On-the-Fly Tests for Non-Ideal True Random Number Generators

Bohan Yang, Vladimir Rožić, Nele Mentens and Ingrid Verbauwhede
ESAT/COSIC and iMinds, KU Leuven,
Kasteelpark Arenberg 10, B-3001 Leuven-Heverlee, Belgium
Email:{bohan.yang, vladimir.rozic, nele.mentens, ingrid.verbauwhede}@esat.kuleuven.be

***Abstract*—Hardware implementations of statistical tests are needed to detect failures and statistical weaknesses of entropy sources in True Random Number Generators on the fly. Current implementations of these tests work under the assumption that the entropy source produces independent, identically distributed (IID) numbers. However, some entropy sources produce non-IID data and rely on compression to provide the full entropy. Currently there are no embedded test implementations suitable for this type of entropy source. We provide the first FPGA implementation of embedded tests that estimate the generated min-Entropy and verify if it is within the expected boundaries.**

I. Introduction

True random number generators (TRNG) are important cryptographic primitives because random bits are used to generate session-keys, initialization vectors, challenges in authentication protocols as well as masks for countermeasures against side-channel attacks. Embedded tests are necessary for on-the-fly monitoring of the generated random bits. These tests are recommended by the NIST [1] and AIS-31 [2] standards.

Statistical tests performed on the generated bit sequences can be used to detect failures or weaknesses of the noise source. Many embedded tests implementations have been put forward in recent years for both FPGAs [3], [4], [5] and ASIC [3], [6]. However, all presented tests are tailored for independent, identically distributed (IID) number generators. Unfortunately, hardware TRNGs rarely produce statistically perfect numbers and they require arithmetic post-processing to produce a full-entropy output. In order to detect TRNG failures quickly and reliably, we need on-the-fly tests that are suitable for non-IID data.

In this work, we present implementations of 4 statistical tests that report the level of entropy, rather than an alarm signal. These test algorithms are originally developed for estimating the min-entropy of non-IID sources during the prototype evaluation [1]. Our contribution is a compact hardware implementation of these tests which can be used for 2 applications:

1) Embedded evaluation of non-IID sources, where an alarm signal is set every time the entropy level drops below a pre-defined value.
2) On-the-fly calibration of the post-processing block, where the compression rate is increased when the entropy level drops in order to provide full-entropy output.

II. Test Algorithms: Description and Simplification

A set of 5 algorithms for min-entropy evaluation is presented in [1]. We find that one of those, namely the compression test, is not suitable for hardware implementation due to large memory requirements. The remaining 4 algorithms can be implemented after some simplifications. The algorithms are very general in the sense that they can be applied to any number of possible output values. We have implemented tests for a generator that produces one bit at a time, which simplifies the algorithms to some extent.

Our tests report 8 different levels of entropy on a scale from 0 to 7. The levels are corresponding to the 3 most significant bits of the binary representation of min-entropy per bit. This approach enables us to provide compact implementations that require storing of a small number of precomputed constants. All statistical tests provide more reliable results when the length of the tested sequence is longer. However, testing very long sequences means that the failures are not detected fast enough. As a compromise solution we have chosen a sequence length of 2^{13} bits.

Since each algorithm tests a sequence for a different type of failure, some statistical weaknesses will remain undetected by some tests (the test overestimates the entropy). For this reason, it is necessary to implement several tests and to observe the minimal result at the output.

A. The Collision Test

We say that a sub-sequence of a dataset contains a collision if it contains two equal data values. Since we are testing a TRNG that produces only 2 distinct data values, a collision can happen after 2 bits (if they are equal) or after 3 bits (if the first 2 bits are different). The whole sequence can be divided into segments that contain exactly one collision. These segments consist of either 2 or 3 bits, as illustrated in Figure 1.

The collision test measures the mean time until the first collision ($\overline{\mu}$). Since a collision always happens after either 2 or 3 generated bits, for a sequence of fixed length min-entropy can be determined using only one parameter. This parameter can be the total number of collisions, the number of 2-bit segments, or the number of 3-bit segments containing a collision. We chose to track the number of 3-bit segments since the maximal value of this parameter is the smallest which results in the most compact counter and comparators. Estimated min-entropy is a monotonically decreasing function of this parameter. Therefore,

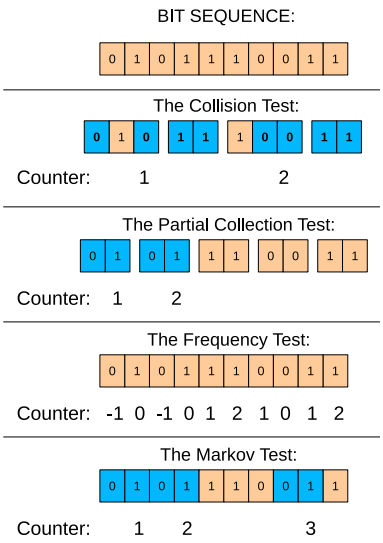

Fig. 1: An example illustrating the principles of the test algorithms.

since the test reports only 8 levels of min-entropy, hardware implementation requires only one counter and 7 cut-off values.

B. The Partial Collection Test

This test measures the number of distinct data values in each block of data. The length of a block is equal to the size of the output space (in this case 2). The concept is shown in Figure 1. The test statistics can be described using only one parameter. In this case it is the number of non-overlapping blocks containing 2 different bits. This test can be implemented using one counter and comparing the final value with the precomputed cut-off values.

C. The Frequency test

The purpose of this test is to find the frequency of the most common value in a data set (in this case either 0 or 1). This test can be implemented by using an up/down counter to track $|C_1 - C_0|$ and by comparing the final result with the precomputed cut-off values.

D. The Markov Test

This test relies on approximating the TRNG with a Markov process. This is a process where the output depends not only on the current state but also on the previous states. This test requires to estimate the probabilities of all states and all state transitions, and then to estimate the probability p_{max} of the most likely bit sequence. The min-entropy is then given by $H_{min} = -log_2(p_{max})$. In order to calculate H_{min}, 6 parameters have to be obtained from the bit-sequence, namely a count of occurrences of different states (C_0 and C_1) and counts of all state transitions (C_{00},C_{01},C_{10},C_{11}).

In order to reduce the number of parameters we make the following observations:

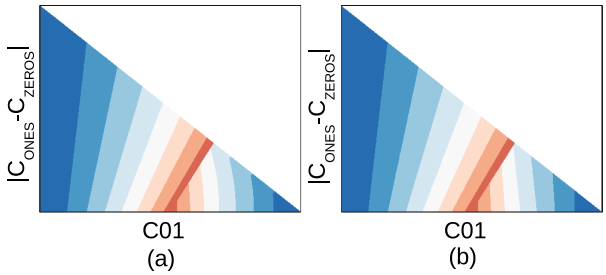

Fig. 2: Heatmap diagrams showing entropy levels for the Markov test: (a) Before approximations. (b) After approximations.

- $C_{01} \approx C_{10}$ These 2 values can differ by at most 1 which is negligible compared to the length of the sequence.
- $C_0 + C_1 = N$
- $C_{01} + C_{00} = C_0$
- $C_{11} + C_{10} = C_1$

Since there are 6 parameters with 4 constraints, we only need to keep track of 2 parameters. We have chosen to track $|C_1 - C_0|$ which is already available from the frequency test, and C_{01}. The entropy level was calculated for all possible parameter values and the results are shown in the heatmap diagram in Figure 2 (a). Different colors are used to denote entropy levels (blue for low entropy, red for high entropy), values above the diagonal never appear because $C_{01} \leqslant min(C_0, C_1)$.

Borders between different entropy level regions can be approximated with straight lines, as shown in Figure 2 (b). There are 15 lines in total, and each line is determined by 2 parameters: 15-bit slope coefficient and 19-bit value at zero. In total only 15 constants of 34 bits have to be precomputed. The entropy level can be determined using at most 15 multiplications, additions and comparison operations:

$$|C_1 - C_0| < C_{01} \cdot k_i + n_i \quad (1)$$

where k_i and n_i are the precomputed constants corresponding to the linear coefficients of the 15 borders between entropy regions.

III. Test Validation

In order to verify the functionality of the simplified tests, 100 sequences of 2^{13} bits were generated using simulated non-IID sources and the test results were compared with the theoretical min-entropy values. Two types of non-IID sources have been used for this purpose: a biased generator without bit-dependencies and an unbiased generator with correlation between the consecutive bits.

Figure 3 shows the test results obtained by testing the biased generator without bit-dependencies. We used 21 different biased generators with the probability of 0 varying from 0% to 100% in steps of 5%. For each bias value, 100 sequences were generated and the test results were recorded. Minimal

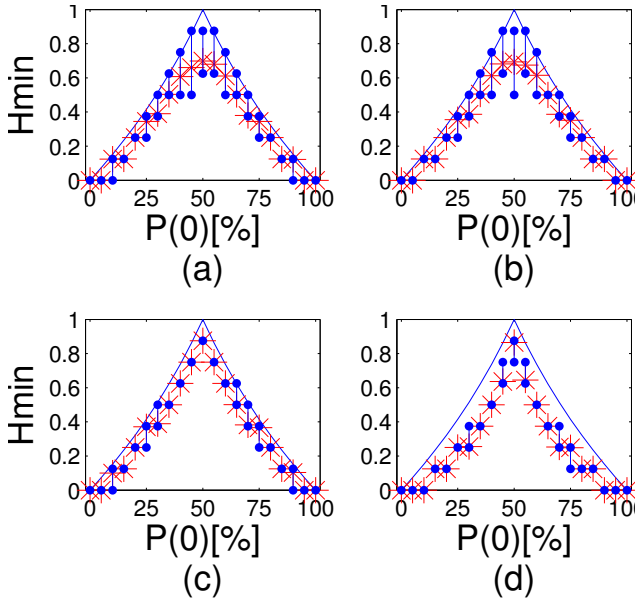

Fig. 3: Test results of the sequences generated by the biased generators: (a) The collision test. (b) The partial collection test. (c) The frequency test. (d) The Markov test.

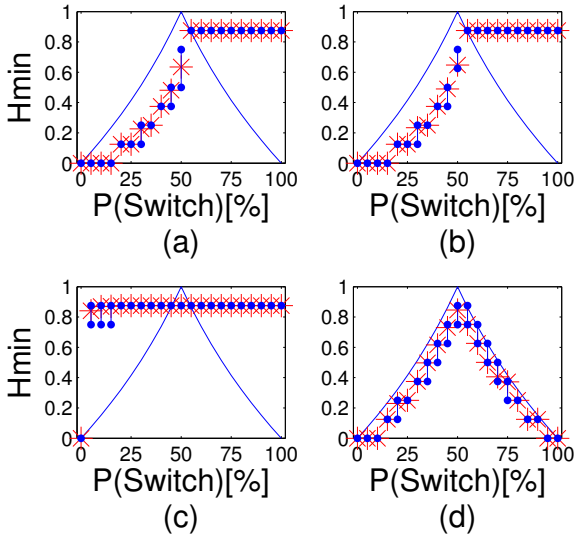

Fig. 4: Test results of the sequences generated by the generators with bit-dependencies: (a) The collision test. (b) The partial collection test. (c) The frequency test. (d) The Markov test.

and maximal values are shown as dots, and the average value is represented using the star symbol. The theoretical value of entropy is shown as a continuous line. As expected, the entropy estimations of all 4 tests follow the theoretical value of min-entropy.

Figure 4 shows the test results obtained using an unbiased generator with dependencies. For this experiment, we varied the probability of switching the bit value. Again, this probability was varied in steps of 5% and 100 sequences were generated for each value. It can be seen that the frequency test almost always reports the full entropy, which is expected because this test is not designed to detect this type of failure. The collision test and the partial collection test detect weaknesses only when the probability of switching is low. The only test that reliably estimates the min-entropy in the presence of bit-dependencies is the Markov test.

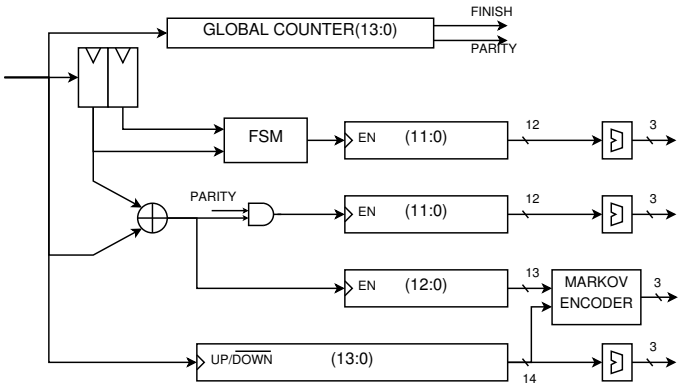

Fig. 5: Architecture overview of the hardware module containing all 4 tests running in parallel.

We note that, even though the Markov test can successfully estimate entropy for both types of failures that we tested, there are failures that are only detectable by other tests. For example, a sequence of alternating patterns of 2 zeros and 2 ones (00110011...) which can appear when oversampling the ring-oscillator based TRNG, will not cause failing the Markov test which reports the highest entropy level. However, the collision test and the partial collection test correctly estimate the entropy level as 0. Therefore, it is important to have a variety of tests because no single test can detect all possible weaknesses.

IV. Implementation and Result

A. Implementation

Figure 5 shows the general architecture of the implemented hardware module. The clock signal and some enable signals are omitted for better clarity. Input signals coming from the TRNG are the data bit and the data valid signal. The tests operate on bit sequences of length 2^{13}. The proposed design takes 15 clock cycles to calculate the min-entropy level.

The implementation boundaries of each individual test were removed to obtain a better unified hardware implementation. A global counter is used to indicate the end of the sequence and to provide the parity bit used by the partial collection test. The up/down counter result is used by both the frequency test and the Markov test. Collisions are counted using a 2-bit finite state machine.

Each test result, with the exception of the Markov test, is computed by comparing the available counter values with the precomputed boundaries corresponding to each level. The Markov test uses a 2-parameter function which requires a more complex encoder. The result obtained from the counters corresponds to a single location in the heat-map diagram and

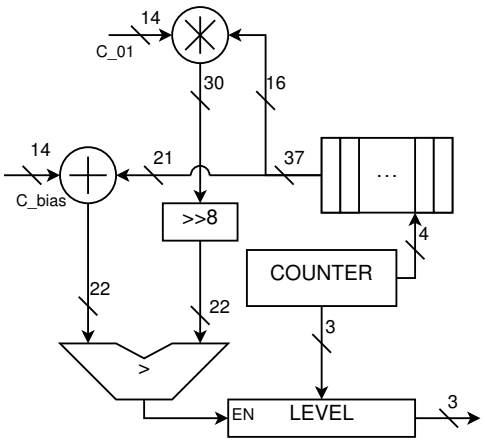

Fig. 6: Architecture of the encoder used in the Markov test.

TABLE I: Implementation Results

FPGA:				
Slices	FFs	LUTs	DSP48A1s	MaxFreq(MHz)
60	81	174	1	130
ASIC:				
Area(GE)			Energy/bit(pJ) @$1MHz$	
2394			1.4	

the min-entropy level is determined by finding the region on the diagram that contains this point. This is achieved by solving 15 linear equations of which each one requires one multiplication and one addition. The data-path used by the Markov encoder is shown in Figure 6. Coefficients of the linear equations are stored as 15 words of 37 bits. The comparison operation from Equation 1 was implemented as:

$$|C_1 - C_0| - n_i < C_{01} \cdot k_i \tag{2}$$

This implementation enables us to ignore the fractional part of the product $C_{01} \cdot k_i$ (the last 8 bits) which results in a more compact multiplier. The test goes through the precomputed constants, and reports the entropy level when it finds a match.

B. Result

We implemented our hardware designs in Verilog HDL and used Mentor Graphics Modelsim SE PLUS 6.6d for functional simulation. Our proposed hardware design in this paper was synthesized using Xilinx ISE14.7 on a Spartan-6 XC6SLX45 FPGA and Synopsys Design Compiler D-2010.03-SP4 applying UMC's $0.13\mu m$.1P8M Low Leakage Standard cell Library with the following typical values: voltage of $1.2V$ and temperature of $25\,^{\circ}\mathrm{C}$.

The implementation result is given in Table I. With the utilization of 60 slices and 1 DSP48A1 slice, our design has a maximum working frequency of 130MHz on FPGA. In other words, it can handle an input bit rate up to $130Mbit/s$, which is sufficient for most of the TRNGs on FPGA. Our design is also suitable for ASIC. The area consumption is 2394GE (Gate Equivalent). The most area consuming part is the multiplier in the Markov encoder. The power consumption is estimated at the gate level by PowerCompiler, based on the switching activities generated by a real testbench. The power strongly depends on the clock frequency and technology. In order to draw a fair comparison, we use energy per bit to represent the energy efficiency. The result shows that our design has a rather low energy consumption.

V. Conclusion

In this paper, we presented a compact hardware implementation of statistical tests suitable for monitoring non-IID entropy sources. We demonstrated that the presented tests are suitable for estimating the min-entropy of biased generators and generators with bit dependencies. ASIC implementations are very compact consuming around 2.4 kGE, and FPGA implementations on Xilinx Spartan-6 consume less than 1% of the available resources.

Since most entropy sources used in TRNGs don't produce a full-entropy output, the proposed testing module can be widely applied. Different tests detect different types of failures, therefore, test results can be used to detect exactly which type of failure has happened. This can be useful to distinguish active attacks from failures due to aging. Our future work will focus on exploring the failure conditions of different TRNGs in the presence of active attacks (through temperature changes, voltage drops or oversampling). The ultimate goal is to determine which tests are most suitable for each TRNG, and how the test results should be interpreted.

VI. Acknowledgement

This work was supported in part by the Research Council KU Leuven: GOA TENSE (GOA/11/007). In addition, this work is supported in part by the Flemish Government through FWO G.0550.12N, G.0130.13N and FWO G.0876.14N, the Hercules Foundation AKUL/11/19, and by a grant from Intel. In addition, this work was supported in part by the Scholarship from China Scholarship Council (No.201206210295).

References

[1] E. Barker and J. Kelsey, "Recommendation for the entropy sources used for random bitgeneration," ser. NIST DRAFT Special Publication 800-90B, 2012.

[2] W. Killmann and W. Schindler, "A proposal for: Functionality classes for random number generators," ser. BDI, Bonn, 2011.

[3] R. Santoro, O. Sentieys, and S. Roy, "On-line monitoring of random number generators for embedded security." in *ISCAS*, 2009, pp. 3050–3053.

[4] F. Veljković, V. Rožić, and I. Verbauwhede, "Low-cost implementations of on-the-fly tests for random number generators," in *DATE*, 2012, pp. 959–964.

[5] A. Vaskova, C. López-Ongil, E. San Millán, A. Jiménez-Horas, and L. Entrena, "Accelerating secure circuit design with hardware implementation of diehard battery of tests of randomness," in *IOLTS*, july 2011.

[6] V. B. Suresh, D. Antonioli, and W. P. Burleson, "On-chip lightweight implementation of reduced nist randomness test suite," in *HOST*, 2013, pp. 93–98.