNG Core Implemented in FPGA

6.6.1 TRNG design

The block diagram of the DC-TRNG core architecture is depicted in Fig. 6.7. A 3-element ring-oscillator RO1, implemented using 3 LUTs, is used as the entropy source. Each LUT is placed in a separate slice. The entropy is extracted using 3 tapped delay lines and a priority encoder. Each tapped delay line is implemented using a delay chain of 9 slices. In each slice a CARRY4 element and 4 flip-flops are utilized. CARRY4 elements from the same delay line are placed in neighboring slices along the y-axis. Placement constraints are specified within the Verilog description. Half of all slices available on Spartan-6 contain CARRY4 primitives.

Another ring oscillator (RO2) generated the clock of $125MHz$, which was used to generate the time base. Two configurations were tested, one with jitter accumulation time of 2 clock cycles ($N = 2$) and one with accumulation time of 4 clock cycles ($N = 4$).

6.6.2 Implementation Results

The results of implementation of the DC-TRNG core in the Xilinx Spartan 6 FPGA family are presented in Table 6.4.

6.6.3 Discussion

Several facts concerning the DC-TRNG core can be derived from implementation results:

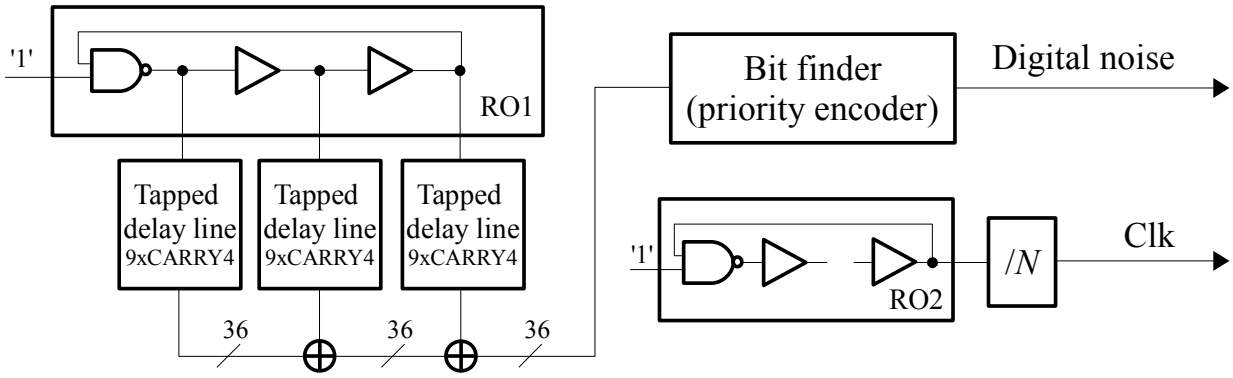

Figure 6.7: Architecture of the DC-TRNG

Table 6.4: Results of implementation of the DC-TRNG in the Xilinx Spartan 6 FPGA family

Configuration	RO1 freq. [MHz]	RO2 freq. [MHz]	Bit Rate [Mbits/s]	Area (LUT/Reg/L&R)	Power cons. [mW]	Entropy [per bit]
$t_A = 16ns$	$\simeq 280$	$\simeq 125$	62.5	(65/0/110)	9.66	0.110
$t_A = 32ns$	$\simeq 280$	$\simeq 125$	31.25	(65/0/112)	10.02	0.353

- For the implementation of tapped delay lines, the design uses FPGA primitives that ar neither LUTs nor Registers. In the area column, these results are repported as fully occupied slices (all 4 LUTs and 4 FFs).
- Placement constrains are needed for the ring oscillator (each LUT is placed in a separate slice).
- The core has a compact implementation occupying only 54 slices on Spartan-6 FPGA.
- The architecture of the TRNG is very simple. Placement constraints are required but they can be specified within the Verilog file. No routing constraints are required.
- The output bit rate is very high. Entropy per bit can be improved by using a longer accumulation time at the cost of the decreased output bit rate.

6.7 Study of the TERO-TRNG Core Implemented in FPGA

6.7.1 TRNG design

The block diagram of the transition effect ring oscillator based TRNG (TERO-TRNG) core implemented in the Spartan 6 FPGA is depicted in Fig. 6.8. The TERO cell is built up using 12 buffers and 1 NAND gate per branch. The control signal is generated with a ring oscillator (NAND+12 buffers) connected to a 7-bit synchronous counter. The duty cycle of the TRNG control signal, which is the most significant bit of the counter, is 50%. The duty cycle can be changed in the future to improve the bitrate – only a small part of the control period is needed to reset the counter.

We also implemented the TERO core with 8 buffers per branch, but the number of oscillations was very low and relatively stable – not enough entropy could be accumulated during few oscillation periods. Even with 12 buffers, the number of oscillations did not change too much and consequently, the entropy rate was relatively low (see results presented in the following section).

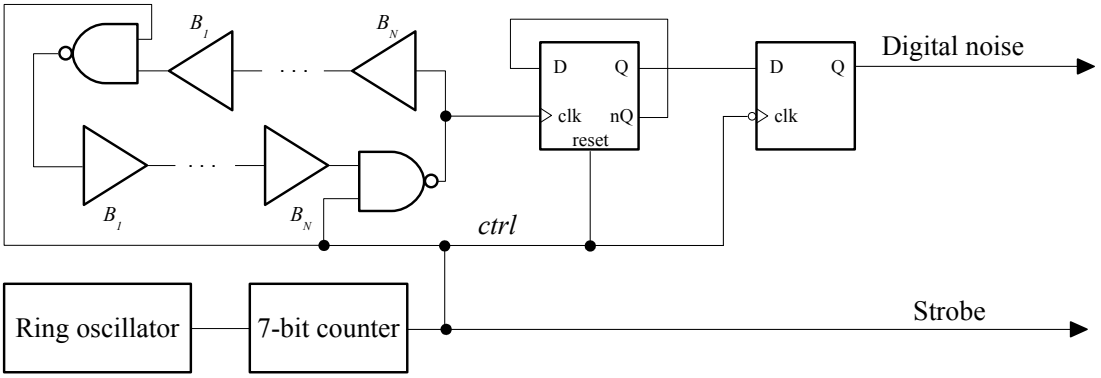

Figure 6.8: Architecture of the TERO-TRNG core as implemented in Spartan 6 FPGA

6.7.2 Implementation Results

Results of implementation of the TERO-TRNG core in the Xilinx Spartan 6 FPGA family are presented in Table 6.5.

Table 6.5: Results of implementation of the TERO-TRNG in the Xilinx Spartan 6 FPGA family

Configuration	RO fr. [MHz]	Number [of oscill.]	Bit rate [Mbits/s]	Area (LUT/Reg/L&R)	Power cons. [mW]	Entropy [per bit]
(1 NAND + 12 Buf) x 2	215.26	36	0.419	38/0/7	1.0	0.275

6.7.3 Discussion

Several facts concerning the TERO-TRNG core can be derived from implementation results:

- The area of the TRNG is very small, while the output bit rate can be relatively high.
- The architecture of the TRNG is simple, but in order to achieve sufficient entropy rate, the two TERO cell branches must be balanced in such a way that the number of oscillation periods will be between 100 and 200. This is difficult to obtain repeatedly. Perfectly balanced TERO cell will oscillate permanently and very unbalanced cell will feature very few oscillation periods.
- The number of delay elements does not have direct impact on the power consumption, since in each TERO configuration at any time, two events (two rising edges or falling edges) pass across the TERO cell. However, it seems that higher number of delay elements makes the TERO cell design easier, then the balance can be set up in smaller steps.

6.8 Study of the STR-TRNG Core Implemented in FPGA

6.8.1 TRNG design

The block diagram of the STR-TRNG core implemented in the Spartan 6 FPGA is depicted in Fig. 6.9. As explained earlier, the STR must work in the evenly spaced mode. Therefore, the Muller cells must feature additional reset and preset inputs, which allow to set up the correct number of events during initialization of the self-timed ring.

The sampling clock is generated by an additional ring oscillator. Each STR stage is sampled in the D flip-flop and then they are XOR-ed together. The XOR output is sampled again in another D flip-flop at the same frequency.

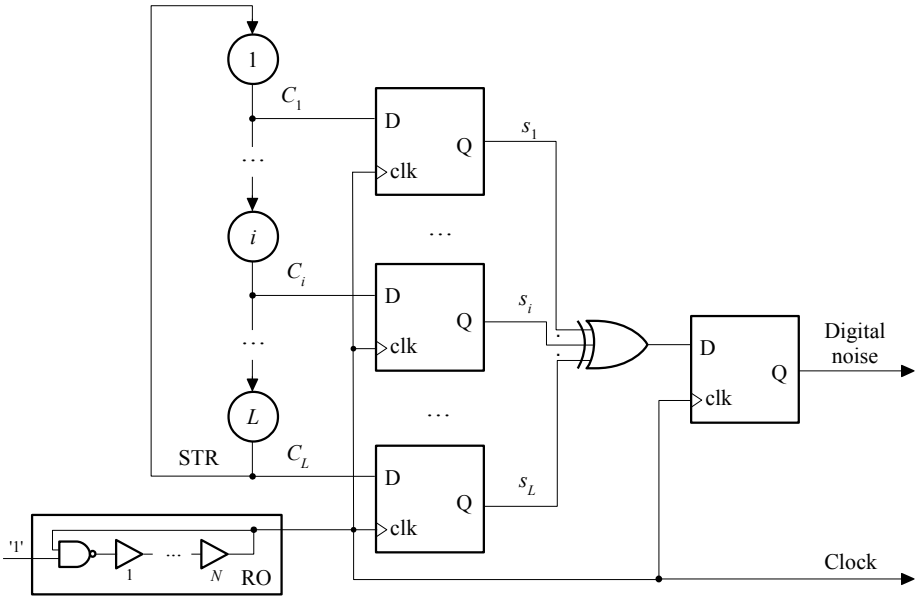

Figure 6.9: Architecture of the STR-TRNG core as implemented in Spartan 6 FPGA

6.8.2 Implementation Results

Results of implementation of the STR-TRNG core in the Xilinx Spartan 6 FPGA family are presented in Table 6.6.

Table 6.6: Results of implementation of the STR-TRNG in the Xilinx Spartan 6 FPGA family

Configuration	STR freq. [MHz]	RO freq. [MHz]	Bit rate [Mbits/s]	Area (LUT/Reg/L&R)	Power cons. [mW]	Entropy [per bit]
127 Muller cells	307.6	143	143	7/0/170	18.96	0.9305

6.8.3 Discussion

Several facts concerning the STR-TRNG core can be derived from implementation results:

- Although the implementation of the Muller cell in LUT based FPGAs (most of currently available FPGAs) is relatively simple, the STR-TRNG needs careful placement and routing to guarantee the evenly spaced mode (and thus precise timing between events) and to maintain the frequency as high as possible (since the number of events circulating inside the ring is constant and guaranteed by the initialization of Muller cell at start up, the phase difference between events is reduced with reduced clock period and, consequently, the entropy rate is thus higher).
- The output bit rate is extremely high at the expense of high power consumption.
- The entropy rate is relatively high regarding the high bit rate.

6.9 Study of the RO-PUF and TERO-PUF Core Implemented in FPGA

6.9.1 PUF design

The architecture of the RO-PUF implemented in FPGA is presented in Fig. 6.10. It is composed of two groups of 16 ring oscillators having the topology presented in Fig. 2.5 (right panel): each RO contains one NAND gate and twelve non-inverting buffers. The controller generates the time base signal from the input clock (*clk*). Depending on this time base signal, it selects one of sixteen couples of ROs using two multiplexers.

Number of periods of the clock signal present at the output of each multiplexer is counted in two synchronous 10-bit counters during each time base period. The arbiter detects which counter reaches its maximum value as the first one (i.e. the most significant bit of the corresponding counter toggles as the first one). If it is the one of Channel A (upper ROs in Fig. 2.5), the output of the arbiter is set to one, if it is the one of Channel B, the arbiter output is reset to zero.

Since in the first version of the RO-PUF all ring oscillator oscillated permanently, the power consumption was relatively high. However, at any time, just two out of 32 rings were needed. Therefore, in the power optimized version depicted in Fig. 6.11, the unused rings were stopped using demultiplexers. The rest of the PUF circuitry remained unchanged.

The TERO-PUF implemented in selected FPGA is depicted in Fig. 6.12. It is composed of two groups of 16 transition effect ring oscillators: each TERO contains one NAND gate and six non-inverting buffers in both branches. The controller generates the time base signal from the input clock (*clk*). Depending on this time base signal, it selects one of sixteen couples of TEROs using two multiplexers.

Number of oscillation periods of the two TERO cells selected by multiplexers is counted in two synchronous 10-bit counters during each measurement interval (the time base period generated by the controller). The counter values are then subtracted to give up to 3-bit response of the PUF. The counters are reset at the end of the measurement period and another couple of TERO cells is selected for measurement.

6.9.2 Implementation Results

The results of implementation of the RO-PUF and TERO-PUF and their cells are presented in Table 6.7. First, an RO cell (one ring oscillator containing twelve buffers and one NAND gate)

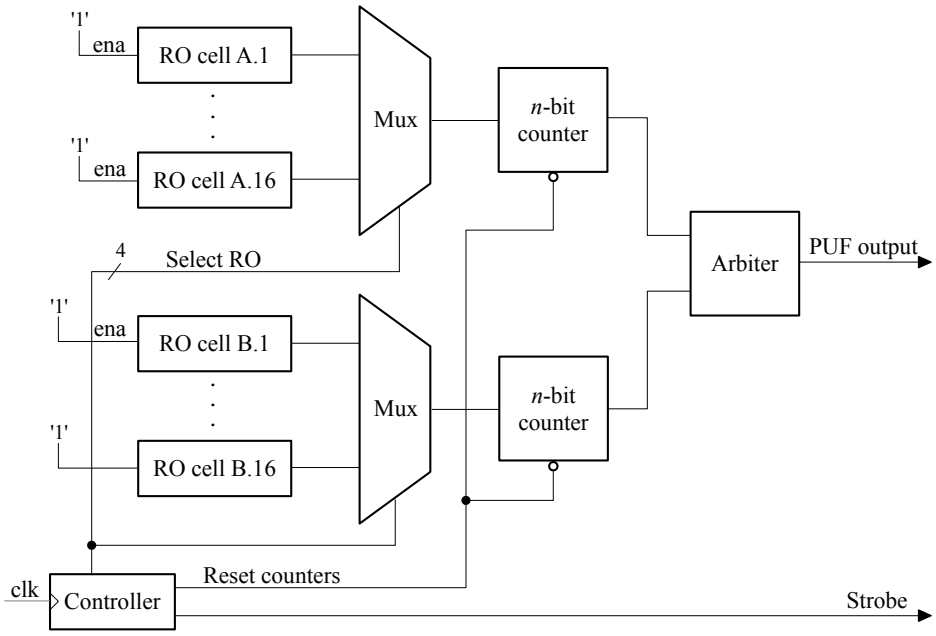

Figure 6.10: Architecture of the RO-PUF as implemented in Spartan 6 FPGA

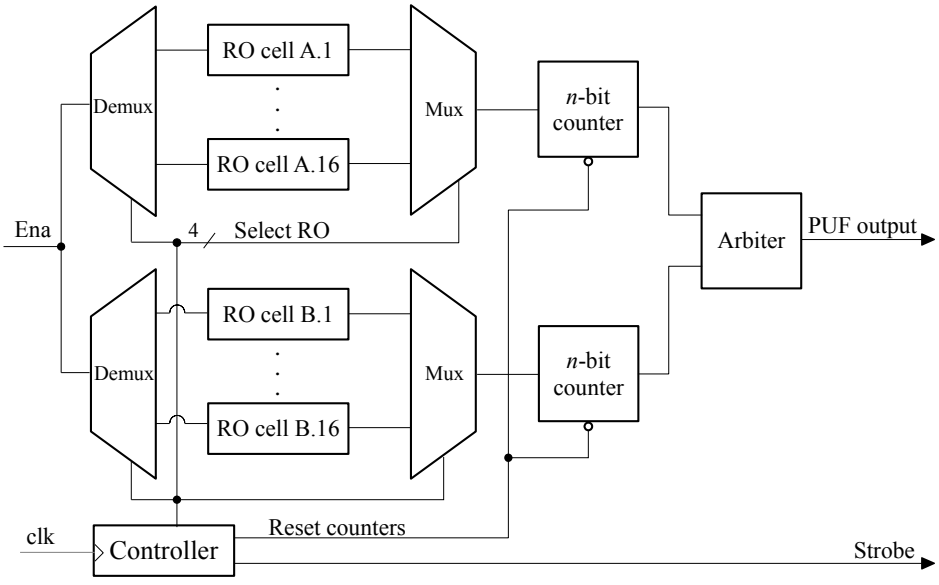

Figure 6.11: Power-optimized architecture of the RO-PUF as implemented in Spartan 6 FPGA

and a TERO cell (one transition-effect ring oscillator composed of two branches containing six buffers and one NAND gate each) were implemented in the device and observed.

The RO cell oscillated at about 80 MHz and the TERO cell at about 180 MHz. Although the total number of the delay elements (inverters and NAND gates) in two kind of cells (RO and TERO) was similar, the power consumption was quite different (see the first two lines in the table). This is mainly due to the fact that the RO cell oscillates permanently and the

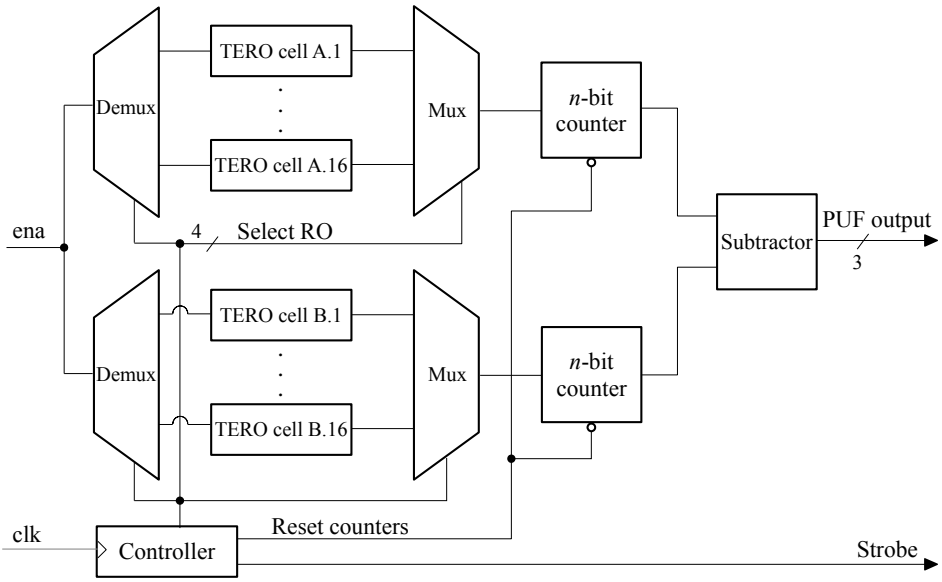

Figure 6.12: Power-optimized architecture of the TERO-PUF as implemented in Spartan 6 FPGA

TERO cell stops to oscillate after some time. When it stops to oscillate, the power consumption significantly reduces. Of course, using shorter measurement periods in the TERO-PUF, would increase the power consumption, but also decrease the measurement latency. In order to get comparable results, the measurement period was set up for both RO-PUF and TERO-PUF to about 16 μs.

As can be observed in Table 6.7, the RO-PUF optimized for the power consumption consumes four times less power than the straightforward architecture of the RO-PUF at the expense of slightly more than 10% area extension, which is due mainly to the use of demultiplexers.

Table 6.7: Results of implementation of the RO-PUF and TERO-PUF in the Xilinx Spartan 6 FPGA family

DUT	**Area** (LUT/Reg/L&R)	**Power consumption** [mW]	**Nb bits per challenge**
RO cell (1 NAND + 12 Buf.)	13/0/0	1.4	–
TERO cell (1 NAND + 6 Buf.) x 2	14/0/0	0.5	–
RO-PUF (2 x 16 ROs)	177/0/37	14.2	1
RO-PUF optimized for power	191/0/39	2.9	1
TERO-PUF (2 x 16 TEROs)	343/1/38	3.8	up to 3

The TERO-PUF has almost the same structure as the RO-PUF. The only difference is that instead of using simpler arbiter, it uses more complex subtractor, which occupies more area and consumes more energy. In return, the PUF response has up to three bits. However, it must be taken into account that according to [14], the quality of the PUF in term of steadiness, uniqueness and randomness deteriorates when using three bits per challenge.

6.9.3 Discussion

Several facts concerning the RO-PUF and TERO-PUF core can be derived from implementation results:

- The RO-PUF and TERO-PUF have very similar structure, but give different results.
- The TERO-PUF consumes more power, probably because TERO cells oscillate at higher frequency and the counters counting at higher frequency consume more power. However, it must be taken into account that the power consumption of the TERO-PUF can be reduced by increasing the period of the control signal (the oscillation will last relatively shorter time period regarding the total measurement time).
- When using only one bit per challenge in the TERO-PUF, the RO-PUF and TERO-PUF give similar statistical results. However, it seems that the TERO-PUF is more difficult to manipulate (to attack).

6.10 Discussion on FPGA Implementation of Selected TRNG Cores

As explained above, the aim of implementation of selected cores of TRNGs in FPGA was to fairly evaluate their design and operation when operating in the same conditions. Thanks to the flexibility of FPGAs, many DUT configurations and topologies could be tested in a relatively short time. We have to stress again that we are aware that some of the obtained results and observations (e.g. availability of logic and routing resources or feasibility) cannot be directly generalized to ASICs and that other strategies of evaluation must be adopted to them.

In the next paragraphs, we will evaluate and briefly discuss the first results obtained in the Xilinx Spartan 6 family. In order to fairly evaluate selected designs, we propose to compare the next parameters:

- *Area* – the final area will be given in logic elements (i.e. logic cells) occupied by the design (in the context of results presented in previous sections, the area will be proportional to the sum of LUTs, registers and combined LUTs and registers, since each of these objects occupies one logic element).
- *Power consumption* in mW – this parameter will give the power consumption only of the core of the TRNG.
- *Bit rate* in Mbits/s – since the data interface is faster (400 MBits/s) than all available TRNG designs, the speed of the acquisition card will not limit the speed of the data transfer.
- *Entropy rate* per output bit – the entropy will be estimated using the NIST SP 800-90B procedure.
- *Entropy and bit rate product* – since the bit rate and the entropy rate at the output of the generator are closely related (output with high bit rate and low entropy rate can be post-processed, in order to increase entropy at the expense of decrease of the bit rate), they should be evaluated together using the same parameter. We use the product of the entropy rate and of the bit rate.

- *Feasibility and repeatability* – this evaluation parameter will reflect the difficulty of the design and its repeatability across selected FPGA family. The parameter value is divided into six levels, which will be explained later.

In the next step, we propose to attribute a score to all above mentioned TRNG parameters. The proposed score scale is between 0 (the lowest score) to 5 (the highest score) according to Table 6.8.

Table 6.8: Scoring intervals for TRNG parameters

Parameter	**5**	**4**	**3**	**2**	**1**	**0**
Area [Logic cells]	< 20	20 – 99	100 – 199	200 – 499	500 – 999	> 999
Power [mW]	< 0.01	0.01 – 0.09	0.1 – 0.9	1.0 – 9.9	10 – 99.9	> 99.9
Bit rate [Mbits/s]	> 99	99 – 10	9.9 – 1	0.99 – 0.1	0.099 – 0.01	< 0.01
Entropy rate	> 0.997	0.997 – 0.99	9.989 – 0.9	0.899 – 0.5	0.499 – 0.1	< 0.1
Entropy * Bit rate	> 99	99 – 10	9.9 – 0.9	0.9 – 0.05	0.05 – 0.001	< 0.001
Feasib. & repeatability	Automatic all families	Man. setup per family	Man. setup iterative	Man. setup complex	Man. setup per device	Random results

While the score scale of the first five parameters is quite straightforward, the scale of the *Feasibility and repeatability* needs some explanation.

The highest score (5) will be given to designs that do not need any manual intervention and the obtained results are always satisfactory, independently from the family. The results are repeatable for all devices.

A score of 4 will be given to designs that need some simple manual setup depending on the family (e.g. manual placement) and the obtained results are repeatable between devices inside the same family.

A score of 3 will be given to designs that need a manual setup (optimization of parameters) in an iterative manner. This manual optimization cannot be automatically translated from one family to another, but remains the same for all devices in the same family (repeatability).

A score of 2 will be given to designs that necessitate the use of some complex algorithms (optimization of the topology and routing). The obtained results are the same for all devices in the same family (repeatability).

A score 1 will be given to designs that need manual setup for each device individually, but a satisfactory solution can always be obtained.

A score 0 will be given to designs, in which satisfactory results cannot be guaranteed (they appear randomly).

Table 6.9 presents a summary of implementation results of selected TRNG designs. It can be observed that the ELO-TRNG occupies relatively small area and consumes relatively low power. It is also very easy to implement (highest feasibility and repeatability). It also reaches the highest entropy rate between evaluated generators at the expense of very small bit rate.

The COSO-TRNG occupies the smallest area and consumes the lowest power. It has an interesting bit rate. However it is difficult to implement (low feasibility and repeatability). It also reaches a high entropy rate and relatively high entropy and bit rate product (EBR).

We obtained very interesting results with the STR-TRNG. It has the highest bit rate and relatively high entropy and thus very high EBR. On the other hand, it has the highest power consumption and it is relatively difficult to implement.

Table 6.9: Summary of implementation results of selected TRNGs

TRNG type	Area	Power	Bit rate	Entropy	E * BR
	(LUT/Reg/L&R)	[mW]	[Mbits/s]	per bit	
ELO	41	1.38	0.003	0.986	0.003
COSO	8	0.80	1.32	0.960	1.267
PLL	28	10.5	0.355	0.860	0.305
DC	175	9.66	62.5	0.1105	6.906
TERO	45	1	0.419	0.275	0.115
STR	177	18.9	143	0.931	133.062

Table 6.10 presents scores of implementation of selected TRNG designs. As could be expected from the previous analysis, the STR-TRNG obtained the highest score, followed by COSO-TRNG and DC-TRNG, but the differences are very small.

Table 6.10: Scores of selected TRNGs

TRNG	Area	Power	Bit rate	Entropy	E * BR	Feas. & Repeat.	Total score
ELO	4	2	0	3	1	5	15
COSO	5	3	3	3	3	1	18
PLL	4	1	2	2	2	4	15
DC	3	2	4	1	3	4	17
TERO	4	2	2	1	2	1	12
STR	3	1	5	3	5	2	19

Obtained results can be better analyzed using the circular area charts of individual TRNGs presented in Fig. 6.13.

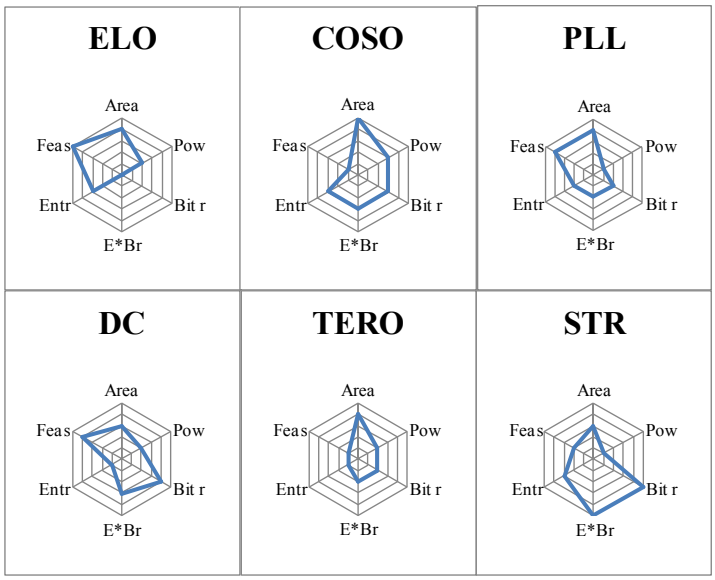

Figure 6.13: Circular area charts of scores of evaluated TRNGs in Spartan 6 FPGA

The presented designs were implemented in Spartan 6 family. We can expect that implementation in other FPGA families will give similar results. Nevertheless, we cannot exclude that some designs, which will be easy to implement in one FPGA family will be more difficult or even impossible to implement in another family. Therefore, we plan to implement selected designs also in Altera Cyclone V and Microsemi Smart Fusion 2 families in the near future.

6.11 Discussion on FPGA Implementation of Selected PUF Cores

Similarly to the TRNG evaluation, we will evaluate and briefly discuss the first results in implementation of selected PUF cores in the Xilinx Spartan 6 family. At this stage of the project, we compare the next parameters:

- *Area* – the final area will be given in logic elements (i.e. logic cells) occupied by the design (in the context of results presented in previous sections, the area will be proportional to the sum of LUTs, registers and combined LUTs and registers, since each of these objects occupies one logic element).
- *Power consumption* in mW – this parameter will give the power consumption only of the core of the PUF.
- *Number of bits per challenge* – gives number of bits obtained at the output of the PUF function as response to the challenge.
- *Power consumption per output bit* – this parameter characterizes the power needed per generation of one output bit.
- *Feasibility and repeatability* – this evaluation parameter will reflect the difficulty of the design and its repeatability across selected FPGA family. Like in TRNG implementations, the parameter value is divided into six levels.

In the next step, we propose to attribute a score to all above mentioned PUF parameters. The proposed score scale is between 0 (the lowest score) to 5 (the highest score) according to Table 6.11.

Table 6.11: Scoring intervals for PUF parameters

Parameter	**5**	**4**	**3**	**2**	**1**	**0**
Area [Logic cells]	< 20	20 – 99	100 – 199	200 – 499	500 – 999	> 999
Power [mW]	< 0.01	0.01 – 0.09	0.1 – 0.9	1.0 – 9.9	10 – 99.9	$> 99{,}9$
Nbits/ch	> 4	4	3	2	1	0
Power/bit	< 0.001	0.001 – 0.05	0.05 – 0.9	0.9 – 9.9	10 – 99	> 99
Feasib. & repeatability	Automatic all families	Man. setup per family	Man. setup iterative	Man. setup complex	Man. setup per device	Random results

Parameters like *Area*, *Power*, and *Feasibility and Repeatability* have the same scoring as in the previous section. The scoring of *Number of bits per challenge* is quite straightforward. The scoring of parameter *Power/bit* is derived from the scoring of the previous two parameters

Table 6.12: Summary of implementation results of selected PUFs

PUF type	Area	Power	Nbits/ch	Power/bit
	(LUT/Reg/L&R)	[mW]		[mW]
RO	214	14.2	1	14.2
RO Power Opt.	230	2.9	1	2.9
TERO	382	3.8	3	1.27

(*Power* and *Number of bits per challenge*). Table 6.12 presents a summary of implementation results of selected PUF designs. It can be observed that the RO-PUF occupies relatively small area, but consumes relatively high power. It is relatively easy to implement (highest feasibility and repeatability), but it gives only one bit per challenge.

The RO-PUF optimized for power occupies slightly bigger area, but the power consumption is considerably lower. Although the TERO-PUF consumes a little bit more power, it gives more bits per challenge. The power consumption per bit is thus lower.

Table 6.13 presents scores of implementation of selected PUF designs. As could be expected from the previous analysis, the TERO-PUF obtained the highest score, followed by RO-PUF optimized for power and RO-PUF without optimization.

Table 6.13: Scores of selected PUFs

PUF type	Area	Power	Nbits/ch	Power/bit	Feas. & Repeat.	Total score
RO	2	1	1	1	4	9
RO Power Opt.	2	2	1	2	4	11
TERO	2	2	3	3	3	13

Obtained results can be better analyzed using the circular area charts of individual PUFs presented in Fig. 6.14.

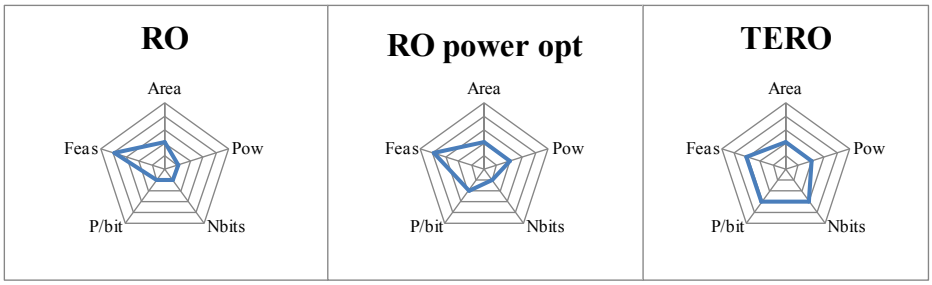

Figure 6.14: Circular area charts of scores of evaluated PUFs in Spartan 6 FPGA

Similarly to TRNGs, the presented PUF designs were implemented in Spartan 6 family. Since we cannot exclude that some designs, which will be easy to implement in one FPGA family will be more difficult or even impossible to implement in another family, we plan to implement selected PUF designs also in Altera Cyclone V and Microsemi Smart Fusion 2 families in the near future.

Chapter 7

Conclusions

According to the analysis presented in this deliverable, it is clear that some ideal TRNG or PUF design does not and will not exist. Consequently, designers will always be obliged to do some compromises according to the requirements of their targeted application.

In order to cover the largest spectrum of requirements and to avoid the case that no practical solution would be available, several TRNG and PUF principles have been pre-selected at this first stage of the HECTOR project (Task 2.1).

The TRNG and PUF evaluations and comparisons given above were first made according to results published in journal papers and conference proceedings, and also according to practical experiments realized in the past few years by some partners involved in the HECTOR project. In order to make these evaluations more objective, selected principles were implemented in the Spartan 6 FPGA family using the Evariste hardware/software tools. Once the HECTOR evaluation boards will be available, the same designs will also be implemented Cyclone V and Smart Fusion 3 families and the obtained results will be compared with those obtained for the Spartan 6 family.

According to obtained results, the list of selected TRNGs and PUF should be reduced. The final candidates should be then studied in more details, and the stochastic models, embedded tests and post-processing functions should be proposed in Task 2.2 and 2.3 and implemented in Task 2.4.

Chapter 8

List of Abbreviations

ASIC	Application specific integrated circuit
D-FF	D flip-flop
DRNG	Deterministic random number generator
EMA	Electromagnetic analysis
EMI	Electromagnetic interference
FIPS	Federal information processing standard
FPGA	Field programmable logic array
HDA	Helper data algorithm
HDRNG	Hybrid deterministic random number generator
HTRNG	Hybrid true random number generator
IID	Independent and identically distributed
IP	Intellectual property
LVDS	Low voltage differential signaling
MPV	Manufacturing process variability
NIST	National institute of standards and technology
PDF	Probability distribution function
PLL	Phase-locked loop
PTRNG	Physical true random number generator
PUF	Physical unclonable function
RNG	Random number generator
RO	Ring oscillator
STR	Self-timed ring
TERO	Transition effect ring oscillator
TRNG	True random number generator
VCO	Voltage controlled oscillator

Bibliography

[1] Hint project. http://www.hint-project.eu/.

[2] Intrinsic id. https://www.intrinsic-id.com/.

[3] F. Armknecht, R. Maes, A. Sadeghi, B. Sunar, and P. Tuyls. Memory leakage-resilient encryption based on physically unclonable functions. In *Advances in Cryptology - ASIACRYPT 2009, 15th International Conference on the Theory and Application of Cryptology and Information Security, Tokyo, Japan, December 6-10, 2009. Proceedings*, pages 685–702, 2009.

[4] E. Barker and J. Kelsey. Recommendation for the Entropy Sources Used for Random Bit Generation, NIST Special Publication 800-90B. [online] Available from http://csrc.nist.gov/publications/drafts/800-90/draft-sp800-90b.pdf, 2012.

[5] M. Baudet, D. Lubicz, J. Micolod, and A. Tassiaux. On the security of oscillator-based random number generators. *Journal of Cryptology*, 24(2):398–425, 2011.

[6] P. Bayon, L. Bossuet, A. Aubert, and V. Fischer. Electromagnetic analysis on ring oscillator-based true random number generators. In *Circuits and Systems (ISCAS), 2013 IEEE International Symposium on*, pages 1954–1957, 2013.

[7] F. Bernard, V. Fischer, and B. Valtchanov. Mathematical model of physical rngs based on coherent sampling. *Tatra Mountains Mathematical Publications*, 45(1):1–14, 2010.

[8] M. Bhargava and K. Mai. An efficient reliable PUF-based cryptographic key generator in 65nm CMOS. In *Design, Automation & Test in Europe Conference & Exhibition, DATE 2014, Dresden, Germany, March 24-28, 2014*, pages 1–6, 2014.

[9] N. Bochard, F. Bernard, V. Fischer, and B. Valtchanov. True-randomness and pseudo-randomness in ring oscillator-based true random number generators. *International Journal of Reconfigurable Computing*, 879281(2010):1–13, 2010.

[10] C. Bösch, J. Guajardo, A. Sadeghi, J. Shokrollahi, and P. Tuyls. Efficient helper data key extractor on FPGAs. In *Cryptographic Hardware and Embedded Systems - CHES 2008, 10th International Workshop, Washington, D.C., USA, August 10-13, 2008. Proceedings*, pages 181–197, 2008.

[11] L. Bossuet, Xuan Thuy Ngo, Z. Cherif, and V. Fischer. A PUF Based on a Transient Effect Ring Oscillator and Insensitive to Locking Phenomenon. *Emerging Topics in Computing, IEEE Transactions on*, 2(1):30–36, March 2014.

[12] R.G. Brown. Dieharder: A Random Number Test Suite. [online] Available from http://www.phy.duke.edu/ rgb/General/dieharder.php, 2015.

[13] A. Cherkaoui. *Générateurs de nombres véritablement aléatoires à base d'anneaux asynchrones : conception, caractérisation et sécurisation.* PhD thesis, 2014. Université Jean Monnet de Saint Etienne.

[14] A. Cherkaoui, L. Bossuet, and C. Marchand. Design, Evaluation and Optimization of Physical Unclonable Functions based on Transient Effect Ring Oscillators. Cryptology ePrint Archive, Report 2015/623. [online] Available from http://eprint.iacr.org/, 2015.

[15] A. Cherkaoui, V. Fischer, A. Aubert, and L. Fesquet. A self-timed ring based true random number generator. In *IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC 2013)*, pages 99–106, 2013.

[16] A. Cherkaoui, V. Fischer, L. Fesquet, and A. Aubert. A very high speed true random number generator with entropy assessment. In G. Bertoni and J.-S. Coron, editors, *Cryptographic Hardware and Embedded Systems (CHES 2013)*, volume 8086 of *LNCS*, pages 179–196. Springer, 2013.

[17] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, 2008.

[18] Y. Dodis, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. In *Advances in Cryptology - EUROCRYPT 2004, International Conference on the Theory and Applications of Cryptographic Techniques*, pages 523–540, 2004.

[19] J.C. Ebergen, S. Fairbanks, and I.E. Sutherland. Predicting performance of micropipelines using charlie diagrams. In *IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC 1998)*, pages 238–246, 1998.

[20] O. Elissati, E. Yahya, S. Rieubon, and L. Fesquet. A novel high-speed multi-phase oscillator using self-timed rings. In *International Conference on Microelectronics (ICM 2010)*, pages 204–207, 2010.

[21] A. Fahim. *Clock generators for SoC processors.* Cluwer Academic Publisher, MA, USA, 2004.

[22] S. Fairbanks. High Precision Timing using Self-timed Circuits, university of cambridge, computer laboratory. Technical report No. UCAM-CL-TR-738, 2009.

[23] R.C. Fairfield, R.L. Mortenson, and K.B. Coulthart. An LSI random number generator (RNG). In GeorgeRobert Blakley and David Chaum, editors, *Advances in Cryptology*, volume 196 of *LNCS*, pages 203–230. Springer, 1985.

[24] L. Feiten, A. Spilla, M. Sauer, T. Schubert, and B. Becker. Implementation and analysis of ring oscillator pufs on 60 nm altera cyclone fpgas. *Inf. Sec. J.: A Global Perspective*, 22(5-6):265–273, November 2013.

[25] NIST FIPS. 140-1: Security Requirements for Cryptographic Modules. [online] Available from http://csrc.nist.gov/publications/fips/fips140-1/fips1401.pdf, 1994.

[26] V. Fischer. A closer look at security in random number generators design. In *Constructive Side-Channel Analysis and Secure Design (COSADE 2012)*, pages 167–182. Springer, 2012.

[27] V. Fischer, F. Bernard, N. Bochard, and M. Varchola. Enhancing security of ring oscillator-based rng implemented in FPGA. In *International Conference on Field Programmable Logic and Applications (FPL 2008)*, pages 245–250, 2008.

[28] V. Fischer and M. Drutarovsky. True random number generator embedded in reconfigurable hardware. In *Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems (CHES 2002)*, volume 2523 of *LNCS*, pages 415–430. Redwood Shores, CA, USA, Springer Verlag, 2002.

[29] V. Fischer and D. Lubicz. Embedded evaluation of randomness in oscillator based elementary trng. In Lejla Batina and Matthew Robshaw, editors, *Cryptographic Hardware and Embedded Systems (CHES 2014)*, volume 8731 of *LNCS*, pages 527–543. Springer, 2014.

[30] P. Haddad, V. Fischer, F. Berdnard, and J. Nicolai. A Physical Approach for Stochastic Modeling of TERO-based TRNG. In *Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems (CHES 2015), Saint-Malo, France*, volume 9293 of *LNCS*, pages 357–372. Springer Verlag, 2015.

[31] J. Hamon, L. Fesquet, B. Miscopein, and M. Renaudin. High-level time-accurate model for the design of self-timed ring oscillators. In *IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC 2008)*, pages 29–38, 2008.

[32] C. Helfmeier, C. Boit, D. Nedospasov, and J. Seifert. Cloning physically unclonable functions. In *International Symposium on Hardware-Oriented Security and Trust (HOST 2013)*, pages 1–6, 2013.

[33] M. Hiller and G. Sigl. Increasing the efficiency of syndrome coding for PUFs with helper data compression. In *Design, Automation & Test in Europe Conference & Exhibition, DATE 2014, Dresden, Germany, March 24-28, 2014*, pages 1–6, 2014.

[34] M. Hiller, M. Weiner, L.R. Lima, B. Birkner, and G. Sigl. Breaking through fixed PUF block limitations with differential sequence coding and convolutional codes. In *TrustED'13, Proceedings of the 2013 ACM Workshop on Trustworthy Embedded Devices, Co-located with CCS 2013, November 4, 2013, Berlin, Germany*, pages 43–54, 2013.

[35] M. Hofer and C. Böhm. An alternative to error correction for sram-like PUFs. In *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings*, pages 335–350, 2010.

[36] D.E. Holcomb, W.P. Burleson, and K. Fu. Power-Up SRAM State as an Identifying Fingerprint and Source of True Random Numbers. *IEEE Transactions on Computers*, 58(9):1198–1210, 2009.

[37] Y. Hori, T. Yoshida, T. Katashita, and A. Satoh. Quantitative and Statistical Performance Evaluation of Arbiter Physical Unclonable Functions on FPGAs. In *Proceedings of the International Conference on Reconfigurable Computing and FPGAs (ReConFig 2010), Cancun, Mexico*, pages 298–303, 2010.

[38] S. Katzenbeisser, Ü. Koçabas, V. Rozic, A. Sadeghi, I. Verbauwhede, and C. Wachsmann. Pufs: Myth, fact or busted? A security evaluation of physically unclonable functions (pufs) cast in silicon. In *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*, pages 283–301, 2012.

[39] W. Killmann and W. Schindler. A proposal for: Functionality classes for random number generators, Version 2.0. BSI, Germany. [online] Available from https://www.bsi.bund.de, 2011.

[40] P. Kohlbrenner and K. Gaj. An embedded true random number generator for FPGAs. In *Proceedings of the ACM/SIGDA 12th International Symposium on Field Programmable Gate Arrays, FPGA 2004, Monterey, California, USA, February 22-24, 2004*, pages 71–78, 2004.

[41] S.S. Kumar, J. Guajardo, R. Maes, G.-J. Schrijen, and P. Tuyls. The butterfly PUF protecting IP on every FPGA. In *Proceedings of the IEEE International Workshop on Hardware-Oriented Security and Trust (HOST 2008).*, pages 67–70, June 2008.

[42] D. Lim, J.W. Lee, B. Gassend, G.E. Suh, M. van Dijk, and S. Devadas. Extracting secret keys from integrated circuits. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 13(10):1200–1205, Oct 2005.

[43] R. Maes. An accurate probabilistic reliability model for silicon pufs. In *Cryptographic Hardware and Embedded Systems - CHES 2013 - 15th International Workshop, Santa Barbara, CA, USA, August 20-23, 2013. Proceedings*, pages 73–89, 2013.

[44] R. Maes, V. Rozic, I. Verbauwhede, P. Koeberl, E. van der Sluis, and V. van der Leest. Experimental evaluation of Physically Unclonable Functions in 65 nm CMOS. In *Proceedings of the European Solid-State Circuits Conference (ESSCIRC 2012)*, pages 486–489, Sept 2012.

[45] R. Maes, P. Tuyls, and I. Verbauwhede. Low-overhead implementation of a soft decision helper data algorithm for SRAM PUFs. In *Cryptographic Hardware and Embedded Systems - CHES 2009, 11th International Workshop, Lausanne, Switzerland, September 6-9, 2009, Proceedings*, pages 332–347, 2009.

[46] R. Maes, P. Tuyls, and I. Verbauwhede. A soft decision helper data algorithm for SRAM PUFs. In *IEEE International Symposium on Information Theory, ISIT 2009, June 28 - July 3, 2009, Seoul, Korea, Proceedings*, pages 2101–2105, 2009.

[47] R. Maes, A. Van Herrewege, and I. Verbauwhede. PUFKY: A fully functional PUF-based cryptographic key generator. In *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*, pages 302–319, 2012.

[48] A. Maiti, J. Casarona, L. McHale, and P. Schaumont. A large scale characterization of RO-PUF. In *Proceedings of the IEEE International Symposium on Hardware-Oriented Security and Trust (HOST 2010)*, pages 94–99, June 2010.

[49] G. Marsaglia. DIEHARD: Battery of Tests of Randomness. [online] Available from http://stat.fsu.edu/pub/diehard/, 1996.

[50] D. Merli, J. Heyszl, B. Heinz, D. Schuster, F. Stumpf, and G. Sigl. Localized electromagnetic analysis of ro pufs. In *Hardware-Oriented Security and Trust (HOST), 2013 IEEE International Symposium on*, pages 19–24, 2013.

[51] D. Merli, D. Schuster, F. Stumpf, and G. Sigl. Semi-invasive em attack on fpga ro pufs and countermeasures. In *Proceedings of the Workshop on Embedded Systems Security*, WESS '11, pages 2:1–2:9, New York, NY, USA, 2011. ACM.

[52] Y. Oren, A. Sadeghi, and C. Wachsmann. On the effectiveness of the remanence decay side-channel to clone memory-based PUFs. In *Cryptographic Hardware and Embedded Systems - CHES 2013 - 15th International Workshop, Santa Barbara, CA, USA, August 20-23, 2013. Proceedings*, pages 107–125, 2013.

[53] E. Öztürk, G. Hammouri, and B. Sunar. Towards robust low cost authentication for pervasive devices. In *PerCom*, pages 170–178, 2008.

[54] R.S. Pappu. *Physical One-Way Functions*. PhD thesis, Cambridge, MA, USA, 2001.

[55] Z.S. Paral and S. Devadas. Reliable and efficient PUF-based key generation using pattern matching. In *HOST*, pages 128–133, 2011.

[56] A. Rényi. On measures of entropy and information. In *Proceedings of the Berkeley Symposium on Mathematical Statistics and Probability*, pages 547–561, 1960.

[57] L.M. Reyneri, D. Del Corso, and B. Sacco. Oscillatory metastability in homogeneous and inhomogeneous flip-flops. *Solid-State Circuits, IEEE Journal of*, 25(1):254–264, 1990.

[58] V. Rozic, B. Yang, W. Dehaene, and I. Verbauwhede. Highly efficient entropy extraction for true random number generators on FPGAs. In *Proceedings of the 52nd Annual Design Automation Conference, San Francisco, USA*, pages 116:1–116:6. ACM, 2015.

[59] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, and S. Vo. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications, NIST Special Publication 800-22, Revision 1a. [online] Available from http://csrc.nist.gov/publications/nistpubs/800-22-rev1a/SP800-22rev1a.pdf, 2010.

[60] I. Schaumüller-Bichl, A. Kolberger, M. Deutschmann, C. Heuberger, W. Müller, and B. Hackl. D1.1: SWOT Analysis of Existing PUF Security Modules, September 2013. CODES Project.

[61] B. Skoric, P. Tuyls, and W. Ophey. Robust key extraction from physical uncloneable functions. In *Applied Cryptography and Network Security, Third International Conference, ACNS 2005, New York, NY, USA, June 7-10, 2005, Proceedings*, pages 407–422, 2005.

[62] G. E. Suh, C. W. O'Donnell, I. Sachdev, and S. Devadas. Design and implementation of the AEGIS single-chip secure processor using physical random functions. In *32st International Symposium on Computer Architecture (ISCA 2005), 4-8 June 2005, Madison, Wisconsin, USA*, pages 25–36, 2005.

[63] G.E. Suh and S. Devadas. Physical unclonable functions for device authentication and secret key generation. In *Proceedings of the 44th ACM/IEEE Design Automation Conference (DAC '07).*, pages 9–14, June 2007.

[64] B. Sunar, W.J. Martin, and D.R. Stinson. A provably secure true random number generator with built-in tolerance to active attacks. *IEEE Transactions on Computers*, 56(1):109–119, January 2007.

[65] I.E. Sutherland. Micropipelines. *Communications of the ACM*, 32(6):720–738, 1989.

[66] G. Taylor and G. Cox. Behind Intels New Random-Number Generator. [online] Available from http://spectrum.ieee.org/computing/hardware/behind-intels-new-randomnumber-generator, 2011.

[67] R. van den Berg. Entropy Analysis of Physically Unclonable Functions. Master's thesis, Department of Mathematics and Computer Science Eindhoven University of Technology, 2012.

[68] V. van der Leest, B. Preneel, and E. van der Sluis. Soft decision error correction for compact memory-based PUFs using a single enrollment. In *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*, pages 268–282, 2012.

[69] A. Van Herrewege, S. Katzenbeisser, R. Maes, R. Peeters, A. Sadeghi, I. Verbauwhede, and C. Wachsmann. Reverse fuzzy extractors: Enabling lightweight mutual authentication for PUF-enabled RFIDs. In *16th International Conference on Financial Cryptography and Data Security (FC 2012)*, pages 374–389, 2012.

[70] A. Van Herrewege, A. Schaller, S. Katzenbeisser, and I. Verbauwhede. Inherent pufs and secure prngs on commercial off-the-shelf microcontrollers. In *2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4-8, 2013*, pages 1333–1336, 2013.

[71] M. Varchola and M. Drutarovský. New high entropy element for FPGA based true random number generators. In *Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems (CHES 2010), Santa Barbara, USA. Proceedings*, volume 2162 of *LNCS*, pages 351–365. Springer Verlag, 2010.

[72] A. Wild and T. Güneysu. Enabling sram-pufs on xilinx fpgas. In *24th International Conference on Field Programmable Logic and Applications, FPL 2014, Munich, Germany, 2-4 September, 2014*, pages 1–4, 2014.

[73] A. Winstanley and M. Greenstreet. Temporal properties of self-timed rings. In Tiziana Margaria and Tom Melham, editors, *Correct Hardware Design and Verification Methods*, volume 2144 of *LNCS*, pages 140–154. Springer, 2001.

[74] M.M. Yu and S. Devadas. Secure and robust error correction for physical unclonable functions. *IEEE Design & Test of Computers*, 27(1):48–65, 2010.

[75] M.M. Yu, D. M'Raïhi, R. Sowell, and S. Devadas. Lightweight and secure PUF key storage using limits of machine learning. In *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, pages 358–373, 2011.

[76] M.M. Yu, R. Sowell, A. Singh, D. M'Raïhi, and S. Devadas. Performance metrics and empirical results of a PUF cryptographic key generation ASIC. In *2012 IEEE International Symposium on Hardware-Oriented Security and Trust, HOST 2012, San Francisco, CA, USA, June 3-4, 2012*, pages 108–115, 2012.