On-Line Monitoring of Random Number Generators for Embedded Security

Renaud Santoro, Olivier Sentieys, Sébastien Roy

On-Line Monitoring of Random Number Generators for Embedded Security

Renaud Santoro*, Olivier Sentieys* and Sébastien Roy‡
*IRISA - University of Rennes
6 rue de Kerampont
22300, Lannion, France
{santoro,sentieys}@irisa.fr
‡Dept. of Electrical and Computer Engeneering
Université Laval, Québec
Qc, Canada, G1K7P4
sebasroy@gel.ulaval.ca

***Abstract*— Many embedded security chips require a high-quality Random Number Generator (RNG). Unfortunately, hardware RNG randomness can vary in time due to implementation defects or certain kinds of attacks. To overcome this issue, this paper presents the implementation of a battery of statistical test for randomness. The battery is selected for its efficient implementation, making the area and power consumption insignificant. Performance and cost of the hardware implementation are given for FPGA and VLSI targets. Results show that statistical tests can easily be implemented in low-cost embedded security circuits and can enhance on-line monitoring of RNG randomness to prevent RNG failures.**

I. Introduction

The objective of a random number generator (RNG) is to produce random binary numbers which are statistically independent, uniformly distributed and unpredictable. RNGs are necessary in many applications like cryptography, communication, VLSI testing, probabilistic algorithms, and so on. RNG randomness evaluation is performed by using a battery of statistical tests. Several such batteries are reported in the literature including Diehard [1], NIST [2], FIPS 140-1, FIPS 140-2 [3] and the AIS 31 [4] batteries. They are all implemented using high-level software programming. When an RNG is evaluated, designers put a huge bit stream into memory and then submit it to software tests. If the bit stream successfully passes a certain number of statistical tests, the RNG is said to be sufficiently random.

However, the number of hardware applications requiring RNGs is continuously increasing, specially in embedded circuits (e.g. Field Programmable Gate Array (FPGA) and System-on-Chip). As a consequence, many applications need an embedded RNG. Generally, a hybrid RNG comprising a True Random Number Generator (TRNG) and a Pseudo Random Number Generator (PRNG) is used. PRNGs are based on deterministic algorithms. They are periodic, and must be initialized by a TRNG. TRNGs are based on a physical noise source (e.g. radioactive decay, thermal noise or free running jitter oscillators) and depend strongly on their implementation quality. Most of the TRNGs implemented in FPGA or ASIC (Application-Specific Integrated Circuit) use phase jitter produced by a free running oscillator or a Phase-Locked Loop (PLL). Jitter is the deviation of a signal from its ideal behavior. Jitter is caused by deterministic noise (power supply noise, cross-talk noise, pink noise) and by random noise (substrate noise, temperature) and strongly depends on the oscillator generator used in the TRNG (usually ring oscillator or PLL) [5]. In practice, jitter can be influenced by noise external to the FPGA (power supply noise, temperature) and by chip activity. This dependence is a weakness exploitable by hackers. In [6], the TRNG proposed in [7] was tested against an active non-invasive attack. It was shown therein that the generator randomness can vary by changing the temperature of the FPGA chip. If the TRNG configuration is not correctly chosen, the generator fails the statistical tests. In cryptography, security is usually based on the randomness quality of a key generated by an RNG. Some PRNGs are recognized to produce high quality random numbers [8]. However, their quality depend on TRNG seed randomness. Consequently, TRNG failures must be prevented in real time for applications requiring a high level of security. AIS 31 [4] defines a TRNG evaluation methodology based on the use of 8 statistical tests. Depending on the required randomness level, the TRNG noise source is tested by the statistical tests when the generator is switch on or/and during operation. As a result, the tests prevented TRNG failures. The solution that we propose in this paper is on-chip monitoring of the TRNG by using a hardware battery of four statistical tests. In [9] the implementation of the four tests has been realized for a targeted Xilinx Virtex 2 FPGA. However, as presented in this paper, our design is more efficient. Moreover, performance and cost of the hardware implementation are given for FPGA and VLSI targets. Furthermore, in contrast to [9], the paper is not only focused on test implementation. After the design presentation, two recognized TRNGs are evaluated by the on-chip monitoring in order to evaluate their randomness quality.

This paper is organized as follows. In Section 2, the statistical tests selection and implementation is described. Then, the relevance of the continuous TRNG monitoring is demonstrated

978-1-4244-3828-0/09/$25.00 ©2009 IEEE

in Section 3. Finally, conclusions and perspectives on the TRNG on-chip monitoring are given.

II. Statistical tests

A. Statistical test selection

Many batteries of statistical tests exist in the literature. The most widely recognized batteries are Diehard [1] and NIST [2]. These include 15 and 16 statistical tests respectively. The AIS 31 is composed of 8 statistical tests. Hardware implementation of some algorithms in these batteries would consume too much in terms of arithmetic and memory units. Consequently, less demanding (in terms of hardware resources) statistical tests must be chosen. Thus, four standard statistical tests were selected for implementation: the *frequency* test, the *poker* test, the *run* test and the *long-run* test. These tests analyze a 2×10^4-bit stream. The tests provide explicit bounds that the computed results must satisfy. If the binary stream fails any of the tests, the generator is considered not truly random. The tests were used in FIPS 140-1 and in FIPS 140-2 [3]. Moreover, they are currently included in AIS 31 [4]. Although these tests are less powerful than the Diehard and NIST batteries, the hardware implementation will be highly efficient. As a result, the hardware tests will have a high operating frequency allowing the analysis of a RNG in real-time to prevent deviation from its ideal behavior.

B. Statistical test implementation

1) Frequency test: This test verifies that the 2×10^4-bit stream is uniformly distributed. The bias of the sequence is analyzed. The number of 1s (n_1) is counted and must lie in $[9726, 10274]$.

2) Poker test: The bit stream is decomposed into four-bit non-overlapping subsequences and 5×10^3 blocks are created. The poker test verifies that subsequences are uniformly distributed. Occurrences of the 16 possible subsequences are counted. Let n_i be the number of occurrences of the integer i binary representation, with $0 \leq i \leq 15$. The value of X_2 given by

$$X_2 = \frac{2^4}{5 \times 10^3} \left(\sum_{i=0}^{15} n_i^2 \right) - 5 \times 10^3, \qquad (1)$$

follows a Chi-Square distribution with 15 degrees of freedom and must lie in $]2.16; 46.17[$ to pass the poker test.

In hardware, computing equation (1) is an area-consuming task. An alternative is to compute the summation $\sum_{i=0}^{15} n_i^2$ and to compare the result with the required interval $]Y_{min}, Y_{max}[$. In $\sum_{i=0}^{15} n_i^2$, the maximal value of a counter (n_i) must be computed as a function of the other counter values ($n_j, j \neq i$). The optimization problem

$$\begin{array}{ll} max(n_i) & \text{subject to} \\ & Y_{min} < \sum_{j=0}^{15} n_j^2 < Y_{max} \\ & \sum_{j=0}^{15} n_j = 5 \times 10^3 \\ & n_j \geq 0,\ 0 \leq j \leq 15 \end{array} \qquad (2)$$

i	Required interval for n_i^0 and n_i^1
1	2343-2657
3	1135-1365
3	542-708
4	251-373
5	111-201
≥ 6	111-201

TABLE I
Run test required interval.

Area (μm^2)	Power (mW)	Critical Path (ns)
29981.6	7.37	6.08 (164.47 MHz)

TABLE II
Total area, total dynamic power and critical path in 130 nm, 1.2V CMOS technology.

must be solved and n_i is maximal when $\sum_{j=0, j\neq i}^{15} n_j^2$ is minimal and the constraints are satisfied. The maximal value of a counter $max(n_i)$ is 428. This optimization reduces the counter bit width and the operator size in the Data-Flow Graph (DFG) as only one 9×9 multiplier and one 21-bit by 21-bit adder are required.

3) Run test: The run test determines if the number of runs of various lengths is as expected in a random sequence. Let n_i^1 and n_i^0 be respectively the number of 1-runs (sequence of consecutive 1s) and 0-runs (sequence of consecutive 0s) of length i. For $1 \leq i \leq 6$, n_i^1 and n_i^0 are counted. If the occurrences satisfy Table I, the random sequence passes the run test. Only twelve counters are required to memorize the occurrence of each run.

4) Long run test: This test verifies that there are no runs of length 26 or more, requiring only a 5-bit counter.

It has been shown in this section that the selected tests need very few arithmetic and memory hardware resources. Consequently, even if the four selected tests are less powerful than a stringent battery such as Diehard, results show that these tests provide a good trade-off between their ability to detect TRNG weaknesses and hardware efficiency. As demonstrated in the next section, the hardware tests will have a high operating frequency allowing the analysis of a TRNG in real-time to prevent deviation from its ideal behavior.

C. Implementation results

Statistical tests have been synthesized with the Synopsys Design Compiler tool targeting 130 nm CMOS technology. Table II summarizes the total area, the estimated dynamic power and the critical path results of the battery hardware implementation. Power results have been obtained with an operating voltage of 1.2V. Then, the battery has been implemented in a Xilinx Virtex 2 FPGA to compare the design with [9] and in a Xilinx Virtex 5 FPGA. Table III shows the total number of Look-Up Tables (LUT) and the design maximum frequency. The maximal frequency obtained by [9] is 113 MHz

targeted FPGA	Xilinx Virtex 2 XC2V1000-6	Xilinx Virtex 5 XC5VLX50T-3
Total LUTs	626 (6%)	482 (1%)
Maximum frequency (MHz)	134.7 MHz	189.4 MHz

TABLE III

BATTERY IMPLEMENTATION RESULTS IN XILINX VIRTEX 2 AND VIRTEX 5 FPGAS.

FPGA Temperature (°C)	Fischer and al. [7]	Schellekens and al. [11]
30	79	100
50	86.9	100
70	91.3	100
90	99.7	100
110	99.7	100
130	99.7	99.8

TABLE IV

SUCCESS PERCENTAGES (%) OF THE TWO TESTED TRNGS AS A FUNCTION OF THE FPGA TEMPERATURE (°C).

and is consequently inferior than the frequency of our design.

Tables II and III show that implementation is extremely efficient and needs few logic resource, suggesting that such batteries can be easily implemented into low-cost embedded circuits. Moreover, the operating frequency is high, making possible to test the high data rate TRNG noise source in real-time.

After the presentation of the statistical test selection and implementation, their interest in TRNG evaluation is presented in the next section.

III. TRNG RANDOMNESS EVALUATION

To show the TRNG on-chip monitoring interest, two recognized TRNGs are tested against temperature attacks, in real time using the proposed hardware statistical tests. These TRNGs have been chosen because they are well documented and they can be implemented in FPGA circuits. The first TRNG is based on sampling the phase jitter in a ring oscillator and a model to compute the number of ring oscillators satisfying a given randomness level. Figure 1 presents the principle of the TRNG [10]. Since the number of ring oscillators is too high, the actual number of oscillators implemented is reduced and a post-processing unit is added to correct the bias produced by the generator. It was shown in [10] that 114 ring oscillators of 13 inverters are necessary to have a secure TRNG. The post-processing unit is a $[256, 16, 113]$ BCH code used as a resilient function.

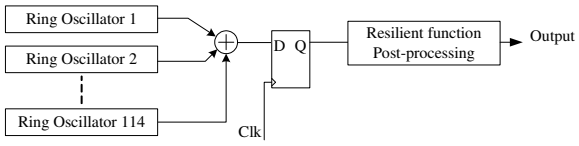

Fig. 1. Oscillator sampling proposed in [10].

Authors of [11] have described a TRNG based on [10] with reduced numbers of ring oscillators and inverters (110 ring oscillators of 3 inverters). Therefore, the TRNG presented in [11] is considered as the first tested TRNG. The frequency of the CLK signal in Figure 1 is taken equal to 40 MHz. As a result, the TRNG data rate is 2.5 Mb/s.

The second TRNG has been presented in [7]. Figure 2 gives the principle of the generator. One clock signal is sampled by another clock signal. The two clocks are generated using two PLLs. A simple XOR extractor is required to reduce the bias of the output bit stream. The principle is to perform a XOR on K_D successive bits of the output bit stream. In [7], the TRNG is implemented with $K_D = 203$ and the following PLL parameters, $M_{CLK}/D_{CLK} = 29/10$, $M_{CLJ}/D_{CLJ} = 27/7$.

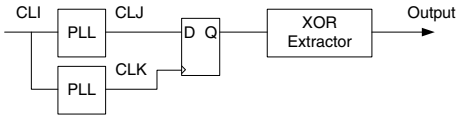

Fig. 2. Oscillator sampling proposed in [7].

Each TRNG is implemented in an Altera Stratix DSP board with an EP1S25 FPGA. Table IV gives the percentage of success for the two TRNGs tested when they are submitted to 1×10^3 statistical test realizations. On each realization the TRNG is declared truly random when the four tests are satisfied. If one test is not succeeded by the TRNG, the analyzed sequence is rejected. These measures are realized when the FPGA temperature is varying from 30 to 130°C. The recommended operating FPGA temperature is between 0 to 55°C. In order to show the dependance between TRNG behavior and its environment, the FPGA has been intentionally heated upper than 55°C. As presented in Table IV, the test success percentage is not always equal to 100%. At temperature below than 90°C, the second TRNG [7] presents some deficient random bit streams. However, by increasing the FPGA temperature, TRNG [7] randomness quality is improving. On the contrary, the on-line statistical test has a small efficiency on TRNG [11]. At 130°C, the tests filter only two bit streams on the 1×10^3 analyzed. According to the experiment, Table IV allows to conclude that TRNG [11] has a better randomness than the TRNG [7].

For TRNG [7], Figure 3 presents the test efficiency on the bit stream filtered by the tests. When the FPGA temperature is less or equal to 90°C, the TRNG defects are almost all detected by the frequency test. At 110°C, the run test and the poker test discern the generator failures while the frequency test is totally inefficient. Finally, at 130°C, the TRNG weaknesses are discovered by the run and the frequency test. This study

shows that the frequency test, the poker test and the run test should be preferred in order to monitor the TRNG [7].

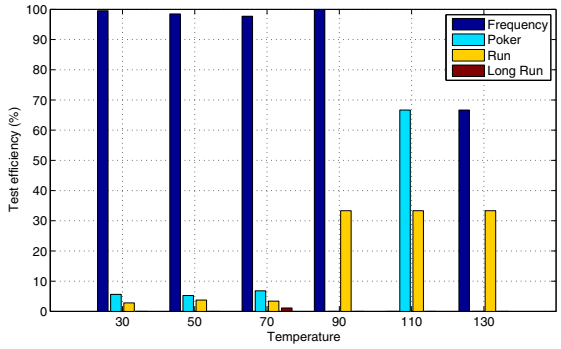

Fig. 3. Test efficiency in percent (%) on TRNG [7] as a function of the FPGA temperature (°C).

To show the benefit of the on-line monitoring on TRNG [7], the 8-bit entropy is computed for the bit stream produced by the generator (X) and the same sequence filtered by the tests ($X_{filtered}$). Results for FPGA temperature between 30°C to 130°C are reported in Figure 4. As shown, for temperature less than 90 °C, the on-line monitoring improves the bit stream entropy and demonstrates the methodology benefit.

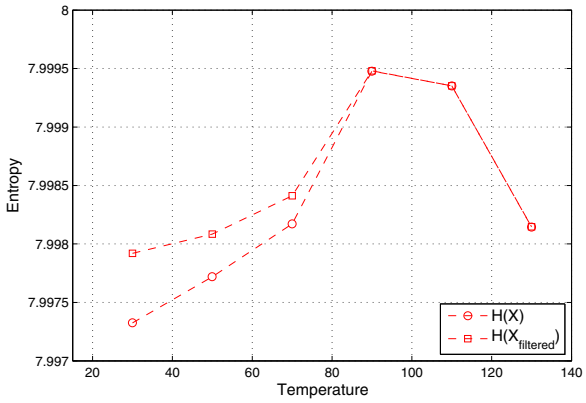

Fig. 4. Entropy computation as a function of the FPGA temperature (°C).

The experiments showed that TRNGs are sensitive to their environment such as temperature variations. However, the on-line monitoring allows to filter cases in which TRNGs are presenting some weaknesses. Consequently, even if the targeted device is used in an inappropriate environment, the design is able to improve the bit stream randomness.

IV. Conclusion

After the selection of an efficient battery of statistical tests, this paper presented its optimized hardware implementation. Each test has been studied to minimize the resources required by the design. Implementation results have been given for FPGA and VLSI targets. Implementation is extremely efficient, suggesting that such batteries can be implemented into low-cost embedded circuits. Using hardware statistical tests in secure circuits allows on-line monitoring of random number generator and to detect defects due to implementation quality, various types of attacks, and environment variations. The methodology improves the randomness quality of the bit stream produced by the TRNG [7]. Moreover, after having study the statistical test efficiency to detect TRNG weaknesses, the battery can be reduced to three tests, the frequency, the poker and the run test. The implementation is very useful for RNG designers, and considerably simplifies the randomness evaluation, a time-consuming task when using high-level software statistical tests. In futur works, more stringent tests will be added to the four statistical tests. As a result, TRNG weaknesses could be detected more efficiently. Then, the experiments will be extended to other recognized TRNGs, such as the generator described in [12] and [13].

References

[1] G. Marsaglia, "Diehard: A battery of tests of randomness," Florida State University, Tallahassee, FL, USA, Tech. Rep., 1996. [Online]. Available: http://stat.fsu.edu/pub/diehard/

[2] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, and D. Banks, "A statistical test suite for random and pseudorandom number generators for statistical applications," *NIST Special Publication in Computer Security*, pp. 800–22, 2001.

[3] FIPS, "Security requirements for cryptographic modules, FIPS PUB 140-2," p. 58 pages, 1999. [Online]. Available: http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf

[4] W. Killmann and W. Schindler, "A proposal for: Functionality classes and evaluation methodology for true (physical) random number generators," T-Systems debis Systemhaus Information Security Services and Bundesamt für Sicherheit in der Informationstechnik (BSI), Tech. Rep., 2001.

[5] A. M. Fahim, *Clock Generators for SOC Processors: Circuits and Architectures (Text, Speech & Language Technology)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2005.

[6] M. Simka, "Active non-invasive attack on true random number generator," in *In Proceedings of the VI. PhD student conference*, 2006.

[7] V. Fischer and M. Drutarovský, "True random number generator embedded in reconfigurable hardware," in *CHES '02: Revised Papers from the 4th International Workshop on Cryptographic Hardware and Embedded Systems*. London, UK: Springer-Verlag, 2003, pp. 415–430.

[8] P. L'Ecuyer and R. Simard, "Testu01: A c library for empirical testing of random number generators," *ACM Trans. Math. Softw.*, vol. 33, no. 4, p. 22, 2007.

[9] A. Hasegawa, S.-J. Kim, and K. Umeno, "Ip core of statistical test suite of fips 140-2," Communications Research Laboratory and ChaosWare, Inc., http://www.design-reuse.com/articles/7946/ip-core-of-statistical-test-suite-of-fips-140-2.html, Tech. Rep., 2003.

[10] B. Sunar, W. J. Martin, and D. R. Stinson, "A provably secure true random number generator with built-in tolerance to active attacks," *IEEE Trans. Comput.*, vol. 56, no. 1, pp. 109–119, 2007, member-Berk Sunar.

[11] D. Schellekens, B. Preneel, and I. Verbauwhede, "Fpga vendor agnostic true random number generator," in *Field Programmable Logic and Applications, 2006. FPL '06*, 2006, pp. 1–6.

[12] M. Dichtl and J. D. Golic, "High-speed true random number generation with logic gates only," in *CHES*, 2007, pp. 45–62.

[13] I. Vasyltsov, E. Hambardzumyan, Y.-S. Kim, and B. Karpinskyy, "Fast digital trng based on metastable ring oscillator." in *CHES*, ser. Lecture Notes in Computer Science, E. Oswald and P. Rohatgi, Eds., vol. 5154. Springer, 2008, pp. 164–180.