High-Speed True Random Number Generation with Logic Gates Only

Markus Dichtl[1] and Jovan Dj. Golić[2]

[1] Siemens AG, Corporate Technology, Munich, Germany
markus.dichtl@siemens.com
[2] Telecom Italia, Security Innovation, Turin, Italy
jovan.golic@telecomitalia.it

Abstract. It is shown that the amount of true randomness produced by the recently introduced Galois and Fibonacci ring oscillators can be evaluated experimentally by restarting the oscillators from the same initial conditions and by examining the time evolution of the standard deviation of the oscillating signals. The restart approach is also applied to classical ring oscillators and the results obtained demonstrate that the new oscillators can achieve orders of magnitude higher entropy rates. A theoretical explanation is also provided. The restart and continuous modes of operation and a novel sampling method almost doubling the entropy rate are proposed. Accordingly, the new oscillators appear to be by far more effective than other known solutions for random number generation with logic gates only.

Key words: Random number generation, ring oscillators, generalized ring oscillators, logic gates, true randomness.

1 Introduction

Unpredictable random numbers are essential for the security of cryptographic algorithms and protocols and their implementations, especially for generating the underlying secret keys. Ideally, they should be truly random and hence unpredictable in terms of high entropy content even by an opponent with unlimited computational power. Practically, they may also be allowed to be only pseudo random and hence unpredictable by an opponent with a limited computational power, but then they contain low entropy and their unpredictability is intrinsically heuristic.

Digital true random number generators (TRNGs or RNGs), which can be implemented by using only logic gates in digital semiconductor technology, would be very practical in terms of cost effectiveness and flexibility, but are not sufficiently robust and are not able to produce high entropy rates. A common type of such RNGs utilizes unpredictable variations in the phase and frequency (jitter) of free-running oscillators implemented as ring oscillators, which are here also called classical ring oscillators. A ring oscillator consists of an odd number of logic inverters connected cyclically to form a ring. Typically, a high-frequency

ring oscillator is sampled at a much lower speed by an independent (system) clock through a D-type flip-flop. If the sampling clock is generated by another ring oscillator, then there is a tendency of the ring oscillators to couple with each other, thus significantly reducing the amount of randomness produced. Accordingly, it has been suggested to produce the clock by a slow, possibly external oscillator based on analog elements (e.g., see [7], [10], and [16]).

In [15], it is suggested to use ring oscillator signals to clock linear feedback shift registers (LFSRs) and then sample the produced output signal at a lower speed by the system clock, thus combining randomness with pseudo randomness. However, it is demonstrated in [4] that such a scheme is not secure in that the RNG sequence may be predictable by guessing the limited phase or frequency uncertainties and by solving the linear equations.

In [1], it is proposed to introduce a feedback signal for synchronizing the slow and fast ring oscillators so that the fast one is sampled close to its edges, i.e., transition points. This approach, which requires a considerable amount of hardware with very precise timing, may increase the sensitivity to phase jitter at the expense of introducing some statistical dependences. In [2], it is suggested to restart the two oscillators and the sampling D-type flip-flop from the same state, for each new random bit to be produced. Under a reasonable assumption regarding the absence of long-term correlations in the underlying noise process, this would ensure statistical independence of the random bits produced, but cannot increase the speed. We look forward to seeing experimental data showing how these methods work in practice.

Recently, a TRNG based on a multitude of ring oscillators combined by XOR logic gates was suggested in [14], but its security proof turns out to be based on highly unrealistic assumptions. The statistical results [13] for this design may be caused by pseudo random behavior and, hence, do not allow one to judge the amount of entropy produced.

Another type of digital RNGs exploits the metastability of RS latches and edge-triggered flip-flops based on RS latches such as the D-type flip-flop (e.g., see [6]). The metastability essentially results from an even number of logic inverters connected in a loop. For example, the input and clock signals for a D-type flip-flop can be produced by ring oscillators. Since the metastability events are relatively rare and are sensitive to manufacturing variations and temperature and voltage changes, the resulting designs are slow and not very reliable.

Two new types of ring oscillators called Fibonacci and Galois ring oscillators are proposed in [8] and it is suggested that much higher entropy rates can thus be achieved in comparison with other existing RNG proposals based on digital logic circuits only, even when implemented in FPGA technology. This would of course be of great practical interest, but no firm experimental evidence is provided, possibly due to the paradigm of mixing randomness with pseudo randomness. The main objective of this work is to evaluate and analyze the amount of true randomness produced by these oscillators. This is achieved by using the restart approach, which consists in repeating the experiments from identical starting

conditions.[1] In this way, it is practically possible to distinguish between true and pseudo randomness. In addition, the restart approach practically ensures mutual statistical independence of the random bits produced [2] and, as such, enables simple on-line testing of randomness properties. For comparison, similar experiments are also conducted for classical ring oscillators and a significant difference in performance is observed.

A short description of Fibonacci and Galois ring oscillators is provided in Section 2. The experimental results of the restart approach for distinguishing between true and pseudo randomness produced by these oscillators are presented in Section 3, whereas a comparison with classical ring oscillators is given in Section 4. Section 5 explains why the ring oscillator based TRNG designs from [14] and [13] fail. The TRNG designs resulting from Fibonacci and Galois ring oscillators including the restart and continuous modes of operation are proposed and discussed in Sections 6.1 and 6.2, respectively, a new sampling method almost doubling the entropy rate is introduced in Section 6.3, and the FPGA implementation details are given in Section 7. Section 8 contains a theoretical explanation of the improved true randomness and the conclusions are pointed out in Section 9.

2 Fibonacci and Galois Ring Oscillators

Fibonacci and Galois ring oscillators [8] (FIRO and GARO, respectively) are both defined as generalizations of a ring oscillator (RO). They consist of a number, r, of inverters connected in a cascade together with a number of XOR logic gates forming a feedback in an analogous way as in the well-known Fibonacci and Galois configurations of an LFSR (see Figures 1 and 2). The difference is that the delay synchronous units in an LFSR, i.e., synchronously clocked D-type flip-flops are replaced by the inverters. A FIRO or GARO is thus defined by the binary feedback coefficients or, equivalently, by the associated feedback polynomial $f(x) = \sum_{i=0}^{r} f_i x^i$, $f_0 = f_r = 1$. The output signal could be taken from any inverter in the cascade.

It is shown in [8] that to make sure that the inverter outputs cannot get stuck at a fixed state, the feedback polynomial should be chosen to have a form $f(x) = (1 + x)h(x)$, with $h(1) = 1$ for a FIRO and with r odd for a GARO. It is also suggested to choose a primitive polynomial $h(x)$, as then in both cases the state-transition diagram of the associated synchronously operated oscillator contains one long cycle of length $2^r - 2$ and one short cycle of length 2, which is metastable in the asynchronous operation.

It is claimed in [8] that the high-speed output oscillating signal has both pseudo and true randomness properties, where the latter result from unpredictable variations in the delay of internal logic gates which get propagated and enhanced through feedback, possibly in a chaotic manner, and also from internal metastability events. It is suggested that further randomness due to metastability may be induced within a sampling unit (e.g., a D-type flip-flop) as well as

[1] M. Dichtl used restart methods in TRNG simulations for certification since 2003.

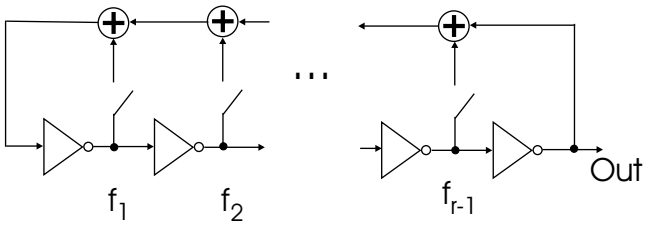

Fig. 1. Fibonacci ring oscillator.

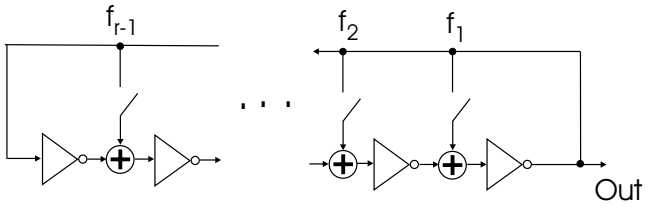

Fig. 2. Galois ring oscillator.

that the mutual coupling effect between the oscillating and sampling signals may be significantly reduced by the pseudo random noise-like form of the oscillating signal. To increase randomness and robustness, it is also proposed to use an XOR combination of a FIRO and a GARO (FIGARO).

3 Distinguishing between True and Pseudo Randomness

In order to assess the quality of TRNGs based on FIROs or GAROs, we need to distinguish the amount of true randomness contained in a pseudo random oscillating signal. We can do this by repeating the experiments from identical starting conditions, that is, by restarting a TRNG from the same initial states of all the logic gates. Pseudo randomness is deterministic and hence shows identical behavior in each repetition of the experiment. True randomness, on the other hand, behaves differently in repetitions, despite the identical starting conditions. To a minor extent, true randomness may also be present in the starting conditions, which are not ideally identical. We conducted experiments in the FPGA technology making sure that the initial conditions are essentially identical, with the all-zero state as the initial state. For implementation details, see Section 7.

As an example, Figure 3 shows the oscillograms of repeated restarts of a FIRO of length 15, from identical starting conditions. In the figure, the horizontal axis is the time, the period of time shown for each restart is 80 ns, the vertical axis is the output voltage, and only 25 curves of 1000 recorded are shown. The sampling rate on the oscilloscope was 20 Gsamples/s. It is clearly visible that many different curve forms occur in the figure. They are identical or similar only

in the beginning and then they diverge from each other surprisingly quickly. The FIRO thus produced true randomness in a form of random analog signals.

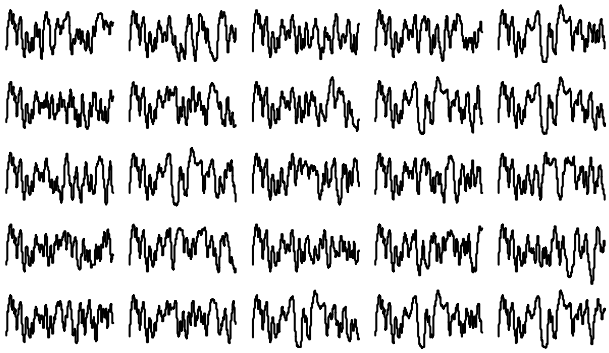

Fig. 3. Output voltages of 25 restarts, each 80 ns long, of a FIRO with feedback polynomial $x^{15} + x^{14} + x^7 + x^6 + x^5 + x^4 + x^2 + 1$. The occurrence of various curve forms shows that true randomness is produced.

The amount of randomness in the obtained curves that is relevant for entropy extraction by sampling can be measured by the standard deviation of the output voltage as a function of time. More precisely, if this standard deviation is relatively large, then extracting one bit of true randomness by sampling is easy and reliable. On the other hand, if this standard deviation is relatively small, then the extracted random bit will be heavily biased and the bias will strongly depend on the implementation. Accordingly, we computed the standard deviation and the mean value of the output voltage as functions of time for the 1000 curves, recorded for a longer period of time. The results are displayed in Figure 4, for the standard deviation, and in Figure 5, for the mean value.

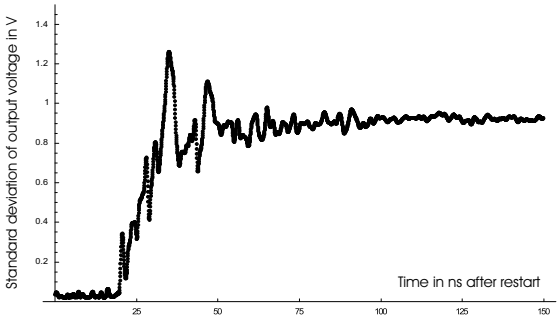

Fig. 4. Standard deviation of the output voltage of 1000 restarts of a FIRO with feedback polynomial $x^{15} + x^{14} + x^7 + x^6 + x^5 + x^4 + x^2 + 1$.

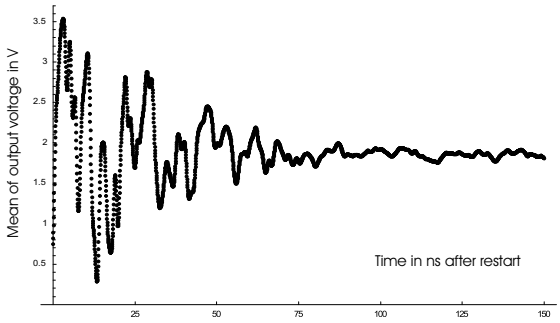

Fig. 5. Mean output voltage of 1000 restarts of a FIRO with feedback polynomial $x^{15} + x^{14} + x^7 + x^6 + x^5 + x^4 + x^2 + 1$.

In another example, analogous experiments were conducted for a GARO of length 31 and the results obtained are shown in Figures 6, 7, and 8.

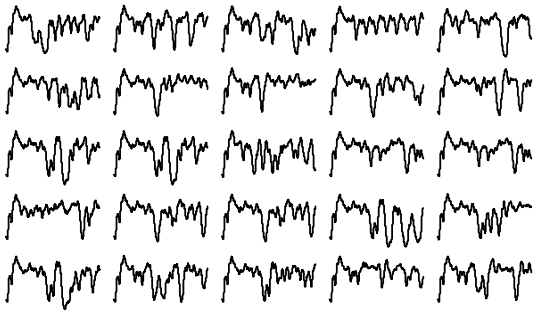

Fig. 6. Output voltages of 25 restarts, each 80 ns long, of a GARO with feedback polynomial $x^{31} + x^{27} + x^{23} + x^{21} + x^{20} + x^{17} + x^{16} + x^{15} + x^{13} + x^{10} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + x + 1$. The occurrence of various curve forms shows that true randomness is produced.

The obtained experimental results clearly show that both FIROs and GAROs are capable of producing true randomness. After about 25-30 ns, the standard deviation becomes significantly large to enable an extraction of 1 bit of entropy via sampling, at least in principle. After about 50 ns, as both the means and the standard deviations achieve relatively stable values, the entropy extraction becomes fairly robust and reliable. To be precise, these observations pertain to the restart mode of operation examined in the experiments. Similar observations also hold for a more random and more robust FIGARO.

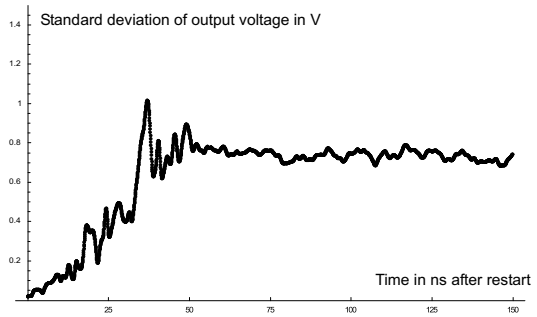

Fig. 7. Standard deviation of the output voltage of 1000 restarts of a GARO with feedback polynomial $x^{31} + x^{27} + x^{23} + x^{21} + x^{20} + x^{17} + x^{16} + x^{15} + x^{13} + x^{10} + x^9 + x^8 + x^6 + x^5 + x^4 + x^3 + x + 1$.

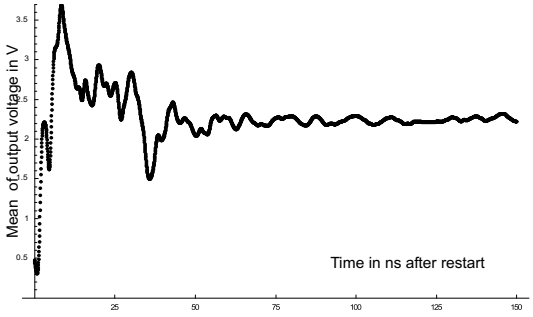

Fig. 8. Mean output voltage of 1000 restarts of a GARO with feedback polynomial $x^{31}+x^{27}+x^{23}+x^{21}+x^{20}+x^{17}+x^{16}+x^{15}+x^{13}+x^{10}+x^9+x^8+x^6+x^5+x^4+x^3+x+1$.

4 Comparison with Classical Ring Oscillators

In order to assess the practical suitability of FIROs and GAROs for the generation of true random numbers, we now compare them with a classical RO composed of three inverters implemented in the same FPGA technology. For the same reasons as in Section 3, in order to determine the amount of randomness generated by ROs, we use the restart approach. The frequency of the RO was about 296 MHz. We recorded the output voltage in the first 80 ns after restarting from the all-zero state, but the curves were so similar that no useful information about the phase jitter could be derived. Instead, we recorded a time frame from 490 to 510 ns after restarting, by sampling at a rate of 20 Gsamples/s, for 1000 restarts.

Figure 9 shows the first 100 of these curves in one plot. To get a numerical measure for the jitter, we also evaluated the 1000 curves statistically. We computed the average output voltage U_{av} over all 401000 samples, which was 1.7143 V. For each curve i, the time t_i is defined as the first time greater than 500 ns at which the output voltage was larger than U_{av}. Graphically, these times

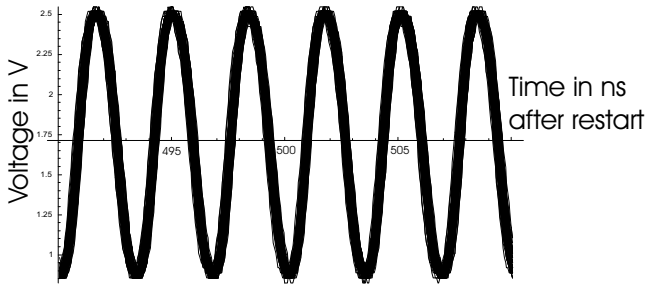

Fig. 9. Output voltages of 100 restarts of the RO. The vertical position of the horizontal axis is the mean voltage U_{av}.

can be seen in Figure 9 as the points to the right of the number 500, where the curves cut the horizontal axis. The minimum t_i from the 1000 curves was 500.7 ns, the maximum was 501.3 ns. The standard deviation of the 1000 t_is was 0.1005 ns. This is only about 3% of the period of the RO. This low standard deviation shows clearly that even after about 148 periods, the RO had accumulated only a very small amount of phase jitter, whereas the FIRO and GARO of Figures 3 and 6, respectively, started to produce very different curves after only 25-30 ns.

Consequently, the arithmetic mean and the standard deviation of the output voltages as functions of time after the restart of the RO are both computed for a much longer time frame. The graphs for the standard deviation and the mean value are shown in Figures 10 and 11, respectively. Zoomed in details from Figure 10 are shown in Figure 12. The observed oscillations are due to the fact that the variations of the output voltage are much larger around the edges of the oscillating signal.

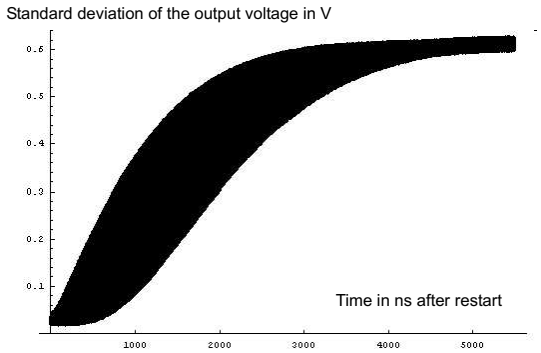

Fig. 10. Standard deviation the output voltage of 1000 restarts of the RO. The curve oscillates very much, so that it smears to the black area.

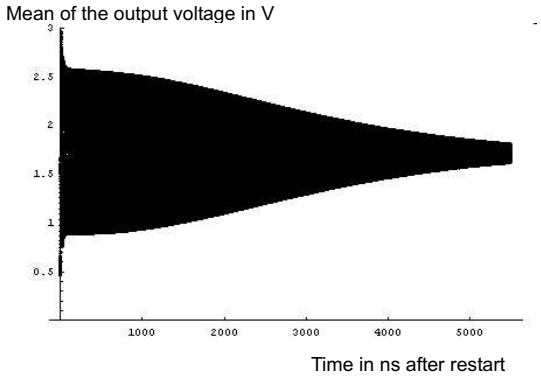

Fig. 11. Mean output voltage of 1000 restarts of the RO. The curve oscillates very much such that it smears to the black area.

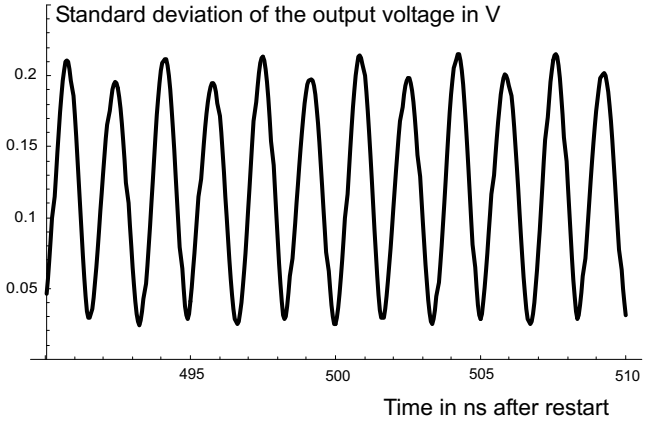

Fig. 12. Zoomed in detail of Figure 10.

These figures show very clearly that the classical ROs need more than 5 μs until they reach an approximately stable value of the standard deviation of the output voltage, that is, until the output voltage is in a completely random phase. In contrast, the FIROs and GAROs achieve a more or less stable standard deviation of their output voltages already after about 50 ns. Accordingly, the entropy rate achievable by FIROs, GAROs, and FIGAROs is orders of magnitude higher than that of classical ROs.

Moreover, one may extrapolate that similar conclusions also hold for the continuous mode of operation, without restarts from the same state, as the obtained results are independent of the initial state chosen. On the conservative side, the shown experimental results at least serve as a more or less firm indication for the achievable entropy rates of the continuous mode of operation.

5 A 'Provably Secure' TRNG Based on Ring Oscillators

Section 4 shows that a ring oscillator can produce randomness only at a relatively low rate. To overcome the problem, one may be tempted to use a very large number of ring oscillators instead of a single one. In [14], it is thus suggested to combine the outputs of a large number of ROs of equal length by an XOR operation, and then to sample the resulting signal and use the binary samples as inputs to a resilient postprocessing function. The resulting design has a large gate count and a high power consumption. For concrete implementations, it is suggested to use 114 ROs of length 13. The authors claim their design to be provably secure, with respect to the amount of true randomness produced.

The basic idea of the security proof is that transitions in the RO signals lead to transitions in the XOR output signal. If sampling occurs close enough to a jittering transition, then the sampling result is assumed to be random, as previously already suggested in [1]. One RO period is split up into 100 time slots of equal length. The sampled bit is considered random if the sampling occurs in a time slot with a transition. The probability of this happening is analyzed in an urn model.

However, it turns out that the security claim [14] is not justified, as its proof relies on several highly unrealistic assumptions. As such, the security proof [14] cannot be considered relevant. A criticism of the underlying assumptions is briefly presented in the sequel, whereas a detailed analysis of the TRNG design [14] is given in [5]. Note that the statistical results reported in [13], for another instance of the design using 210 ROs of length 3, provide no evidence that the design produces substantial amounts of true randomness, because a large number of ROs may also be a good source of pseudo randomness.

Unrealistic Probabilistic Model of Jitter The following assumption for an individual RO with average period T is stated in [14]. In any open time interval $(mT-T/4, mT+T/4)$, there is a unique point t where the signal crosses $(L+H)/2$ volts and this t behaves as a normally distributed random variable with mean mT and some variance σ^2. Here, L and H stand for the voltages that represent the logic low and high values, respectively. This assumption essentially means that a RO has a built-in perfect clock of period T and that jittering only occurs around the transition times of this perfect clock. This assumption is obviously very unrealistic. It would imply that the ROs cannot accumulate phase jitter, but Figure 10 shows clearly that this is not the case.

Interaction of Ring Oscillators In the urn model [14] for the transitions in the XOR output signal, it is assumed that the transitions in individual RO signals are uniformly and independently distributed among the chosen 100 time slots the period T is divided into. In [14], it is claimed that [3] shows that the phase drift is independent from one ring oscillator to another. However, no such result could be found in [3]. The whole paper analyzes jitter in individual ROs, and it never mentions having implemented two ROs on the same FPGA

simultaneously. Hence, [3] does not provide any insight into the statistical independence of transitions of several ROs implemented on the same chip. Our own experiments [5] show clearly that ROs implemented on the same FPGA interact strongly and are hence not statistically independent. Accordingly, as ROs implemented on the same chip interact strongly, it is not justified to assume that their transitions occur in statistically independent time slots.

Unrealistic Speed The security proof of [14] is implicitly based on the assumption that each transition (0-1 or 1-0) in each RO signal leads to a transition in the XOR output signal to be sampled. For the suggested design with 114 ROs of length 13, this implies that in the RO period of 26 gate delays, 228 transitions need to occur. This means 8.77 transitions per gate delay, independently of the gate technology used. This is not feasible with any technology known today.

The practical implementation from [13] has even much more severe speed problems. There, a 'robust' FPGA implementation of the design with 210 ROs of length 3, which oscillate at frequencies of about 333 MHz, is suggested. This means 70 transitions per gate delay or an average frequency of about 69.9 GHz in the XOR output signal.

Violation of Operating Conditions for Sampling Flip-Flop Even if the high-speed signal of the XOR of a large number of RO signals could be computed, it could not be sampled correctly. For flip-flops implemented in different technologies, the numerical values of the required setup- and hold-times vary, but no flip-flop can reliably sample signals with 8 transitions per gate delay. The Virtex II Pro FPGA used in [13] requires a signal to be sampled to be constant for 0.17 ns. During this time the XOR output signal would, if it could be computed, make about 23.8 transitions. The sampling flip-flop would thus be very far away from its specified operating conditions and, hence, cannot be assumed to work correctly.

Now, one might object that violating the required hold- or setup-times of a flip-flop can bring the flip-flop into a metastable state, which itself can be a source of randomness. Indeed, this is true, but whether metastability is really achieved depends to a large extent on small manufacturing variations and also on environmental conditions like supply voltage and temperature. Therefore, a security proof can hardly be based on the metastability of sampling flip-flops.

6 TRNGs Based on Fibonacci and Galois Ring Oscillators

By repeatedly restarting FIROs and GAROs, we have seen that they indeed generate true randomness. There are several ways of using this randomness in a practical random number generator that produces random bits sequentially.

6.1 Restart Mode of Operation

Since we have seen that a FIRO or GARO behaves differently each time even when restarted from identical starting conditions, we can use this restart method

also in the practical implementation of a TRNG. A FIRO or GARO is normally in a static reset state. Only when a random bit is needed, the oscillator is allowed to run for a short period of time. After sampling, the oscillator is stopped and reset to its initial state. A D-type flip-flop used for sampling should also be reset to a fixed state. An obvious advantage of the restart mode of operation is a low power consumption.

The main advantage of the restart method is that the bits generated in this way are statistically independent. More precisely, this is true under a reasonable assumption that, after restarting, there are no residual long-term statistical dependencies in the underlying noise process causing the true randomness. In fact, the long-term statistical dependences are very unlikely to exist also without restarting. This is very important for satisfying the evaluation criteria such as [11], as in this case the on-line testing reduces to statistically testing the bias of the bits generated. Instead of testing the bias, one may only apply an adaptive method for producing unbiased bits, such as the well-known von Neumann extractor, possibly in a faster generalized form [9]. So, if something goes wrong with the internal randomness, but not with the independence of repeated runs, then the output speed is thus automatically reduced, while keeping a true random output.

For this independence, however, it must be assumed that the starting state of the oscillator is independent of the bit generated previously. To achieve this independence for all the logic gates in the oscillator circuit, one has to wait a sufficiently long time after having stopped the oscillator, before restarting it. In this time, the oscillator can return to its static initial state. We discuss this waiting time in more detail in Section 7.

So, the independence is achieved at a cost of reducing the speed, because this waiting time has to be added to the running time guaranteeing a sufficiently large standard deviation of the output voltage (e.g., 25-50 ns), in order to obtain a lower bound on the sampling period. If the waiting time is sufficiently large, but the running time is too short, then the standard deviation of the output voltage becomes relatively low and, as a result, the output random bits have an increased bias, while remaining statistically independent.

If a D-type flip-flop is used for sampling, then the output bits produced may be biased, i.e., may have a deviation of the probability of zeros from 1/2. To get more balanced output bits, one may toggle the state of an intermediate flip-flop at each 0-1 transition in the oscillator signal and then sample the state when a random bit is needed. This is equivalent to counting the number of 0-1 transitions in the oscillating signal and using the count reduced modulo 2 as the output bit. The edge-triggered toggle flip-flop also has to be reset during the restart.

In our experiments, we managed to generate statistically independent random bits at a speed of 7.14 (6.25) Mbits/s, with a small bias of zeros of about 0.0162 (0.0056), by the sampling method with toggling and the FIRO used to generate Figure 3. In the implementation, the FIRO runs after the restart for 60 ns. Then it is stopped and the resulting bit is sent to and kept on an output line for 40 ns. The waiting time before restarting is 40 (60) ns. The independence is measured

by the chi-square statistical test comparing the empirical distribution of 142858 (125000) 4-bit blocks of successive bits with the theoretical distribution, with respect to a given bias, and in both cases the test was satisfied with a significance level of more than 10%.

6.2 Continuous Mode of Operation

One may run a FIRO, GARO, or FIGARO continuously and sample them when random bits are needed. Alternatively, one may restart them from a fixed state, as in the restart mode, each time a sequence of random bits is needed and then run them only as long as needed. The latter approach consumes less power and may imply statistical independence of successive runs if implemented properly. The two sampling methods described above for the restart mode of operation, namely, with or without an intermediate toggle flip-flop, are also applicable in this case. An XOR combination of a FIRO and GARO, FIGARO, together with an appropriately chosen sequential circuit for postprocessing are thus proposed in [8] for generating random bits at a high speed. A drawback of the continuous mode of operation relates to high-security applications where it is required to control the entropy rate by on-line testing, e.g., according to [11]. Namely, this appears to be a non-trivial task due to mixing true with pseudo randomness.

Another problem is determining the maximum sampling rate. To this end, one may refer to the restart method from a fixed initial state, but the corresponding results regarding the standard deviation of the output voltage should be taken with some caution, because the space of achievable (analog) internal states is larger than the space of the restart states. If the sampling period is chosen to be too short, then the successive samples produced may become statistically dependent. Namely, at each time, the statistical dependence of a current sample on the previous state, at a time when the preceding sample was produced, increases if the sample period decreases, whereas this previous state is clearly statistically dependent on the preceding sample. A statistical dependence among successive samples may then result as a consequence. Of course, by increasing the sampling period, such a statistical dependence diminishes. However, as statistical dependences may also result from a sampling D-type flip-flop, it may be prudent to always restart this flip-flop from the same state, for any sample produced.

The FIRO used to generate Figure 3 was allowed to run for 100 μs and its analog output was recorded. The autocorrelation function computed from this record is shown in Figure 13. It drops to about zero surprisingly quickly, that is, after about 20 ns, but this may also be due to a combined effect of true and pseudo randomness. We implemented this FIRO in the continuous mode and observed that at the speed of 25 Mbits/s, the statistics of the 4-tuples did not pass the chi-square independence test with respect to the significance level of 0.01%, but at 12.5 Mbits/s, the test was satisfied with a significance level of more than 10% and the bias of zeros was about 0.0192.

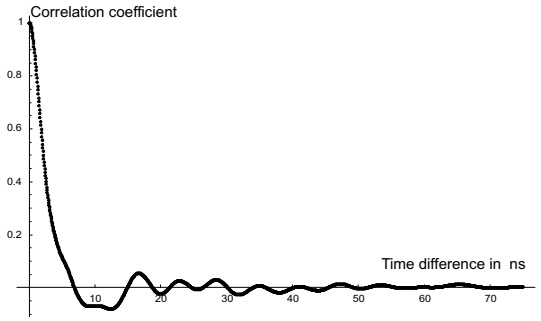

Fig. 13. Autocorrelation function of the analog output voltage for a continuously running FIRO with feedback polynomial $x^{15} + x^{14} + x^7 + x^6 + x^5 + x^4 + x^2 + 1$.

6.3 Almost Doubling the Entropy Rate

Instead of extracting one random bit at a time, by sampling with a D-type flip-flop, either with or without an intermediate toggle flip-flop, one may also extract two random bits at a time, by sampling with a D-type flip-flop, both with and without an intermediate toggle flip-flop. As these two bits result from two different, a sort of complementary properties of the oscillating signal, it is reasonable to expect that the statistical dependence between them is relatively weak. Namely, the bit sampled without toggling depends only on the signal value at the sampling time, while the bit sampled with toggling essentially depends on the number of transitions in the signal since the last restart, for the restart mode, or since the preceding sampling time, for the continuous mode. If their biases are both small or, more generally, comparable in magnitude, then the entropy rate could thus be almost doubled, which, of course, would be practically significant.

We checked experimentally how much the entropy rate can be increased by this approach. For example, with the same FIRO as above, running in the restart mode of operation, the speed is thus increased from 7.14 to 14.28 Mbits/s, for raw random data. The Shannon entropy estimates obtained on 571432 2-bit samples are about 0.987 and 0.961 for individual bits, where the higher entropy corresponds to the sampling method with toggling, and about 1.933 for both bits jointly. The mutual information measuring their statistical independence is thus quite low, i.e., about 0.015.

Thus, we get a theoretical output rate of 13.8 Mbits/s of unbiased and statistically independent random bits. In practice, this rate can approximately be achieved by postprocessing algorithms, which should take into account the statistical dependence, albeit weak, between the two bits obtained by sampling. For example, by using the algorithm from [9], the theoretical output rate can be approached at the cost of increased processing complexity, by increasing the number of 2-bit samples processed simultaneously. The restart mode of operation is especially suitable for this algorithm since the processed 2-bit samples are then statistically independent.

7 FPGA Implementation

In this section, we give more details on how the experimental results presented in previous sections are achieved. The FIROs and GAROs were experimentally tested by using a Xilinx Spartan-3 Starter Kit board based on the Xilinx FPGA XC3S200-4FT256C.

In our experiments, we observed a considerable cross-talk between different signals on the FPGA. In principle, this is a problem for the restart method since the random oscillating signals generated by a FIRO or GARO are intrinsically analog, as is clear from Figures 3 and 6. These analog signals can be disturbed very easily by analog cross-talk from other signals on the board.

We implemented the circuit on the FPGA very carefully in such a way that the oscillations of the FIROs and GAROs were not disturbed by other signals. The cleanest approach would be to have no other signals on the FPGA. However, there has to be some mechanism for timing when the oscillators have to be restarted periodically. We used a quartz clock available on the board and a counter implemented on the FPGA. There may have been analog cross-talk from the counter to a FIRO or GARO, but this does not invalidate our experimental proof that true randomness was really generated. The counter was designed in such a way that it followed the same sequence of states for each restart and run of a FIRO or GARO. So, if the counter influences a FIRO or GARO, then it does it in an identical way at each run of the FIRO or GARO. Therefore, the occurrence of a varying behavior of the oscillators cannot be attributed to pseudo random disturbances from the counter, but is caused by true randomness.

The FIROs and GAROs may not only be influenced by different signals on the FPGA, but also by their own state from the previous run. The only way to solve this problem is to keep the oscillators, after having stopped them, for a sufficiently long time in a constant state so that all the transitory voltages can settle down to a constant value. Of course, we would theoretically have to wait infinitely long, because of an exponential decay. Since the timing analysis of FPGA implementations can be very complex, it is difficult to give precise estimates for the waiting times required. The signal stopping the oscillators may ripple through several logic gates, especially in the implementation of the many-input XOR needed for FIROs. Since the gate delay for logic functions on the FPGA is about 1 ns, about 10 ns are sufficient to account for the gate delay. From our observations of FPGA transitory voltages on the oscilloscope, we concluded that additional 20 ns were sufficient for residual voltages to settle down to such a low value that they do not have noticeable influence on subsequent restarts of the oscillators. For the experiments reported in Sections 3 and 4, to be on the safe side, we chose a waiting time of 4960 ns.

8 A Theoretical Rationale for Improved Randomness

Why do FIROs and GAROs perform so much better than classical ROs? Here we provide a number of theoretical reasons for this phenomenon. Of course, it

remains to be further investigated if a more precise theoretical analysis would be possible.

The primary source of randomness are random delays and transition times of the logic gates in the circuit, which are due to various internal and external noise factors such as thermal noise and unpredictable short-term or long-term fluctuations in voltage and temperature. The amount of primary randomness generated per time unit can thus be measured by the product of the total number of logic gates and their average switching frequency, and this product is roughly proportional to the power consumption. In a classical RO, this product is independent of the number of inverters used, as the average switching frequency is inversely proportional to this number. On the other hand, in a FIRO or GARO, this product increases as the number of inverters, r, or the number of feedback logic gates increases. This is because the average switching frequency does not decrease with r, due to a more complex feedback. Accordingly, a FIRO or GARO generates more primary randomness than a RO. *Equivalently, one may say that the amount of phase jitter is thus effectively increased.*

During the oscillations in a FIRO or GARO, additional true randomness may be generated due to internal metastability events resulting from the feedback loops involving chains of inverters, but the frequency of these events is difficult to estimate and the resulting impact on entropy rate is hence difficult to quantify.

Another and, perhaps, the main advantage of FIROs and GAROs over classical ROs, which is evident from the oscillating waveforms shown in Figures 3, 6, and 9, is also a consequence of a more complex feedback signal. Namely, each random variation of a delay or a transition time gets transformed and propagated through feedback logic gates in a pseudo random or chaotic manner, and all such random variations combined hence result in a high-frequency noise-like oscillating signal, which inherently possesses both analog and binary properties. So, the more complex feedback cannot introduce new randomness as such, but can and does transform the primary randomness produced by individual logic gates, including those in the feedback, into a form more suitable for extraction by sampling. *Equivalently, one may say that the sensitivity to phase jitter is thus effectively increased.*

In a classical RO, the random delay variations of inverters just add up together in a regular manner so that it is much more difficult to extract each new bit of true randomness by sampling. A theoretical model of entropy build-up in ROs is given in [12]. Note that the oscillating nature of the standard deviation curves in Figures 10 and 12 means that it is in principle easier to extract randomness by sampling near the edges of the oscillating signal. However, this is difficult to implement in practice, and [1] is as a step in this direction.

In conclusion, the sampling frequency can be made much higher without essentially reducing the entropy per bit in the sampled sequence, and this results in a much higher entropy rate achievable by FIROs, GAROs, and FIGAROs, in comparison with classical ROs.

In addition, in a FIRO or GARO, the irregularity of a high-frequency oscillating signal, which is random, pseudo random, and chaotic on the binary as

well as analog level, reduces the mutual coupling effect between the oscillating and sampling signals, which is the main weakness of classical ring oscillators. This irregularity may also increase the frequency of metastability events in the sampling circuit such as an edge-triggered D-type flip-flop. The two phonemena and the resulting impact on true randomness are interesting topics for future experimental investigations.

9 Conclusions

We demonstrated that a carefully implemented restart method is useful not only for designing TRNGs with testable true randomness properties, but also for distinguishing between true and pseudo randomness in TRNGs using logic gates only, such as those based on classical ring oscillators and on the so-called Fibonacci or Galois ring oscillators. The experimental evaluation and analysis based on the restart method clearly show that the latter are capable of producing orders of magnitude higher entropy rates than the former. This is mainly because a more complex feedback, on one hand, maintains a high switching frequency while increasing the number of inverters and, on the other hand, transforms the original randomness into a form more suitable for extraction by sampling.

Consequently, TRNGs based on Fibonacci or Galois ring oscillators are thus very convenient for high-speed applications, in both FPGA and ASIC technologies. The restart mode of operation is recommended for high-security applications, with an on-line testing of true randomness properties. The continuous mode of operation can achieve higher speeds, but the true randomness properties do not seem to be directly testable. A new sampling method almost doubling the entropy rate is also proposed.

References

1. H. Bock, M. Bucci, and R. Luzzi, "Offset-compensated oscillator-based random bit source for security applications," Cryptographic Hardware and Embedded Systems - CHES 2004, *Lecture Notes in Computer Science*, vol. 3156, pp. 268–281, 2004.
2. M. Bucci and R. Luzzi, "Design of testable random bit generators," Cryptographic Hardware and Embedded Systems - CHES 2005, *Lecture Notes in Computer Science*, vol. 3659, pp. 147–156, 2005.
3. W. R. Coppock and C. R. Philbrook, "A mathematical and physical analysis of circuit jitter with application to cryptographic random bit generation," Worcester Polytechnic Inst., Major Qualifying Project Report, Apr. 2005.
4. M. Dichtl, "How to predict the output of a hardware random number generator," Cryptographic Hardware and Embedded Systems - CHES 2003, *Lecture Notes in Computer Science*, vol. 2779, pp. 181–188, 2003.
5. M. Dichtl, "A closer look at a provably secure true random number generator," unpublished paper, submitted to and rejected from CHES 2007.
6. M. Epstein, L. Hars, R. Krasinski, M. Rosner, and H. Zheng, "Design and implementation of a true random number generator based on digital circuits artifacts," Cryptographic Hardware and Embedded Systems - CHES 2003, *Lecture Notes in Computer Science*, vol. 2779, pp. 152–165, 2003.

7. R. C. Fairfield, R. L. Mortenson, and K. B. Coulthart, "An LSI random number generator (RNG)," Advances in Cryptology - Crypto '84, *Lecture Notes in Computer Science*, vol. 196, pp. 203–230, 1985.
8. J. Dj. Golić, "New methods for digital generation and postprocessing of random data," *IEEE Trans. Computers*, vol. 55(10), pp. 1217–1229, Oct. 2006.
9. A. Juels, M. Jakobsson, E. Shriver, and B. K. Hillyer, "How to turn loaded dice into fair coins," *IEEE Trans. Information Theory*, vol. 46(3), pp. 911–921, May 2000.
10. B. Jun and P. Kocher, "The Intel random number generator," White paper for Intel Corporation, Cryptography Research Inc., Apr. 1999, available at http://www.cryptography.com/resources/whitepapers/IntelRNG.pdf.
11. W. Killmann and W. Schindler, *AIS 31: Functionality Classes and Evaluation Methodology for True (Physical) Random Number Generators*, version 3.1, Bundesamt fur Sicherheit in der Informationstechnik (BSI), Bonn, 2001.
12. W. Schindler, "A stochastical model and its analysis for a physical random number generator presented at CHES 2002," Cryptography and Coding - 9th IMA International Conference, *Lecture Notes in Computer Science*, vol. 2898, pp. 276–289, 2003.
13. D. Schellekens, B. Preneel, and I. Verbauwhede, "FPGA vendor agnostic true random number generator," *Proc. 16th Int. Conf. Field Programmable Logic and Applications - FPL 2006*, Aug. 2006, to appear.
14. B. Sunar, W. Martin, and D. Stinson, "A provably secure true random number generator with built-in tolerance to active attacks," *IEEE Trans. Computers*, vol. 56(1), pp. 109–119, Jan. 2007.
15. T. E. Tkacik, "A hardware random number generator," Cryptographic Hardware and Embedded Systems - CHES 2002, *Lecture Notes in Computer Science*, vol. 2523, pp. 450–453, 2002.
16. K. S. Tsoi, K. H. Leung, and P. H. W. Leong, "Compact FPGA-based true and pseudo random number generators," *Proc. 11th IEEE Annual Symposium on Field-Programmable Custom Computing Machines*, p. 51, Apr. 2003.