Design and Testing Methodologies for True Random Number Generators Towards Industry Certification

Josep Balasch*, Florent Bernard[†], Viktor Fischer[†], Miloš Grujić*, Marek Laban[‡], Oto Petura[†],
Vladimir Rožić*, Gerard van Batum[§], Ingrid Verbauwhede*, Marnix Wakker[§], Bohan Yang*

*KU Leuven COSIC, Kasteelpark Arenberg 10 bus 2452, B-3001 Heverlee, Belgium

{josep.balasch, milos.grujic, vladimir.rozic, ingrid.verbauwhede, bohan.yang}@esat.kuleuven.be

[†]Hubert Curien Laboratory, UMR CNRS 5516, University of Lyon, 42000 Saint-Etienne, France

{florent.bernard, fischer, oto.petura}@univ-st-etienne.fr

[‡]MICRONIC, Sliachská 2/C, 83102, Bratislava, Slovak Republic

laban@micronic.sk

[§]Brighsight B.V., Brassersplein 2, 2612 CT Delft, The Netherlands

{battum, wakker}@brighsight.com

Abstract—The objective of this paper is to provide insight on the design, evaluation and testing of modern True Random Number Generators (TRNGs) aimed towards certification. We discuss aspects related to each of these stages by means of two illustrative TRNG designs: PLL-TRNG and DC-TRNG. Topics covered in the paper include: the importance of formal security evaluations based on a stochastic model of the entropy source, the development of suitable and lightweight embedded tests to detect failures, the implementation and testing of TRNGs in dedicated FPGA platforms, and a robustness assessment to environmental and/or physical modifications.

I. INTRODUCTION

Random numbers are essential in cryptography. They are widely used as confidential keys, initialization vectors and padding values. They are also used in cryptographic challenge/response protocols and even in countermeasures against side channel attacks. Random Number Generators (RNGs) constitute one of basic cryptographic primitives. They must generate random numbers that have good statistical properties and the generated sequences must not be predictable or vulnerable to manipulation. Strict unpredictability (and thus security) can be guaranteed only if the randomness is coming from some non-manipulable physical process. For this reason, the implementation of True Random Number Generators (TRNGs) directly impacts the security of the whole cryptographic system.

In contrast to other cryptographic primitives (e.g. ciphers or hash functions), TRNGs are not standardized. The reason is that unlike algorithmic cryptographic primitives, the evaluation of TRNGs cannot be dissociated from the way they are physically implemented in target technologies. In other words, when considering the security of TRNGs, both their principle and their implementation in hardware must be taken into account. Because the physical implementation cannot be known in advance, principles of key generators such as TRNGs cannot be standardized.

The traditional TRNG design and evaluation approach is depicted in Figure 1. The main blocks are a *digital noise source*, which exploits some physical source of randomness,

followed by an optional *post-processing* block. 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. Historically, during the design and the security evaluation

Fig. 1. Traditional TRNG design and 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 [1], NIST SP 800-22 [2] or DIEHARD [3]. The generator was not certified for the practical use if the statistical tests did not succeed.

This approach is no longer suitable for modern data security systems for several reasons: 1) the algorithmic post-processing can mask considerable weaknesses of the source of randomness; 2) standard statistical tests can only evaluate the statistical quality of generated numbers and not their entropy (which should guarantee unpredictability as the main security parameter); and 3) standard high-end statistical tests are complex and thus expensive and slow, needing huge data sets. Consequently, they are executed only occasionally or on demand and only on selected sets of data.

In addition, the importance of security-by-design in random number generation is further illustrated by recent advances in attacking TRNGs. For example, in [4] Marketos and Moore

could manipulate “random” numbers generated in a secure commercial processor by injecting a periodic signal to its power supply. In [5], Bayon et al. could manipulate numbers generated by a set of 50 ring oscillators even to a bigger extent by using strong electromagnetic fields. And in [6], Martin et al. investigated the impact of power and clock glitches, temperature and underpowering on a TRNG design implemented on an FPGA. Physical attacks are considered to be very dangerous, as they can act temporarily during a time interval when the generator is used but not tested. Therefore, TRNGs need to incorporate sound mechanisms to detect such attacks.

A. Paper organization

In the remaining of this paper we describe the modern approach towards the design and evaluation of TRNGs for security applications and we present a high security TRNG, which was designed in the framework of the European HECTOR project, as an example. In Section II we review the main elements of a typical TRNG architecture. In Section III we present current approaches in security evaluation and certification of TRNGs. In Section IV we illustrate the design approach of the HECTOR RNG, a hybrid TRNG that contains two independent sources of digital noise: the PLL-TRNG and the DC-TRNG. In Section V we provide results of experiments in which we verify the robustness of these designs to temperature variations. Lastly, we conclude the paper in Section VI.

II. GENERATION OF RANDOM NUMBERS FOR CRYPTOGRAPHY

In electronic circuits, various physical phenomena can be explored as sources of randomness. These phenomena range from thermal noise in transistors, metastability, chaos, to timing jitter of clock signals generated in ring oscillators, self-timed rings or phase-locked loops. Both TRNGs as presented in Section IV extract randomness from the timing jitter.

A modern TRNG architecture as depicted in Figure 2 consists of the *digital noise source*, the *post-processing* and the *embedded tests*. The digital noise source contains two components: the *entropy source* and the *digitization module*. The entropy source is the only component generating true randomness, while all other modules are purely deterministic. When the output of the entropy source is in the form of analog signals, the digitization module is required to convert it into the digital form. The *raw random signal* at the output of the digitization module doesn’t always have perfect statistical properties. For this reason, a post-processing is utilized to enhance statistical properties of the raw random numbers usually by a suitable data compression method. The output of the post-processing is denoted as *internal random numbers*. Embedded tests are lastly required to monitor the health of the entropy source, e.g. Total failure test and on-line tests in Figure 2.

The TRNG principles can be evaluated according to two groups of criteria: security criteria and design criteria [7]. The security criteria include availability of the stochastic model,

entropy rate at output, testability, and robustness. The main design criteria include area, output bit rate, power consumption, feasibility in selected technology (FPGA and/or ASIC), repeatability, and portability.

TRNG evaluation standards use two types of entropy to quantify randomness. *Min-entropy*, used by the NIST standard [8] is the most conservative measure of unpredictability. It is equal to the information content of the most likely outcome. *Shannon entropy*, used by the AIS 20/31 [9], is equal to the average information content of the random variable. If not specified otherwise, in the remainder of this paper we will use the term entropy to denote the Shannon entropy.

III. MODERN METHODS IN SECURITY EVALUATION AND CERTIFICATION OF RANDOM NUMBER GENERATORS

During the last years, the American National Institute for Standards and Technologies (NIST) and the German Federal Office For Information Security (BSI) worked independently on new approaches in security evaluation and certification of random number generators. The difference in these approaches comes mainly from differences in evaluation schemes in the USA and in Europe. While the European certification authority is involved in the certification process, the American NIST is responsible for the standard, but not for its application. As a consequence, the American standard is currently less stringent as far as entropy rate estimation is concerned: the stochastic model of the generator is not required and the entropy rate is estimated using selected black box statistical tests.

A. German and European Approach in RNG Security Evaluation and Certification

In 2001, the German Federal Office for Information Security (BSI) proposed the first version and in 2011 an updated version of an evaluation methodology for random number generators (AIS 20/31) [9], which should help designers to better consider security aspects in their design and which should help evaluators of generators in the evaluation process.

The AIS 20/31-compliant design and evaluation is depicted in Fig. 2. Comparing to the traditional approach depicted in

Fig. 2. AIS 20/31-compliant TRNG design and evaluation

Fig. 1, the AIS 20/31 methodology [9]:

- 1) requires construction of a statistical model aimed at entropy estimation and management;
- 2) 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 increase eventually the entropy per bit (usually by some data compression method);

- 3) 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;
- 4) defines on-line testing strategy by executing permanently a simple and fast total failure test and by executing at initialization, regularly or on demand embedded online tests
- 5) defines two groups of off-line tests: Procedure B tests, which are aimed at testing raw binary signal (essentially for entropy) and Procedure A tests for testing statistical quality of internal random numbers;
- 6) requires proof of robustness of the entropy source against variations of environmental conditions and/or detection of situations, in which these conditions are out of allowable intervals.

Nowadays, the German methodology in the RNG design and evaluation has become the de facto standard in Europe.

B. NIST Approach in RNG Security Evaluation and Certification

The American NIST specifies requirements on the RNG design, security evaluation and certification in the NIST SP 800-90 standard, which is composed of three parts, A, B, and C. Part A deals with recommendations for random number generation using deterministic random bit generators [10], Part B gives recommendations for the (physical and non-physical) entropy sources [8], and Part C specifies random bit generation constructions, describing how to connect two above mentioned generators to a robust hybrid random number generator [11].

The NIST SP 800-90B compliant design and evaluation is depicted in Fig. 3. Comparing to the AIS20/31 approach, the

Fig. 3. NIST SP 800-90B compliant TRNG design and evaluation

NIST approach:

- 1) does not require construction of a statistical model, instead, two sets of statistical tests are used to estimate the entropy rate – one set estimates entropy if the generated numbers are independent and identically distributed (IID) and the second one serves to estimate entropy if the generated numbers are non-IID;
- 2) defines an on-line testing strategy by executing continuously at least two specified health tests: the Repetition count test and the Adaptive proportion test.

We can conclude, that in order to make the generator compliant with both standards, the designer has to satisfy requirements of both standards and namely all requirements of AIS 20/31 and the requirement of implementation of NIST continuous health tests. We will illustrate this approach on the HECTOR RNG design.

IV. HECTOR TRNG DESIGN APPROACH

In this section we illustrate a modern RNG design, which is compliant with both AIS20/31 and NIST SP 800-90 requirements, with the HECTOR RNG. The HECTOR RNG is a hybrid true random number generator (HTRNG) of security level PTG.3 (see Fig. 4). It contains two independent sources of digital noise implemented in two separate FPGAs: Intel Cyclone V FPGA and Xilinx Spartan 6 FPGA. The cryptographic post-processing block of security level DRG.3, the control unit, and data interfaces are implemented in a third FPGA: the Microsemi SmartFusion 2 system on chip device featuring an ARM M3 microcontroller.

Fig. 4. Functional diagram of HECTOR HTRNG

A. PLL-based TRNG: PLL-TRNG

The TRNG exploiting the jitter introduced by the PLL, was first proposed in [12]. The PLL-based TRNG (PLL-TRNG) uses coherent sampling to generate a stream of random bits. The PLL plays two roles in the generator:

- The random jitter caused by electric noises inside the PLL serves as a source of randomness.
- The PLL guarantees the following relationship between its input and output frequencies:

$$f_1 = f_0 \cdot \frac{K_M}{K_D}, \quad (1)$$

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

The block diagram of the PLL-TRNG is depicted in Fig. 5. The clock signal clk_1 is sampled in a D flip-flop (DFF) using the reference clock signal clk_0 . The output of the flip-flop is decimated in the decimator, in which K_D samples (outputs of the DFF) are added modulo 2 to form one raw random bit at the output of the generator.

Fig. 5. PLL-based TRNG

In this first configuration of the PLL-TRNG, the source of randomness is the tracking jitter of the PLL, i.e. the difference in phases between the reference clock (ideally jitter-free) and the jittery output clock of the PLL. Because of the PLL principle, the tracking jitter of the PLL 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) [13].

Figure 6 depicts an example of input/output waveforms of the PLL-TRNG, in which the multiplication factor is $K_M = 5$ and the division factor is $K_D = 7$. It can be observed that the rising edges of the reference clock signal (clk_0) are placed in seven positions during one period $T_Q = K_D T_0 = K_M T_1$. In two of them, the rising edges of clk_0 appear when the sampled signal is equal to one (samples 3 and 6 taken in the first half of the sampled clock period). At the moment when two other rising edges occur, the sampled signal is equal to zero (samples 1 and 4 in the second half of the sampled period). At one rising edge of the reference clock the sampled clock rises from 0 to one (sample 0). Finally, the last two of the 7 samples belonging to the T_Q period appear close to the falling edge of the sampled signal (samples 2 and 5). The position of the seven samples repeats in all periods T_Q causing a pattern with few unstable bits at the DFF output.

The decimator from Fig. 5 can be seen as a one-bit counter counting bits equal to one during each period T_Q . The decimator value at the end of the T_Q period representing the TRNG output depends thus on the number of unstable (i.e. random) bits. However, we can remark that it also depends on the duty cycle of the sampled signal, which should remain stable.

Fig. 6. Example of the PLL-TRNG input/output waveforms

It was shown in [12] that if the standard deviation of the PLL output jitter (σ_{jit}) fulfills the following condition:

$$\sigma_{jit} > \text{MAX}(\Delta T_{min}), \quad (2)$$

at least one sample will be random during each period T_Q . The term $\text{MAX}(\Delta T_{min})$ in Condition (2) represents the worst case (the longest) distance between the rising edges of the clock signal clk_0 and rising or falling edges of clk_1 during the T_Q period. It is given by

$$\text{MAX}(\Delta T_{min}) = \frac{T_0}{4K_M} \text{GCD}(2K_M, K_D), \quad (3)$$

where GCD means the Greatest Common Divisor.

As shown in [12], if K_M and K_D are relatively prime and K_D is odd, the TRNG output bit rate is $R = T_Q^{-1} = f_0/K_D$ and the sensitivity to jitter is $S = \Delta^{-1} = K_D/T_1$. 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_0 should be as high as possible,
- to increase R , K_D should be as low as possible,
- to increase S , K_M should be as high as possible.

From a One-PLL to a Two-PLL TRNG Design

In some technologies, Condition (2) 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 the sensitivity to jitter by increasing the multiplication and division factors (see the top panel and the bottom panel in Fig. 7, respectively). Although the effect of increasing the final multiplication and

Fig. 7. PLL-TRNG using two PLLs in series or in parallel

division factors is similar in both configurations of the PLL-TRNG, they differ significantly in the size of the exploited jitter [14]. In the cascaded connection of the two PLLs (top panel in Fig. 7), the jitter introduced by PLL0 is filtered out by PLL1. This is clearly not the case in the parallel configuration. Therefore, the parallel configuration of the PLL-TRNG is preferable and will be used in our design.

HECTOR PLL-TRNG

The PLL-TRNG proposed in the framework of the HECTOR project uses the differential jitter between two clocks generated in two independent PLLs as a source of randomness. It is depicted in Fig. 8. The generator uses two PLLs to generate one reference clock signal ($clk_0 = 192.7$ MHz) and two sampled clock signals (clk_{10} and clk_{11}), one of which is delayed by 90 degrees. The input clock signal frequency is $f_{in} = 125$ MHz and the synthesized frequencies are $f_0 = 192.7$ MHz and $f_1 = 475$ MHz, so the multiplication factor

Fig. 8. Schematic diagram of HECTOR PLL TRNG

of the PLL-TRNG is $K_M = 456$ and the division factor is $K_D = 185$.

Two outputs of PLL1 are sampled in two flip-flops and their outputs are XORed together and sampled to obtain sampled jitter clock signal featuring a pattern, which depends on the clock frequency ratio. The decimator is a one-bit counter, which counts number of samples (present at the output of the last flip-flop) during K_D periods of the reference clock. At the end of this interval, the counter gives the random bit value. So the bit rate of the generator is $R = f_0/K_D = 1.0416$ Mbits/s.

The generator is accessible in two modes: Evaluator mode and User mode.

Evaluator Mode

In the Evaluator mode of the PLL-TRNG, two kinds of RNG core outputs can be obtained to verify its correct operation, including operation of the two PLLs, of the two sampling flip-flops, and of the sampled output of the XOR gate:

- 1) *Sampled output of the XOR gate* – can be used to characterize the jitter and to verify correct operation of the two PLLs,
- 2) *Raw binary signal* – access to this signal is needed to apply Procedure B of the TRNG evaluation, in which tests T6 to T8 of AIS 20/31 need to be applied on the generated raw binary signal.

Availability of the Evaluator mode is very important since it makes security evaluation of the generator much easier. It thus reduces manufacturing costs and simplifies the security evaluation process.

The type of the TRNG core output is selected using the *Mux_sel* signal (*Mux_sel* = '0' to output the sampled XOR signal and *Mux_sel* = '1' to output the raw random signal). By default, the raw random data output is selected.

User Mode

In the User mode of the HECTOR RNG, the raw random bitstream featuring entropy rate per bit of at least 0.997 (as required by AIS 20/31) is sent to the cryptographic post-processing block implemented in the SmartFusion[®]2 device. In this mode, the user has thus access only to the cryptographically post-processed output signal.

Embedded Parametric Statistical Tests

The sampled output of the XOR gate is used internally by

both the Total failure and Online tests. The Total failure test [9] (test T0 in HECTOR RNG) counts the number of random samples appearing in each of 255 periods T_Q (one period T_Q is composed of K_D periods of clk_0). If no random sample was found, the Total failure alarm (*Err_t0*) is triggered and a system interrupt is requested.

Since the latency of the Total failure test is 255 periods T_Q , two 128-bit buffers are placed between the TRNG core output and the post-processing block input. The first one is depicted in Fig. 8. The second is a part of the serial-to-parallel converter inside the cryptographic post-processing unit and it is therefore not depicted in this figure. The 256-bit capacity of the two buffers ensures that all the bits entering the post-processor are tested by the Total failure test before usage.

Two Online tests [9] evaluating the output of the XOR gate sampled by the following flip-flop (the DFF output) are implemented:

- the Online test T1 computes parameter P_1 corresponding to the number of random samples during subsequent periods T_Q (a period T_Q is the period of pattern appearing at the sampled output of the XOR gate, which depends on frequency ratio of the clock signals generated in PLLs),
- the test T2 computes parameter P_2 corresponding to the variance of the clock jitter measured during 4080 periods T_Q .

Computed values P_1 and P_2 are compared with thresholds obtained from the stochastic model and required for the Shannon entropy rate per bit of 0.997 ($4 < P_1 < K_D/4$; $228 < P_2 < 1280$). Both Online tests have identical probability of the false alarm equal to 2^{-42} , which means that for the given bit rate (approximately 1 million bits per second), the false alarm can appear once in 4 million seconds, i.e. once in almost two months. Note that values of parameters P_1 and P_2 obtained in practice are usually much higher than the low thresholds ($P_1 \sim 14$, $P_2 \sim 400$) so the probability of the false alarm is even smaller. Based on the model, the values of parameters P_1 , P_2 indicate that the entropy rate at generator output is much higher than required (0.997).

The execution time of the Online test (4080 periods T_Q , i.e. about 4 ms) represents a compromise between precision of entropy estimation and reactivity of the test. While guaranteeing a sufficient precision, it is much shorter than that of the fastest general-purpose statistical tests FIPS-140-1 [1], which need at least 20 000 random bits (i.e. almost 5 times more bits).

Implementation Results

The PLL-TRNG can be implemented on both Intel Cyclone V and Xilinx Spartan-6 FPGAs of the HECTOR evaluation boards. In Intel Cyclone V FPGA, the design consumes 273 ALMs and achieves a throughput of 1.04 Mbps. In Xilinx Spartan-6 FPGA, the design occupies 190 slices including three embedded test and data interfaces. Although the two PLLs are set up differently comparing to generator implemented in Intel FPGA (the synthesized frequencies are $f_0 = 234.375$ MHz and $f_1 = 241.667$ MHz, so $K_M = 232$,

Fig. 9. Block diagram of the DC-TRNG design

$K_D = 225$), the generator achieves the same throughput $R = f_0/K_D = 1.0416$ Mbits/s.

B. Delay-Chain based TRNG: DC-TRNG

The Delay-Chain based TRNG (DC-TRNG) uses jitter accumulated in a free-running ring oscillator as a source of randomness. It is depicted in Figure 9. The original principle of the DC-TRNG and a simple stochastic model of it was published in [15]. A refined stochastic model was introduced in [16] to account for the non-linearity of the delay-chain.

HECTOR DC-TRNG

The ring oscillator is sampled by a tapped delay chain with a timing resolution around $17ps$. A system clock sourced by a quartz oscillator is used for circuit synchronization and triggering the sampling. The sampled result is converted to a raw binary bit using a priority encoder followed by a decimator. Raw binary bits can be sent to the mother board directly for offline evaluations. Raw binary bits are compressed using another decimator to improve entropy-per-bit. Outputs of the free-running oscillator and the tapped delay chain are tested for total failure. Two types of online tests are applied to the raw binary bits.

Two frequency dividers are used to generate different clock signals. The frequency divider 1 is chosen based on the jitter accumulation time. The divider 2 is used to generate the data-clk signal when the post-processing is applied. An implementation of the digital noise source for Xilinx Spartan6 FPGA is shown in Figure 10. The free-running oscillator is implemented using a single look-up table (LUT). A primitive called CARRY4 is used to construct the tapped delay chains.

The generator can work in two modes: evaluation mode and user mode. In the evaluation mode, the raw random bit can be collected to derive parameters for online tests and used as input for Procedure B of the TRNG evaluation. Tests T6 to T8 of AIS 20/31 are applied on the raw random bits.

In the user mode, the random bit sequence after the algorithmic post-processing can be obtained as the input to the cryptographic post-processing implemented in the SmartFusion2 device. The random bit sequence meets the AIS 20/31 requirement, according to which this input needs to feature a 0.997 Shannon entropy per bit as the lower bound.

Fig. 10. Implementation of the digital noise source of DC-TRNG

In order to provide an additional security margin while fulfilling the throughput requirement of the output, a decimator is used to improve the robustness of the TRNG core against potential active attacks.

Embedded tests

Two types of embedded tests were implemented: total failure tests and online tests.

1) *Total failure tests*: In order to check whether the entropy source has totally broken down, total failure tests are included. The source of randomness can fail if:

- the free-running ring oscillator is missing or not working,
- there is no edge sampled by the tapped delay chain.

We implement two total failure tests. The first total failure test is designed to detect whether the free-running ring oscillator is toggling or not. An edge detector is connected to the output of the ring oscillator. The edge detector is reset every two cycles. It triggers an alarm if the input signal has not changed over these two cycles.

The second total failure test is designed to generate an alarm if there is no edge sampled in the tapped delay chain. During normal operation, all taps cannot be all zeros nor all ones. Therefore, the equality of all captured values in the delay chain is used to generate the alarm signal.

2) *Online tests*: The design methodology proposed in [17] is followed to design two online tests for DC-TRNG. These two statistical tests detect long-term weaknesses in the generated raw binary bits with different false-alarm rates. On-line test 1 (Sensitive test) has a higher false-alarm rate but is more efficient in detecting attacks. On-line test 2 (Robust test) is less efficient in detecting attacks but it is more robust against false alarms. Both on-line tests are working on the raw random numbers, but different alarm policies are applied.

On-line test 1 operates on a sequence of 512 consecutive raw bits. The test statistics is the count N_{111} of the template 111 in the sequence. This statistics is computed on-the-fly using a 9-bit counter. After every 512 bits, the counter value is compared with the pre-computed (empirically determined) upper and lower boundaries. If the counter exceeds these boundaries, an alarm signal is generated. The false alarm rate of this test is 1%. For on-line test 1, the following alarm policy is applied to provide high flexibility. When this alarm is triggered four

consecutive times, a warning should notify the user without blocking the output of the TRNG. The DC-TRNG will stop outputting and be restarted when there are five consecutive alarms generated by on-line test 1. For on-line test 1, given the throughput of raw random numbers (2.2Mbps), 5 consecutive false alarms appear once in 27 days.

On-line test 2 also operates on a sequence of 512 consecutive raw bits. The test statistic C_1 is computed as follows:

$$C_1 = \sum_{i=1}^{511} b_i \oplus b_{i+1}, \quad (4)$$

where b_i denotes the i -th bit of the sequence. This statistics is computed using an xor gate and a 9-bit counter. After every 512 bits, value C_1 is computed and compared with the pre-computed boundaries. The alarm signal is generated if the counter value exceeds these boundaries. Afterwards, the counter is reset and the next 512 bits are tested. The false alarm rate of this test is 10^{-9} . Since on-line test 2 is designed to be robust, a different alarm policy is used. DC-TRNG is blocked from outputting and restarted whenever the alarm is triggered. The false alarm for on-line test 2 appears once in around two days and 16 hours.

Implementation Results

The DC-TRNG is implemented on both Xilinx Spartan-6 and Intel Cyclone V FPGAs of the HECTOR evaluation board. For Xilinx Spartan-6 FPGA, the design occupies only 128 slices while achieving a throughput of 1.1 Mbps. For Intel Cyclone V FPGA, the design consumes 273 ALMs and achieves a throughput of 1.116 Mbps.

C. Cryptographic Post-processing, Known Answer Test and Continuous Tests

Besides the control logic accessible from the local micro-controller, the FPGA fabric in the HECTOR RNG control device contains several hardware blocks belonging to the RNG (see Fig. 11): two serial-to-parallel converters, two sets of the NIST SP 800-90B continuous tests (one for each TRNG core), and a cryptographic post-processing block including the known answer test (KAT).

The serial data received from the two TRNG cores are converted into 32-bit words and synchronized to the system clock in two serial-to-parallel converters. Serial data from the two TRNG cores are input to serial inputs of the cryptographic post-processing block, in which they are converted to two 128-bit input vectors. The cryptographic post-processing block uses three AES ciphers to ensure security level DRG.3 [9]. It generates one 128-bit block from two input blocks and reads 32-bit chunks out of 128-bit output blocks. The speed of the post-processed data bit rate is defined by the slower TRNG, which is currently slightly more than 1 Mbit/s. The KAT is executed after every reset of the post-processing block.

Two kinds of 32-bit words can be saved in the data memory:

- unprocessed data coming from one of the two TRNG cores (raw random numbers in Fig. 11),

Fig. 11. Block diagram of RNG blocks placed in the HECTOR RNG control device

- cryptographically post-processed data (output of the cryptographic post-processing block, i.e. internal random numbers in Fig. 11).

The format of the unprocessed TRNG core output depends on the TRNG core itself. For example, for the PLL-TRNG, using the *Mux_sel* signal from Fig. 8, the user can read output of the XOR gate or the Decimator output. For the DC-TRNG from Fig. 9, two kinds of signals can be stored depending on the value of the *Mux_sel* signal: the Encoder output or the algorithmic post-processing output.

The cryptographic post-processing block features also two serial outputs: the two serial bit streams passing across the input serial-to-parallel converter are output and used for continuous testing. Two NIST continuous tests ensure compliance with requirements of NIST SP 800-90B [8]: the Repetition count test and Adaptive proportion test. Both tests evaluate continuously the raw random bit stream just before the cryptographic post-processing block (as required by NIST SP 800-90B). Consequently, they test correct operation of all RNG blocks preceding the cryptographic post-processing block (and namely the decimator, 128-bit output buffer and LVDS I/Os for the PLL-TRNG, and the algorithmic post-processing and LVDS I/Os for the DC-TRNG, which are not tested by dedicated tests of the two TRNG cores). The execution time of the NIST SP 800-90B Continuous tests is similar to that of Online tests (about 4 ms).

Finally, we wish to stress that since the cryptographic post-processing generates 128 bits from 256 high quality raw random bits, the HECTOR generator generates full entropy random data according to NIST SP 800-90.

D. TRNG Hardware Design

As shown in Figure 4, the HECTOR RNG is implemented in three FPGA devices. Two of them contain one independent TRNG core including embedded tests and the third one contains the cryptographic post-processing functions and related blocks and the main control unit.

The TRNG cores are connected to the main control device (SF2) using a fast low voltage differential signaling (LVDS) interface aimed at high-speed data acquisition and a slow synchronous serial interface used as a control interface.

Raw random bit streams coming from the two TRNG cores are acquired (after being cryptographically post-processed or not) directly in the 64 MB RAM memory. The memory is divided in two halves: one is used for data acquisition and the second one is used as a solid disk, which is accessible from the host computer (a PC) as a data storage medium (mass storage device). Thanks to the direct memory access and the use of fast data interface, no data is lost during data acquisition. The maximum size of the acquired data block is 30 MB (few data blocks are left free for the file system).

The block diagram of the HECTOR RNG hardware is depicted in Fig. 12. Besides the three FPGA devices and the external memory presented in the previous paragraph, it

Fig. 12. Block diagram of the HECTOR RNG hardware

contains a USB hub, and two USB interfaces (a USB to UART converter and a USB physical layer device used for the mass storage device), an SD card connector, and other data and power connectors.

The SD card can contain device drivers, which can be downloaded to the host PC. HECTOR RNG communicates with the host PC via two USB ports: a UART serial interface (virtual COM port) and a mass storage device interface. The UART port is used to send commands or small data blocks and to receive demonstrator state words or small data blocks. The mass storage device port is used to transfer high volume data files (up to 30 MB) from HECTOR RNG to the host PC.

Special attention was paid to the desing of power supplies: to avoid generation of a deterministic noise, only linear power regulators and high quality filters are used. The control device monitors continuously operating conditions (temperature and power voltage). The controller creates a log file, in which it saves information about all security critical events: alarms and warnings.

The HECTOR RNG hardware is depicted in Fig. 13. Three FPGA areas visible in the picture are separated by a ground plate. They are closed in three separated chambers inside the metallic shielding.

V. VERIFICATION TESTING OF TRNG DESIGNS

In this section we provide results of the verification testing of our designs implemented in the HECTOR RNG hardware.

Fig. 13. HECTOR RNG hardware

In particular, we perform a robustness test to check the behaviour of the designs over a certain temperature range.

Methodology

The structure of the test proceeds as follows:

- The temperature of the target device is varied between -40°C and 80°C in steps of 10°C .
- Once the temperature is set and stabilized, three sets of 2 MB of random data are acquired at the TRNG outputs.
- The data sets are processed by an AIS 20/31 statistical analysis tool, developed according to the specifications of BSI [9].

The TRNG output quality is verified by running the AIS 20/31 test suite. This test suite consists of nine individual statistical tests T0 to T8, each covering a statistical aspect of the data. A description of the tests follows:

- **Test T0** (disjointness test): a sequence of groups of six bytes passes the disjointness test if the subsequent members are pairwise different.
- **Test T1** (monobit test): the bit sequence $b_1, \dots, b_{20000}$ passes the monobit test if X , the number of values 1, satisfies: $9654 < X < 10346$.
- **Test T2** (poker test): the bit sequence $b_1, \dots, b_{20000}$ passes the poker test if groups of four bits do not repeat significantly more than statistically expected.
- **Test T3** (run test): a run is a maximum sub-sequence of consecutive zeroes or ones. The bit sequence $b_1, \dots, b_{20000}$ passes the run test if the number of occurring run lengths lies within the permitted intervals, as specified in [9].
- **Test T4** (long run test): a run of length ≤ 34 is called a long run. The bit sequence $b_1, \dots, b_{20000}$ passes the long run test if no long run occurs.
- **Test T5** (autocorrelation test): for $\tau \in \{1, \dots, 5000\}$, Z_{τ} is the of number of times a bit from $\{1, \dots, 5000\}$ differs from the bit τ positions further. The bit sequence $b_1, \dots, b_{20000}$ passes the autocorrelation test (with shift τ) if $2326 < Z_{\tau} < 2674$. (Note that the sub-sequence $b_{10001}, \dots, b_{20000}$ is not used in the test variable.)
- **Test T6** (uniform distribution test): the sequence $w_1, \dots, w_n \in \{0, 1\}^k$ passes the uniform distribution

TABLE I

ENTROPY TEST (T8) RESULTS AT DIFFERENT TEMPERATURES, ROUNDED TO 7 DECIMAL PLACES (COLUMNS 2 TO 4). TEST FAILURE FOR DIFFERENT TESTS AT DIFFERENT TEMPERATURES, NUMBER OF FAILURES OUT OF THE 257 TESTS (COLUMNS 5-9).

TEMP	PLL-TRNG	DC-TRNG	DC-TRNG	PLL-TRNG		DC-TRNG		DC-TRNG
	(raw)	(raw)	(post-processing)	(raw)		(raw)		(post-processing)
	T8	T8	T8	T0-T5	T6-T8	T0-T5	T6-T8	T0-T8
-40° C	7.9998891	7.995574	8.0037523	5 T1, 1 T2	Pass	3 T1	Pass	Pass
-30° C	7.9974187	8.001387	8.0028506	53 T1, 4 T2, 1 T3	Pass	7 T1, 1 T2	Pass	Pass
-20° C	7.9917223	7.995273	8.0011306	118 T1, 20 T2, 1 T3	Pass	1 T1	Pass	Pass
-10° C	7.9956523	7.995273	8.0010628	57 T1, 6 T2	Pass	1 T1	Pass	Pass
0° C	8.0006363	7.998265	7.9975243	Pass	Pass	3 T1, 1 T2	Pass	Pass
10° C	7.9964841	7.995066	7.9979476	Pass	Pass	5 T1, 1 T2	Pass	Pass
20° C	8.0005664	7.992955	7.9978494	Pass	Pass	Pass	Pass	Pass
30° C	7.9946934	7.991739	8.0025181	Pass	Pass	31 T1, 7 T2, 1 T3	Pass	Pass
40° C	8.0006823	7.994178	7.9985431	Pass	Pass	30 T1, 21 T2, 8 T3	Pass	Pass
50° C	7.9992708	7.982312	8.0033354	1 T1	Pass	80 T1, 75 T2, 25 T3	Pass	Pass
60° C	7.9989801	7.999119	7.9981832	Pass	Pass	Pass	Pass	Pass
70° C	7.9945552	7.989778	7.9923489	Pass	Pass	58 T1, 36 T2, 8 T3	Pass	Pass
80° C	8.0017179	7.990528	8.0003341	Pass	Pass	1 T1	Pass	Pass

test with parameters (k, n, a) if none of the $x \in \{0, 1\}^k$ occurs more than $n(2^{-k} + a)$ or less than $n(2^{-k} - a)$. Comment: for $k = 1$, $n = 20000$ and $a = 0.0173$, the uniform distribution test corresponds to the monobit test T1.

- **Test T7** (comparative test for multinomial distributions): this test checks that the occurrence of a specific value for elements of a sequence is approximately χ^2 -distributed over different samples.
- **Test T8** (entropy test): the entropy test is performed in accordance with Coron [18]. The bit sequence $b_1, \dots, b_{(Q+K)L}$ is segmented into non-overlapping output words $w_1, \dots, w_{Q+K}$ of length L . A_n is the distance from w_n to its predecessor with the same value, which is used for the Coron test.

Tests T0 to T5 are applied on the internal numbers, while tests T6 to T8 are applied on the raw output data. A test will pass when complete runs of the AIS 20/31 statistical test suite passes (three times). A test fails when two out of three runs of the statistical tests contain failing tests. The procedure for the statistical tests is described in [9] (paragraph 210). It comes down to: The test suite is in principle performed once and all basic tests must pass in order to pass Test procedure A. If one of the 1285 basic tests (e.g. a single mono-bit or poker test) fails, then the test procedure can be repeated once on new random data. That second time all 1285 tests must have the verdict pass to formally pass Test procedure A. In all other cases Test procedure A fails. Note that due to the nature of randomness one or more of the individual tests of the test suite may fail. Depending on the severity of the fail and the importance of the test, this may or may not cause the whole run of the test suite to fail. It can also happen that online tests detect output anomalies, which cause the TRNG to stop operating. In such cases the TRNG will not provide output and the test will fail. The expected outcome of the test is that all runs of the AIS 20/31 statistical test suite pass over the temperature range of -40° C to +80° C.

Test Results

All AIS 20/31 tests were executed. The monobit test, poker test, run test, long run test, and autocorrelation test were performed 257 times, by applying them on different parts of the data collected. The disjointness test, uniform distribution test, comparative test for multinomial distributions, and entropy test were performed once on the whole data.

The first 3 columns in Table V show the results of the entropy test. When the entropy is above 7.976 bits entropy per byte (meaning that the entropy of 256 bit seed is at least 255 bits), it is above the threshold required by the standard. It can be seen that all entropy tests passed for all three TRNG output streams for all temperatures. The entropy value above eight can be explained by rounding errors in the calculation (summation of $p \cdot \log_2(p)$ many times). In fact the real entropy values are indistinguishably close to eight in the precision used.

The last 5 columns in Table V show the tests that failed for each output stream and at which temperatures. It can be observed that, in case of the post-processed DC-TRNG output, all tests passed. For the raw outputs of the two TRNGs it can be seen that some monobit tests fail. However, the bias is rather minor and quite stable and the number of ones or zeros does not exceed much the allowed interval. When the difference between the number of zeros and ones gets higher, other tests are also being impacted. When this is the case, the poker and run tests start failing also. When the monobit test fails on the raw) PLL-TRNG, it is always due to the number of ones being higher than the allowed maximum. The number of zeros and ones should be between 9654 and 10346 in order to pass the test. When the test fails, the number of ones is above 10346. On the acquired data the highest number of ones for any of the failing tests was around 10500. Therefore, on the acquired data, the number of ones was always below 101.5% of the allowed maximum, which can be considered as a small bias. When the monobit test fails on the (raw) DC-TRNG, it is most of the time because of the number of ones being lower than the allowed minimum (9654). On the acquired data, the lowest

number of ones for failing tests was around 9000. Therefore, on the acquired data, the number of ones was always above 93.2% of the allowed minimum.

The results show that for the DC-TRNG no obvious impact of the temperature on the quality of the random numbers could be observed. For the PLL-TRNG, more failures could be observed at low temperatures. Failures of the tests on the raw data of the two TRNGs could be observed; some monobit, poker and run tests did not pass, making the whole test not pass. This is caused by a small bias in the raw output streams. Despite the small bias, it is to notice that the quality of the raw random data is high, as the obtained entropy values are high.

Overall, the verification testing shows that the principles used in the TRNGs can generate sufficient randomness. Further fine-tuning of parameters will be useful for removing a small bias towards zero bits and making statistical test results more stable over different operational conditions.

VI. CONCLUSIONS

In this paper we have documented the modern approach towards design and testing of TRNGs by means of our work on the development of the HECTOR RNG. In particular, we have described the design strategy followed in two different TRNG architectures (PLL-TRNG and DC-TRNG) in order to fulfill the requirements of both AIS20/31 and NIST SP 800-90.

We demonstrated that while the area occupied by the two TRNGs including embedded tests was relatively small, the output bit rate was sufficiently high (more than 1 Mbit/s required by HECTOR).

We also showed that both TRNGs were feasible in selected technology and that they thus fulfilled the portability requirement. For practical use, it is important that the generators give repeatable results device by device.

The availability of the stochastic model, very high entropy rate at the output, testability, and robustness make them ideal candidates for industrial use in high-security cryptographic applications.

We combined the dedicated embedded tests of sources of randomness with general-purpose black box NIST continuous tests to test TRNG cores and the complete data path to test all RNG components in real time and thus to enhance security.

A critical point which is currently being investigated by long lasting testing experiments is to ensure that the embedded tests are truly effective and efficient against various types of attacks.

ACKNOWLEDGMENT

The research leading to these results has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 644052 (HECTOR - Hardware Enabled Crypto and Randomness).

REFERENCES

- [1] NIST, "FIPS 140-1: Security Requirements for Cryptographic Modules," [online] Available from <http://csrc.nist.gov/publications/fips/fips140-1/fips1401.pdf>, 1994.
- [2] 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.
- [3] G. Marsaglia, "DIEHARD: Battery of Tests of Randomness," [online] Available from <http://stat.fsu.edu/pub/diehard/>, 1996.
- [4] A. T. Markettos and S. W. Moore, "The frequency injection attack on ring-oscillator-based true random number generators," in *Cryptographic Hardware and Embedded Systems - CHES 2009*, ser. LNCS, C. Clavier and K. Gaj, Eds., vol. 5747. Springer, 2009, pp. 317–331.
- [5] P. Bayon, L. Bossuet, A. Aubert, V. Fischer, F. Pouchet, B. Robisson, and P. Maurine, "Contactless electromagnetic active attack on ring oscillator based true random number generator," in *Constructive Side-Channel Analysis and Secure Design - COSADE 2012*, ser. LNCS, W. Schindler and S. A. Huss, Eds., vol. 7275. Springer, 2012, pp. 151–166.
- [6] H. Martn, T. Korak, E. San Milln, and M. Hutter, "Fault Attacks on STRNGs: Impact of Glitches, Temperature, and Underpowering on Randomness," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 2, pp. 266–277, 2015.
- [7] V. Fischer, "A Closer Look at Security in Random Number Generators Design," in *Proceedings of Constructive Side-Channel Analysis and Secure Design - COSADE 2012*, ser. LNCS, vol. 7275. Springer-Verlag Berlin Heidelberg, 2012, pp. 167–182.
- [8] M. Turan, E. Barker, J. Kelsey, K. McKay, M. Baish, and M. Boyle, "Recommendation for the Entropy Sources Used for Random Bit Generation, NIST Special Publication 800-90B," [online] Available from <http://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP-800-90B.pdf>, 2018.
- [9] W. Killmann and W. Schindler, "AIS20/AIS31. 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
- [10] E. B. Barker and J. M. Kelsey, "SP 800-90A. Recommendation for Random Number Generation Using Deterministic Random Bit Generators," Gaithersburg, MD, United States, Tech. Rep., 2012.
- [11] —, "SP 800-90C. Recommendation for Random Bit Generator (RBG) Constructions," Gaithersburg, MD, United States, Tech. Rep., 2016.
- [12] 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.
- [13] A. Fahim, *Clock generators for SoC processors*. Cluwer Academic Publisher, MA, USA, 2004.
- [14] V. Fischer, M. Drutarovsky, M. Simka, and N. Bochard, "High performance true random number generator in altera stratix fplds," in *Field-Programmable Logic and Applications (FPL 2004)*, ser. LNCS, vol. 3203. Springer, 2004, pp. 555–564.
- [15] 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 (DAC), San Francisco, USA*, 2015, pp. 116:1–116:6.
- [16] M. Grujić, V. Rožić, B. Yang, and I. Verbauwhede, "A Closer Look at the Delay-Chain based TRNG," in *ISCAS*, 2018, pp. 1–5.
- [17] B. Yang, V. Rožić, N. Mentens, W. Dehaene, and I. Verbauwhede, "TOTAL: TRNG on-the-fly testing for attack detection using Lightweight hardware," in *DATE*, 2016, pp. 127–132.
- [18] J. Coron, "On the security of random sources," in *Practice and Theory in Public Key Cryptography, PKC '99*, ser. LNCS, H. Imai and Y. Zheng, Eds., vol. 1560. Springer, 1999, pp. 29–42.