A Highly-Portable True Random Number Generator based on Coherent Sampling

Adriaan Peetermans, Vladimir Rožić and Ingrid Verbauwhede
imec-COSIC, KU Leuven, Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium
Email: {adriaan.peetermans, vladimir.rozic, ingrid.verbauwhede}@esat.kuleuven.be

***Abstract*—True Random Number Generators (TRNGs) are indispensable in modern cryptosystems. Unfortunately, in order to guarantee high entropy of the generated numbers, many TRNG designs require a complex implementation procedure, often involving manual placement and routing. In this work, we introduce a dynamic calibration mechanism for the Coherent Sampling Ring Oscillator based TRNG (COSO-TRNG) enabling easy integration of the entropy source into complex systems. The TRNG setup procedure automatically selects a configuration that guarantees the security requirements. In the experiments, we show that the proposed mechanism is capable of assuring correct TRNG operation even when an automatic placement is carried out and when the design is ported to another FPGA family. We generated random bits on both a *Xilinx Spartan 6* and a *Microsemi SmartFusion2* implementation that, without post processing, passed AIS-31 statistical tests at a throughput of 3.30 Mbit/s and 1.47 Mbit/s respectively.**

***Index Terms*—Security, Entropy, TRNG, AIS-31, NIST SP 800-90B, Coherent Sampling**

I. Introduction

TRUE Random Number Generators (TRNGs) are used in cryptographic applications to create random keys, parameters in challenge-response protocols, padding values and masks. Implementing TRNGs on Field-Programmable Gate Arrays (FPGAs) is challenging due to limited TRNG-specific resources and techniques available to the designer. Due to the availability of only digital resources in FPGAs, these TRNGs are usually based either on the unpredictability of metastable memory elements [1], [2] or on the timing jitter present in free-running oscillators [3], [4].

The output of a TRNG should pass statistical tests, such as NIST SP 800-22 [5], NIST SP 800-90B [6] or FIPS 140-2 [7]. However, simply passing statistical tests is not sufficient to guarantee the security of a TRNG. According to the AIS-31 [8] and NIST SP 800-90B [6] recommendations, the security of a TRNG should be justified by a stochastic model of the entropy source.

From an implementation aspect, TRNG designers need to consider the feasibility of the entropy source on an FPGA, portability across different FPGA families and vendors, design constraints, and design effort. Unlike most digital designs, TRNGs usually require some manual setup or placement and routing constraints. For example, designs based on Self-Timed Ring oscillators (STRs) [9], and delay chains [10], [11] require placement constraints that have to be set up for each FPGA family, thus limiting the portability of these designs. In addition, some entropy sources [4] don't work correctly on all locations on an FPGA. Therefore, a search procedure is required for each individual device until a suitable placement is found.

This work focuses on the COherent Sampling ring Oscillator based TRNG (COSO-TRNG) [12]. A stochastic model of this entropy source was first developed by [13] for Phase-Locked Loop (PLL) based implementations. A more general stochastic model, proposed in [14], doesn't specify the source of the random jitter, as this model is based on the period difference between two oscillators of any type.

The existing COSO-TRNG implementations can achieve a throughput of the order of 1 Mbit/s, requiring only minimal chip area. The entropy source consists of a structure that generates two oscillating signals with similar periods. In practice, a generic way of creating these two signals is by using two identically designed Ring Oscillators (ROs). Process and interconnect delay variations, however, make it very challenging to match the periods of the two ROs in an FPGA [15]. For this reason, a search procedure has to be applied until two well matched ROs are found. This high effort renders the COSO-TRNG implementation unpractical as this procedure has to be repeated for every device, even from the same FPGA family.

In this paper, we propose a highly-portable, easy-to-use architecture of a COSO entropy source without requiring any device-specific blocks such as PLLs, carry-chains or DSPs, without any placement and routing constraints, and without the need for a manual search procedure. The main contributions are the following:

- We propose a new entropy source with reconfigurable ring oscillators, which enables matching two oscillating periods with a precision of a few picoseconds. The proposed source uses only LookUp Tables (LUTs) without requiring device-specific resources, making the design portable across different families and vendors.
- We provide experimental confirmation that this matching is possible even without placement and routing constraints. The Verilog source code of this TRNG is open

This work was supported in part by the Research Council KU Leuven: C16/15/058. In addition, this work is supported in part by the Hercules Foundation AKUL/11/19, and by the European Commission through the Horizon 2020 research and innovation programme Cathedral ERC Advanced Grant 695305. Adriaan Peetermans is funded by an FWO fellowship and Vladimir Rožić is an FWO postdoctoral researcher.

TABLE I
NOTATION.

Symbol	Definition
$\mathrm{E}[\cdot]$	Expected value, defined as: $\mathrm{E}[X] = \int_{-\infty}^{\infty} x f_X(x)dx$ for a continuous random variable X with probability density function: $f_X(x)$ or $\mathrm{E}[X] = \sum_{i=1}^{\infty} x_i p_i$ for a discrete random variable X with probability masses: $Pr(X = x_i) = p_i$.
$\mathrm{Var}[\cdot]$	Variance, defined as: $\mathrm{Var}[X] = \mathrm{E}[(X - \mathrm{E}[X])^2]$.
$\mathcal{N}(\mu, \sigma^2)$	The normal probability distribution, describing a continuous random variable X with probability density function: $f_X(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{(x-\mu)^2}{2\sigma^2})$. It can be shown that: $\mathrm{E}[X] = \mu$ and $\mathrm{Var}[X] = \sigma^2$.

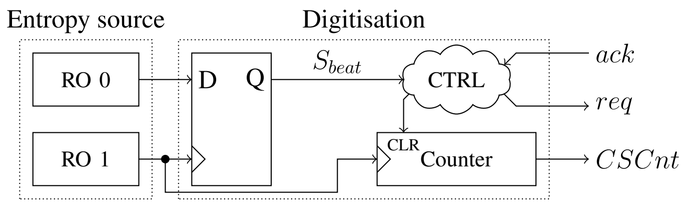

Fig. 1. Architecture of the COSO-TRNG.

source and publicly available[1].

- We design a control circuit that monitors the TRNG health, dynamically adjusts the matching conditions, and notifies the user if suitable matching could not be obtained.

II. Background

The Coherent Sampling (CS) designs use two oscillators in the entropy source. Their periods should satisfy one of the two conditions: either the ratio of the periods is equal to a known rational fraction, which is the case in PLL-based designs, or this ratio is tuned to a value very close to one, which is the case when they are generated by ROs. Here, we focus on the latter case because we want to avoid using any device-specific components such as PLLs. The notation used in this text is summed up in Table I.

A. Stochastic model

Figure 1 shows the architecture of the COSO-TRNG. A Data Flip-Flop (DFF) performs the sampling and outputs a low frequency beat signal (S_{beat}). The period of this signal is measured using a counter clocked by the sampling signal, reset every period of S_{beat}. The output of this counter ($CSCnt$) is a discrete random variable due to the independent random jitter that is present in both oscillators. According to the model in [14], the mean and the variance of this random variable are:

$$\mathrm{E}[CSCnt] = \frac{\mathrm{E}[T_{RO0}]}{\mathrm{E}[\Delta]}, \tag{1}$$

$$\mathrm{Var}[CSCnt] = \mathrm{E}[CSCnt]\frac{\mathrm{Var}[\Delta]}{\mathrm{E}[\Delta]^2}. \tag{2}$$

[1]https://github.com/KULeuven-COSIC/COSO-TRNG

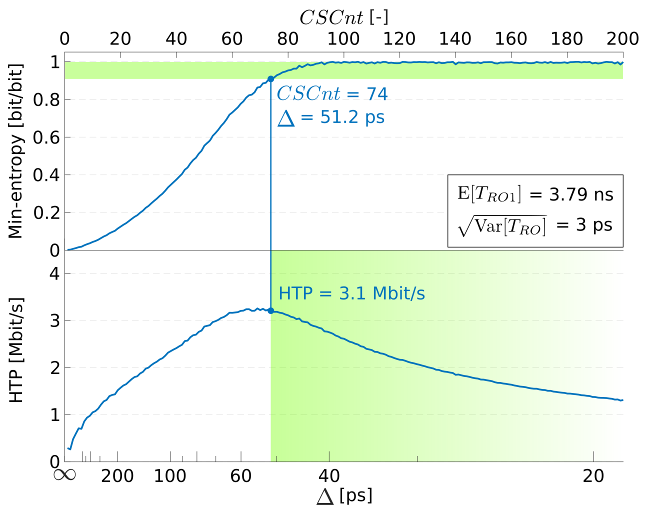

Fig. 2. Estimated min-entropy and HTP versus $\mathrm{E}[\Delta]$ and $\mathrm{E}[CSCnt]$ for a *Spartan 6* implementation.

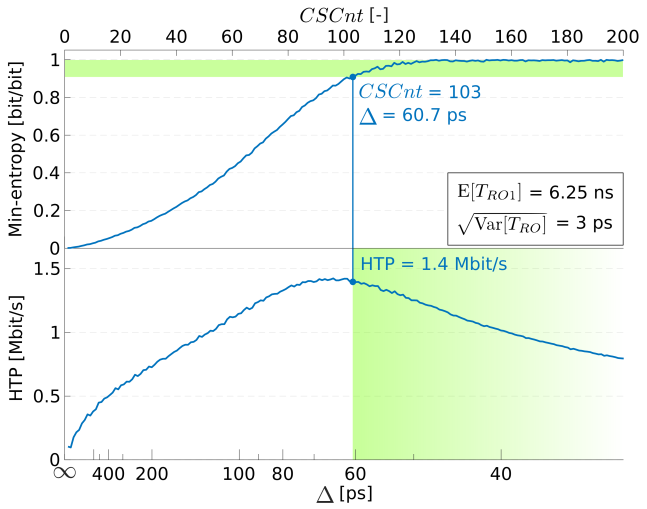

Fig. 3. Estimated min-entropy and HTP versus $\mathrm{E}[\Delta]$ and $\mathrm{E}[CSCnt]$ for a *SmartFusion2* implementation.

Where T_{RO0} and T_{RO1} denote the oscillator period lengths. Mean and variance of the period difference, Δ, are given by:

$$\mathrm{E}[\Delta] = |\mathrm{E}[T_{RO1}] - \mathrm{E}[T_{RO0}]|, \tag{3}$$

$$\mathrm{Var}[\Delta] = \mathrm{Var}[T_{RO0}] + \mathrm{Var}[T_{RO1}]. \tag{4}$$

The Least Significant Bit (LSB) can now be used from the output of the counter to generate random bits.

B. Entropy rate optimisation

To calculate the entropy per generated bit, knowledge of the exact distribution of $CSCnt$ is required. Derivation of an analytical expression for this distribution starting from the stochastic model is a difficult problem. Yang et. al. in [14] mitigate this problem by assuming that the resulting distribution is Gaussian. This assumption reduces the stochastic model

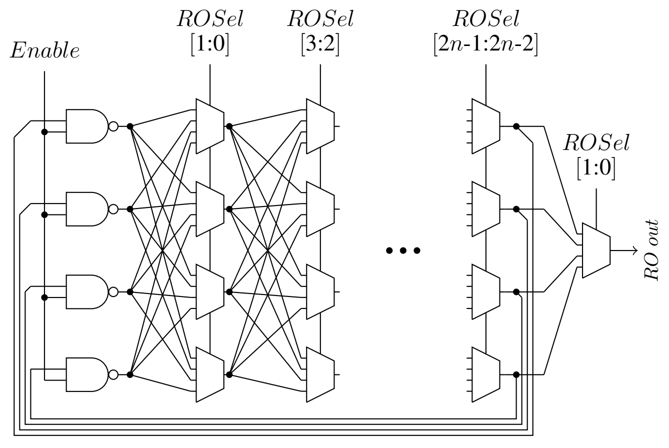

Fig. 4. Architecture of the configurable RO.

to the elementary RO-TRNG model presented by Ma et. al. in [16]. However, our results show that the approximation used by Yang et. al. [14] doesn't hold in case when the oscillators are very well matched, i.e. when the period difference is the same order of magnitude as the jitter. For this reason, we estimate the distribution of $CSCnt$ by running an event-driven simulation of the COSO-TRNG in *MATLAB*.

Variables for this model are:

- The sampled signal's average period length: $\mathrm{E}[T_{RO0}]$
- The average period length difference: $\mathrm{E}[\Delta]$
- The jitter strength: $\mathrm{Var}[T_{RO0}]/\mathrm{E}[T_{RO0}]$

In the model, we assumed both random variables T_{RO0} and T_{RO1} to be independent and Gaussian distributed:

$$T_{RO0} \sim \mathcal{N}(\mathrm{E}[T_{RO0}], \mathrm{Var}[T_{RO0}]), \tag{5}$$

$$T_{RO1} \sim \mathcal{N}(\mathrm{E}[T_{RO1}], \mathrm{Var}[T_{RO1}]). \tag{6}$$

Another assumption is made by equating the variance of both oscillating signals:

$$\mathrm{Var}[T_{RO0}] = \mathrm{Var}[T_{RO1}]. \tag{7}$$

It can be justified by the fact that both ROs are implemented in the same technology (have the same jitter strength) and the period difference ($\mathrm{E}[\Delta]$) is small.

From the estimated $CSCnt$-distribution, we estimate the probability of a zero/one (p_0/p_1) as:

$$p_i = \sum_{j=0}^{\infty} \Pr(CSCnt = 2j + i), \text{ with } i \in \{0, 1\}. \tag{8}$$

The min-entropy is then calculated as:

$$H_\infty = -\log_2(\max_{i\in\{0,1\}} p_i). \tag{9}$$

The throughput is equal to:

$$T = \frac{1}{\mathrm{E}[CSCnt] \cdot \mathrm{E}[T_{RO1}]}. \tag{10}$$

These equations enable us to estimate the min-entropy-throughput product (HTP). Figures 2 and 3 show the simulation results for variables obtained for a *Spartan 6* and *SmartFusion2* implementation respectively. The upper part depicts the min-entropy, which should be higher than 0.91 according to AIS-31 [8] as indicated by the shaded region. The point with minimal $CSCnt$ that meets this requirement is indicated by the vertical line and all values of $CSCnt$ or Δ to the right of it give configurations that comply with the AIS-31 standard. This region is indicated by a shade with gradient, because configurations further to the right have reduced throughput and are therefore less desired. To maximise throughput, a configuration as close as possible to the vertical line is wanted. The lower part shows the HTP. It shows a maximum for certain values of $\mathrm{E}[\Delta]$. This maximum is a trade-off between long accumulation time to provide sufficient entropy and keeping the throughput high.

III. Architecture

To be able to control the matching of two ROs with a few picoseconds of precision, a configurable design is necessary. We propose the architecture, shown in Fig. 4. Each RO stage is implemented as a combination of four multiplexers (MUXs) in the same vertical column. For each of the stages, the controller is able to choose one MUX from every column. In this way an RO is obtained with $n+1$ number of stages, where n equals the number of columns containing a MUX. An additional column containing only NAND gates is necessary to disable/enable the RO.

This topology enables 4^n possible combinations, each with a slightly different oscillation period due to on-chip delay variability. Both the ROs that make up the entropy source, shown in Fig. 1, are implemented using this architecture to increase matching symmetry resulting in $(4^n)^2$ combinations in total.

Simplified pseudocode describing the controller is depicted in Alg. 1. Its function is monitoring if the $CSCnt$-signal is still within predefined boundaries (L and H in the algorithm, representing the lower and upper bound respectively) and selecting a new RO combination ($ROSel$) if necessary. This $\mathrm{E}[CSCnt]$ is directly related to the matching of the two ROs ($\mathrm{E}[\Delta]$) via Eq. (1). The controller sequentially goes through the possible combinations until a suitable one is found. Optimal boundaries have to be chosen to maximise the throughput and provide sufficient entropy from Fig. 2 and 3. A smaller range $[L, H)$ enables finer control, but increases the controller latency to find a suitable configuration.

This controller activates at start-up to find a configuration that validates the stochastic model. After start-up, the controller remains actively checking the produced $CSCnt$ values and dynamically recalibrates the ROs when necessary.

IV. Experimental results

In this section, we prove the following claims:

- The architecture is feasible, it can obtain a wide range of $CSCnt$ values.

Algorithm 1 Controller

```
Input: $CSCnt[7:0]$, $req$
Output: $ROSel[2n\text{-}1:0]$, $matched$
Global constant: $L$, $H$
1:  $goodSamples[6:0] \leftarrow 0$
2:  $sampleCnt[6:0] \leftarrow 0$
3:  $ROSel[2n\text{-}1:0] \leftarrow 0$
4:  $matched \leftarrow 0$
5:  while $true$ do
6:      if $req$ then
7:          if $L \leq CSCnt[7:0] < H$ then
8:              $goodSamples[6:0] \leftarrow goodSamples[6:0] + 1$
9:              $matched \leftarrow 1$
10:         if $sampleCnt[6:0] == 2^7 - 1$ then
11:             if $goodSamples[6:0] == 0$ then
12:                 $ROSel[2n\text{-}1:0] \leftarrow ROSel[2n\text{-}1:0] + 1$
13:                 $matched \leftarrow 0$
14:             $goodSamples[6:0] \leftarrow 0$
15:         $sampleCnt[6:0] \leftarrow sampleCnt[6:0] + 1$
```

- Global Placement (GP) or searching for an optimal location is not required.
- Local Placement (LP) or relative placement constraints are not required.
- The architecture is portable.

Each of these claims is verified in the following subsections. First the experimental set-up is further clarified and at the end, we also explore the optimal number of RO stages.

All the figures depicting a $CSCnt$ distribution have a shade with gradient representing the configurations with high enough entropy per bit.

A. Experimental set-up

We first implemented the proposed entropy source and digitisation as depicted in Fig. 1 on a *Xilinx Spartan 6* FPGA. The number of configurable stages (n) is set to four, which gives a total of 256 combinations for each RO. Every four-input MUX fits inside a six-input LUT, the NAND gates are also implemented by one LUT each. As for realistic values of $\mathrm{Var}[T_{RO}]$ and $\mathrm{E}[\Delta]$, $\mathrm{E}[CSCnt]$ is below 256, an 8-bit asynchronous counter can be used in the digitisation, but initially we used a 16-bit counter in the experiments. This counter is reset every period of S_{beat} and read by the controller. The LSB of this counter is used as the generated random bit.

B. Feasibility of the architecture

We first implemented the entropy source and digitisation, with manual placement on a *Xilinx Spartan 6* FPGA. The LUTs containing the RO stages are placed in a symmetrical way to facilitate matching the two ROs. The upper part of Fig. 5 shows a box plot of the frequencies for both ROs, each for the 256 possible configurations. A box plot visualises the data distribution, where 50 % of the data is contained within the box. A data point is considered as an outlier (marked with

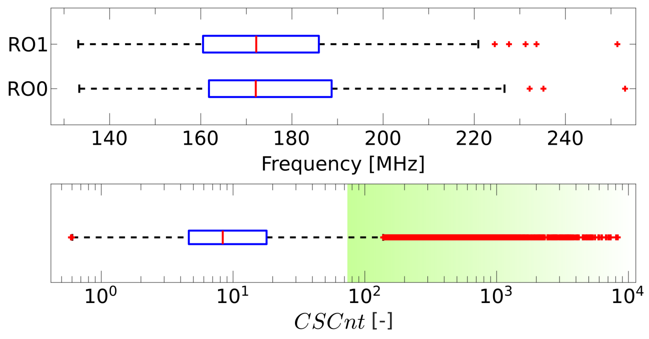

Fig. 5. Obtainable $CSCnt$ values for a fixed placement on *Spartan 6*.

a cross in the plot) if it is situated further than 1.5 times the interquartile range from the closest quartile. In the lower part of the figure, a box plot is given for the $256^2 = 65536$ obtained average $CSCnt$ realisations. A wide range up to 10^4 is achievable (Δ ranging down to 0.4 ps), which enables the controller to find a configuration close to the optimal HTP.

C. Global placement

To prove that this design is capable of matching the ROs independent from GP, the previous experiment is carried out on 25 locations, manually chosen to cover the entire chip. The LP constrains are maintained to improve matching. Figure 6 shows a box plot for each of the tested locations. The $CSCnt$ values in this and subsequent experiments are calculated based on the measured RO frequencies, instead of testing every possible RO pair, to reduce measurement time. This experiment shows that indeed sufficient matching can be obtained at every tested location, which is in strong contrast with previous designs that use coherent sampling with ROs [14], [15], [17], where a large design effort was needed to manually find an FPGA location with sufficient matching between the ROs.

D. Local placement and portability

The next experiment shows that even the LP constraints are not required for the correct operation of the TRNG. Figure 7 depicts the RO frequencies and corresponding calculated average $CSCnt$ values for an implementation on the *Xilinx Spartan 6* FPGA, without any specified placement constraints. The figure shows that matching can still be obtained.

To prove the portability, this experiment is repeated on a *Microsemi SmartFusion2* FPGA. Figure 8 shows the results, which are similar to those obtained on the *Spartan 6* FPGA. The *SmartFusion2* FPGA only features four-input LUTs, while the *Spartan 6* FPGA has six-input LUTs. The smaller LUTs mean that after porting, one four-input MUX does not fit into one LUT anymore. The design is slightly adapted by redefining the used primitives (only DFFs and LUTs) to account for the change in library naming at different FPGA vendors. The synthesis tool automatically constructs the four-input MUXs using multiple four-input LUTs. These results show that the portability claim is valid.

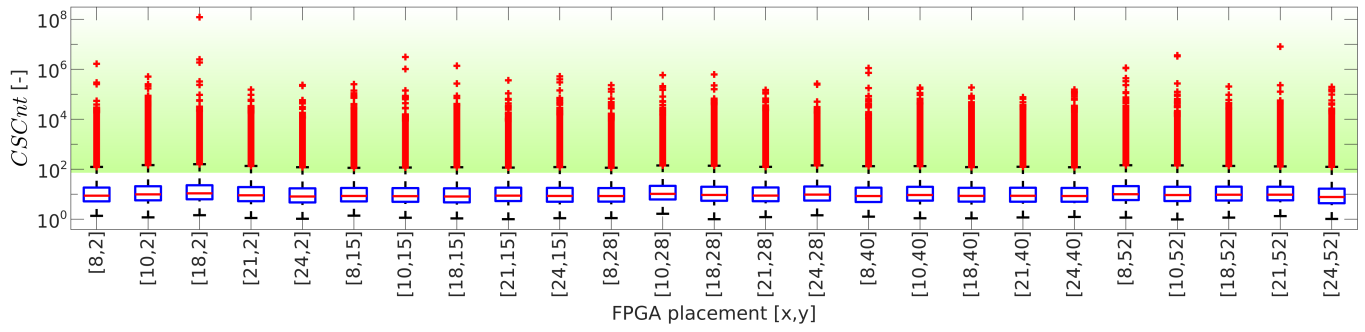

Fig. 6. Calculated $CSCnt$ values at different locations on *Spartan 6*.

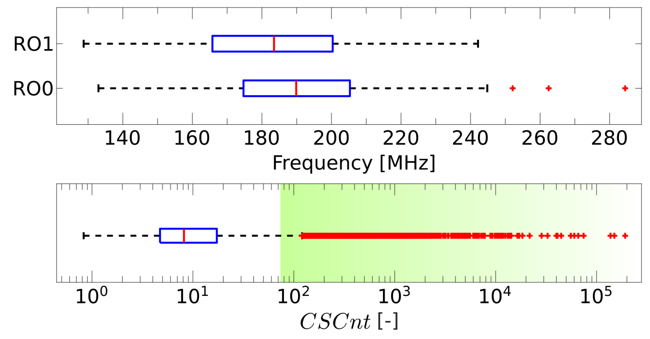

Fig. 7. Calculated $CSCnt$ values for omitted LP constraints on *Spartan 6*.

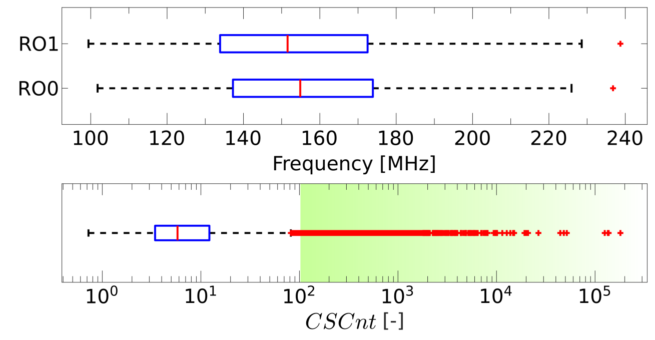

Fig. 8. Calculated $CSCnt$ values for omitted LP constraints on *SmartFusion2*.

E. Optimal number of stages

As a last experiment, we varied the number of stages and monitored the achievable $CSCnt$ values. More stages offer a greater number of configurations and increase the chance of finding sufficient matching. Figure 9 shows the results of this experiment on a *Spartan 6* with omitted placement constraints. Indicated next to the box plots are the number of configurations that obtain a $CSCnt$ higher than 73 and the percentage with respect to the total number of configurations. The topologies with one or two stages show none or only a few $CSCnt$ values that fall into the shaded region. These with

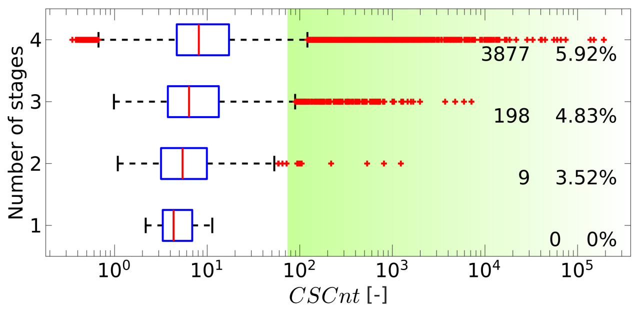

Fig. 9. Calculated $CSCnt$ values for different number of RO stages on *Spartan 6*.

three or more stages show a large amount of $CSCnt$ values in this region and are therefore preferred. A topology with three stages consumes less area and produces higher oscillation frequencies than one with four, therefore it is optimal.

The start-up time required by the controller to obtain a $CSCnt$ value in the range of $[74, 127]$ on a *Spartan 6* implementation is on average 1.5 ms over 100 experiments.

V. Results and Comparison

To get a theoretical estimate of the entropy per bit, the magnitude of the jitter strength $(\mathrm{Var}[T_{RO0}]/\mathrm{E}[T_{RO0}])$ is needed. We use the values obtained in [15], as similar FPGA devices are used there. To get a conservative entropy estimate, we take the minimal jitter strength over all frequencies tested in [15]. The obtained jitter strength value is 1.6 fs, which gives a period jitter equal to approximately 3 ps for frequencies in the range $[160\text{ MHz}, 264\text{ MHz}]$. At this jitter strength, the acceptable $CSCnt$ range is given by the shaded area in the lower part of Figs. 2 and 3.

Both the *Spartan 6* and *SmartFusion2* implementations obtain a configuration with an average $CSCnt$ value in this range, close to the maximal HTP: 81.12 and 107.85 respectively, producing a throughput of 3.30 Mbit/s and 1.47 Mbit/s. The Shannon entropy should be larger than 0.997 per bit as required by [8]. Converted to min-entropy, it should be higher than 0.91 per bit. Calculated from the model at a measured oscillation frequency of 160 MHz and 264 MHz,

TABLE II
COMPARISON WITH OTHER WORK.

Architecture	FPGA family	Area [DFFs/LUTs]	Throughput [Mbit/s]	Statistical test	Design effort
This work	***Spartan 6***	**39/108**[2]	**3.30**	**AIS-31 T6-T8**	**-**
	SmartFusion2	**38/111**[2]	**1.47**	**AIS-31 T6-T8**	**-**
Original COSO [15]	*Spartan 6*	3/18	0.54	AIS-31 T8	MP
	Cyclone V	3/13	1.44	AIS-31 T8	MP
	SmartFusion2	3/23	0.328	AIS-31 T8	MP
COSO: one bit per half cycle [17]	*Actel Fusion AFS600*	7/24[1]	2	NIST SP 800-22	MP & MR
	Spartan 3	7/18[1]	1.6	NIST SP 800-22	MP & MR
COSO: mutual sampling [17]	*Actel Fusion AFS600*	14/29[1]	4	FIPS 140-2	MP & MR
	Spartan 3	14/23[1]	3.2	FIPS 140-2	MP & MR
COSO: parameter adjustment [14]	*Virtex-5*	109 slices	4.08	NIST SP 800-22	MP & MR
DC-TRNG [18]	*Spartan 6*	128 slices	1.1	AIS-31 T6-T8	MP
	Cyclone V	273 ALMs	1.116	AIS-31 T6-T8	MP & MR
PLL-TRNG [18]	*Spartan 6*	190 slices[3]	1.0416	AIS-31 T6-T8	PLL required
	Cyclone V	273 ALMs	1.04	AIS-31 T6-T8	PLL required
ES-TRNG [11]	*Spartan 6*	5/10	1.15	AIS-31 T0-T5	MP
	Cyclone V	6/10	1.067	AIS-31 T0-T5	MP
TERO [15]	*Spartan 6*	12/39	0.625	AIS-31 T8	MP & MR
	Cyclone V	12/46	1	AIS-31 T8	MP & MR
	SmartFusion2	12/46	1	AIS-31 T8	MP & MR
STR [15]	*Spartan 6*	256/346	154	AIS-31 T8	MP & MR
	Cyclone V	256/352	245	AIS-31 T8	MP & MR
	SmartFusion2	256/350	188	AIS-31 T8	MP & MR

[1] Values calculated using shown circuit diagram.
[2] Values calculated for three-stage ROs, controller hardware included.
[3] Including embedded tests and data interface.

the obtained min-entropy is 0.95 per bit and 0.93 per bit for the *Spartan 6* and *SmartFusion2* respectively. The bit streams pass test procedure B (T6 - T8), described in AIS-31 [8] using 120 Mbit of generated data and have an estimated entropy of 0.999 per bit and 0.997 per bit, using the results from test T8. Both implementations are therefore PTG.3 compliant if cryptographic post processing is added. We utilise CBC-MAC post processing as specified in the NIST SP 800-90B standard [6]. After post processing, the throughput is lowered by a factor of two and both pass the NIST SP 800-22 tests.

A comparison to previous COSO-TRNGs and other designs that comply with the AIS-31 standard is given in Table II. Compared to other designs, this work offers a throughput higher than 1 Mbit/s and a larger area, but still a lot smaller than the design proposed in [14]. However, this design comes with a built-in on-line test which alarms the user in case that no sufficient matching can be found. According to both standards [6], [8], an on-line testing module is a necessary requirement for certification and therefore increases the area usage of the other designs. The design effort is also lowered in this design. Previous work required at least Manual Placement (MP) and often also Manual Routing (MR). This effort had to be repeated every time when the design is implemented on an other device, even from the same FPGA family.

If higher throughput is required, the techniques proposed in [17] can still be utilised with our proposed entropy source. These techniques boost the throughput by a factor of four. However the stochastic model has to be extended to provide a valid entropy estimation when using mutual sampling.

VI. Conclusion and future work

In this work, we proposed a new entropy source for the COSO-TRNG that significantly reduces the design effort. We experimentally verified the feasibility of the architecture and showed that it is always possible to meet the entropy requirements from the stochastic model even when placement constraints are omitted or the design is ported to another FPGA family. This property renders the TRNG highly suitable for integration into larger cryptosystems. Random bits are generated with a throughput of 3.30 Mbit/s and 1.47 Mbit/s on a *Spartan 6* and a *SmartFusion2* FPGA respectively and are able to pass the statistical tests. The design only has a modest area requirement, but comes with a cost of increased latency when the controller is searching for an optimal configuration.

Future work will focus on investigating whether this methodology is also applicable to other TRNG designs that require a large design effort, optimising the searching strategy of the controller to improve the induced latency, and observing the behaviour of this entropy source under active manipulation attacks.

References

[1] J. L. Danger, S. Guilley, and P. Hoogvorst, “High speed true random number generator based on open loop structures in FPGAs,” *Microelectronics journal*, vol. 40, no. 11, pp. 1650–1656, 2009.

[2] P. Z. Wieczorek and K. Golofit, “Dual-metastability time-competitive true random number generator,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 1, pp. 134–145, 2014.

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

[5] L. E. Bassham III, A. L. Rukhin, J. Soto, J. R. Nechvatal, M. E. Smid, E. B. Barker, S. D. Leigh, M. Levenson, M. Vangel, D. L. Banks *et al.*, "A statistical test suite for random and pseudorandom number generators for cryptographic applications," *NIST Special Publication 800-22 rev. 1a.*, 2010.

[6] M. S. Turan, E. Barker, J. Kelsey, K. A. McKay, M. L. Baish, and M. Boyle, "Recommendation for the entropy sources used for random bit generation," *NIST Special Publication 800-90B*, 2018.

[7] NIST, "Security requirements for cryptographic modules," *FIPS PUB 140-2*, 2001.

[8] W. Killmann and W. Schindler, "A proposal for: Functionality classes for random number generators," https://cosec.bit.uni-bonn.de/fileadmin/user_upload/teaching/15ss/15ss-taoc/01_AIS31_Functionality_classes_for_random_number_generators.pdf, 2011, [Online; accessed 22-May-2019].

[9] 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 - 15th International Workshop, Santa Barbara, CA, USA, August 20-23, 2013. Proceedings*, 2013, pp. 179–196.

[10] V. Rožić, 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, CA, USA, June 7-11, 2015*, 2015, pp. 116:1–116:6.

[11] B. Yang, V. Rožić, M. Grujić, N. Mentens, and I. Verbauwhede, "ES-TRNG: A high-throughput, low-area true random number generator based on edge sampling," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2018, no. 3, pp. 267–292, 2018.

[12] 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.

[13] 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.

[14] J. Yang, Y. Ma, T. Chen, J. Lin, and J. Jing, "Extracting more entropy for TRNGs based on coherent sampling," in *International Conference on Security and Privacy in Communication Systems*. Springer, 2016, pp. 694–709.

[15] O. Petura, U. Mureddu, N. Bochard, V. Fischer, and L. Bossuet, "A survey of AIS-20/31 compliant TRNG cores suitable for FPGA devices," in *2016 26th International Conference on Field-Programmable Logic and Applications (FPL)*. IEEE, 2016, pp. 1–10.

[16] Y. Ma, J. Lin, T. Chen, C. Xu, Z. Liu, and J. Jing, "Entropy evaluation for oscillator-based true random number generators," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2014, pp. 544–561.

[17] B. Valtchanov, V. Fischer, and A. Aubert, "Enhanced TRNG based on the coherent sampling," in *2009 3rd International Conference on Signals, Circuits and Systems (SCS)*. IEEE, 2009, pp. 1–6.

[18] J. Balasch, F. Bernard, V. Fischer, M. Grujić, M. Laban, O. Petura, V. Rožić, G. van Battum, I. Verbauwhede, M. Wakker, and B. Yang, "Design and testing methodologies for true random number generators towards industry certification," in *2018 IEEE 23rd European Test Symposium (ETS)*, vol. 2018. IEEE, 2018, pp. 1–10.