New Paradigms for Digital Generation and Post-Processing of Random Data[*]

Jovan Dj. Golić
Access Network and Terminals
Telecom Italia Lab, Telecom Italia
Via G. Reiss Romoli 274, 10148 Turin, Italy
jovan.golic@tilab.com

Abstract

A new method for digital true random number generation based on asynchronous logic circuits with feedback is introduced. In particular, a concrete technique using the so-called Fibonacci and Galois ring oscillators is developed and experimentally tested in FPGA technology. The generated random binary sequences inherently have a high speed and a very high and robust entropy rate in comparison with previous proposals for digital random number generators. A new method for digital post-processing of random data based on non-autonomous synchronous logic circuits with feedback is also introduced and a concrete technique using a self-clock-controlled linear feedback shift register is proposed. The post-processing can provide both randomness extraction and computationally secure speed increase of input random data.

Key words. True random number generators, asynchronous circuits with feedback, digital free-running oscillators, clock-controlled syhchronous circuits.

1 Introduction

In many applications of computers and other electronic devices there is a need for a physical source of true random numbers, for example, in computer simulations of various probabilistic algorithms, computer games, and, most notably, in cryptographic algorithms and protocols whose security relies on the ability to generate unpredictable secret keys. In addition, high-speed truly random sequences are needed for countermeasures against side-channel attacks on microelectronic devices such as the integrated chip cards. For example, such countermeasures include random masking of cryptographic functions as well as generating secret keys for the encryption of internal links and memories in such devices.

The output of a random number generator (RNG) is typically a binary sequence which is required to be unpredictable in the information-theoretic sense. Equivalently, the RNG

[*] The contributions of this paper are covered by two patent applications by Telecom Italia S.p.A..

output should ideally be statistically modeled as a purely random sequence, that is, as a sequence of mutually independent uniformly distributed binary random variables (bits), with the maximal possible entropy per bit. A necessary condition is that it should be computationally infeasible to distinguish the RNG output sequence from a purely random sequence and, in particular, to predict the RNG output sequence. These conditions are also necessary for pseudorandom number generators (PRNGs), which produce pseudorandom sequences from a random seed according to a state-transition function updating their internal state. A PRNG is called computationally secure if its output sequence is computationally unpredictable.

1.1 Previous Work on Digital RNGs

For practical reasons, RNGs implemented in the solid-state, semiconductor technology are preferable. A conventional type of hardware-based RNGs exploit thermal noise in resistors and/or shot noise in PN-junctions, but include analog elements and are hence not cost effective for many applications. Another class of RNG's exploit analog voltage-controlled oscillators (for example, see [5]). A more practical type of RNGs can be implemented by digital integrated circuits only. They are usually based on unpredictable variations in the phase and frequency jitter of free-running oscillators implemented as ring oscillators, where a ring oscillator consists of an odd number of inverter logic gates connected together to form a ring (for example, see [3], [2], and [11]). Typically, such RNGs consist of a number of ring oscillators that are combined linearly, by using an XOR operation and then sampled at a much lower speed by an independent (system) clock through a D-type flip-flop. The main disadvantages of this type of RNGs are a low speed and a tendency of the ring oscillators to couple with each other and with the system clock too, thus reducing the amount of randomness produced.

Another type of digital RNGs exploit the metastability of RS latches and edge-triggered flip-flops based on RS latches such as the D-type flip-flop (for example, see [8]). Namely, the output of such a flip-flop may become unpredictable if the input and clock signals are such that the setup and/or hold times are violated, for example, if the data input signal is forced to change at nearly the same time as the clock signal. The output signal then stabilizes on a random, typically biased value after a random amount of time. The metastability of D-type flip-flops can possibly be exploited together with the jitter of the underlying ring oscillator signals by using D-type flip-flops for sampling the ring oscillator signals. In any case, the metastability events are relatively rare and are sensitive to temperature and voltage changes [8], so that the RNGs solely based on metastability are relatively slow and do not appear to be very reliable.

Accordingly, in spite of many (patented) proposals for hardware-based RNGs, finding an efficient and robust method for high-speed generation of true random numbers that can be implemented by using only logic gates in digital semiconductor technology still remains a challenge. The method should be efficient in terms of the gate count, achievable speed, and power consumption, and should preferably be suitable for integration in both FPGA (field programmable gate array) and ASIC (application specific integrated circuit) technologies.

1.2 Previous Work on Digital Post-Processing

Typically, every true RNG essentially consists of two parts, namely, a physical source of randomness producing a raw binary sequence which is random, but not purely random, and a post-processing part, possibly integrated in the physical source of randomness, which produces the final output sequence. For example, ring oscillator signals can be used to clock linear feedback shift registers (LFSRs) [12] or can be XOR-ed with data inputs to particular flip-flops in LFSRs [3], [2], [11] (or in linear congruential generators [9]). In the former case, the output signal produced by LFSRs should be sampled at a lower speed by the system clock, while in the latter case, the slower system clock should be used to clock the LFSRs. In both cases, randomness is combined with pseudorandomness, and in the latter case, new randomness can possibly be introduced by effectively sampling multiple ring oscillator signals at various points in the LFSR circuit. However, as demonstrated in [4], if the ring oscillator signals are used only to clock linear circuits, then the RNG sequence may be predictable by guessing the limited phase or frequency uncertainties and by solving the linear equations.

If the raw binary sequence has a high speed, then it typically has a statistical weakness that consists in the bias of individual bits, meaning that the probabilities of the output values, 0 and 1, are not equal, and/or in the correlation of bits that are relatively close to each other in time. The main objective of the post-processing part is to eliminate statistical weaknesses from the raw binary sequence in the information-theoretic or only computational sense. In the first case, each output bit should approximately convey one bit of information or entropy, and this is possible achieve only by reducing the speed. Such post-processing is sometimes referred to as randomness extraction. In the second case, the output sequence does not have to be purely random, but should be computationally difficult or infeasible to distinguish from a purely random sequence. In this case, the speed does not have to be reduced.

Another important objective of post-processing is to provide robustness of the statistical properties of the RNG output sequence with respect to changes in temperature and other environmental conditions, and also with respect to physical attacks on the physical source of randomness. In particular, one may also wish to deal with the case when the raw binary sequence is heavily biased. If the post-processing is adaptive, this would then imply that the output speed varies and that it can be reduced significantly. Alternatively, one may keep the same speed, but the RNG output sequence may then be unpredictable only in the computational rather than information-theoretic sense.

Randomness extraction techniques can be based on linear transformations only. However, they may become insecure if the speed reduction is not sufficiently large. A well-known block linear transformation is one used in the von Neyman extractor, which, together with decimation, removes the bias from a raw binary sequence, provided that the input bits are statistically independent. The usage of more general block linear transformations is suggested and analyzed in [1]. The technique is based on a randomly chosen and fixed linear function, which is applied to blocks of input data, taken from any RNG, to produce reduced-size blocks of output data. As the blocks have to be long, the implementation cost in terms of the gate count is considerable. By using some known results on the so-called universal hash functions, it is proven that under certain conditions almost all such functions give rise to nearly uniform distributions of the corresponding blocks of output data. However, one can argue that this result is not very meaningful for a fixed linear function and also that a nearly uniform distribution of output blocks does not imply a nearly uniform distribution of every (linear) transformation of output blocks. Further, this sort of randomness extraction is not sufficiently robust, namely, it does not provide computational unpredictability of output data if the entropy of input data is significantly reduced by the adversary's action on the underlying random number generator. To achieve robustness and computational security, the post-processing technique should also incorporate nonlinear transformations.

The entropy rate of raw binary sequences generated by digital circuits usually is not high. A well-known technique for producing a high-speed (pseudo)random sequence is to apply a randomness extractor to the raw binary sequence thus reducing its speed and then to use this sequence to produce a random seed for a PRNG, which is then clocked at a high speed. Such a PRNG should be computationally secure and hence relatively complex. For example, it can be based on iterated cryptographic hash functions.

1.3 Main Objectives and Results

The first objective of this paper is to introduce a new method and the corresponding techniques for generating high-speed and high-entropy raw binary sequences by using only logic gates in digital semiconductor technology. A purely random RNG output sequence can then be obtained by further post-processing of the generated raw binary sequence. The second objective of this paper is to introduce a new method and the corresponding techniques for digital post-processing of raw binary sequences that provide randomness extraction, computational security, as well as an increase in speed if required. Both methods should be implementable by standard digital library units and practically independent of the fabrication technology.

The main principle of the proposed general method for digital random number generation is to replace the standard ring oscillators by more complex asynchronous logic circuits with feedback and thus produce high-speed pseudorandom sequences with much more inherent randomness due to more complex feedback signals. To guarantee oscillations, the state-transition function of the corresponding autonomous finite-state machine is required not to have any fixed points. Other conditions regarding the randomness and pseudorandomness properties of the output signal may be imposed too. In particular, two practically important generalizations of the ring oscillator structure called the Fibonacci and Galois ring oscillators are developed. As the generated high-speed oscillating signal has randomness on the digital as well as analog level, the independent sampling signal can have a similar frequency thus forcing the sampling D-type flip-flop into frequent metastability conditions and hence producing a high-speed raw binary sequence with a high entropy rate, unlike any other previously proposed purely digital RNG. The technique is implemented in FPGA technology and experimentally tested.

The main principle of the proposed general method for digital post-processing of random data is to take any computationally secure PRNG, which is typically a synchronous logic circuit with feedback implementing an autonomous finite-state machine, and operate it in a non-autonomous mode with an input taken from the physical source of randomness. The input should be incorporated so as to satisfy the condition that, for each initial state, the output sequence of the PRNG should be purely random if the input sequence is purely random. In the binary case, the input can be introduced into the PRNG through XOR logic gates in a similar way as the plaintext input for stream ciphers with plaintext memory [6]. This is in spirit similar to the data scrambling technique used in digital communications and is a generalization of previously proposed linear techniques (see also [9]).

It is proposed that the clock frequency for the PRNG is the same as or higher than the speed of the raw binary sequence, in bits per second, and that the speed reduction, for randomness extraction, is achieved through self-clock-controlled irregular clocking of the PRNG. Note that a higher clock frequency means that some of the input bits are effectively repeated. Irregular clocking of the PRNG means that some of its output bits are discarded. Altogether, the output speed can be higher than the input speed if the clock frequency is sufficiently high, without a need to use a separate randomness extraction technique. In a concrete proposal, the PRNG is chosen to be a self-clock-controlled LFSR in the so-called Galois configuration, which is simple to be implemented in hardware and yet practically secure in a sense that its output sequence cannot be easily predicted in the autonomous mode of operation.

The general method for digital random number generation based on asynchronous logic circuits with feedback is presented in Section 2, whereas the concrete techniques introducing the Fibonacci and Galois ring oscillators are proposed in Section 3. The general method for digital post-processing of random data based on non-autonomous synchronous logic circuits with feedback is introduced in Section 4 and a concrete technique using a self-clock-controlled LFSR is described in Section 5. Conclusions are given in Section 6.

2 Asynchronous Circuits with Feedback

A generic circuit for generating a raw oscillating signal that possesses the properties of pseudorandomness and randomness is shown in Fig. 1. A logic circuit implementing the state-transition function of an autonomous finite-state machine without fixed points is operated asynchronously, without a clock, by feeding back the outputs, representing the next state, to the respective inputs, representing the current state. In this asynchronous operation, the identity function should be implemented by a delay element, that is, by a cascade of an even number of inverters. The condition that there are no fixed points guarantees that the asynchronous logic circuit with feedback cannot get stuck in a fixed state and thus output a constant sequence. The state sequence is thus ensured to oscillate among a number of states. The output signal is generated by applying another logic circuit implementing the output function of the autonomous finite-state machine to the outputs of the first logic circuit producing the state sequence. The output function should effectively depend on the parts of the internal state that cannot get stuck at a fixed value.

The amounts of pseudorandomness and randomness contained in the oscillating signal as well as their robustness depend on the asynchronous logic circuit used. To this end, one may impose a further condition that the state-transition diagram contains only long cycles and metastable short cycles, so that the state sequence cannot in practice get stuck in one of the latter. Also, the numbers of 0's and 1's per cycle should be approximately balanced.

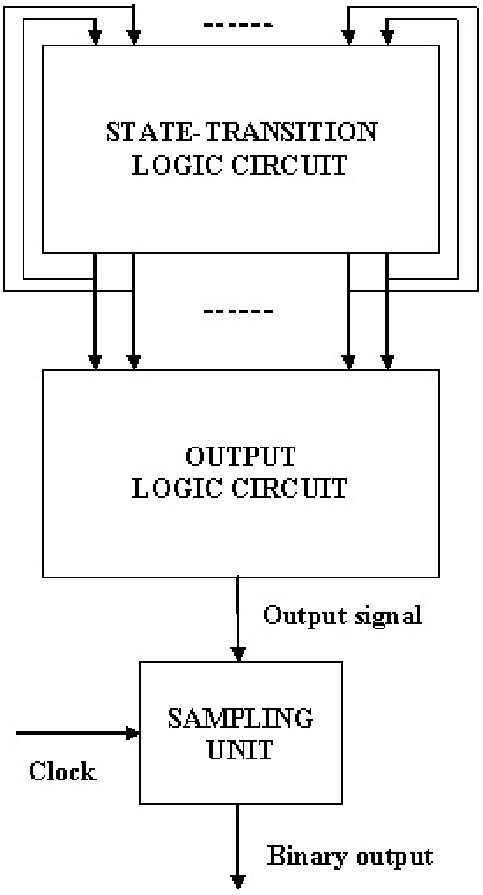

Figure 1: A generic asynchronous circuit with feedback.

The output signal is expected to have pseudorandom properties similar to those corresponding to the synchronous operation, but also random properties due to unpredictable variations in the delay of internal logic gates which get propagated and enhanced through feedback and possibly also due to internal metastability events. The delay variations are due to various internal and external noise factors possibly through unpredictable fluctuations in voltage and temperature. If the delays of the internal logic gates are not long, then the output signal should have a high speed and should exhibit randomness also on the analog level, as many transitions cannot be completed to the full digital level. Since the oscillating signal is then a sort of analog noise, further randomness due to metastability is induced within a sampling unit (e.g., implemented as a D-type flip-flop), especially if the clock signal, possibly generated by an independent ring oscillator, has a similar frequency as the oscillating signal. The mutual coupling effect is eliminated by the pseudorandomness of the oscillating signal. The output sequence obtained after sampling is the desired raw binary sequence. It can have a very high speed and can contain a high amount of entropy per bit. The final purely random output sequence can be obtained by further post-processing to remove the residual bias and other redundancy such as correlation, at the expense of a somewhat reduced output speed.

3 Fibonacci and Galois Ring Oscillators

The general principle introduced in Section 2 does not directly yield practical constructions that guarantee good randomness and pseudorandomness properties of the output signal. In the sequel, we propose two practically important concretizations which can be regarded as generalizations of the ring oscillator structure. Both generalizations are based on a cascade of inverters and essentially consist in replacing the simple circular feedback defining a ring oscillator by a more complex feedback incorporating a number of XOR gates in a way corresponding to the well-known Fibonacci or Galois configurations of an LFSR. Note that the inverter gates are used instead of the delay elements. The feedback connections are chosen so as to satisfy the basic condition that there are no fixed points in the corresponding state-transition function. It turns out that such feedback connections can be characterized nicely. The delays of individual inverters can be adjusted by composing each of them as a cascade of an odd number of elementary inverter logic gates. If the number of inverters is only one, then both generalizations reduce to the conventional ring oscillator.

Accordingly, the so-called Fibonacci ring oscillator, shown in Fig. 2, consists of a number, r, of inverters connected in a cascade so that the output of each but the last inverter is directly used as the input to the next inverter. The feedback connections are specified by the binary coefficients f_i with the convention that the corresponding switch is closed if $f_i = 1$ and is open if $f_i = 0$, in which case the corresponding 2-input XOR gate is not present. The feedback coefficients and the corresponding inverters are indexed from left to right. The output of the last inverter together with the outputs of the preceding inverters specified by the coefficients f_i being equal to 1 are then XOR-ed together by using, for example, 2-input XOR gates to form the feedback signal defining the input to the first inverter in the cascade. The output signal could be taken from any inverter in the cascade.

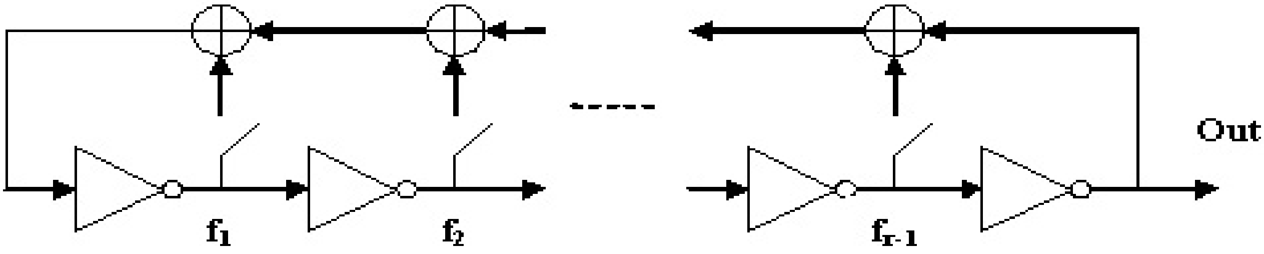

Figure 2: Fibonacci ring oscillator.

For both the configurations, it is convenient to represent the feedback coefficients by a binary polynomial $f(x) = \sum_{i=0}^{r} f_i x^i$, $f_0 = f_r = 1$, called the feedback polynomial.

Theorem 1 *A Fibonacci ring oscillator does not have a fixed point if and only if*

$$f(x) = (1+x)h(x) \quad \text{and} \quad h(1) = 1. \tag{1}$$

The condition (1) means that $f(x)$ is divisible by $1+x$, i.e., $f(1) = 0$, and that the quotient polynomial $h(x)$ is itself not divisible by $1+x$. It is interesting that the degree r of $f(x)$ can be odd or even, but it is necessary that $r \neq 2$.

Theorem 2 *The state-transition diagram of a synchronously operated Fibonacci ring oscillator satisfying the condition* (1) *contains one short cycle of length* 2*, composed of the all-zero and all-one states, and a number of longer cycles whose lengths depend on the polynomial* $h(x)$*. If* $h(x)$ *is a primitive polynomial, i.e., an irreducible polynomial with the maximal period* $2^{r-1} - 1$*, then there is only one long cycle of length* $2^r - 2$*.*

The choice of a primitive polynomial $h(x)$ is recommended to ensure good pseudorandom properties of the output signal. It follows that the short cycle is metastable in the asynchronous operation, which means that in practice the state sequence can spend only a very short time in this cycle.

The so-called Galois ring oscillator, shown in Fig. 3, consists of a number, r, of inverters connected in a cascade so that the output of each but the last inverter is used to form the input to the next inverter and the output of the last inverter directly defines the feedback signal. The feedback connections are specified by the binary coefficients f_i with the convention that the corresponding switch is closed if $f_i = 1$ and is open if $f_i = 0$, in which case the corresponding 2-input XOR gate is not present. The feedback coefficients and the corresponding inverters are indexed from right to left. The input to the first inverter in the cascade is directly defined by the feedback signal. If $f_i = 0$, then the input to the i-th inverter is directly defined by the output of the $(i+1)$-th inverter. If $f_i = 1$, then the input to the i-th inverter is formed by XOR-ing the output of the $(i+1)$-th inverter with the feedback signal. In Fig. 3, the output signal is taken from the last inverter in the cascade.

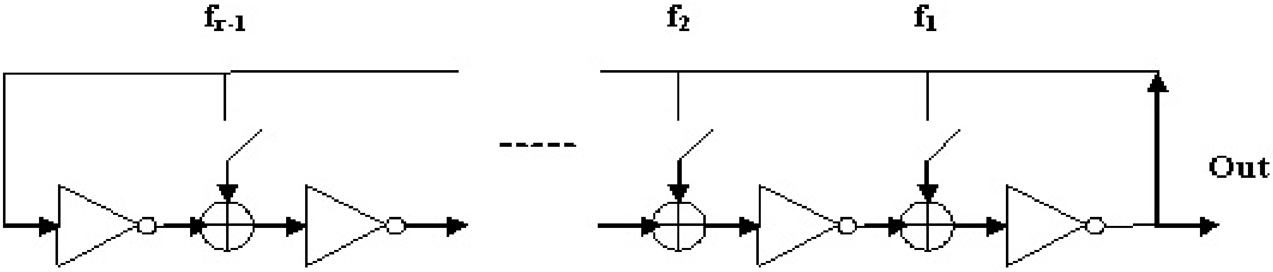

Figure 3: Galois ring oscillator.

Theorem 3 *A Galois ring oscillator does not have a fixed point if and only if*

$$f(1) = 0 \quad \text{and} \quad r \text{ is odd.} \tag{2}$$

The condition (2) means that $f(x) = (1 + x)h(x)$ and that the degree of $f(x)$ is odd.

Theorem 4 *The state-transition diagram of a synchronously operated Galois ring oscillator satisfying the condition* (2) *contains one short cycle of length* 2 *if and only if an additional condition* $h(1) = 1$ *is satisfied. The number and lengths of the longer cycles then depend on the polynomial* $h(x)$*. If* $h(x)$ *is a primitive polynomial, then the additional condition is satisfied and there is only one long cycle of length* $2^r - 2$*.*

The choice of a primitive polynomial $h(x)$ is recommended to ensure good pseudorandom properties of the output signal. One can prove that the short cycle of length 2 is composed of the two states of the following form. Divide the r inverters in $2n + 1$ groups separated by the effective feedback connections, that is, by the $2n$ nonzero feedback coefficients and let the inverter output bits define the state vector. Then the two states have the forms $c_{n+1}^* A_n c_n^* A_{n-1} \cdots A_2 c_2^* A_1 c_1^*$ and $\overline{c_{n+1}}^* A_n \overline{c_n}^* A_{n-1} \cdots A_2 \overline{c_2}^* A_1 \overline{c_1}^*$, respectively, where A_i is an alternating string, c_i^* is a constant string obtained by repeating a constant c_i, and $\overline{c_i}^*$ is a constant string defined by the binary complement $\overline{c_i}$. All the strings are uniquely determined by the following rule: if $c_1 = 0$, then the last bit of A_i equals c_i and the first bit of A_i equals $\overline{c_{i-1}}$. It thus follows that the transition between these two states is metastable in the asynchronous operation, unless maybe if all the constant strings have the same odd length, for example, equal to one. In particular, since the output signal is taken from the last inverter in the cascade, it is recommended that the length of the first constant string, c_1^*, should be even. For example, this length is equal to two if $f_1 = 0$ and $f_2 = 1$.

For both configurations, the frequency of the output signal is determined by the total inverter delay in the shortest feedback loop, that is, by the smallest index i such that $f_i = 1$. So, to obtain a high-speed signal, it is desirable that such an index i is small, e.g., equal to 1 or 2 for the Fibonacci configuration and equal to 2 for the Galois configuration as well as that the (odd) number of elementary inverter logic gates comprising each inverter is relatively small. However, the inverter delay for the Fibonacci configuration should be somewhat larger because of the feedback signal being produced with a delay not present in the Galois configuration. A relatively small inverter delay is also desired for the delay-to-rise time ratio to be relatively small which in turn contributes to the randomness of the output signal on the analog level. Also note that the internal states containing constant strings of lengths at least two that start immediately after the points where the feedback signal is fed back into the cascade are susceptible to metastability. The Galois configuration is better in this regard than the Fibonacci configuration as it contains more than just one such point.

The randomness as well as robustness can further be increased by XOR-ing the outputs of two oscillators, one being in the Fibonacci and the other in the Galois configuration. The corresponding elementary RNG with a sampling unit included, implemented by a D-type flip-flop, is shown in Fig. 4. The lengths of the two oscillators minus one should preferably be mutually prime, firstly, for the period of the corresponding pseudorandom sequence to be maximized and, secondly, for the interlocking or coupling effect to be minimized. In particular, it is suggested that the lengths differ only by one, where the even length corresponds to the Fibonacci ring oscillator. In practice, the lengths could be around 20.

The experiments conducted in FPGA technology (XILINX xc2v3000-4) by using a logic analyzer confirm the ability of this RNG, as well as of individual Galois and Fibonacci ring

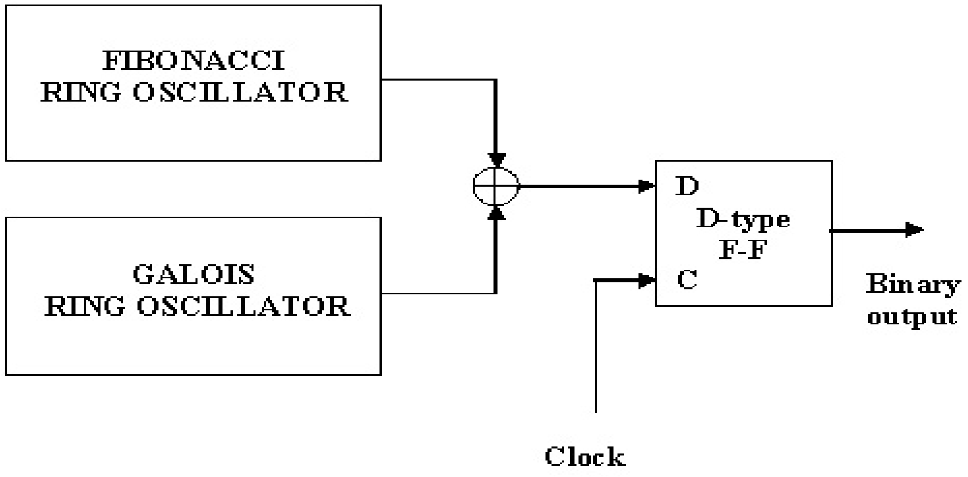

Figure 4: An elementary digital RNG.

oscillators, to generate high-speed and high-entropy raw binary sequences. Importantly, this is also true if the oscillators are short (e.g., of length about 6), in which case the effect of pseudorandomness is small. The amount of randomness for the same speed is orders of magnitude higher than for classical ring oscillators. The type of redundancy detected in the raw binary sequence consists in a slight bias of individual bits and in a slight correlation of several (e.g., 2 to 3) adjacent bits. As the bias and this type of correlation are eliminated by the pseudorandom properties of the oscillating signal, this is an indication of frequent metastability events in the sampling D-type flip-flop, which can then be regarded as the most important source of enhanced randomness. The redundancy can easily be removed by digital post-processing that reduces the output speed by a small factor. The observed robustness of randomness properties is emphasized by the fact that in FPGA technology we did not have any control over the layout of logic gates.

4 Synchronous Non-Autonomous Circuits with Feedback for Post-Processing of Random Data

A generic circuit for digital post-processing of raw random data is shown in Fig. 5. The raw binary sequence generated at the output of an RNG is used as the input sequence to a PRNG which is implemented as a synchronous non-autonomous finite-state machine. It is described in terms of the state-transition and output functions and operates synchronously with a clock signal. The state-transition and the output functions should be chosen so as to satisfy the *computational unpredictability criterion:* it should be difficult (ideally, computationally infeasible) to predict the PRNG output sequence when the input is the all-zero sequence or, more generally, when the input sequence is fixed. This implies that the PRNG internal state bit size should not be small. The state-transition function can be implemented as a composition of a logic circuit and a parallel combination of D-type flip-flops, used as delay elements, whereas the output function is implemented by another logic circuit.

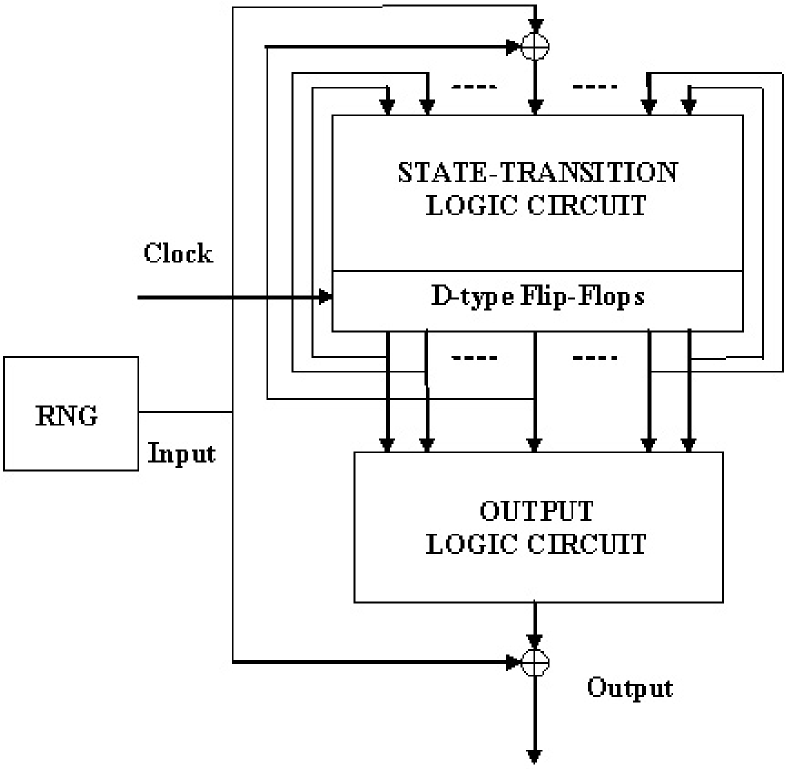

Figure 5: A generic post-processing circuit.

The input sequence is introduced through an XOR operation, which, at each time, combines an input bit with one or more internal state bits, which are then fed back into the logic circuit along with other internal state bits to produce the internal state at the next time. For simplicity, only one such internal state bit is shown in Fig. 5. These internal state bits are chosen so as to satisfy the *propagation criterion:* for any initial state, a change of the first input bit should give rise to a pseudorandom change of the subsequent output bits, namely, to a change difficult (ideally, computationally infeasible) to predict. The output function, depending on the current internal state bits, is combined with the current input bit to produce the output bit so as to satisfy the *randomness criterion:* the output sequence should be purely random if the input sequence is purely random. To satisfy this criterion, it is sufficient to combine the current input bit and the output function by another XOR operation, as shown in Fig. 5. This corresponds to the way in which the plaintext input is introduced into stream ciphers with plaintext memory [6]. The PRNG initial state can be fixed or randomly chosen.

The described generic circuit implements a reversible sequential transform, provided that the initial state is known. This implies that any statistical distinguisher for the input sequence can be converted into a distinguisher for the output sequence, by guessing the internal state at any time and recovering the input sequence. Consequently, an important component of the generic circuit is irregular clocking described below.

The clock signal for triggering the D-type flip-flops should have the same or higher frequency as the speed of the input sequence, in bits per second. It can be synchronous with a clock used in the RNG to produce the raw binary sequence. The reduction in speed is proposed to be achieved by irregular clocking in the self-clock-controlled manner according to the clock-control sequence produced by applying another function to the internal state sequence. This function should be chosen so as not to violate the computational unpredictability criterion and is, for simplicity, not shown in Fig. 5. If the clock-control sequence is binary, then, at each time when an output bit is produced, the value of the clock-control bit determines the number of times, δ_1 or δ_2, the D-type flip-flops should be clocked before the next output bit is produced. If the clock-control bit is not biased, then the expected reduction in speed due to irregular clocking is $(\delta_1 + \delta_2)/2$ times. Note that if $\delta_1 = \delta_2$ and if the PRNG is linear, then it may be possible to recover linear transforms of the input sequence from the resulting uniformly decimated output sequence. This in turn may give rise to statistical distinguishers for the output sequence, but the deviation from the uniform distribution is significantly reduced as linear transforms of the input sequence are expected to involve a large number of terms.

5 Self-Clock-Controlled LFSRs for Post-Processing

A concretization of the generic circuit from Section 4 that is suitable for hardware implementations is proposed in the sequel. Essentially, it is a self-clock-controlled LFSR in the Galois configuration that is operated non-autonomously. Such an LFSR is a cascade of D-type flip-flops where the output of the last flip-flop defines the feedback signal that is used as input to the first flip-flop and is XOR-ed with outputs of some flip-flops to form the inputs of the succeeding flip-flops in the cascade. It is well known that binary sequences with a long period and good statistical properties can be produced if the feedback connections are chosen according to a primitive or irreducible polynomial.

Of course, an LFSR, when clocked regularly, is a linear sequential circuit whose output is easily predictable by solving the corresponding linear equations. The linear predictability can be destroyed if an LFSR is clocked irregularly, that is, if some of its output bits are discarded according to a clock-control signal. So, in order to define a computationally secure PRNG, one can use one or more LFSRs, possibly irregularly clocked, and combine their outputs by a suitable function. Such, LFSR-based PRNGs are very suitable for hardware implementations.

The simplest PRNG of this type contains a single self-clock-controlled LFSR in the Galois or Fibonacci configuration. For the Galois configuration, the basic structure of the corresponding post-processing circuit is shown in Fig. 6. The LFSR consists of a number, r, of D-type flip-flops connected in a cascade so that the output of each but the last flip-flop is used to form the input to the next flip-flop and the output of the last flip-flop directly defines the feedback signal. All the flip-flops are clocked synchronously by the same clock signal.

The feedback signal is XOR-ed into the flip-flops in an analogous way as in the Galos ring oscillator described in Section 3. In order to ensure the maximal period, $2^r - 1$, and good statistical properties it is recommended to choose a primitive binary feedback polynomial.

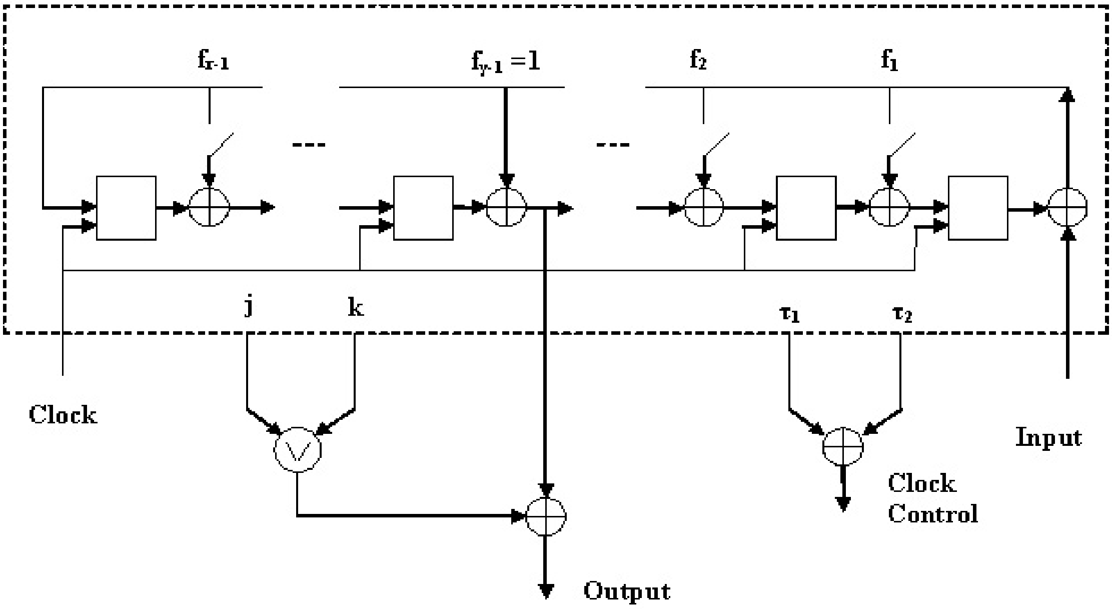

Figure 6: An LFSR-based post-processing circuit.

The LFSR operates in the non-autonomous manner by XOR-ing the input signal with the feedback signal. The output is formed by XOR-ing the output of the XOR gate connected to the output of the γ-th flip-flop, where $f_{\gamma-1} = 1$, with the output of the 2-input OR gate applied to the outputs of the j-th and k-th flip-flops. Formally, if one takes $\gamma = 1$, then the output is formed by XOR-ing the feedback signal with the output of the OR gate.

The clock-control sequence is binary, so that, at each time when an output bit is produced, the value of the clock-control bit determines the number of times, δ_1 or δ_2, the LFSR should be clocked in order to produce the next output bit. The clock control can be implemented in hardware by a counting logic and another D-type flip-flop by using its clock-enable input. The clock-control bit is produced by XOR-ing the outputs of the τ_1-th and τ_2-th flip-flops. It is suggested that $\min(i, j, \gamma) > \max(\tau_1, \tau_2)$. Since the clock-control bit is not biased, the expected speed of the output sequence is $(\delta_1 + \delta_2)/2$ times smaller than the clock frequency.

The security of the proposed PRNG is depends on the LFSR length r. For the RNG post-processing applications, it is suggested that $64 \leq r \leq 128$ may be sufficient. Also, the security is affected by the number of nonzero feedback coefficients, that is, by the weight of the feedback polynomial, which is suggested to be close to $r/2$. In addition, the weight of low-degree polynomial multiples of the feedback polynomial should not be small. In particular, under the assumption that the structure of the LFSR-based post-processing circuit is known, these conditions should practically prevent the types of statistical distinguishers described in [7], even if the input sequence is heavily biased. The security level can be further increased, at the expense of an increased gate count, by using a number of LFSRs, which are irregularly clocked in the self-clock-control or mutual-clock-control manner.

The proposed technique can be made adaptive by making the speed ratio $(\delta_1 + \delta_2)/2$ roughly inversely proportional to an estimate of the entropy rate of the input raw binary sequence, where this estimate can be computed on line by using a simple statistic such as the bias of individual bits and/or the autocorrelation function on relatively short segments.

The post-processing circuit from Fig. 6 was implemented in FPGA technology together with the RNG circuit from Fig. 4 and the produced binary sequences were subjected to statistical tests. For example, the choice of $r = 64$ in the post-processing circuit was sufficient to satisfy the DIEHARD suite of tests [10] and standard statistical tests for binary sequences such as the frequency, serial, poker, autocorrelation, runs, and Maurer tests. This was the case even for the all-zero input sequence and $\delta_1 = \delta_2 = 1$, but then in order to satisfy the linear complexity test, we have to use irregular clocking, that is, $\delta_1 \neq \delta_2$.

In practice, the choice of $\delta_1 = 3$ and $\delta_2 = 5$ may be appropriate. In this case, the tests are also satisfied for relatively small r (e.g., at most 30), and are not satisfied for the all-zero input sequence. The average reduction in speed of about four times may be sufficient for randomness extraction according to the following experiment. For a simplified post-processing circuit in which $r = 1$ and $\delta_1 = \delta_2 = 4$, which is equivalent to taking every 4-th bit of the sequence whose each bit is equal to the XOR sum of the current input bit and all the preceding input bits, the tests were satisfied for most RNG configurations chosen.

6 Conclusions

A new paradigm for generation of true random numbers by using only digital circuits is introduced and experimentally tested in FPGA technology. The new method is robust and can achieve orders of magnitude higher entropy rates than other known digital techniques for random number generation. The main point is to replace the free-running ring oscillators by new oscillators generating high-speed pseudorandom signals with enhanced randomness properties on the digital as well as analog level and to sample these signals by an independent ring oscillator signal of a similar frequency thus causing frequent metastability events in the sampling D-type flip-flop and eliminating the mutual coupling effect. The method is supported by mathematical analysis. A more systematic and detailed experimental analysis both in FPGA and ASIC technologies is an interesting area for future investigations. A new method for digital post-processing of raw random data which can provide both randomness extraction and computationally secure increase in speed is also introduced.

Acknowledgment

The author is grateful to Loris Bollea from Telecom Italia Lab for conducting the experiments in FPGA technology.

References

[1] B. Barak, R. Shaltiel, and E. Tromer, "True random number generators secure in a changing environment," Cryptographic Hardware and Embedded Systems - CHES 2003, *Lecture Notes in Computer Science*, vol. 2779, pp. 166-180, 2003.

[2] W.-T. Chuang and S. C. Hsu, "Enhanced random number generator," US patent No. US 6,240,432 B1, May 2001.

[3] K. B. Coulthart, R. C. Fairfield, and R. L. Mortenson, "Random number generator," US patent No. 4,641,102, Feb, 1987.

[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] V. Fischer and M. Drutarovský, "True random number generator embedded in reconfigurable hardware," Cryptographic Hardware and Embedded Systems - CHES 2002, *Lecture Notes in Computer Science*, vol. 2523, pp. 415-430, 2002.

[6] J. Dj. Golić, "Modes of operation of stream ciphers," Selected Areas in Cryptography - SAC 2000, *Lecture Notes in Computer Science*, vol. 2012, pp. 233-247, 2001.

[7] J. Dj. Golić and R. Menicocci, "A new statistical distinguisher for the shrinking generator," *Cryptology ePrint Archive*, Report 2003/041, Mar. 2003.

[8] 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.

[9] P.-Y. Liardet, "Random number generating circuit and process," US patent No. US 6,581,078 B1, Jun 2003.

[10] G. Marsaglia, "DIEHARD: A battery of tests of randomness," 1996, avalilable at http://stat.fsu.edu/∼geo.

[11] R. V. M. Oerlemans, "Digital true random number generator circuit," US patent No. US 2002/0156819 A1, Oct. 2002.

[12] 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.