Chapter 4
True Random Number Generators for Cryptography

Berk Sunar

4.1 Introduction

Random numbers and randomization techniques are critical for modern-day cryptography. Random numbers are used to initialize key bits for secret- and public-key algorithms, seed pseudo-random number generators, provide challenges, nonces, padding bits, as well as initialization vectors in cryptographic primitives and protocols. For cryptographic applications it is crucial to generate pseudo-random bits which will be unpredictable to the adversary even at the exposure of partial information. The literature is filled with protocols that are built around state-of-the-art cryptographic primitives, yet fail in practice, due to a weak random number generator (cf. [1]).

In this chapter, we focus on practical TRNG designs that are suitable for manufacturing on common ASIC silicon process or to be implemented on reconfigurable logic platforms (e.g., FPGA, CPLD, etc.). Hence, esoteric designs and software TRNGs (e.g., TRNGs that use randomness in RAM or Disk access times [2]) are not discussed. Unfortunately, the literature of TRNG designs is rather scattered. Some designs appear in academic articles fragmented into a number of fields which specialize in digital design techniques, integrated circuits, and even physics. Many designs are simply patented and otherwise not published. Therefore, our survey of TRNG design will be incomplete. We survey a number of selected designs and discuss them in terms of their performance, weaknesses, scalability, and versatility.

In the remainder of this chapter we first discuss the building blocks of common TRNGs. We then present a potpouri of TRNG designs; incomplete, yet chosen to expose the diversity in design techniques. This is followed by a survey of post-processing techniques. Finally, we present several new research problems motivated by real-life needs.

Department of Electrical and Computer Engineering, Worcester Polytechnic Institute,
e-mail: sunar@wpi.edu

Ç.K. Koç (ed.) *Cryptographic Engineering*, DOI 10.1007/978-0-387-71817-0_4,

4.2 TRNG Building Blocks

A true random number generator (TRNG) is a device that utilizes physical processes to generate a random bit stream. Although there is a zoo of TRNGs available, the most popular and useful ones are commonly built from the following three components:

Entropy Source: Numerous TRNG designs have been proposed in the literature for collecting randomness from physical processes such as thermal and shot noise in circuits, jitter and metastability in circuits, Brownian motion, atmospheric noise, or even nuclear decay. The entropy source is perhaps the most critical component as it determines the available entropy. On the other hand, it should be clear that sources such as atmospheric noise [3] and nuclear decay are not viable except for fairly restricted applications or online distribution services. Furthermore, some sources exhibit biases which should be eliminated in the collection or postprocessing steps. Quantification of the available entropy and its exact statistical properties is another significant design task. Another issue is considering long-term effects which may cause the breakdown in the entropy source. Active monitoring techniques for detecting total breakdown are available. However, more subtle failures are difficult to detect in practice.

Harvesting Technique: The entropy source is tapped using a harvesting technique that ideally does not disturb the physical process above, yet collects as much entropy as possible. A large number of designs have been proposed to realize this step. Since blackbox analysis of TRNGs other than statistical tests and simple true randomness tests (Tot and restart[1]) are impossible, the harvesting mechanism should come with rigorous justification.

Postprocessing: Although this component is not needed in all designs, good design practice dictates the use of a postprocessor. The goal is to make the TRNG design more robust by postprocessing the output bits. A postprocessor may be applied to hide or eliminate biases and/or dependencies in the entropy source or harvesting mechanism. A secondary goal, which has gained quite a bit of importance due to active fault and side-channel attacks, is to provide resilience to environmental changes and to tampering by adversaries. A postprocessor may be as simple as a von Neumann corrector [4] or may be as complicated as an extractor function [5] or a one-way hash function such as SHA-1 [6]. Although one-way hash functions such as SHA-1 or MD5 provide a safety net when used for postprocessing, they make the analysis of the output distribution very difficult.

Finally, we would like to note that postprocessing algorithms do not merely improve the output distribution and make the design more robust but also bring a degree of flexibility into the design. For instance, postprocessing techniques with

[1] Briefly stated, Tot tests check for a total breakdown of the entropy source of an RNG usually caused due to material ageing effects or extreme fluctuations in the operating conditions. Restart tests verify generation of randomness by restarting the RNG from nearly identical operating conditions.

quantifiable properties allow trade-offs to be made between the quality of the output bits and the throughput of the TRNG.

4.3 Desirable Features

Thus far a large number of TRNG designs have been proposed. These designs vary significantly according to their entropy sources and the harvesting techniques they employ. Each design has its strengths and weaknesses. Some of these properties are related to performance and some are related to security and robustness. We summarize below some of the features we would like TRNGs to have.

- From a practical standpoint it is essential that TRNGs are built using a commonly available cheap silicon process. Moreover, it is highly desirable to implement TRNGs using purely digital design technique. This allows for easier integration with digital microprocessors, and also makes it possible to implement TRNGs on popular reconfigurable platforms (i.e., FPGAs and CPLDs).
- Compact and efficient design with high throughput per area and energy spent. Use of amplifiers or other analog components should be avoided, if possible. Analog components tend to consume more energy and make the analysis difficult. Note that, since we are not allowing analog components, we have to sample variations in the time domain (such as the design in [7] does) rather than the variations in the voltage levels. If strictly followed, this criterion also means that we should avoid complicated postprocessing schemes (e.g., SHA-1) or at least implement them in the software.
- It is desirable to have a mathematical justification of the entropy collection mechanism, with all assumptions empirically verified. The design should be sufficiently simple to allow rigorous analysis. To validate the output of TRNGs the DIEHARD [8] or NIST Test Suites [9] are commonly employed. These statistical tests are necessary but not sufficient. Recently, Schindler and Killman [10] sketched a methodology for evaluating true random number generators and outlined the pioneering standardization efforts of the BSI as described in [11]. They advocate rigorous testing of TRNGs and note that a statistical blackbox testing strategy may not be employed for this purpose. The AIS document provides clear evaluation criteria for TRNGs and also allows TRNG designers to present their own criteria.

4.4 Survey of TRNG Designs

In this section we present a survey of TRNG designs. The survery is certainly not exhaustive and there are many other interesting designs available. Considering that a large number of designs first appeared in patents and not in academic articles, it is also likely that many innovative designs are simply kept as trade secrets. In any case, we find it useful to present chosen representative designs to expose alternative TRNG construction techniques.

4.4.1 Baggini and Bucci

The early design introduced by Baggini and Bucci [12] as shown in Figure 4.1 uses a combination of analog and digital components for amplification and sampling of white noise. The design is build to resist variations in operating conditions and component behavior. Reference [12] gives an analytical model for the TRNG which captures the relationship between the maximum bit correlation to the output bit-rate and therefore claims that it is unnecessary to use statistical testing. The reference does not report any implementation results.

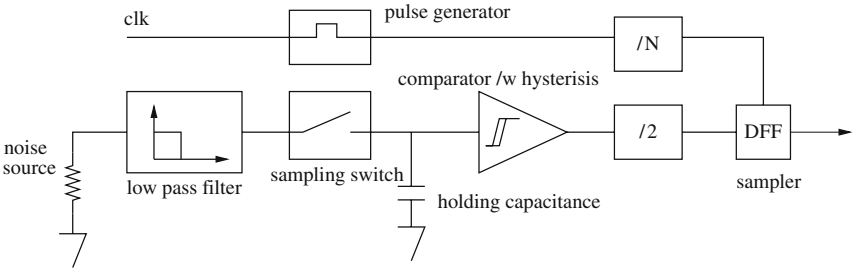

Fig. 4.1 The Baggini and Bucci TRNG Design.

4.4.2 The Intel TRNG Design

The Intel TRNG Design shown in Figure 4.2 was discussed in [6]. The entropy source of the design is thermal noise on a junction. The design uses two resistors in differential configuration to make the design more robust against power supply and environmental variations. The differential thermal noise is amplified and used to drive a voltage controlled oscillator (VCO). The VCO is then sampled by another oscillator. The output sequence is postprocessed using the von Neumann corrector and then hashed using SHA-1. As an added safety measure, the software driver that interfaces with the TRNG, implements the NIST 140-1 randomness tests monobit, runs, and poker. Jun and Kocher in [6] who have analyzed the TRNG output using 16

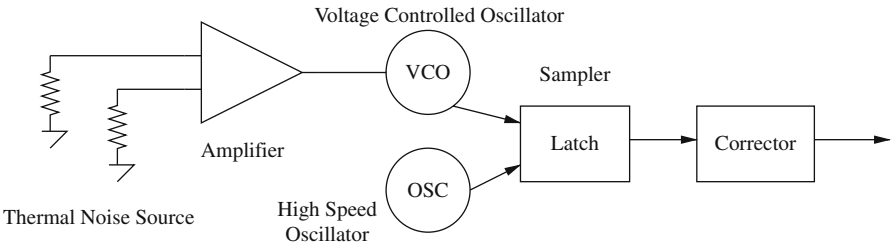

Fig. 4.2 The Intel Random Number Generator.

specialized tests and the NIST FIPS 140-1 test suite report that no weaknesses were found in the TRNG output before processing with SHA-1. The reference, however, notes that the von Neumann postprocessing technique is essential for eliminating biases in the output stream.

Since reference [6] gives only details of the security analysis we know nothing about the performance of the design, i.e., footprint, throughput, and power consumption. We may speculate that the footprint will be low due to the simplicity of the design since SHA-1 is implemented on the software side. With respect to security, we only have the blackbox analysis of Jun and Kocher. On the other hand, since the design is relatively simple, by modeling the junction noise and the oscillator jitter, one should be able to analyze the quality and performance of the TRNG output. Finally, the design has analog components, i.e., noise amplifier and voltage controlled oscillator, and therefore does not lend itself for implementation on a reconfigurable platform.

4.4.3 The Tkacik TRNG Design

The innovative design introduced in [7] randomly samples the XOR of bits chosen from a linear feedback shift register (LFSR) and a cellular automata shift register (CASR). The randomness comes from the jitter in the two free-running oscillator circuits which are used to clock the two deterministic circuits. The design is shown in Figure 4.3. The TRNG outputs 32 bits at a time. The author states that it is used with minor variations at Motorola for a number of years.

A positive aspect of the design is in its diversification. The output stream is verified using the DIEHARD [8], NIST 140-1 [9] and the Crypt-X suites [13]. The author shows that the output of the entire design has far better statistical behavior when compared to the LFSR or CASR output alone. There are no details given with regard to the performance aspects of the design. In [14] Dichtl outlines an attack on this particular TRNG construction based on two weaknesses of the design:

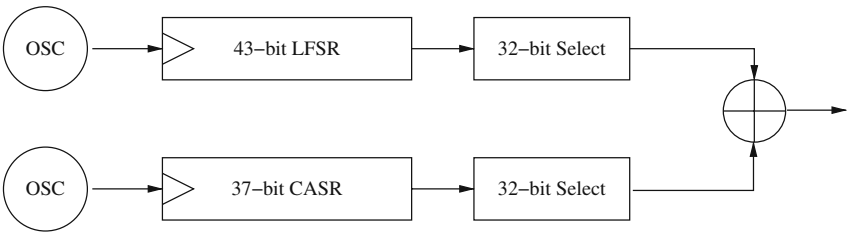

Fig. 4.3 The Tkacik TRNG Design.

- The source of entropy is fairly limited (only two oscillators are used). In fact, the LFSR and the CASR act as a pseudo-random number generator seeded with only two low-entropy oscillators.
- The design uses linear components (e.g., LFSR), and therefore the attacker can build a linear model and solve it.

The attack allows an adversary to predict the output bits assuming he/she had access to earlier bits. The treatment is theoretical and thus it is unclear if the attack would work in practice. Also, the assumption that the attacker knows some of the previously generated bits will make it impractical for many applications. On the other hand, the attack points to a dependency between output bits, and casts serious doubts about the reliability of the Tkacik TRNG. Finally, the design can be made robust by significantly lowering the output rate and/or including non-linear components. In [15] Schindler further analyzes the Tkacik design under a formulated stochastical model and develops lower and upper entropy bounds on the random output bits. Schindler also shows that the output bits carry sufficient entropy when the output is sampled 60,000 times more slowly than suggested in [7].

4.4.4 The Epstein et al. TRNG Design

In [16], a simple architecture based on bi-stable circuits is proposed. Figure 4.4 shows the basic component of the TRNG design which simply lays out many such units and computes the XOR of their output bits. A unit consists of two multiplexers and two inverters put together in a configuration that gives a metastable circuit. Note that if the select input is logic 0, then the circuit reduces to two separate single inverter oscillator rings. Alternatively, if the select input is set to logic 1, the circuit becomes functionally identical to two cascaded inverters. In the first mode, we have two free-running oscillators and in the second mode a stable circuit with no switch-

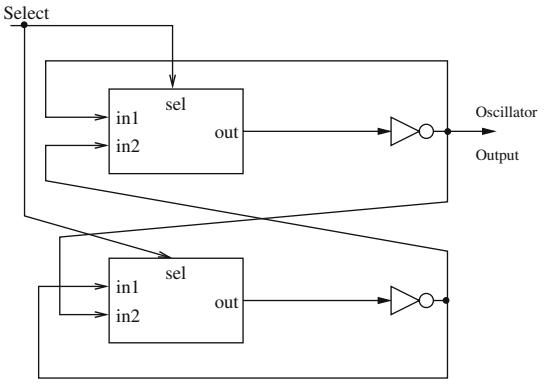

Fig. 4.4 Bi-stable memory component of the Epstein et al. TRNG design.

ing activity. Now, consider the case when the select input transitions from logic 0 to 1. Then, the two free-running oscillators may not be in the same phase and we obtain a bi-stable circuit with uncertainty in the output signal until the transitions settle.

The TRNG design which was composed of 15 instances of the components shown in Figure 4.4 and an additional 14 XOR gates was manufactured using a 0.18μ CMOS technology. All output sequences passed the DIEHARD tests after being postprocessed by the von Neumann corrector. Being constructed only from digital components, the design could be implemented on reconfigurable logic as well. Also, the design is fairly compact and should be power efficient as well. Unfortunately, reference [4] gives no information about the performance of the design. The output is verified using statistical tests. A security analysis is not provided.

4.4.5 The Fischer–Drutarovský Design

The design introduced by Fischer and Drutarovský [17] samples the jitter in a phase locked loop (PLL) on a specialized reconfigurable logic platform. The design is unique in the sense that it was the first TRNG proposal targeting FPGAs. The reference implementation targeted a particular Altera field programmable logic device family that comes with a PLL (e.g., APEX E and APEX II families) as shown in Figure 4.5. The jitter of the clock signal generated by the on-chip PLL is sampled via delay cascaded samplers organized in the configuration shown in Figure 4.6. The key idea is to use multiple samplers to be able to sample near the transition zone that is influenced by the jitter which according to [17] is of the order of only several tens of picoseconds. The multiple samples taken at regular intervals which are then XOR-ed together gives a sample from an area of the waveform that has the desired uncertainty. Finally, the output of the XOR is then downsampled using a decimator. The authors of [17] give a fairly detailed summary of the actual implementation and the design choices made. For instance, the authors note that resources need to be locked in place in the FPLD to obtain the desired routing configuration.

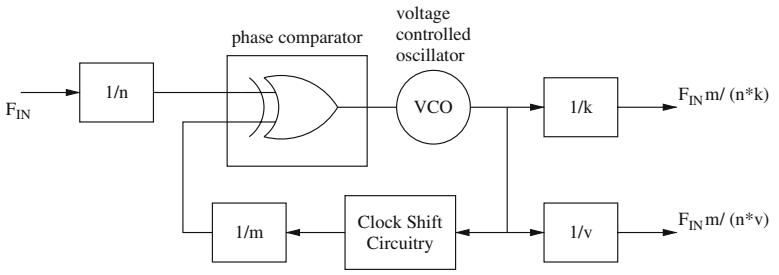

Fig. 4.5 Architecture of the programmable PLL used as the entropy source in the Fischer–Drutarovský TRNG design.

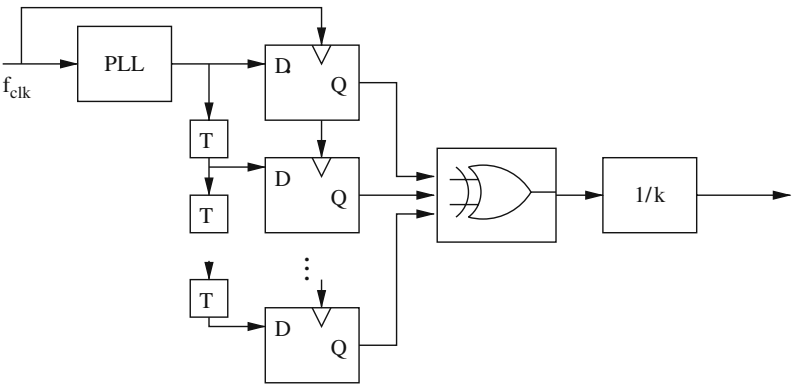

Fig. 4.6 The Fischer–Drutarovský TRNG Design.

This is especially important for the delayed samplers. The reported implementation yielded a bit-rate of nearly 70 Kbits/s. However it is not clear whether this was the upper limit of reliable operation, or whether this is merely a design choice. The generated random bit sequence was verified for statistical behavior using the NIST tests.

All in all, the Fischer–Drutarovský is important in the sense that it highlights the importance of TRNGs for reconfigurable platforms. The authors introduce the novel cascaded delayed sampler and also provide a mathematical model that allows them to pick operating points to increase the likelihood of collecting bits near the transition zones.

4.4.6 The Golić FIGARO Design

The Fibonacci oscillator [18] is shown in Figure 4.7. Basically, the structure is identical to an LFSR except for the delay elements being replaced by inverters. The feedback positions are labeled by switch values f_i. If $f_i = 1$ then the switch is closed and otherwise it is open. The switch values can be represented more conveniently in terms of the feedback polynomial which is given as follows.

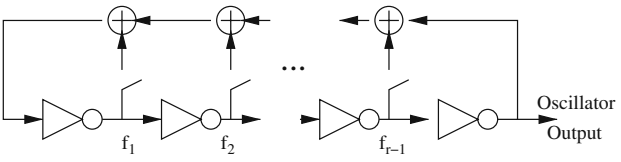

Fig. 4.7 The Fibonacci Oscillator Design.

$$f(x) = \sum_{i=0}^{r} f_i x^i \text{ where } f_0 = f_r = 1.$$

It is important that the oscillator is not stuck in a single fixed state. The necessary and sufficient conditions are given in Theorems 4.1 and 4.2.

Theorem 4.1 ([18]). *A Fibonacci ring oscillator does not have a fixed state if and only if*

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

Theorem 4.2 ([18]). *A Galois ring oscillator does not have a fixed state if and only if*

$$f(1) = 1 \text{ and } r \text{ is odd.}$$

Furthermore, for both kinds of oscillators, if $h(x)$ is chosen to be a primitive polynomial, we are guaranteed to have two cycles: a short cycle of only 2 states and a long cycle which includes the remaining $2^r - 2$ states. The Galois configuration of the oscillator ring is shown in Figure 4.8. The FIGARO (**Fi**bonacci-**Ga**lois-**R**ing-**O**scillator) TRNG design simply XORs the output of a Figaro oscillator with the output of a Galois oscillator and samples the XOR output. To eliminate local correlations and biases the author also proposes to use a self-controlled LFSR for post-processesing of the output. Later on the performance was analyzed by Dichtl and Golić (see Section 4.4.12).

4.4.7 The Kohlbrenner–Gaj Design

Similar to earlier design the Kohlbrenner–Gaj design [19] uses jitter in ring oscillators as the entropy source. What makes this design different is that, it is designed to perfectly match the CLB architecture of a Xilinx Virtex-II FPGA. The oscillator, for instance, is build into a CLB. The oscillator signal passes twice through the CLB structure and is flipped in only one of the passes (in LUT1) as shown in Figure 4.9. For clarity the clk and reset signals are not shown in the figure. The oscillation frequency is determined by the delay elements on the oscillator path, i.e., two lookup tables, four multiplexers, and two memory cells. Kohlbrenner notes that, this particular configuration gives a sufficiently stable 130 MHz oscillator signal. The TRNG samples one such oscillator with another one.

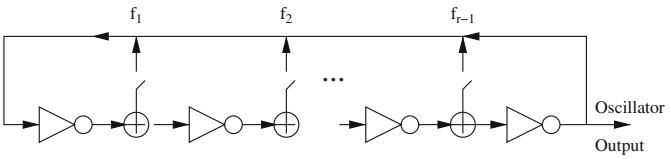

Fig. 4.8 The Galois Oscillator Design.

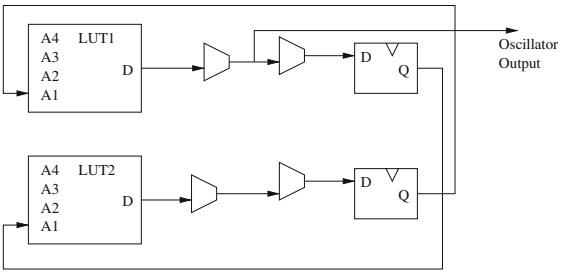

Fig. 4.9 Oscillator/CLB structure of the Kohlbrenner–Gaj Design.

The TRNG output is also postprocessed with a simple succesive XOR scheme, to eliminate biases. The reported bit-rate is of the order of several hundred kilobits/s. The exact rate depends on the strength of the XOR postprocessing scheme. Although the rate is relatively low, the design is fairly compact and its bit-rate will be sufficient in many applications. The output sequence was statistically verified using the NIST 140-1 test suite.

4.4.8 The Bucci–Luzzi Testable TRNG Design Framework

Bucci and Luzzi [20] made the observation that it is difficult, and perhaps impossible, to test the quality of TRNG outputs after complex postprocessing techniques have been employed. The authors propose to augment the designs with reset circuits that clear the state of the TRNG. This is done to support a so-called *certification mode* which establishes whether the TRNG is trustworthy. In the certification mode, the TRNG is restarted before the collection of each output bit. The objective behind the restart is to eliminate any dependencies between the collected bits. Then the output of the TRNG is either stuck in a fixed bit and no entropy is generated, or it generates independent bits. The former can be checked via a scheme that simply counts the transitions. If the transition rate is as expected, then biases in the output may be eliminated by using a stateless postprocessor. The stateless postprocessor preserves the independence among output blocks. In principle, the proposed restart approach is applicable mainly to any entropy source that permits a restart. The key point though is that the output diverges quickly from the start state into an unpredictable state. Hence, the amount of time required for an entropy source to produce diverging outputs after reset may be used as a metric.

An important side benefit of the stateless TRNG approach is that it makes detection of forcing attacks much easier when stateless linear postprocessors are used. A non-(pseudo) random bias introduced by the the attacker will be visible at the output due to the independence of the output bits and the linearity of the postprocessor.

4.4.9 The Rings Design

The rings design shown in Figure 4.10 was proposed by Sunar, Martin and Stinson in [21]. The design is very simple. Basically, free-running ring oscillator outputs are combined together via an XOR operation and then sampled. The source of randomness, is phase jitter. The main idea is to populate the output waveform with transition zones and then to sample randomly. The authors provide a mathematical framework and rigorous analysis of the quality of the output of the TRNG based on a set of assumptions at the input. Furthemore, to reduce the number of rings, the authors propose to use a resilient function for postprocessing of the TRNG output. By keeping the degree of the resilient function high, the TRNG develops a quantifiable tolerance against active adversaries. The rings design has two main contributions: the analysis framework and the introduction of resilient functions for postprocessing. The analysis builds a simple jitter model, and computes the minimum number of rings that need to be included in the design to achieve a certain fill-rate in the sampling window, at a certain confidence level. The deterministic bits collected from the unfilled portion of the sampling window are eliminated by a resilient function of appropriate strength.

An initial reference implementation of the Rings design was provided by Schellekens et al. in [22] on a Xilinx Virtex-II FPGA. The implementation produced a stream at a 2.5 Mbps bit-rate with a sampling frequency of 40 MHz and using 110 rings with 13 inverters and the resilient function constructed from the linear cyclic code $(256, 16, 113)$. The output sequence was verified using the DIEHARD and NIST tests. Schellekens et al. also observed that the Rings design is stateless and uses a linear stateless postprocessing technique (a resilient function constructed from a linear code) and therefore satisfies the criteria for testability introduced earlier by Bucci and Luzzi [20].

Finally, we should note that the Rings design received criticism in several aspects from Dichtl and Golić [23]. Among the criticisms are the independence assumption of the ring oscillators and the sampling rate. While the sampling rate may be easily reduced, it is more difficult to verify the independence of the ring oscillators when

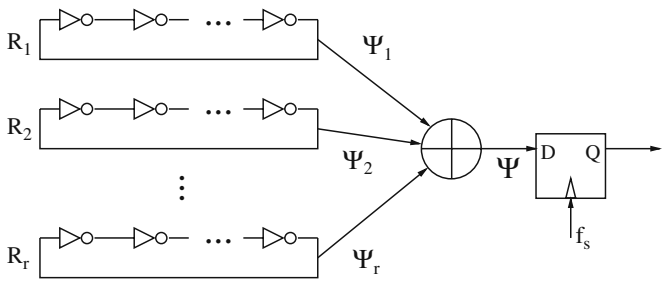

Fig. 4.10 The ring oscillators design.

a large number of rings are used. For a smaller number of rings, careful place and routing may sufficiently isolate the rings from interacting with each other. A much simpler solution is to collect only one sample from one oscillation period. In this case, ring independence is not required. It suffices to check against phase interlock which would reduce the fill-rate.

4.4.10 The PUF–RNG Design

An RNG design based on physically unclonable functions (PUFs) was proposed by O'Donnel et al. in [24]. The RNG design is build around a PUF circuit as shown in Figure 4.11. Under normal operation the output of the PUF circuit is determined by the subtle imprecisions in the delay paths created during the manufacturing process along with the challenge value supplied. Alternatively, for a particular set of challenges the delays will be closely matched and the sampling circuit will enter a meta-stable state. Hence, the output of the device will be unpredictable. While this is good news, a challenge that gives rise to metastability does so only temporarily due to temperature and voltage variations.

Hence, the PUF–RNG design *searches* for meta-stable challenges by repeatedly applying a challenge and checking if a sufficiently unstable output is obtained. Roughly stated, the same challenge is fed to the PUF circuit a fixed number of times with a fixed window length, with the hope of obtaining nearly uniform distribution at the PUF output in one window. If this is not achieved after trying a fixed number of windows, a new challenge is generated with the help of a pseudo-random number generator. When a meta-stable challenge is found, it is used to generate an output string which is further postprocessed using the von Neumann corrector.

The reference reports an implementation based on the PUF integrated into the AEGIS secure processor [25]. The PRNG, as well as the metastable challenge searching technique, is implemented in the software. The output of the RNG is verified using the NIST test suite. Unfortunately, the throughput rate is not given. The primary advantage of this design is that it makes use of an existing PUF component.

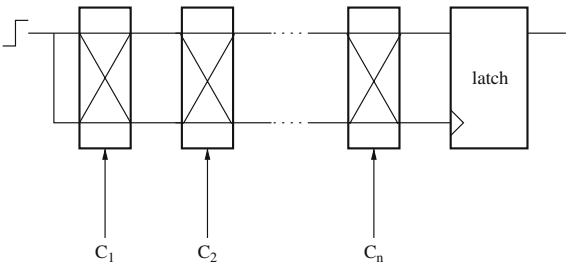

Fig. 4.11 A delay-based PUF design.

PUF circuits have become a popular tool for IC identification and for achieving tamper-resilience. Therefore, it is quite likely that a security device comes with an integrated PUF device.

4.4.11 The Yoo et al. Design

The practical aspects of the Rings design including IC routing effects, and the effects of power supply and temperature variations were investigated in [26]. The authors first note that if the signal is subsampled, then there is a chance especially at low fill-rates that the sampler may end up being stuck in an deterministic portion of the sampling window. The authors therefore recommend sampling at a frequency that is relatively prime to the oscillation frequency. The authors note that IC level effects such as phase interlock, narrow signal rejection in the XOR tree, and narrow signal attenuation affects will limit the scalability and performance of the Rings TRNG design. Furthermore, the same reference shows via experiments performed on an FPGA implementation that by changing the temperature and supply voltage, the oscillation frequency may be shifted to invalidate the relatively prime condition. Hence, the Rings TRNG may be vulnerable to non-invasive temperature and supply voltage variation attacks. Finally, to make the design robust against such attacks, the authors propose to use more than one ring length in the design. A design that features two ring lengths is proposed. The design passes the DIEHARD and NIST tests and delivers a throughput of 67 Mbps at a power consumption less than 300 mW with an area of less than 1000 LUTs. The design is also shown to be robust to temperature and power supply variations.

4.4.12 The Dichtl and Golić RNG Design

Dichtl and Golić investigated Fibonacci and Galois ring oscillators in [23]. Their analysis is primarily based on the restart technique. By restarting the oscillators from the same initial conditions they measure the time it takes to observe a bit change in the otherwise pseudo-random bitstream. Hence, the time it takes to observe a random bit determines the sampling rate and throughput of the RNG. In the same reference, the authors report an FPGA implementation of a Fibonacci ring that achieves a troughput of 6.25 Mbps.

Based on their experiments, Dichtl and Golić claim much higher entropy rates than that of traditional ring oscillators. The authors also note that the restart technique may be used as a mode of operation for the RNG and that the restart approach allows testability. Another contribution of this work is a novel two-level sampler design which reduces the bias introduced by the sampling flip-flop. With its small footprint the design seems to be ideal for embedded systems. The authors provide some preliminary justification for the performance improvement.

4.5 Postprocessing Techniques

There are several postprocessing techniques used in practice. Here we present the most popular ones.

- **Cryptographic Hash Functions:** Perhaps the most popular and most robust postprocessing technique is to run the output of a TRNG design through a cryptographically strong hash function such as SHA-1 or MD5. For instance, the Intel RNG makes use of SHA-1. From a performance perspective, implementing a full hash function for a TRNG seems like an overkill. However, from a security perspective, if properly implemented it has the important side-benefit of falling back to a pseudo-random number generator if a total breakdown occurs in the randomness source. Furthermore, the non-linearity of the hash function becomes useful if a weakness in the collection mechanism is found. A good strategy would be to implement the one-way function as the last step in software.
- **Von Neumann Corrector:** The von Neumann corrector is one of the oldest and best known postprocessing techniques and is used to eliminate localized biases. It takes pairs of bits from the random bit stream. If they are of identical value (i.e., both '0' bits or both '1' bits) it removes them from the random bit stream. If they are different, it uses one of the bits, e.g., the first bit. On average, the bit-rate will be reduced to only about 1/4 of the input bit-rate. The big advantage of the von Neumann corrector is that it is very easy to implement.
- **Extractor Functions:** The use of extractor functions was proposed by Barak, Shaltiel and Tomer in [5] with the purpose of making TRNG designs robust against changing environmental conditions. Extractor functions are powerful stateless functions with quantifiable properties originally developed as a tool for complexity theory. The authors develop a mathematicl model to capture an adversary's influence on the randomness source and give an explicit construction based on universal hash functions which is proven for its output properties even if non-local correlations exists in the input source. We give several definitions relevant to extractor functions as follows.

  Definition 4.1. The statistical distance between two distributions X and Y is defined as

  $$\varepsilon = \frac{1}{2}\sum_{a} |\mathsf{P}robX = a - \mathsf{P}robY = a| \ .$$

  In practice, we say that X is ε-close to Y and vice versa.

  Definition 4.2 (Min-Entropy). A distribution X on $\{0,1\}^n$ is said to have min-entropy k, if for all $x \in \{0,1\}^n$ $\mathsf{P}robX = x \leq 2^{-k}$.

  In general, an extractor is a function characterized with respect to its input-output behavior. An extractor is viewed as taking an input with a certain level of min-entropy k, and guarantees an output distribution that is ε close to uniform distribution. In [5] the authors provide an extension to this definition. The

authors define a function $E : \{0,1\}^n \mapsto \{0,1\}^m$ which is fixed by the choice of a public parameter. They allow an adversary to choose from 2^t distributions $D_1, D_2, \ldots, D_{2^t}$ over $\{0,1\}^n$ such that the min-entropy of each D_i is greater than k for all $i = 1, 2, \ldots, 2^t$. A public parameter π is chosen at random and independently of the choices of D_i. The adversary chooses one of the distributions, i.e., D_u. The user evaluates the extractor function using the public parameter π and a value drawn from the chosen distribution D_u. A t-resilient extractor function is defined as follows:

Definition 4.3 (t-Resilient Extractor Function, [5]). Given m, k, ε, and t, an extractor $E : \{0,1\}^n \mapsto \{0,1\}^m$ is t-resilient if with probability at the most $1 - \varepsilon$ over the choice of the public parameter π, the statistical distance of the output distribution of $E^{\pi}(X)$ to the uniform distribution is at the most ε.

In a practical setting, this means that the adversary is assumed to have control over t binary values (or 2^t internal states) of the TRNG through control of voltage, temperature, operating frequency, etc. Despite the adversary's ability, the output distribution is biased away from the uniform distribution by at the most ε. This construction gives great power to TRNG designers, since it implicitly captures *any* kind of influence by the adversary. On the other hand, from a design point of view, it is not clear how to quantify the advery's abilities and therefore it is difficult to choose design parameters for the extractor (or for the underlying universal hash function family).

- **Resilient Functions:** Resilient functions were proposed by Sunar, Martin, and Stinson in [21] as the postprocessing step for the Rings Design. The goal was to filter any deterministic bits by using the resilient function. Treating bits effected by the adversary as deterministic bits, enables one to study the tolerance properties of resilient functions against active adversaries. The reference recommends using higher resiliency degrees than necessary to remove deterministic bits. The difference between the degree of the resilient function and the number of deterministic bits expected in a sampling window quantifies the tolerance (in bits) of the TRNG to active adversaries. Resilient functions are formally defined as follows:

  Definition 4.4 (t-Resilient Function). An (n,m,t)-resilient function is a function

  $$F(x_1, x_2, \ldots, x_n) = (y_1, y_2, \ldots, y_m)$$

  from $\mathbb{Z}_2^n$ to $\mathbb{Z}_2^m$ enjoying the property that, for any t coordinates $i_1, \ldots, i_t$, for any constants $a_1, \ldots, a_t$ from $\mathbb{Z}_2$ and any element y of the codomain

  $$\mathsf{P}robF(x) = y | x_{i_1} = a_1, \ldots, x_{i_l} = a_l = \frac{1}{2^m}.$$

  In the computation of this probability, all x_i for $i \notin \{i_1, \ldots, i_t\}$ are viewed as independent random variables each of which takes on the value 0 or 1 with probability 0.5.

In more informal terms, if up to any t of the input bits are deterministic and the remaining bits are random, the output of the resilient function will be perfectly random (or unpredictable). From a cryptographic viewpoint, knowledge of any t values of the input to the function does not allow one to make anything better than a random guess at the output. Resilient functions are used in a number of cryptographic applications where the adversary is assumed to have captured or determined a number of of the key bits.
A simple technique for constructing resilient functions is given in the following theorem:

Theorem 4.3. (e.g., [27]) *Let G be a generator matrix for an $[n,m,d]$ linear code C. Define a function $f : \{0,1\}^n \mapsto \{0,1\}^m$ by the rule $f(x) = xG^T$. Then f is an $(n,m,d-1)$-resilient function.*

For more information on resilient functions, and their connections to codes and designs see [28] and [29].
When compared to extractor functions, resilient functions appear to be much more limited in their capabilities of eliminating the effects of active adversaries on the output stream. The reason for this is that resilient functions are defined to work on either perfectly random or perfectly deterministic bits. In contast, extractor functions assume only a specific min-entropy at the input. On the positive side, resilient functions give perfect output distribution ($\varepsilon = 0$) and are easily constructed from codes. When linear codes are used for the construction the resilient function is also linear and therefore allows testability of the TRNG design in the sense of Bucci and Luzzi [20].

4.6 Exercises

Whenever a TRNG is to be built, several questions come to mind. Here we give an incomplete list of these questions. The reader should extend the list further by considering the context of the implementation, the platform, and development environment.

1. How small can we build it? Low footprint TRNGs are crucial for constrained applications such as RFIDs, smartcards and sensor networks. Usually only a tiny fraction of the chip area is available for the TRNG.
2. Does it scale? Trade-offs between throughput and the quality of the TRNG output are important to optimally meet application requirements at a wide variety of design points.
3. Is it robust? Robustness is an important issue especially in embedded applications, e.g., smartcards, where the user (potential attacker) has full access to the device.
4. Will we know when it fails? There is a great need for online tests. Robustness of the test circuit is also important.

References

1. I. Goldberg and D. Wagner. Randomness in the Netscape Browser. *Dr. Dobb's Journal*, January 1996.
2. D. Davis, R. Ihaka, and P. P. Fenstermacher, Cryptographic randomness from air turbulence in disk drives. In Y. Desmedt editor, *Advances in Cryptology (Crypto 94)*, vol. 839, pp. 114–120, Heidelberg, Germany: Springer-Verlag, 1994.
3. Random.org. *True random number service v2.0 beta*. www.random.org
4. J. von Neumann. *Various techniques for use in connection with random digits*, von Neumann's Collected Works, vol. 5, Pergamon, pp. 768–770, 1963.
5. B. Barak, R. Shaltiel, and E. Tomer. True Random Number Generators Secure in a Changing Environment. In Ç. K. Koç and C. Paar, editors, *Workshop on Cryptographic Hardware and Embedded Systems—CHES 2003*, pp. 166–180, Berlin, Germany, Lecture Notes in Computer Science, Vol. 2779 2003. Springer-Verlag, 2003.
6. B. Jun and P. Kocher. *The Intel random number generator*, White Paper Prepared for Intel Corporation, April 1999.
7. T. E. Tkacik. A Hardware Random Number Generator In B. S. Kaliski Jr., Ç. K. Koç, C. Paar, editors, *Workshop on Cryptographic Hardware and Embedded Systems—CHES 2002*, pp. 450–453, Berlin, Germany, Lecture Notes in Computer Science, Vol. 2523. Springer-Verlag Berlin Heidelberg, 2003.
8. G. Marsaglia. *DIEHARD: A Battery of Tests of Randomness*, http://stat.fsu.edu/∼geo, 1996.
9. NIST. *A Statistical Test Suite for Random and Pseudorandom Numbers*. Special Publication 800-22, December 2000.
10. W. Schindler and W. Killmann. Evaluation Criteria for True (Physical) Random Number Generators Used in Cryptographic Applications. In B. S. Kaliski Jr., Ç. K. Koç, C. Paar, editors, *Proceedings of the Workshop on Cryptographic Hardware and Embedded Systems – CHES 2002*, Lecture Notes in Computer Science, Vol. 2523, pp. 431–449, Springer-Verlag Berlin Heidelberg, August 2002.
11. Anwendungshinweise und Interpretationen zum Schema (AIS). AIS 32, Version 1, *Bundesamt fr Sicherheit in der Informationstechnik*, 2001.
12. V. Bagini and M. Bucci. A Design of Reliable True Random Number Generator for Cryptographic Applications. In Ç. K. Koç and C. Paar, editors, *Workshop on Cryptographic Hardware and Embedded Systems—CHES 1999*, pp. 204–218, Berlin, Germany, Lecture Notes in Computer Science, Vol. 1717. Springer-Verlag, 1999.
13. Crypt-X. http://www.isi.qut.edu.au/resources/cryptx/.
14. M. Dichtl. How to Predict the Output of a Hardware Random Number Generator, In C. D. Walter, Ç. K. Koç, C. Paar, editors, *Proceedings of the Workshop on Cryptographic Hardware and Embedded Systems – CHES 2003*, Lecture Notes in Computer Science, Vol. 2779, pp. 181–188, Springer-Verlag Berlin Heidelberg, 2003.

15. W. Schindler. A Stochastical Model and Its Analysis for a Physical Random Number Generator In K. G. Paterson editor, *Cryptography and Coding—IMA 2003*, Springer, Lecture Notes in Computer Science, vol. 2898, 276–289, Berlin, 2003.
16. M. Epstein, L. Hars, R. Krasinski, M. Rosner and H. Zheng. Design and Implementation of a True Random Number Generator Based on Digital Circuit Artifacts. In C. D. Walter, Ç. K. Koç, C. Paar, editors, *Workshop on Cryptographic Hardware and Embedded Systems—CHES 2003*, Lecture Notes in Computer Science, Vol. 2779, pp. 152–165. Springer-Verlag Berlin Heidelberg, 2003.
17. V. Fischer and M. Drutarovský. True Random Number Generator Embedded in Reconfigurable Hardware In B. S. Kaliski Jr., Ç. K. Koç, C. Paar, editors, *Workshop on Cryptographic Hardware and Embedded Systems—CHES 2002*, pp. 415–430, Berlin, Germany, Lecture Notes in Computer Science, Vol. 2523. Springer-Verlag Berlin Heidelberg, 2003.
18. J. Dj. Golić,. New methods for digital generation and postprocessing of random data. *IEEE Transactions on Computers 55(10)*: 1217–1229, 2006.
19. P. Kohlbrenner and K. Gaj. An embedded true random number generator for FPGAs International Symposium on Field Programmable Gate Arrays. In *Proceedings of the 2004 ACM/SIGDA 12th international symposium on Field programmable gate arrays*, PP. 71–78, ACM Press, New York, NY, 2004.
20. M. Bucci and R. Luzzi. Design of Testable Random Bit Generators, In J. R. Rao and B. Sunar, editors, *Proceedings of the Workshop on Cryptographic Hardware and Embedded Systems – CHES 2005*, Lecture Notes in Computer Science, Vol. 3659, pp. 131–146, Springer-Verlag Berlin Heidelberg, August 2005.
21. B. Sunar, W. J. Martin, and D. R. Stinson. *A Provably Secure True Random Number Generator with Built-in Tolerance to Active Attacks*, IEEE Transactions on Computers, vol. 58, no 1, p. 109–119, January 2007.
22. D. Schellekens, B. Preneel, and I. Verbauwhede FPGA Vendor Agnostic True Random Number Generator. In *Proceedings of the 16th International Conference on Field Programmable Logic and Applications*. pp. 1–6, August, 2006.
23. M. Dichtl and J. Dj. Golić. High-Speed True Random Number Generation with Logic Gates Only. Pascal Paillier, Ingrid verbauwhede, editors, *Proceedings of the Cryptographic Hardware and Embedded Systems – CHES 2007, 9th International Workshop*, Vienna, Austria, September 10–13, 2007. Lecture Notes in Computer Science, vol. 4727, pp. 45-62, Springer Verlag, 2007.
24. C. W. O'Donnell, G. E. Suh, and S. Devadas. *PUF-Based Random Number Generation*. Technical Report 481, MIT CSAIL, November 2004. Available at http://www.csg.csail.mit.edu/pubs/publications.html.
25. G. E. Suh, C. W. ODonnell, I. Sachdev, and S. Devadas. *Design and Implementation of the AEGIS Single-Chip Secure Processor Using Physical Random Functions*. Technical report, MIT CSAIL CSG Technical Memo 483, November 2004.

26. S.-K. Yoo, B. Sunar, D. Karakoyunlu, and B. Birand. *Practical Aspects of the Rings Design*, Available at http://ece.wpi.edu/∼sunar/preprints/rings.pdf.
27. B. Chor, O. Goldreich, J. Håstad, J. Friedman, S. Rudich, and R. Smolensky. The bit extraction problem or t-resilient functions, *26th IEEE Symposium on Foundations of Computer Science*, pp. 396–407, 1985.
28. C. J. Colbourn, J. H. Dinitz and D. R. Stinson. Applications of combinatorial designs to communications, cryptography and networking, *Surveys in Combinatorics*, 1999, pp. 37–100, (1999 British Combinatorial Conference).
29. D. R. Stinson and K. Gopalakrishnan. Applications of Designs to Cryptography, In C. D. Colbourn, and J. H. Dinitz, editors, *CRC Handbook of Combinatorial Designs*, CRC Press 1996.
30. R. A. Schulz. *Random Number Generator Circuit*. United States Patent, Patent Number 4905176, February, 27, 1990.