Evaluation of AIS-20/31 compliant TRNG cores implemented on FPGAs

Oto Petura, Ugo Mureddu, Nathalie Bochard, Viktor Fischer, Lilian Bossuet

Evaluation of AIS-20/31 compliant TRNG cores implemented on FPGAs

Oto Petura, Ugo Mureddu, Nathalie Bochard, Viktor Fischer, Lilian Bossuet
Hubert Curien Laboratory, UMR 5516 CNRS,
Jean Monnet University Saint-Etienne
18, rue Pr. Lauras, 42000 Saint-Etienne, France
email: (oto.petura, ugo.mureddu, nathalie.bochard, fischer, lilian.bossuet)@univ-st-etienne.fr

***Abstract*—FPGAs are widely used to integrate cryptographic primitives, algorithms, and protocols in cryptographic systems-on-chip (CrySoC). As a building block of CrySoCs, True Random Number Generators (TRNGs) exploit analog noise sources in electronic devices to generate confidential keys, initialization vectors, challenges, nonces, and random masks in cryptographic protocols. TRNGs aimed at cryptographic applications must fulfill the security requirements defined in the German Federal Bureau for Information Security's (BSI) recommendations AIS-20/31, which has become a *de facto* standard in Europe. Many TRNG cores have already been published, only a few of which are suitable for FPGAs and even fewer comply with AIS-20/31. Here we present the results of the implementation of AIS-20/31 compliant TRNG cores in three FPGA families: Xilinx Spartan 6, Altera Cyclone V and Microsemi SmartFusion 2. In addition to common design parameters like area, bit rate and power/energy consumption, we compare and discuss the feasibility of generator cores in different FPGAs and the statistical quality of their output. These results will help designers select the best generator and the device family to match the requirements of the data security application. To ensure reproducibility of the results, the open source VHDL code of all generators adapted to individual families can be downloaded from the dedicated web page.**

I. Introduction

True Random Number Generators (TRNG) are used in cryptography to generate confidential keys, initialization vectors, challenges, nonces, and random masks in side channel attack countermeasures. They exploit intrinsic noise sources in electronic devices as a source of randomness.

FPGAs are widely used to integrate cryptographic primitives, algorithms and protocols in the cryptographic systems on chip (CrySoC). As a building block of the CrySoC, the TRNG must meet strict security requirements [1].

TRNGs are typically composed of an analog physical source of randomness, a digitizer, and an optional entropy conditioning block. The source of randomness, digitization, and the entropy harvesting mechanism depend to a large extent on the selected technology, a standard or even a recommended TRNG does not exist. Depending on the characteristics of the source of randomness and the quality of the digital noise, designers select the entropy conditioning method that will enhance the statistical properties of generated numbers.

In the past, during the design and the security evaluation and certification process, the principle of the TRNG and its implementation were only evaluated statistically: the generated numbers were tested using standard test suites.

However, this approach is not suitable for modern data security systems for several reasons: 1) post-processing can mask considerable weaknesses in the source of randomness; 2) generic statistical tests can only evaluate the statistical quality of the numbers that are generated and not their entropy; 3) high-end standard statistical tests are complex and hence both expensive and slow, plus they require huge data sets. Consequently, they are only executed occasionally or on demand and only on selected sets of data of limited size.

The German Federal Office for Information Security recently proposed a methodology of evaluation of random number generators (AIS-20/31) [2]. Currently, all TRNGs aimed at cryptographic applications that require a security certificate for use in European union must comply with AIS-20/31.

Many TRNG cores have already been published, but only a few of them are suitable for FPGAs, and even fewer comply with AIS-20/31. Our aim was to select such generators and to fairly evaluate the difficulties related to their implementation in different FPGA technologies, their area, output bit rate, power requirements, and the statistical quality of their output.

To compare TRNG principles and their implementations in different FPGA families as fairly as possible, the evaluation boards should have the same topology and should use as few components introducing deterministic noise as possible (e.g. should be powered using low noise power supplies).

The paper is organized as follows. In Section II, we describe how we selected AIS-20/31 compliant TRNG cores that are suitable for implementation in FPGAs. In Section III, we describe the strategy of implementation and evaluation of the TRNG cores. In Section IV, we describe the implementation of selected generators in the selected FPGA families. In Section V, we discuss the results and propose selection criteria to help designers select an appropriate design in the future. Section VI concludes the paper and describes the future outlook.

II. Selection of TRNG Cores

Our objective was to select the TRNG principles that are feasible in all recent FPGA families, so the design must be purely digital. Furthermore the selected principles must comply with the AIS-20/31, which requires a stochastic model.

We pre-selected TRNG cores that use oscillating circuitries: single-event ring oscillators (i.e. standard ring oscillators) [3], [4], [5], multi-event ring oscillators with signal collisions

(i.e. transition effect ring oscillators) [6], multi-event ring oscillators without signal collisions (i.e. self-timed rings) [7], and phase-locked loops (PLLs) [8]. Consequently, all of them should be feasible in recent and future families of FPGAs. They all use simple and comprehensible sources of randomness, their raw random signal is available outside the generator, and the stochastic model of the generator exists or is feasible. Therefore, we can conclude that all of them comply with the AIS-20/31 requirements.

III. Strategy for the Implementation and Evaluation of TRNG Cores

Our objective was to use the same hardware configuration for all TRNG cores and for different FPGA families. The hardware/software system we used had three components: an FPGA device with the target of evaluation (TOE), the acquisition board and the PC running the software. The TOE implemented in FPGA devices was connected to the acquisition card using a simple serial interface – a serial data stream and a data strobe signal was sent to the acquisition board via two low voltage differential signaling (LVDS) links. The generated bit streams were saved in a 4-MB SRAM memory of the acquisition card and sent to the PC using the USB bus.

To reduce the vulnerability of the generators to external manipulations, we did not use external clocks: all the clock signals were generated inside the TOE, for example, using a ring oscillator with appropriate topology.

We preselected three representative FPGA families: Xilinx Spartan 6, Altera Cyclone V, and Microsemi SmartFusion 2.

Since expressing logic area in slices or adaptive logic modules (ALMs), as made often by FPGA vendors, would not allow the fair comparison of designs, we characterize the area of generators using the number of occupied look-up tables (LUTs) and registers.

One of the parameters used for design evaluation is power consumption. The power consumption of TRNGs is relatively low and is mostly comparable to, or even lower 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 only an output multiplexer was implemented inside it (the same multiplexer was used later to keep the generator running, while blocking its output to the input/output circuitry). With this small reference project the Spartan 6, Cyclone V, and SmartFusion 2 devices consumed 3.5 mW, 29.7 mW, and 12.5 mW, respectively. This power was subtracted from the total power consumption measured in all the experiments. The results presented in the following sections are thus the net power consumption of the designs we tested.

To evaluate the statistical quality of the generated numbers, we used Procedure B of the AIS-20/31, which is designed to test raw random signals. We used test T8 of this procedure for a rough estimation of the entropy rate. For a rigorous entropy estimation a stochastic model of the generator should be used. This is out of the scope of our paper.

Since all the presented TRNG designs have many degrees of freedom, in our comparisons, we chose the parameters (number of delay elements, division factors, etc.) giving the highest entropy rate at the output.

IV. Implementation of Selected TRNG Cores in FPGA

A. Ring Oscillator Based Elementary TRNG

The ring oscillator based elementary TRNG (ERO-TRNG) was prosed and modeled in [3]. The block diagram of the ERO-TRNG as implemented in FPGAs is depicted in Fig. 1.

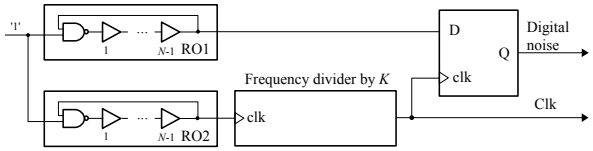

Fig. 1. Architecture of the ERO-TRNG core

The generator uses two identical ring oscillators (RO1 and RO2) as sources of randomness. The output of one ring (a jittery clock signal) is sampled in a D flip-flop after a sufficiently long accumulation period derived from the second clock signal using a frequency divider by K (a 17-bit synchronous counter). Let us note that thanks to the use of two identical oscillators, the impact of the global sources of randomness, which can be easily manipulated, is significantly reduced.

The number of elements of the ring (N) was chosen to get approximately the same clock frequency (300 MHz) in all families: 3 elements (one NAND gate and 2 buffers) were used in Spartan 6 and 5 elements in Cyclone V and SmartFusion 2 family.

The lower entropy bound defined in [3] can be adapted to the elementary TRNG from Fig. 1 as:

$$H_{min} = 1 - \frac{4}{\pi^2 ln(2)} e^{\frac{-\pi^2 \sigma_{th}^2 K T_2}{T_1^3}}, \qquad (1)$$

where σ_{th}^2 is the variance of the jitter due to thermal noise, K is the frequency division factor and T_1, T_2 are periods of the clock signals generated by RO1 and RO2, respectively. Let us note that the oscillation frequency and the size of the jitter differ in each FPGA family, consequently parameter K and hence the output bit rate also differ.

Clock periods of the ring oscillators were about 3 ns in all the devices. The total period jitter exploited in the TRNGs was then approximately 4 ps, 3 ps, and 8 ps for the Spartan 6, Cyclone V, and SmartFusion 2 device, respectively. The frequency division factor K was set up according to Eq. (1) to 80 000, 135 000, and 20 000, respectively.

The two ring oscillators were placed and routed manually in order to ensure the repeatability of the design. Although both rings were placed in close vicinity, apparently, they did not lock.

B. Ring Oscillator Coherent Sampling Based TRNG

The coherent sampling ring oscillator based TRNG (COSO-TRNG) was first proposed in [4]. The block diagram of the

COSO-TRNG core implemented in our devices is depicted in Fig. 2.

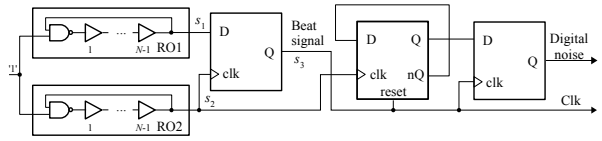

Fig. 2. Architecture of the COSO-TRNG core

The generator uses two oscillators, which have the same number of elements and their topology (placement of the delay elements) is also the same. 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). To extract randomness from the jitter, the difference between two clock periods must fulfill the following condition:

$$\Delta_T < \sqrt[3]{\sigma_T^2 \cdot T} = \Delta_{T_{max}}. \tag{2}$$

Fulfilling this condition is not an easy task. We measured the clock period T and the standard deviation of the period jitter to compute $\Delta_{T_{max}}$. Then we tried different placements and routings to find Δ_T smaller than $\Delta_{T_{max}}$. In our case, we obtained satisfying results with:

- N = 8 in Spartan 6 giving $T = 6.92$ ns, $\sigma \sim 4$ ps, $\Delta_{T_{max}} \sim 50$ ps
- N = 6 in Cyclone V giving $T = 3.17$ ns, $\sigma \sim 2.5$ ps, $\Delta_{T_{max}} \sim 30$ ps
- N = 10 in SmartFusion 2 giving T=5.4 ns, $\sigma \sim 8$ ps, $\Delta_{T_{max}} \sim 70$ ps

C. Multi-Ring Oscillator Based TRNG

The multi-ring oscillator based TRNG (MURO-TRNG) and its stochastic model were originally proposed in [5]. The block diagram of the MURO-TRNG core architecture implemented in our devices is depicted in Fig. 3.

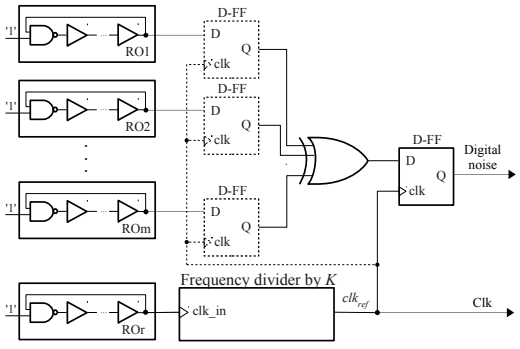

Fig. 3. Architecture of the MURO-TRNG

The generator uses m ring oscillators as sources of randomness. Assuming that the oscillators are independent, their phase is uniformly distributed. Based on the assumption of uniformity, the number of rings must fulfill the following condition:

$$m > \frac{T}{\sigma_{acc}}, \tag{3}$$

where T is the mean value of the clock period and σ_{acc} is the standard deviation of the jitter accumulated during the sampling period. Since the clock phases are uniformly distributed, the probability that the D flip-flop at the output of the generator will sample some clock edges (out of m edges theoretically available at the XOR gate output) is also uniformly distributed.

However, the authors of [9] showed that output of a single m-input XOR gate cannot follow too many high-speed input signals. They proposed using additional flip-flops (dashed lines in Fig. 3), which resolved the problems concerning the speed of the XOR gate. Since the authors of [10] proved that the stochastic model remains valid, we used this modified MURO-TRNG architecture in our study.

D. Coherent Sampling Based TRNG Using PLLs

The coherent sampling based TRNG which uses PLLs (PLL-TRNG), was first published in [8] and the model of the generator was proposed in [11]. The block diagram of the PLL-TRNG core implemented in our devices is depicted in Fig. 4.

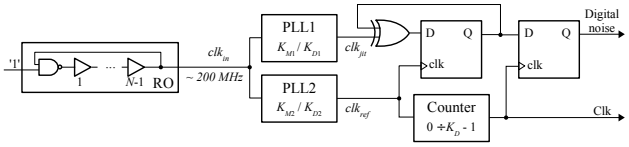

Fig. 4. Architecture of the PLL-TRNG core

The generator is based on the fact that using PLLs, the frequencies of two generated clock are mutually related. Since $f_{jit} = f_{in} \cdot K_{M1}/K_{D1}$ and $f_{ref} = f_{in} \cdot K_{M2}/K_{D2}$, the relationship between f_{jit} and f_{ref} is as follows:

$$f_{jit} = f_{ref} \cdot \frac{K_{M1}}{K_{D1}} \cdot \frac{K_{D2}}{K_{M2}} = f_{ref} \cdot \frac{K_M}{K_D}. \tag{4}$$

Thanks to the coherent sampling principle, the samples of the jittery clock signal obtained in the D flip-flop at the rising edges of the reference clock signal are uniformly distributed over the translated period T_{jit}. The distance between the samples is then $\Delta = T_{jit}/K_D$ and K_D samples must be XOR-ed to obtain one output bit (see Fig. 4).

The output bitrate R of the generator and the sensitivity to the jitter S are defined as:

$$R = f_{ref}/K_D, \tag{5}$$

$$S = \Delta^{-1} = K_D/T_{jit}. \tag{6}$$

To obtain high entropy random bits, the distance between the samples Δ must fulfill the following condition:

$$\Delta \ll \sigma_r, \tag{7}$$

where σ_r is the relative jitter between the two generated clocks. According to Eq. (5) and (6), the objective of a designer is to make Δ as small as possible, while maintaining the output bit rate (R) in an acceptable range by setting the input frequency (f_{in}) and the multiplication and division factors of both PLLs.

We followed the same strategy to determine the multiplication and division factors for given families. The frequency of the input clock signal generated by a ring oscillator was approximately 200 MHz in all cases. The parameters of PLLs and the distance between samples (Δ) are presented in Table I.

TABLE I
PARAMETERS OF PLLS AND CORRESPONDING DISTANCE BETWEEN SAMPLES (Δ) IN SELECTED FPGA FAMILIES

FPGA	PLL1		PLL2		Total		Δ
	K_M	K_D	K_M	K_D	K_M	K_D	[ps]
Spartan 6	37	17	17	7	1 377	259	4.82
Cyclone V	31	29	23	18	667	558	4.25
SmartFusion2	74	162	18	22	729	407	9.10

E. Transition Effect Ring Oscilaltor Based TRNG

The transition effect ring oscillator based TRNG (TERO-TRNG) was proposed in [6] and its stochastic model in [12]. The block diagram of the TERO-TRNG core implemented in our FPGA devices is depicted in Fig. 5.

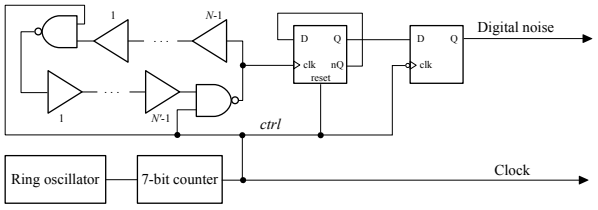

Fig. 5. Architecture of the TERO-TRNG core

The transition effect ring oscillator (TERO) is a multi-event ring oscillator with collisions built as a loop of logic gates. The loop contains an even number of inverting gates and any number of non-inverting gates. Because of the even number of inverting gates, the oscillator has to be restarted regularly – the two events created after each restart circulate inside the loop until a collision occurs, during which the edge that moves faster reaches the slower one.

The difference in the speed of circulating events is caused by differences in delays between inverters in loop branches and by analog phenomena in inverters and buffers. The circulating events create temporary oscillations that disappear after the collision.

The heart of the TERO-TRNG is the TERO cell, which is followed by a counter (in Fig. 5, the counter is represented by a T flip-flop) and an output data register.

The output of the counter represents realizations of the random variable (i.e. the number of oscillations in subsequent control periods). The control signal, which periodically restarts the TERO cell, is generated using a conventional ring oscillator. The control signal defines the output bit rate of the generator.

F. Self Timed Ring Based TRNG

The self timed ring (STR) is a multi-event oscillator without signal collisions. The first TRNG using STRs was proposed in [7] and its model in [13]. The block diagram of the STR-TRNG core implemented in our devices is depicted in Fig. 6.

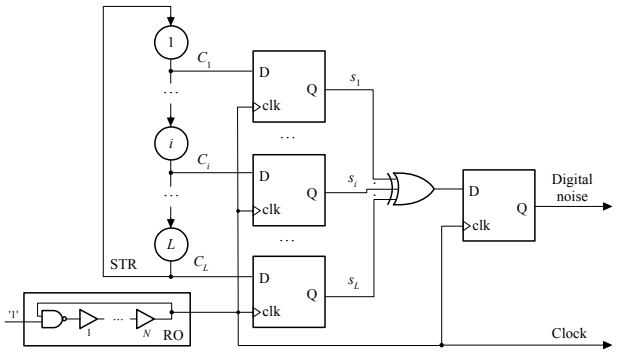

Fig. 6. Architecture of the STR-TRNG core

The STR is composed of L stages, each consisting of a Muller gate and an inverter. The STR stages communicate using the two-phase handshake protocol. In contrast to inverter ring oscillators, several events can propagate without colliding thanks to this handshake protocol, which enables precise, built-in phase control of the internal clock signals.

The ring is initialized with E 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.They either: form a cluster which propagates in the ring (burst oscillation mode), or 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 $E/(L-E)$ with respect to the ratio of forward and reverse static delays).

If E events are confined in L stages and spread evenly around the ring, the phase shift between two stages separated by n stages is [7]:

$$\varphi_n = n \times \frac{E}{L} \times 180. \tag{8}$$

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 [13]:

$$\triangle\varphi = \frac{E}{2L}. \tag{9}$$

The oscillation period of an STR does not depend on the number of its stages, but on the number of propagating events. It is therefore possible to increase the number of ring stages (L) while keeping the frequency constant. Consequently, the phase resolution of an STR can theoretically be set as finely as needed.

In our STR-TRNG, the sampling clock was generated by an additional ring oscillator. Each STR stage was sampled in the D flip-flop. The outputs of the flip-flops were XOR-ed together and the output of the XOR gate was sampled again in another D flip-flop at the same frequency. To guarantee

a sufficient entropy rate at the output of the generator, the following condition must be fulfilled:

$$\triangle\varphi < \sigma_{acc}, \tag{10}$$

where $\triangle\varphi$ is the phase resolution defined in Eq. (9) and σ_{acc} is the standard deviation of the accumulated jitter.

V. Discussion on FPGA Implementation of Selected TRNG Cores

In the following paragraphs, we evaluate and briefly discuss the first results obtained in three selected FPGA families.

We evaluated feasibility and repeatability as follows:

- Score 5 : no need of manual intervention. The results are independent of the device family.
- Score 4 : simple manual setup (e.g. placement). The results are independent of the device family.
- Score 3 : manual per-family optimization needed.
- Score 2 : complex manual setup. The results are repeatable within the device family.
- Score 1 : per-device manual setup.
- Score 0 : the results are not repeatable.

Table II summarizes the results of the implementation of the selected TRNG designs. First of all, it can be observed that the area occupied by individual generators does not differ significantly between the families, regardless of the size of the LUT. This can be explained by the fact that each delay element of rings is implemented using exactly one LUT.

It can be observed that the ERO-TRNG core occupies very small area and consumes relatively little power. It is very easy to implement (highest feasibility and repeatability), but it has a very small bit rate.

The COSO-TRNG core occupies the smallest area and consumes the least power. At the same time, it has an interesting bit rate. It also reaches a high entropy rate and relatively high entropy & bit rate product. However, it is difficult to implement – it must be placed manually in each individual device (low feasibility and repeatability). This disadvantage becomes eliminatory in most practical applications.

The MURO-TRNG core is relatively easy to implement and it features relatively high bit rate at the cost of the area. It has the second highest entropy & bit rate product, but the energy efficiency is not remarkable.

The area occupied by the PLL-TRNG core seems to be small, however, the area occupied by the PLLs is not taken into account in Table II. The main advantage of this generator is related to the fact that PLLs are very well isolated from the rest of the device and therefore more robust.

The TERO-TRNG core seemed to be very promising, but the need of the manual set up of the TERO cell represents an important handicap and its weakest point.

We obtained very interesting results with the STR-TRNG independently from the family. This TRNG core has extremely high bit rate and a high entropy and thus also a very high entropy & bit rate product. While it has the highest power consumption, it maintains a very high energy efficiency. Unfortunately, it occupies huge area and it needs precise placement and routing.

When considering the power and energy consumption, the energy efficiency parameter can be very useful to estimate the energy that must be spent for generating one random bit. This is clearly visible in the case of the STR-TRNG.

On the other end, the use of the entropy & bit rate product does not seem to bring any significant advantage to our evaluation. However, this fact is caused by the strategy of our design: to obtain the highest entropy possible for each TRNG design. We are convinced that if the entropy rate per bit is not close to one (which is the case in many practical applications), the entropy & bit rate product can help in finding the compromise between the entropy rate and the bit rate.

If we compare individual generators from the point of view of different parameters, we can definitely observe that a generator giving the best results in all TRNG parameters does not exist. It can be seen that the ERO-TRNG core wins in feasibility and repeatability, but loses in the bit rate. On the other hand, the COSO-TRNG core obtains perfect results in area, but very bad score in feasibility and repeatability. The MURO-TRNG core can represent a compromise between the bit rate and feasibility, but can be weak from the security point of view – the rings can lock to each other and decrease significantly the entropy. The STR-TRNG core wins in the bit rate, energy efficiency, and entropy & bit rate product and it is certainly the best candidate for the high-speed applications, where the power consumption and difficulty of design are less important.

VI. Conclusions

In this paper, we presented and discussed implementation of selected TRNG cores in three different FPGA families. We showed that all cores comply with the stringent security requirements of the AIS-20/31 standard: they are simple and comprehensible, their stochastic model exists or at least is feasible, and the raw random signal is available for testing.

The results confirm that all the preselected TRNG designs are feasible in all selected families. However, two of evaluated designs are not suitable for use in practice in their current form: the COSO-TRNG and the TERO-TRNG require some manual intervention (placement and routing) for each device individually.

The results also confirm that no ideal TRNG exists – the most suitable generator must be selected according to requirements of the data security application and some compromise must always be done.

It is important to stress that once the designer selects the appropriate generator core, he still has many degrees of freedom in the design and he can adapt the final choice of parameters to the practical needs of the application. The proposed TRNG evaluation parameters (energy efficiency and entropy & bit rate product) can be helpful in this task. Other combined metrics such as area & power consumption product can also be used. However it is more valuable for TRNG implementation in

TABLE II
SUMMARY OF IMPLEMENTATION RESULTS OF THE SELECTED TRNGS

TRNG type	FPGA device	Area (LUT/Reg)	Power cons. [mW]	Bit rate [Mbits/s]	Efficiency [bits/μWs]	Entropy per bit	Entropy * Bit rate	Feasib. & Repeat.
ERO	Spartan 6	46/19	2.16	0.0042	1.94	0.999	0.004	**5**
	Cyclone V	34/20	3.24	0.0027	0.83	0.990	0.003	
	SmartFusion 2	45/19	4	0.014	3.5	0.980	0.013	
COSO	Spartan 6	**18/3**	**1.22**	0.54	442.6	0.999	0.539	1
	Cyclone V	**13/3**	**0.9**	1.44	**1 600**	0.999	1.438	
	SmartFusion 2	**23/3**	**1.94**	0.328	169	0.999	0.327	
MURO	Spartan 6	521/131	54.72	2.57	46.9	0.999	2.567	4
	Cyclone V	525/130	34.93	2.2	62.9	0.999	2.197	
	SmartFusion 2	545/130	66.41	3.62	54.5	0.999	3.616	
PLL	Spartan 6	34/14	10.6	0.44	41.5	0.981	0.431	3
	Cyclone V	24/14	23	0.6	43.4	0.986	0.592	
	SmartFusion 2	30/15	19.7	0.37	18.7	0.921	0.340	
TERO	Spartan 6	39/12	3.312	0.625	188.7	0.999	0.624	1
	Cyclone V	46/12	9.36	1	106.8	0.987	0.985	
	SmartFusion 2	46/12	1.23	1	813	0.999	0.999	
STR	Spartan 6	346/256	65.9	**154**	**2 343.2**	0.998	**154.121**	2
	Cyclone V	352/256	49.4	**245**	**4 959.1**	0.999	**244.755**	
	SmartFusion 2	350/256	82.52	**188**	**2 286.7**	0.999	**188.522**	

application specific integrated circuits (ASICs) which we did not target.

Output parameters of all the tested generators, such as bit rate, power consumption, entropy, etc., depend on the underlying hardware to a great extent. Using the same evaluation boards in the same conditions is very important for a fair comparison.

Presented results, together with the open-source VHDL codes which the designer can download from[1], can help them to make their choice and test the designs on their own hardware.

We believe that most of our conclusions can be extended to implementation of presented TRNG cores in ASICs.

VII. ACKNOWLEDGEMENT

This work has received funding from the European Union's Horizon 2020 research and innovation programme in the framework of the project HECTOR (Hardware Enabled Crypto and Randomness) under grant agreement No 644052.

REFERENCES

[1] V. Fischer, "A closer look at security in TRNGs design," in *Proceedings of Constructive Side-Channel Analysis and Secure Design – COSADE'12*, ser. LNCS, vol. 7275. Springer-Verlag Berlin Heidelberg, 2012, pp. 167–182.

[2] W. Killmann and W. Schindler, "A proposal for: Functionality classes for random number generators, version 2.0," 2011. [Online]. Available: https://www.bsi.bund.de/EN/Home/home_node.htm

[3] M. Baudet, D. Lubicz, J. Micolod, and A. Tassiaux, "On the security of oscillator-based random number generators," *Journal of Cryptology*, vol. 24, no. 2, pp. 398–425, 2011.

[4] P. Kohlbrenner and K. Gaj, "An embedded true random number generator for FPGAs," in *Proceedings of the 2004 ACM/SIGDA 12th international symposium on Field programmable gate arrays*. ACM, 2004, pp. 71–78.

[1]https://labh-curien.univ-st-etienne.fr/cryptarchi/HECTOR_TRNG_designs

[5] B. Sunar, W. Martin, and D. Stinson, "A Provably Secure True Random Number Generator with Built-In Tolerance to Active Attacks," *IEEE Transactions on Computers*, pp. 109–119, 2007.

[6] M. Varchola and M. Drutarovsky, "New high entropy element for FPGA based true random number generators," in *Cryptographic Hardware and Embedded Systems, CHES 2010*. Springer, 2010, pp. 351–365.

[7] 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)*, 2013, pp. 99–106.

[8] 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)*, ser. LNCS, vol. 2523, Redwood Shores, CA, USA. Springer Verlag, 2002, pp. 415–430.

[9] K. Wold and C. Tan, "Analysis and enhancement of random number generator in FPGA based on oscillator rings," in *Proceedings of the International Conference on Reconfigurable Computing and FPGAs (ReConFig'08)*, 2008, pp. 385–390.

[10] 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*, vol. 879281, no. 2010, pp. 1–13, February 2010.

[11] F. Bernard, V. Fischer, and B. Valtchanov, "Mathematical model of physical RNGs based on coherent sampling," *Tatra Mountains Mathematical Publications*, vol. 45, no. 1, pp. 1–14, 2010. [Online]. Available: http://tatra.mat.savba.sk/Full/45/01be-f-v.pdf

[12] 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*, ser. LNCS, vol. 9293. Springer Verlag, 2015, pp. 357–372.

[13] A. Cherkaoui, V. Fischer, L. Fesquet, and A. Aubert, "A very high speed true random number generator with entropy assessment," in *Cryptographic Hardware and Embedded Systems (CHES 2013)*, ser. LNCS, G. Bertoni and J.-S. Coron, Eds., vol. 8086. Springer, 2013, pp. 179–196.