

Technical University of Kosice
Faculty of Electrical Engineering and Informatics
Department of Electronics and Multimedia Communications

Ing. Michal Varchola

**FPGA Based True Random Number
Generators for Embedded Cryptographic
Applications**

Thesis to the dissertation examination

Supervisor: **doc. Ing. Miloš Drutarovský, CSc.**

Program of PhD study: **Infoelectronics**

Field of PhD study: **5.2.13 Electronics**

Form of PhD study: **Daily**

Košice December 2008

Contents

Head Page	i
Contents	iii
List of Abbreviations	iv
List of Symbols	vi
Abstract	xi
Introduction	13
1 General Concept of TRNG	15
1.1 Overview of Concepts	15
1.2 The Practical Concept of TRNG	15
1.2.1 Source of Digital Noise	18
1.2.2 Digitized noise post-processing	20
1.2.3 Output speed	24
1.3 Randomness Extracting	24
1.4 Metastability	25
1.5 Chaos	27
1.6 Jitter	28
1.6.1 Introduction to Jitter	28
1.6.2 The noisy oscillator	30
1.6.3 Jitter Fundamentals	31
1.6.4 Jitter accumulation over the time	33
1.6.5 Jitter decomposition	35
1.6.6 Jitter measuring	35
1.6.7 Jitter in Ring Oscillators	36
2 Testing and evaluation of TRNG	39
2.1 Basic Statistical Tests	39
2.2 Diehard battery of tests	41
2.3 Tests and requirements of the NIST	43
2.3.1 Federal Information Processing Standard Publication FIPS 140	43
2.3.2 NIST 800-22	44
2.4 Tests and requirements of the BSI	46
2.4.1 AIS 20	46
2.4.2 AIS 31	48
2.5 Online Tests	51

3	Implementation Platform Issues	52
3.1	Altera FPGAs	52
3.2	Xilinx FPGAs	52
3.3	Actel FPGAs	53
3.4	Logic Cells Comparison of the Various FPGA Producers	53
3.5	Comparison of the PLLs and the DLLs of Various FPGA Manufacturers	55
3.5.1	Comparison of DFF Metastability of Various FPGA Families .	59
4	Practical Implementations of TRNG	61
4.1	TRNGs based on the noisy clock sampling	61
4.1.1	Sunnar’s type TRNGs	61
4.1.2	TRNGs based on two rationally related clocks	63
4.1.3	Ring Oscillator remake of previous TRNG	64
4.1.4	How to use DLLs instead of PLLs?	65
4.2	TRNGs based on LFSRs and CASRs architectures	66
4.2.1	Tkacik’s TRNG	66
4.2.2	Golic’s TRNG	68
4.3	TRNG based on open delay chain	69
4.4	Stateless TRNG	70
4.5	Generator of Bucci et al.	70
4.6	TRNG based on meta-stability	71
4.7	The Quantum TRNG	79
4.8	TRNGs for FPGAs comparison	79
5	Conclusion and Discussion	82
	ThesisTopics	83
	References	91

List of Abbreviations

ADC	–	Analog to Digital Converter
AIS	–	Application Notes and Interpretation of the Scheme
ASIC	–	Application Specific Integrated Circuit
BSI	–	Bundesamt für Sicherheit in der Informationstechnik
CASR	–	Cellular Automata Shift Register
CLB	–	Configurable Logic Block
DCD	–	Duty Cycle Distortion
DCM	–	Digital Clock Manager
DDJ	–	Data Dependent Jitter
DFF	–	D-type Flip Flop
DFS	–	Digital Frequency Synthesizer
DLL	–	Delay Locked Loop
DRNG	–	Deterministic Random Number Generator
DSS	–	Digital Signature Standard
EMC	–	Electromagnetic Compatibility
EMI	–	Electro-Magnetic Interference
HRNG	–	Hybrid Random Number Generator
FIFO	–	First In First Out
FIPS	–	Federal Information Processing Standard
FPAA	–	Field Programmable Analog Array
FPGA	–	Field Programmable Gate Array
LE	–	Logic Element
LFSR	–	Linear Feed-back Shift Registers
LUT	–	Look-Up Table
MOS	–	Metal Oxide Semiconductor
NIST	–	National Institute of Standards and Technology
PDF	–	Probability Density Function
PJ	–	Periodic Jitter
PLL	–	Phase Locked Loop
PRNG	–	Pseudo Random Number Generator
PUF	–	Physically Unclonable Function
PWAM	–	Piece-Wise-Affine Markov
RL	–	Rotate Left
RMS	–	Root Mean Square
RO	–	Ring Oscillator
QKD	–	Quantum Key Distribution
ROM	–	Read Only Memory
RNG	–	Random Number Generator
SDF	–	Spectral Density Function
SNR	–	Signal to Noise Ratio
SRAM	–	Static Random Access Memory
SSB	–	Single Side Band

SSN	–	Simultaneous Switching Noise
TDC	–	Time-to-Digital Converter
TJ	–	Total Jitter
TRNG	–	True Random Number Generator
VCO	–	Voltage Controlled Oscillator
VHDL	–	VHSIC hardware description language
VHSIC	–	Very-High-Speed Integrated Circuits
WGN	–	White Gaussian Noise
XOR	–	eXclusive OR

List of Symbols

f	–	Frequency
t	–	Time
Λ	–	No Output Digit
\oplus	–	Exclusive OR Operation
$MTFB$	–	Mean Time Between Failures
V_0	–	Nominal Peak Voltage Amplitude
$\epsilon(t)$	–	Deviation from the Nominal Amplitude
$v_0(t)$	–	Nominal Frequency
$\phi(t)$	–	Phase Deviation from the Nominal Phase
$S_x(f)$	–	Spectral Density of Fractional Time Fluctuations
$S_\phi(f)$	–	Spectral Density of Phase Fluctuations
$\mathcal{L}(f)$	–	Single-Sideband Phase Noise
ϕ_{rms}^2	–	RMS Phase Jitter
Φ	–	Phase Jitter
Φ'	–	Period Jitter
Φ''	–	Cycle-to-Cycle Jitter
TIE	–	Time Interval Error
σ_{TIE}^2	–	Variance of Accumulated Jitter
σ_{PER}^2	–	Variance of Period Jitter
σ_{PER}^2	–	Variance of Cycle-to-Cycle Jitter

List of Figures

1.1	The simplified concept of the TRNG [10] with the main components: source of White Gaussian Noise (WGN), low-pass filter (H), analog-to-digital converter (C) and sampler (S).	15
1.2	The concept of TRNG according [11] that is used in AIS 31 testing methodology.	16
1.3	Detail of the Source of Digital Noise with its main components: the Source of Random Process is approximated by the WGN Source and natural low-pass filter and the Randomness extractor, which transforms noise in both discrete amplitude and discrete time.	16
1.4	The practical block diagram of the TRNG.	17
1.5	The central limit theorem: many independent random events converges to a Gaussian distribution [8].	19
1.6	The model of MOS transistor with its noise sources [16].	19
1.7	Implementation of the H function [22]; as is visible, the function can be implemented in FPGA's logic efficiently.	22
1.8	Graphical abstraction of three equilibrium points: Stable 0, Stable 1 and metastable state [36].	26
1.9	The example of the chaotic system [48]: The chaos map (A), States of the chaotic system (B), Reduction of the states (C).	29
1.10	The trajectory of the double scroll attractor [53].	29
1.11	Typical Single Side Band (SSB) phase noise plot of oscillator vs. offset from carrier [8].	32
1.12	Graphical representation of the phase jitter Φ_n , period jitter Φ'_n and cycle-to-cycle jitter Φ''_n	33
1.13	Jitter accumulation over the time.	34
1.14	Jitter decomposition in to random jitter and deterministic components such as: periodic jitter (PJ), Data Dependent Jitter (DDJ), Duty Cycle Distortion (DCD) and unbounded-uncorrelated component.	36
1.15	Tail Fit TM Algorithm for measuring of random jitter [15].	37
1.16	Internal measurement method of the jitter that is suitable for FPGA according [63].	37
1.17	Result of the internal measurement method from [65].	38
3.1	The Logic Element of Altera's Cyclone III FPGA with LUT and dedicated DFF [89].	54
3.2	The part of Configurable Logic Block of Xilinx's Spartan FPGA with LUT and dedicated DFF [90].	54
3.3	The VersaTile of Actel's Fusion FPGA without LUT and dedicated DFF; there are standard gates and switchers instead [41].	55
3.4	The general PLL functional block diagram.	56
3.5	PLL of Altera's FPGAs (Cyclone III) [91].	56

3.6	PLL of Actel's FPGAs (Fusion) [41].	57
3.7	The principle of the DLL [90].	57
3.8	DCM block in the Xilinx's FPGAs [90].	58
3.9	Jitter histogram of the reference crystal oscillator(A), Jitter histogram at the output of Altera's PLL (B), Jitter histogram at the output of Xilinx's DLL (C) [111].	58
3.10	Metastability of Altera's DFF in FLEX FPGAs and MAX CPLDs [36].	59
3.11	Metastability of Xilinx's DFF in Virtex FPGAs [38].	60
3.12	Metastability of Actel's DFF in ProASIC FPGAs [37].	60
4.1	Principle of Sunnar's method; where outputs of huge number ROs are XORED together, sampled by DFF and finally post-processed by resilient function [26].	61
4.2	Principle of TRNG based in two rationally related clocks proposed by Fischer & Drutarovský in [1].	64
4.3	Principle of the TRNG based on two identical ROs proposed in [121] as response to [1].	65
4.4	Principle of the TRNG based on DLL proposed in [108] as response to [1].	65
4.5	Principle of Tkacik's TRNG; Two ROs clock the LFSR and CASR and consequently the 32 bits digest of each is XORED together, what results into 32 bit random number [122].	66
4.6	Plots of LFSR output, CASR output, LFSR and CASR XORED together [122].	67
4.7	Principle of the Golic's TRNG; Outputs of the unique Fibonacci and Galois ROs are XORED together, sampled by DFF and finally post-processed by LFSR [125].	68
4.8	Principle of the TRNG based on open delay chain; Dedicated DFF are replaced by LUTs with DFF functionality [107].	69
4.9	Principle of the stateless TRNG; After extracting single random bit, the whole system is restarted and thus possible correlation could be canceled [11].	71
4.10	Principle of the TRNG based on metastability, which utilize simple integrated circuits such as 74LS74 (DFF) and LM324 (Operational Amplifier).	71
4.11	The metastable circuitry used in principle of TRNG proposed in [40].	72
4.12	Acquiring states of TRNG based on metastability proposed in [40] . .	73
4.13	Principle of the TRNG based on PUF and metastability [127].	74
4.14	Principle of TRNG based on measuring the metastable resolution time [128].	75
4.15	The metastable system of the method of generating random bits proposed in [128].	76
4.16	Method of getting an inverter into the metastable state by making the short circuit of its input and output [35].	77

4.17 Comparison of the accumulated jitter of classic RO(A) and metastable RO(B) [35].	77
4.18 The block diagram of metastable RO TRNG (A), acquiring states of metastable RO TRNG (B) [35].	78
4.19 Principle of the TRNG based on a quantum mechanics [129].	79

List of Tables

1	Colors of noises with various SDF	18
2	FIPS 140 - the Runs Test tresholds, according [13]	44
3	Classification of various PRNGs according class according AIS 20	46
4	Results of NIST800-22 and Diehard test suites for original Sunnar's design for various length of RO.	62
5	Results of NIST800-22 and Diehard test suites for Leuven remake of Sunnar's design for various length and number of ROs.	63
6	Test results of metastable RO TRNG [35].	78
7	Evaluation of TRNG principles using following parameters:	81

Abstract

This thesis deals with possible ways of TRNG implementation into embedded digital hardware. Particularly, the main objective is to highlight FPGA as a considerable platform for TRNG synthesis for cryptographic purposes. Generally, it is not trivial task because FPGAs are not designed for this purpose. This work shows where one can find random processes in the FPGA and how to extract digital random data from them as well as how to enhance its statistical properties by correctors. Next objective of this work is to describe developed statistical and entropy tests for evaluating correct functionality and quality of the output bit-stream. Further, the work summarizes existing TRNG principles and designs suitable for FPGAs which appear much more rare in the literature in comparison to TRNG designs suitable for ASICs or traditional discrete electronics. Finally, the possible Ph.D theses are proposed.

"Any one who considers arithmetical methods of producing random digits is, of course, in a state of sin. For, as has been pointed out several times, there is no such thing as a random number there are only methods to produce random numbers, and a strict arithmetic procedure of course is not such a method."

John Von Neumann

"God does not play dice with the Universe."

Albert Einstein

Introduction

Random Number Generators (RNG) represent basic cryptographic primitives [1]. It is well known that randomness is essential for cryptography, especially for generating the underlying keys [2]. Cryptographic schemes are usually designed under the assumption of availability of an endless stream of fully unpredictable (unbiased and independent) random bits. In security applications, the unpredictability of the output also implies that it must not be possible for any attacker to observe and manipulate the generator [3], even in hostile environment. However, it is not easy to obtain such a stream. If not done properly, this may turn out to be the Achilles heel of an otherwise secure system [4]. That is why RNGs for cryptographic applications must meet stringent requirements [1], namely unpredictability and satisfactory statistical properties, etc. Recent additional security requirements represent inner testability and (provable) security (robustness and resistance against attacks) [5].

Several cases were noticed throughout history where the absences of a suitable random numbers attract considerable public attention [2]. One of the most popular spectacular attack [6] abuses defects found in the random numbers generation for SSL implemented in an early version of Netscape browser. The time and the day used as the only source of "true randomness" did not provide enough entropy [7]. Another example of disputable RNG implementation is from last Iraq war, when key information was eavesdropped thanks to knowing the backdoor behavior of RNG used in the wireless communication equipments.

Unfortunately computers and digital hardware, based on elementary logic functions, can implement only pseudo-random generators by their standard resources. A good Pseudo-Random Number Generator (PRNG) is a deterministic polynomial time algorithm that expands short (hopefully true random and well distributed) seed into long bit sequence, this distribution is polynomially indistinguishable from the uniform probability distribution. PRNGs rely on complexity and their use in cryptography, for example to generate keys, is very critical [1].

Fortunately (for cryptographers, not for high-end audio enthusiasts) there is general natural presence of a noises in the Universe. A random process that follows Gaussian distribution can be found also in semiconductors or passive electronic devices due to thermal noise, flicker noise, shot noise etc [8].

A True Random Number Generator (TRNG) has necessarily to be based on some kind of nondeterministic phenomena that could act as the source of the system randomness electronic noises (and their consequences) are usually the only stochastic phenomena that are suitable in electronic embedded implementations [3].

Despite of the slower speed of TRNGs, they are used more often in cryptographic applications than PRNGs. It is interesting to note that TRNGs are the only cryptographic primitive, which is not standardized. However, the employed principle and the generator implementation inside the cryptographic module has to be validated during the security evaluation of the cryptographic module. [5].

We will consider especially the Field Programmable Gate Array (FPGA) technology in this work because such integrated circuits are usually preferred platform for many cryptography implementation in comparison to the Application Specific Integrated Circuits (ASIC) thanks to their high versatility. Obsolete or weak crypto-algorithms and protocols can be replaced by the new ones with significantly lower efforts using FPGAs [9]. As is highly recommended to implement the whole cryptosystem into one chip, the TRNG has to utilize only available FPGAs' internal resources. The digital TRNGs are implemented by using logic gates or integrated peripherals such as Phase Locked Loop (PLL) only. All of internal components are developed by producer to have deterministic digital behavior as much as possible, but they are still an analog circuit in their nature with noise influence to their functionality within certain conditions.

This work deals where and how to gather randomness from the FPGAs' logic, how to construct reliable TRNG for cryptographic applications and how to evaluate its proper and secure functionality.

1 General Concept of TRNG

As each system is very useful to describe by a block diagram, the general conception of TRNG is described in this section. The fundamental aim of the block diagram is to understand TRNG functionality easily. In order to be able to categorize existing TRNG architectures regarding their structure, the well composed block diagram which suits (ideally) each TRNG is necessary.

1.1 Overview of Concepts

There are several conceptions of TRNG described in the literature. Each of them was more-or-less matched to TRNG architecture, that was described in a particular paper. Simplified structure of TRNG is possible to find in [10] and is depicted in Figure 1.1. WGN is source of White Gaussian Noise; H is low-pass filter which is inseparable in real electronic systems. C is AD converter (e. g. comparator) and S is sampler which creates discrete random bits in discrete time.

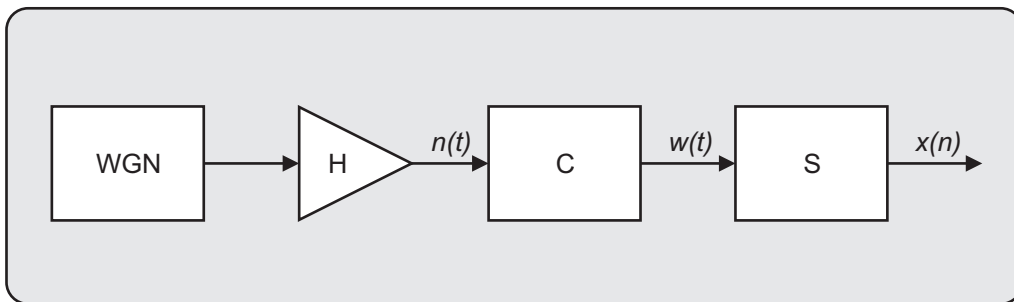


Figure 1.1: The simplified concept of the TRNG [10] with the main components: source of White Gaussian Noise (WGN), low-pass filter (H), analog-to-digital converter (C) and sampler (S).

Next conception shown in [11] was adopted from [12] and is depicted in Figure 1.2. Both conceptions were merged in order to categorize components of observed TRNG architectures more precisely and both are described in the next section.

1.2 The Practical Concept of TRNG

Fundamental idea behind generating random bit-streams by electronic means lies in sampling the stationary wide-band and low power noise (e.g. thermal or shot noise) [10]. This noise is possible to approximate by the WGN which is band-limited due to analog electronic circuitry behavior. Natural band-limitation adds a correlation into WGN which has to be removed before using the random data for cryptographic purposes. The noise has to be discretized before sampling (or vice-versa). Conversion into digital form can be processed in several ways, e.g. by quantizing, comparing, or simple, by letting the WGN to affect directly the edges

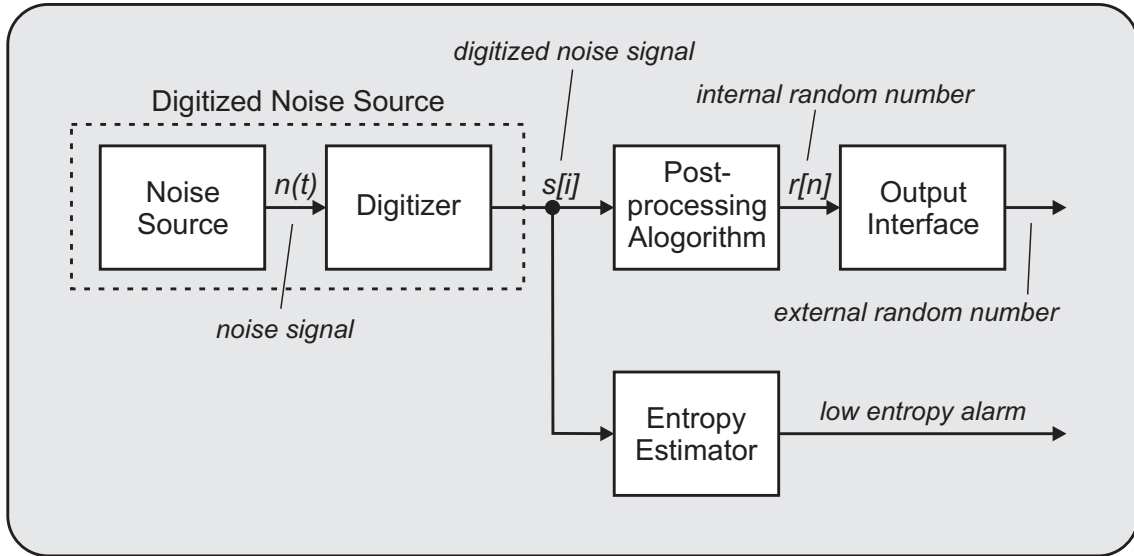


Figure 1.2: The concept of TRNG according [11] that is used in AIS 31 testing methodology.

of digital (oscillatory) signal. The mutual aim of the converter and the sampler is to extract (gather) randomness from the WGN source; therefore we can denote the combination of such two blocks as a randomness extractor. Circuitry described above, the Source of Digital Noise, is depicted in Figure 1.3. The raw output signal of this block is denoted as digitized noise signal $s[i]$.

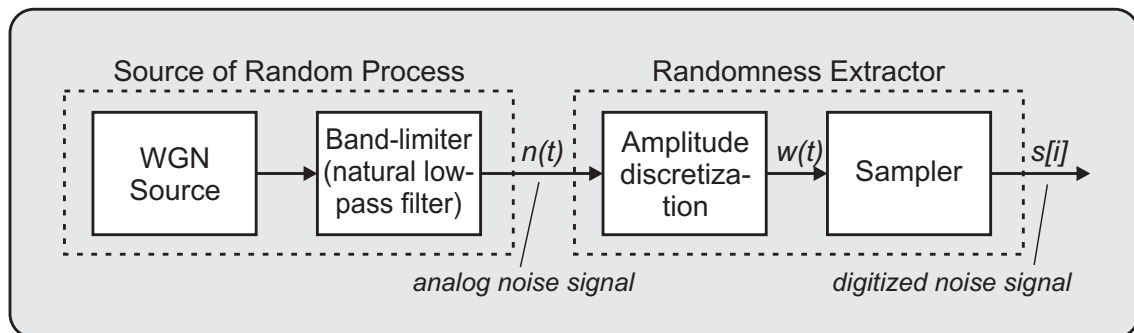


Figure 1.3: Detail of the Source of Digital Noise with its main components: the Source of Random Process is approximated by the WGN Source and natural low-pass filter and the Randomness extractor, which transforms noise in both discrete amplitude and discrete time.

Generally, due to low entropy of the WGN source, statistical properties of the $s[i]$ do not fulfill cryptographic requirements and therefore each practical TRNG has to be composed of several further building blocks, as is depicted in the Figure 1.4, in order to obtain reliable random numbers.

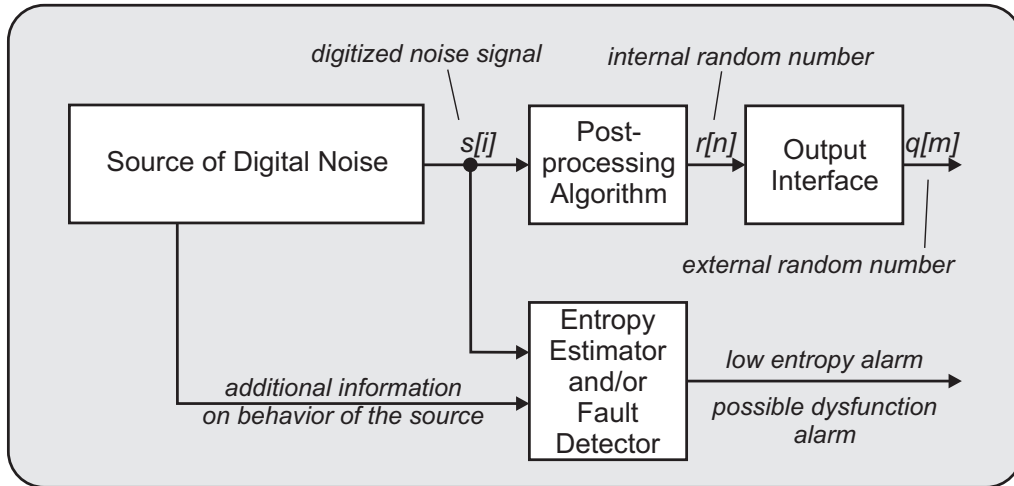


Figure 1.4: The practical block diagram of the TRNG.

As is obvious, Source of Digital Noise is the only necessary block that allows generating random data and it will be more precisely studied in the following chapters. All of rest blocks are more or less optional and their usage usually strengthens security of the specific application. Random data produced by ideal TRNG (identically distributed and independent data) has to be independent with sufficient statistical parameters, what is never the case in the reality. Therefore the aim of the Post-processing Algorithm is to improve statistical properties of the raw $s[i]$ signal e. g. by removing correlation or bias. As nothing is for free, the bit-rate of $r[n]$ could be decreased or even become irregular due to enhancing statistical parameters of raw random signal by means of some kind of postprocessor. Post-processing can also (willingly or unwillingly) adds a pseudorandom pattern into the $r[n]$ signal, what markedly improves statistical parameters or even increase the bit-rate of the generator, but on the other hand, masks lack of entropy of the source or even masks critical dysfunctions faults of the RNG source. We note, that standard randomness statistical tests (e.g. [13], [14]) have been developed for the testing of pseudorandom sequences and such pseudorandom pattern is almost impossible to detect. Therefore, the usage of this postprocessor is quite tricky and should be well considered. Consequently TRNG is degraded to Hybrid RNG (HRNG). It is a very good practice to employ Entropy Estimator and/or Fault Detector as on-line tester in order to notice unwelcome defect behavior on the fly and toggle the alarm to signalize that TRNG is out of required functionality. The unpleasant information is that the realization of entropy estimator is not trivial especially in the reconfigurable hardware. One of the solutions is to tailor the TRNG malfunction decision algorithms on the principle of TRNG particularly. The internal random bit-stream is generally not very useful in applications that need random data and so the bit-stream has to be converted to external random number $q[m]$ according specification of target bus by Output Interface block. All of the blocks mentioned above will be discussed more deeply in

the next chapters.

1.2.1 Source of Digital Noise

The aim of Source of Digital Noise is to create random bit stream $s[i]$ that fully depends on some random stationary process only. The random bit stream has to be unpredictable that means one can guess the logical level of the bit with 50% probability, even if predecessors or successors are known. Moreover random bit stream should not be able to be controlled by any means. First of all we will describe the basic model of source of digital noise and consequently we will show several particular examples how to obtain $s[i]$ mainly in digital hardware such as FPGAs.

As was mentioned Source of Digital noise is possible to describe as composite of four blocks; source of white Gaussian noise, low pass-filter, converter and sampler. It is obvious that source of white Gaussian noise is the most important block. Each electrical engineer subconsciously assumes presence of noise while designing his product regardless whether he is radio-frequency, high-end audio or high-speed communication designer. No matter of their interests a thing which they have in common is that noise is unwelcome and they would like to suppress it. The Signal to Noise Ratio (SNR) and noise power density are the most important relations for them. Thanks to their effort mentioned parameters are well studied. However, task for cryptographers is completely opposite; they would like to gather this noise in order of using it as the secret in their protocols. Therefore deeper study on noise's origin is more than necessary, because quality of the noise could has critical impact to the whole cryptographic system.

We can denote a noise as a white when Spectral Density Function (SDF) of noise is constant along whole band. The adjective white is used because of analogy with white light which has balanced amount of all spectral colors. Other color noises are shown in Table 3:

The noise is Gaussian when its distribution function takes on the characteristic of a Gaussian distribution function. It is due the accumulation of random processes including thermal noise, flicker noise, shot noise. Such noise has origin in thermal vibrations of semiconductor crystal structures, material boundaries having less than perfect valence electron mapping due to semi-regular doping density and process anomalies, thermal vibrations of conductor atoms, and many minor contributors. (cosmic radiation, etc.) [15]. All of these noise sources contribute to the total noise that appears in electronic means. The central limit theorem states that the sum of

Table 1: Colors of noises with various SDF

Color	Purple	Blue	white	Pink	Red-Brown
SDF	f^2	f	1	1/f	$1/f^2$

many independent random events (functions) converges to a Gaussian distribution, as depicted in Figure 1.5 [8].

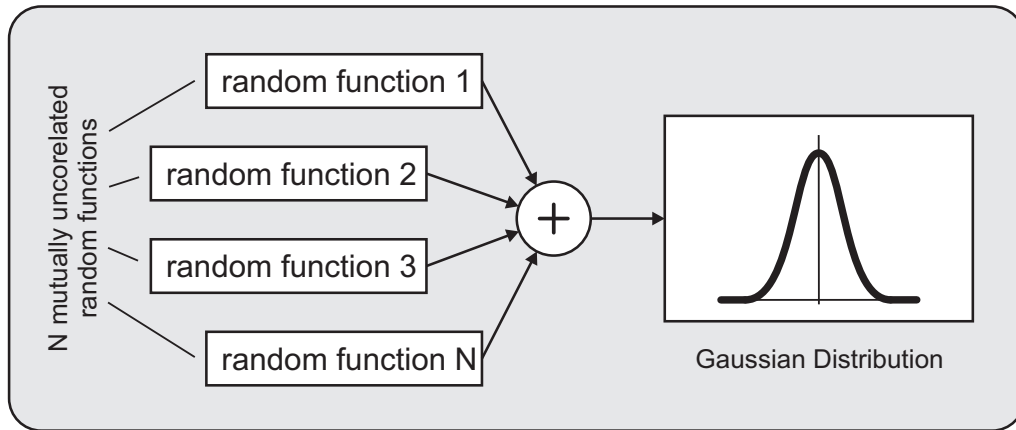


Figure 1.5: The central limit theorem: many independent random events converges to a Gaussian distribution [8].

In a crude sense, all these types of circuits are merely a collection of active devices (transistors) and passive devices (capacitors, inductors, etc.), and nearly every such device will contribute some noise [16].

Most of semiconductor logic devices are fabricated using Metal Oxide Semiconductor (MOS) transistor technology and therefore an example of its equivalent circuit showing some of noise sources is depicted in the Figure 1.6 [16]

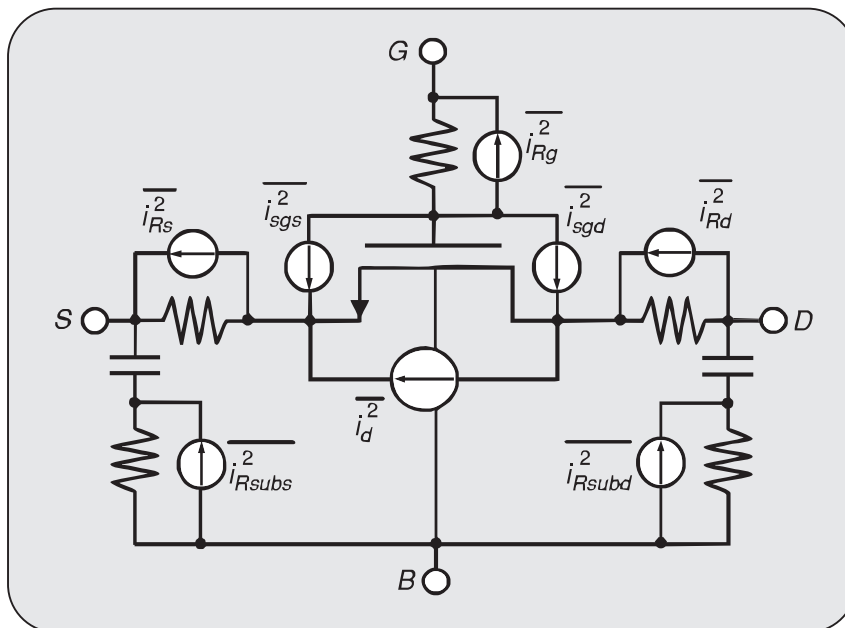


Figure 1.6: The model of MOS transistor with its noise sources [16].

In real electronic systems noise mostly does not take characteristic of white, because it is low-pass filtered by parasite capacitances and inductances, or certain semiconductor features. It is necessary to take in the account, because this filtration adds correlation into noise. For example [16] MOS transistor flicker noise demonstrate both signal dependence and frequency dependence on the order of $1/f$. More deep study of MOS transistor models are described in [16].

1.2.2 Digitized noise post-processing

TRNGs random sources, even if they are stationary, usually do not have output with sufficient statistical properties (some non-zero bias is appearing). Most of them are not stationary as well and the output samples are correlated. "Random" bit-stream produced of such samples is not recommended to use for cryptographic purposes (e.g. AIS 31 requires bias lower then 0.02). Even, as was mentioned in [17] document [18] says that TRNG should not be used for cryptographic purposes without a more complex structure as a PRNG. If one want to employ such random source, he has to employ a post-processing algorithm in order to decorrelate samples and remove the bias at least. As a result, the entropy per bit is increased. This enhancement must compresses input bit-stream. Next, we will discuss the most common linear and non-linear post-processing methods mainly for bias removing. The most of them suppose that the input sequence is independent. The most simple corrector were proposed in [19], [20] and on the other hand complex studies were done in [21], [22], [17], [23] and [24].

XOR Corrector

The simplest solution for bias reducing is so called XOR corrector [19]. It utilizes exclusive-or operation between n -bits to generate one output bit. Bias could be reduced in sufficient amount at the cost of reducing generator's output bit-rate. XOR corrector output bit-rate remains constant, what is important property. The $i - th$ output bit is:

$$y_i = x_{ni} + x_{ni+1} + \dots x_{ni+(n-1)} \pmod{2} \quad (1.1)$$

Von Neumann Corrector

In 1951, Von Neumann [20] described a procedure for generating an output sequence $z_1 z_2 \dots z_m \dots$ of statistically independent and equiprobable binary digits from an input sequence $x_1 x_2 \dots x_n \dots$ generated by a process $X_v(p)$ which chooses x_n from 0, 1 with independence and uniform bias: for all n , $x_n = 1$ with probability p , $x_n = 0$ with probability $q = 1 - p$, p unknown but fixed, $0 < p < 1$. Von Neumann used on each of this pairs $x_1 x_2, x_3 x_4, \dots$ the mapping

$$00 \rightarrow \Lambda, 01 \rightarrow 0, 10 \rightarrow 1, 11 \rightarrow \Lambda \quad (1.2)$$

where Λ represents no output digit. He defined the efficiency of this procedure as the expected number of output digits per input digit. For each input pair the probability of generating a non-null output digit z is $2pq$, so efficiency is just $2pq/2 = pq$ which is $\frac{1}{4}$ at $p = q = \frac{1}{2}$ and less sensitive. The map 1.2 is independent of the value of p , the output 0's and 1's are statistically independent and equiprobable for any p in $(0, 1)$, but the efficiency depends on p . However, this corrector produces output of non-constant bit-rate and preserves correlation of input bit-stream.

H post-processing function

Dichtl in [22] proposes some good way of post-process biased TRNG. He has begun his considerations by attacking corrector based on Quasigroup String Transformations [25]. The post-processing algorithm uses 16 bits from the physical source of randomness in order to produce 8 bit output. The output byte is produced in fixed time.

Let a_0, a_1, \dots, A_{15} be the input bits for post-processing. Let define the 8 bits b_0, b_1, \dots, b_7 by $b_i = a_i \oplus a_{(i+1) \bmod 8}$. The mapping from a_0, \dots, a_7 to b_0, \dots, b_7 defined in this way is not bijective. Only 128 different values for b_0, b_1, \dots, b_7 are possible. When the input is a perfect random number generator, one bit of entropy is destroyed. The output c_0, c_1, \dots, c_7 of the post-processing function is defined by $c_i = b_i \oplus a_{i+8}$. Dichtl names this function H .

If the input 16 bits are split into two bytes $a1$ and $a2$ it is possible to write a pseudocode as

$$H(a1, a2) = a1 \oplus RL(a1, 1) \oplus a2, \quad (1.3)$$

where $RL(x,y)$ is y -position "Rotate Left" operation of x byte.

The design of H post-processing function is shown in the Figure 1.7. Dichtl has also shown that post-processing by H function, achieves better entropy per byte comparing to single XOR corrector with the same compression ratio! This function uses only 16 XOR gates and so the hardware implementation is easy and efficient. Function is suitable for software too, because it uses whole bytes operations which represent basic instructions of common processors (XOR and RL). Dichtl improves the H function by following procedure:

$$H_2(a1, a2) = a1 \oplus RL(a1, 1) \oplus RL(a1, 2) \oplus a2, \quad (1.4)$$

and

$$H_4(a1, a2) = a1 \oplus RL(a1, 1) \oplus RL(a1, 2) \oplus RL(a1, 4) \oplus a2. \quad (1.5)$$

Dichtl shows, that if the bias of any input bits is ϵ , then the lowest power of ϵ in the bias output bytes is 3 for H , 4 for H_2 , and 5 for H_4 .

Author in [17] show simple mathematical proof of Dichtl's results. Any linear corrector has to be represented by a matrix. For $x = (x_1, x_2, \dots, x_n)$ and

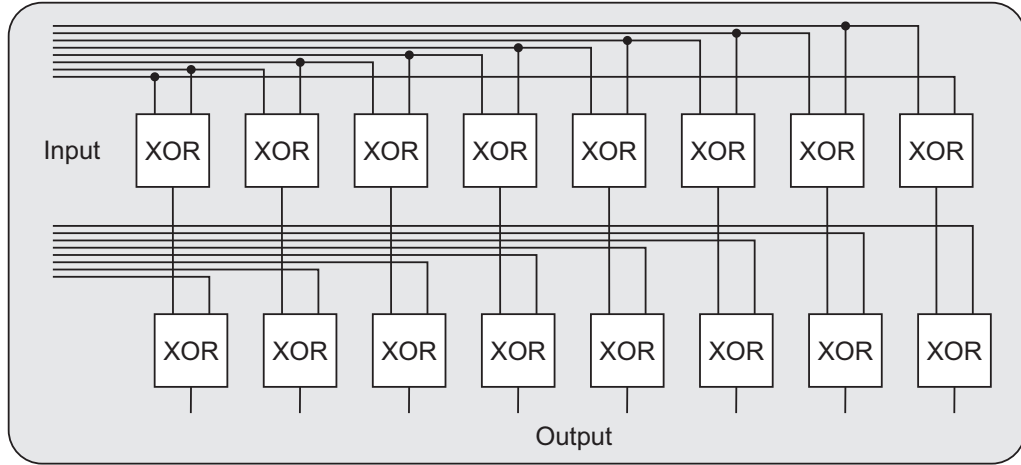


Figure 1.7: Implementation of the H function [22]; as is visible, the function can be implemented in FPGA's logic efficiently.

$y = (y_1, y_2, \dots, y_m)$, any linear binary corrector mapping n bits to m bits, is defined as the product of the vector x by the binary matrix $G = (g_{i,j})$:

$$\begin{pmatrix} g_{1,1} & g_{1,2} & \cdots & g_{1,n} \\ g_{2,1} & g_{2,2} & \cdots & g_{2,n} \\ \vdots & \vdots & \ddots & \\ g_{m,1} & g_{m,2} & \cdots & g_{m,n} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{pmatrix} \quad (1.6)$$

The Theorem, that Author in [17] forms (and proofs), says:

Let G be a linear corrector mapping n bits to m bits and $\epsilon/2$ the bias of the input bits. Then the bias of any non zero linear combination of the output bits is less or equal than $\epsilon^d/2$, where d is minimal distance of the linear code constructed by the generator matrix G .

The theorem assumes independent bits in the input and its result gives an upper bound of the output bias for an arbitrary linear corrector. In particular, the matrix corresponding to H , H_2 and H_4 are respectively generator matrix of $[16, 8, 3]$, $[16, 8, 4]$ and $[16, 8, 5]$ linear codes. Any linear $[n, m, d]$ -code provides a linear corrector with an estimation of its output bias. The compression rate of a corrector mapping m bits to n bits is defined by n/m . According author, there are no linear binary codes of length 16, dimension 8 with minimal distance greater than 5. He says that in those conditions, to minimize output bias, it is necessary to search non linear correctors. Non linear correctors were studied in his paper as well.

S post-processing function

Dichtl in [22] observes linear methods of post-processing reduce powers of input bias ϵ up to $4 - th$. That was reason of search for non-linear method. The S - post-processing function reduces powers of the ϵ up to $5 - th$. This corrector was

found by exhaustive search and the hardware implementation requires considerable amount of chip resources (32 *kB* of ROM).

Resilient Corrector

Resilient functions are specific functions used in cryptography and coding theory. They derive from boolean functions. In more informal terms, resilient functions are suitable for post-processing because "the knowledge of any m values of the input to the function does not allow one to make any better than random guess at the output" [26]. The main problem of that kind of resilient functions is that they produce one bit per n input bits. For more precisions on linear code or resilient functions in error-correcting codes theory, the reader should refer to [27] and [28].

A (n, m, t) -linear function is a function f mapping n bits to m bits such that if t input bits are fixed, there is no influence on the output. According [29] the (n, m, t) -resilient functions is a function f mapping \mathbf{F}_2^n to \mathbf{F}_2^m such that for any coordinates i_1, i_2, \dots, i_t and for any binary constants c_1, c_2, \dots, c_t and for all $y \in \mathbf{F}_2^m$, we have

$$P(f(x) = y | x_{i_1} = c_1, \dots, x_{i_t} = c_t) = 2^{-m}, \quad (1.7)$$

where x_i with $i \notin i_1, \dots, i_t$ verify $P(x_i = 1) = P(x_i = 0) = 0.5$.

In [17] author has shown, that if f is (n, m, t) -resilient function and $\epsilon/2$ is the input bias, the bias of any non zero combination of the output bits is less or equal then $e^{t+1}/2$. According [30] there exists non-linear (16,256,6) Nordstorm-Robinson code that provides a (16,8,5)-resilient function. In this case reducing of bias using resilient function seems to be more effective comparing to linear correctors.

Linear Feed-back Shift Registers (LFSRs)

LFSRs are used in many of the keystream generators that have been proposed in the literature. There are several reasons for this [5]:

- LFSRs are well-suited to hardware implementation;
- they can produce sequences of large period;
- they can produce sequences with good statistical properties; and
- because of their structure, they can be readily analyzed using algebraic techniques.

A LFSR of length L consists of L delay elements each capable of storing one bit and having one input and one output; and a clock which controls the movement of data. During each unit of time the following operations are performed:

- the content of the first delay element is output and forms part of the output sequence;
- the content of element i is moved to stage $i - 1$ for each $i, 1 \leq i \leq L - 1$; and

- the new content of the last delay element is the feedback bit which is calculated by adding together mod 2 the previous contents of a fixed subset of elements (depending on underlying polynomial) [31].

Encryption of the digitized noise signal and Applying a cryptographic hash function

Perfect statistical characteristics of most of encryption algorithms can be used to mask generator imperfections. Another common approach is to extract the randomness by applying a cryptographic hash function (or a block cipher) to the high-entropy source. As there is no mathematical guarantee of security, confidence that such constructions work comes from the extensive cryptanalytic research that has been done on these hash function. However, this research has mostly been concentrated on specific "pseudorandom" properties (e.g., collision-resistance) of these functions. It is not clear whether this research applies to the behavior of such hash functions on sources where the only guarantee is high entropy, especially when these sources may be influenced by an adversary that knows the exact hash function that is used.

1.2.3 Output speed

Nice discussion on the output speed was made in [5]. The speed is a secondary parameter (after security) in many cryptographic applications. Output speed from hundred *kbits* up to 1 *Mbps* are mostly sufficient. However, there are some speed-critical data security applications, where high speed generators are needed. For example, Quantum cryptography [32], [33] needs huge number of random bits (up to hundred of megabits per second) because of a very low efficiency of their transmission (due to underlying principle) over low-power optical channel. High speed telecommunication servers can be given as a second example. They need to generate session keys on a regular high speed basis (tens of megabits per second). For example a 10 *Gbit* Ethernet hub/server would need about 20 *Mbps* random bits to generate one 128 *bit* session key for each 64 *kB* data block in order to be able to face side channel attacks. Another aspect of the output speed is its periodicity. Some generator give random numbers in well defined time intervals, other generate output randomly. In the second case it is necessary to use First In First (FIFO) memory to accumulate generated numbers or to estimate the slowed speed available at the output. The disadvantage of the first solution is that sometimes FIFOs have to be too big, the disadvantage of the second one is that if the estimation is incorrect, random bit-stream can be unavailable in some unknown intervals.

1.3 Randomness Extracting

There are several different views on randomness extractor in the literature. Authors in [4] denote as a randomness extractor such function, "which is applied to

the high-entropy source in order to obtain an output string shorter, but random in the sense that it is distributed according to the uniform distribution”. It is obvious that authors describe more post-processing algorithm in this definition than Randomness Extractor which is adopted for this work - the mechanism that converts filtered analog WGN into a random digital signal (digitized noise signal), which can be processed by digital electronics in fully deterministic way. We can mention an AD converter as a simple example. But in full-digital integrated circuits it is impossible amplify noisy signal and convert it into digital number directly. We have to employ some of the macro-effects of filtered WGN which are possible to recognize in digital hardware. The most used macro-effect is jitter, which appears on the clock signals. Uncertainty of the result of sampling noisy clock signal close to its edges is easily available entropy source even in FPGA. The next effects, that could be used for randomness extracting is metastability or chaos. More precisely unpredictable metastable behavior of digital circuits causes by noise in semiconductors. On the other hand, chaos belongs more to analog circuits. Each of them are described in next chapters.

1.4 Metastability

Generally, standard logic devices have been developed respecting two logic states; a logical one or a logical zero respectively. Both of them are represented by different voltage levels. A forbidden area lies between them in order to resolve precisely such levels. When voltage level is changing on input of logic device, it necessarily has to cross through forbidden area. If the input signal is captured by some kind of flip-flop while it is neither one nor zero, the output of flip-flop could settle in three equilibrium states (Figure 1.8). Two of them are stable and correspond to the correct output states, while third is metastable [34] and circuit behavior becomes totally stochastic and depends on the characteristic of the circuit noise [35].

Such setup or hold violations cause the output of the flip-flop to enter a symmetrically balanced transient (metastable) state. The metastable state is manifested in a bistable device by the outputs glitching, going into an undefined state somewhere between 1 and 0, oscillating, or by the output transition being delayed for an indeterminable time. Once the flip-flop has entered the metastable state, the probability that it will still be metastable later has been shown to be an exponentially decreasing function of time [37]. This effect arises with certain probability whenever asynchronous data is registered by a clocked flip-flop and was well studied in applications such as synchronization or data recovery [37], [38], [36]. The susceptibility of a circuit to reaching this metastable state can be described using a probabilistic equation [37]. The most popular technique for reliability analysis of synchronizing circuitry is Mean Time Between Failures (*MTFB*) testing [37], [38], [36]

$$MTFB = \frac{e^{K2 \times t}}{F1 \times F2 \times K1}, \quad (1.8)$$

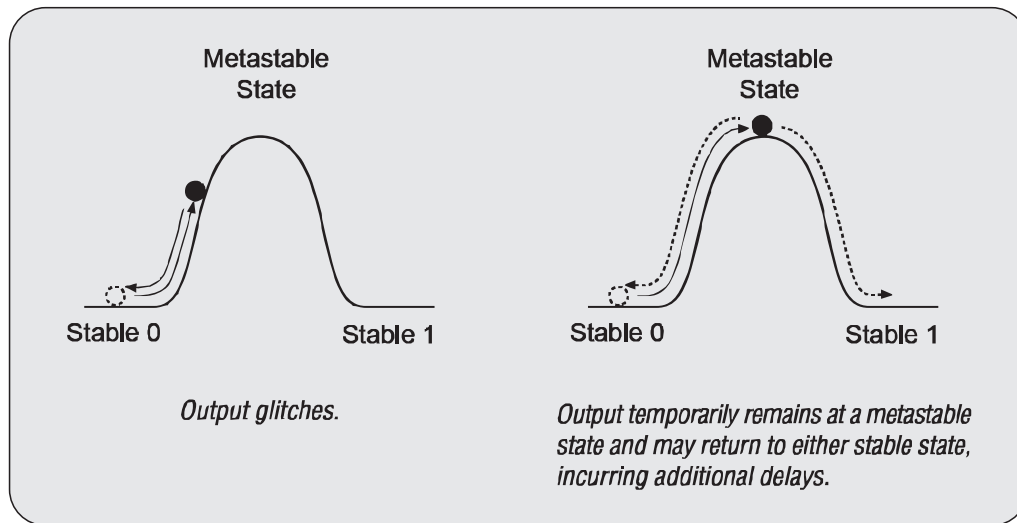


Figure 1.8: Graphical abstraction of three equilibrium points: Stable 0, Stable 1 and metastable state [36].

where:

- $K2$ is an exponent that describes the speed with which the metastable condition is being resolved,
- $K1$ represents the metastability-catching setup time window that describes the likelihood of going metastable,
- $F1$ is the frequency of the asynchronous data input,
- $F2$ is the flip-flop clock frequency, and
- t is the settling time.

Using the principle of level comparing and non regular logic output signal detecting, the number of undefined events for some time is counted. The result is the metastability characteristic which gives the frequency and time duration of metastable state. [39] Thus, a metastable state seems to be perfect entropy source [35]. There are three uncertainties at work here. The first uncertainty is if the circuit will behave normally i.e. attain a logical state after the usual delay for the flip-flop, or will the circuit enter the metastable state. The second uncertainty is the state that the flip-flop will settle into. The third uncertainty is the length of time that the circuit will remain metastable. [40]. But, due to mismatch of transistors, temperature imbalance within chip, ionizing radiation, or any other parasitic fluctuation of the output voltages, the probability that the physical flip-flop circuit will stay in the metastable region is very small. Therefore, straightforward employment

of metastability phenomena in flip-flop circuits for TRNG purpose is inefficient due to the rate occurrence of natural metastability event [35].

Up to now, metastability was very well studied in order to suppress this mysterious effect in synchronizing logic. If designer would like to employ metastability as entropy source for TRNGs, he has to find way how to force it in the logic on the demand. There was proposed only a few architectures how to do it (more discussed in next chapters) and the task still remains open. Forcing metastability in logic, especially in FPGA logic, seems to be more art than science so far.

1.5 Chaos

Gathering randomness by chaos is possible only in the analog systems such as discrete electronics or by means of Field Programmable Analog Array (FPAA) or by quantization error of the Analog to Digital Converter(ADC). On the other side, FPGA manufacturers have begun to produce FPGAs with analog peripherals, particularly Fusion family by Actel [41]. That is why the study on chaos is done in this work as well.

Theory of chaos, as branch of the theory of nonlinear dynamical systems, has brought to designers' attention a somewhat surprising fact; low-dimensional dynamical systems are capable of complex and unpredictable behavior which is intuitively very promising for random number generation. Indeed, there is now a vast number of proposals in this sense [42], [43], [44], [45], [46], [47], [48], [49] and firstly suggested by Ulam and Von Neuman in 1947 [50].

Such systems are possible to materialize by electronics means, known as chaotic circuits. Contrary to traditionally used sources of randomness they use well-defined analog deterministic circuit that exhibits chaos. The dynamics of this circuit are highly sensitive to initial conditions (popularly referred to as the butterfly effect [51]). As a result of this sensitivity, which leads an exponential growth of perturbations in the initial conditions, the behavior of chaotic systems appears to be random. This happens even though these systems are deterministic, meaning that their future dynamics are fully defined by their initial conditions, with no random elements involved. However, the slightest uncertainty about initial state (which is unavoidable in all analogue implementations of chaos systems) causes a very large uncertainty after very short time. This is somehow similar to seeding PRNGs, with the major difference that it brings infinite rather finite entropy and the same seed cannot ever be repeated twice. Additionally, as it has been shown if the state variable of chaos system is not available to the observer, and the chaos-based system map is well designed, the output of the system cannot be predicted at all.

However, it must be remarked that although noise like, the analog outputs of a chaotic source are generally unevenly distributed and always correlated. Hence, it is not possible to use them directly for TRNGs. Consequently, some processing is required for digitalization, de-correlation, and balancing. Algorithms that can

blindly perform a similar task (i.e. extract independent and unbiased symbols from weak source of randomness) are now widely studied in [42].

Chaos theory is usually limited to the class of chaotic models represented by the so-called Piece-Wise-Affine Markov (PWAM) maps for purpose of TRNG designing. Nice analysis was done in [52]. These are one-dimensional systems in which the state variable is updated as:

$$x_{n+1} = M(x_n) \quad (1.9)$$

Where $M : [-1, 1] \rightarrow [-1, 1]$ is such that

- it is non-singular, in the sense that sets of non-null measure cannot be mapped in sets of null measure by M ;
- it is exact, in the sense that any set of non-null measure is eventually expanded by M into a set whose measure is the same as that of $[-1, 1]$;
- an interval partition $X_i, i = 0, \dots, p - 1$ of $[-1, 1]$ exists so that:
 - M is affine in each X_i ,
 - Partition points are mapped into partition points.

Intuitively, properties 1 and 2 assure that the behavior is sufficiently chaotic. More interestingly, property 3 guaranties that the system dynamics embeds Markov chain.

An example of four-partition PWAM map [48] is shown in Figure 1.9(A), and appropriate Markov chain in Figure 1.9(B), and with reduction into two states in Figure 1.9(C).

In many cases chaotic behavior is found only in a subset of phase space. The cases of most interest arise when the chaotic behavior takes place on an attractor. An attractor is a set to which a dynamical system evolves after a long enough time. That is, points that get close enough to the attractor remain close even if slightly disturbed. A trajectory of the dynamical system in the attractor does not have to satisfy any special constraints except for remaining on the attractor. The trajectory may be periodic or chaotic or of any other type. An example of double-scroll attractor is in the Figure 1.10 [53].

Usual analog platforms for implementing chaos based TRNG are: ASICs [44], [45], [46], [47], [48], [49], FPAAAs [42], [43], PSoC [54], ADC [44], [55], FPGA [56] or discrete electronics [53].

1.6 Jitter

1.6.1 Introduction to Jitter

Generally, timing jitter is the unwelcome companion of all electrical systems that use voltage transitions to represent timing information, but as it was mentioned in the

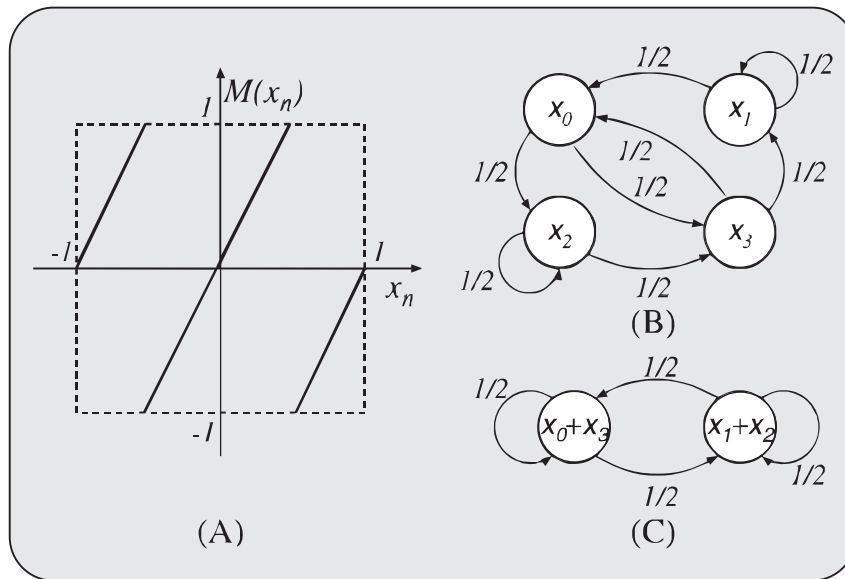


Figure 1.9: The example of the chaotic system [48]: The chaos map (A), States of the chaotic system (B), Reduction of the states (C).

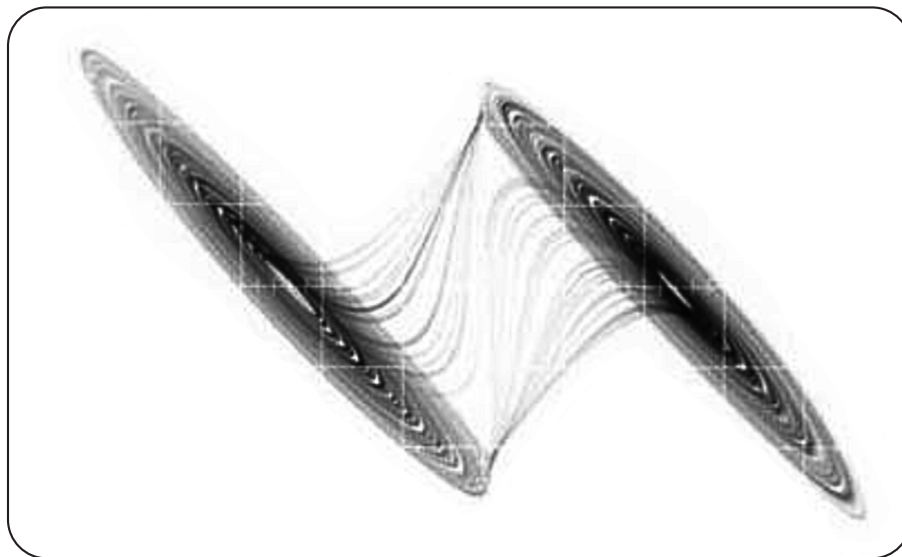


Figure 1.10: The trajectory of the double scroll attractor [53].

previous section, the clock jitter represents one of the best sources of randomness, which can be employed reliably in logic devices. A deeper study of jitter is needed throughout the system design, when jitter is used as randomness source. The jitter is a general term used to specify clock or oscillations uncertainty in the time domain. This simple and intuitive definition is provided by the [57]: "Jitter is defined as the short-term variations of a digital signal's significant instants from their ideal posi-

tions in time.” In general, Jitter is the deviation of timing edges from their ”correct” locations of the clock signal. This definition works well for understanding what jitter is, the problem is that there are various measurement techniques employed in different applications and that several authors define them differently according to their specific needs. As a form of noise, jitter must be treated as a random process and characterized in terms of its statistics. Only by thoroughly analyzing jitter is it possible isolate true random jitter. This analysis takes the form of jitter visualization, decomposition and mathematical model. Understanding what jitter is, and how to characterize it, is the first step to successfully deploying random number generators that meet their security requirements.

1.6.2 The noisy oscillator

Standard IEEE 1139-1988 [58] covers the fundamental metrology for describing random instabilities of importance to frequency and time metrology. Quantities covered include frequency, amplitude, and phase instabilities; spectral densities of frequency, amplitude and phase fluctuations; variances of frequency and phase fluctuations; time prediction and confidence limits when estimating the variance from finite data set.

The instantaneous output of the oscillator can be expressed as:

$$V(t) = (V_0 + \varepsilon(t)) \sin(2\pi v_0 t + \phi(t)) \quad (1.10)$$

Where V_0 is the nominal peak voltage amplitude, $\varepsilon(t)$ is the deviation from the nominal amplitude, v_0 is the nominal frequency, and $\phi(t)$ is the phase deviation from the nominal phase $2\pi v_0 t$. Phase instability, defined in terms of the instantaneous phase deviation $\phi(t)$, can also be expressed in units of time, as:

$$x(t) = \phi(t)/2\pi v_0. \quad (1.11)$$

With this definition, the instantaneous, normalized frequency deviation is

$$y(t) = \frac{dx(t)}{dt}. \quad (1.12)$$

In the frequency domain, frequency, amplitude and phase instabilities can be defined or measured by one-sided spectral densities. The unit of measure of time instability is the spectral density of fractional time fluctuations $S_x(f)$, given by

$$S_x(f) = x^2(f) \frac{1}{BW} \quad (1.13)$$

Where $x(f)$ is the fractional time deviation as a function of Fourier frequency, BW is the measurement system bandwidth in Hz , and the units $S_x(f)$ are in s^2/Hz .

Phase instability can be characterized by the spectral density of phase fluctuations $S_\phi(f)$, given by:

$$S_\phi(f) = \phi^2(f) \frac{1}{BW} = (2\pi v_0)^2 S_x(f). \quad (1.14)$$

The units of $S_\phi(f)$ are rad^2/Hz .

The one of the most common and convenient measurements for phase deviation is known as the single-sideband phase noise $\mathcal{L}(f)$. Phase noise and instability values are related according to

$$\mathcal{L}(f) = \frac{1}{2}S_\phi(f) = \pi v_0 S_x(f). \quad (1.15)$$

Usually, $\mathcal{L}(f)$ is given in dB as $10\log(\mathcal{L}(f))$, and its units are dB below the carrier in the bandwidth of one Hz (dBc/Hz).

The stability of oscillators is typically characterized in terms of either jitter, measured by σ^2 variance in time domain or the phase noise $\mathcal{L}(f)$ measured in frequency domain. Due to the relationship between the time domain and frequency domain fluctuations, the root mean-square time deviation $\overline{x^2(t)}$ is related to the spectral density according to

$$\overline{x^2(t)} = \int_0^\infty S_x(f)df. \quad (1.16)$$

The very common definition for jitter known as Root Mean Square (RMS) phase jitter is therefore related to the oscillator phase noise according to

$$\phi_{rms}^2 = \overline{\phi^2(t)} = \lim_{T \rightarrow \infty} \frac{1}{2T} \int_{-T}^T |\phi(t)|^2 dt = \int_0^\infty S_\phi(f)df = 2 \int_0^\infty \mathcal{L}(f)df. \quad (1.17)$$

The typical single side band phase noise plot of oscillator vs. offset from carrier is shown in Figure 1.11 [8].

1.6.3 Jitter Fundamentals

For clock applications, time-domain measurements are preferable, since most specifications involve time-domain values. There are three basic measurements of the jitter in the time domain: phase jitter, period jitter, and cycle-to-cycle jitter [Wed06].

Phase jitter is defined as the difference between the measured phase advance of the clock or oscillator versus the phase advance from an ideal clock with period T_0 . The phase measurement is made in discrete time intervals nT_0 , where $n = 0, 1, 2, \dots$. An ideal oscillator would have a clock cycle occurrence when $t = nT_0$, but the noisy oscillator with time deviation $x(t)$ will have a time-shift error. The cycle occurrence of the noisy clock using ideal period can be expressed:

$$t_n = nT_0 + x(nT_0). \quad (1.18)$$

Phase jitter Φ_n is defined as the difference between this measured time and the ideal period time $t = nT_0$:

$$\Phi_n = t_n - nT_0 \quad (1.19)$$

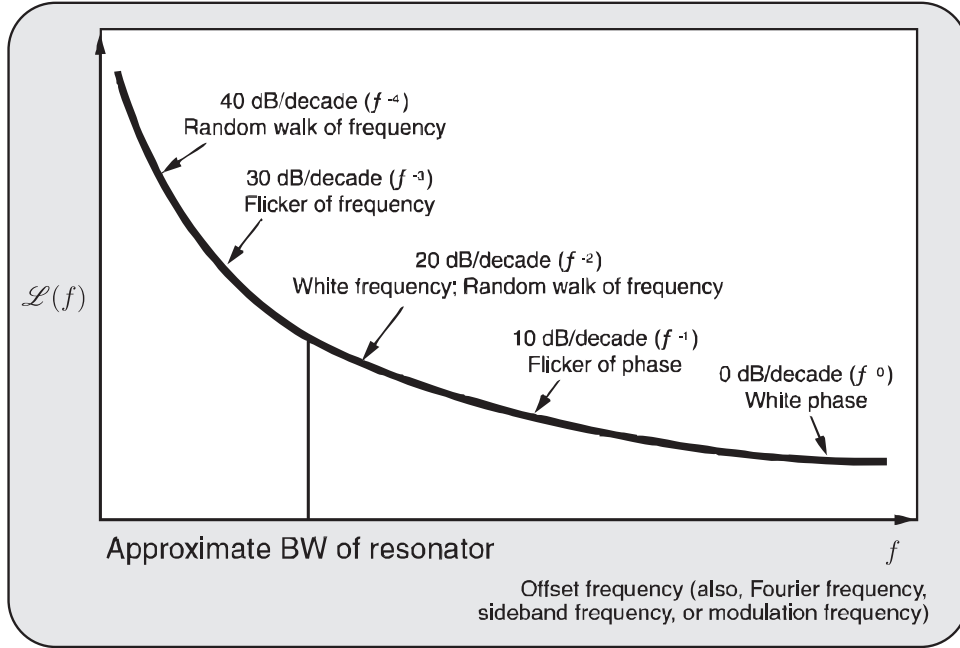


Figure 1.11: Typical Single Side Band (SSB) phase noise plot of oscillator vs. offset from carrier [8].

It is equivalent to a discrete measurement of the time deviation sampled at each ideal clock period.

Period jitter Φ'_n is defined as the difference between measured adjacent clock periods and the ideal clock period T_0 . It is also a discrete measurement, made at period intervals defined as

$$\Phi'_n = (t_n - t_{n-1}) - T_0 = \Phi_n - \Phi_{n-1} \quad (1.20)$$

Period jitter can therefore be considered the first difference function of the phase jitter. Cycle-to-cycle jitter is the measured difference between two successive clock periods

$$\Phi''_n = (t_n - t_{n-1}) - (t_{n-1} - t_{n-2}) = \Phi'_n - \Phi'_{n-1} \quad (1.21)$$

It can be considered the first difference function of the period jitter or the second difference function of the phase jitter. Phase jitter, period jitter, and cycle-to-cycle jitter provide three different, yet related ways to characterize a noisy clock. Figure 1.12 shows their differences when applied to an example clock signal. There is an obvious analogy to distance, speed, and acceleration.

- Phase jitter is a measure of the relative distance the actual (measured) phase has moved from the ideal clock phase.

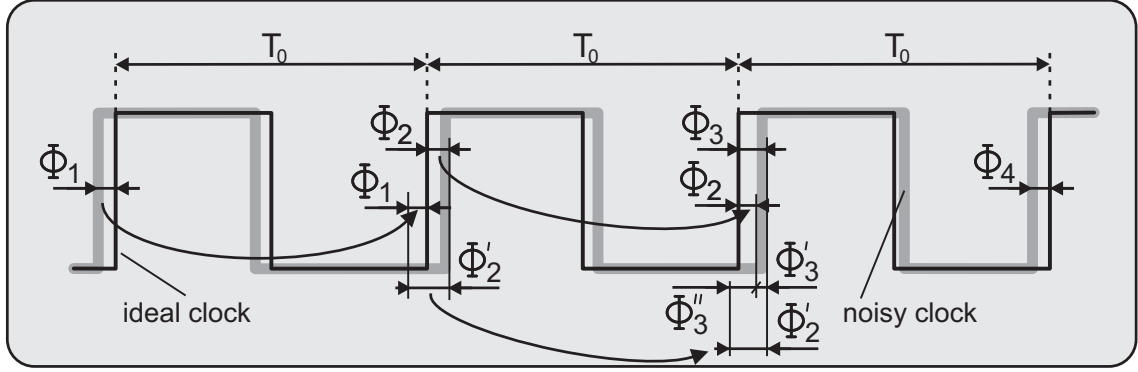


Figure 1.12: Graphical representation of the phase jitter Φ_n , period jitter Φ'_n and cycle-to-cycle jitter Φ''_n .

- Period jitter represents the speed at which the measured phase is changing and the relative distance the measured period has moved from the ideal period.
- Cycle-to-cycle jitter measures the acceleration of the phase away from or towards the ideal phase, as well as the speed at which the measured period is changing.

1.6.4 Jitter accumulation over the time

While phase noise measures clock uncertainty in the frequency domain, the jitter measures the uncertainty in time domain [16].

Jitter will result in clock signal where the actual timing differs from the ideal. Consider the time interval between two arbitrary chosen clock edges separated by N clock cycles. Let the ideal edge delay for the N clock cycles be written as $\tau = N \cdot T_0$. The two actual clock edges and their time separation are now influenced by the fluctuating time deviation $x(t)$ and can be written as

$$t_{edge1} = t + x(t) \quad (1.22)$$

$$t_{edge2} = (t + \tau) + x(t + \tau) \quad (1.23)$$

$$t_{edge2} - t_{edge1} = \tau + [x(t + \tau) - x(t)] \quad (1.24)$$

The accumulated error between the two clock edges is known as the Time Interval Error (*TIE*). The *TIE* for an N clock cycle interval is defined as (Figure 1.13)

Thus, *TIE* is a measure of the accumulated timing error between the endpoints t and $t + \tau$ over the $\tau = N \cdot T_0$ time interval. If the measurement is started then $t = 0$ and $x(t) = 0$. The result is a *TIE* identical to phase jitter $\Phi = x(N \cdot T_0)$.

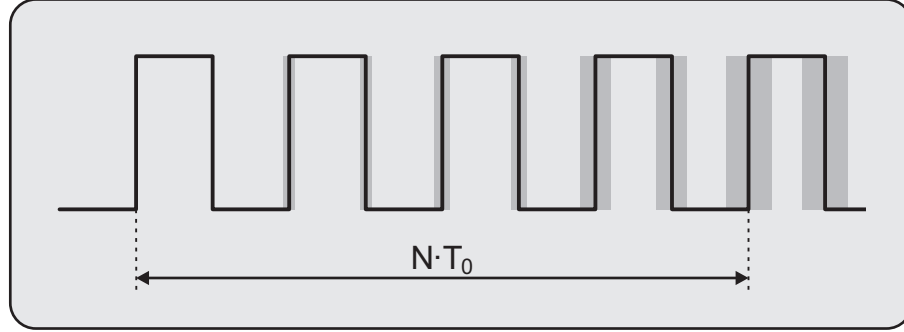


Figure 1.13: Jitter accumulation over the time.

This is why TIE and phase jitter are sometimes used interchangeably. Taking the mean-square average of TIE , which is called Timing jitter, gives:

$$\overline{TIE(t, \tau)^2} = \frac{1}{2\pi\nu_0} [\overline{\phi(t + \tau)^2} - 2\overline{\phi(t + \tau)\phi(t)} + \overline{\phi(t)^2}] \quad (1.25)$$

Typically we can assume that stationary random noise sources will result in average phase deviation values that are independent of time shifts. They therefore share the same RMS value $\phi_{rms}^2 = \overline{\phi^2(t)} = \overline{\phi^2(t + \tau)}$ and we can consider TIE to be a function of just the observation interval i.e. $\overline{TIE(t, \tau)^2} = \overline{TIE(\tau)^2}$. The remaining term represents the autocorrelation function of the phase deviation $R_\phi(\tau) = \overline{\phi(t + \tau)\phi(t)}$. It is also possible to expect, with random noise that mean-square average value for TIE equates to a σ^2 variance for its Gaussian distribution. Consequently, the definition for timing jitter (or accumulated jitter) as the variance equal to $\overline{TIE(\tau)^2}$ and written as

$$\sigma_{TIE}^2(\tau) = \frac{1}{\pi\nu_0} [\phi_{RMS}^2 - R_\phi(\tau)] \quad (1.26)$$

The N -cycle timing jitter is therefore a function of RMS phase jitter, which does not depend on the time interval, and a function of the interval-dependent autocorrelation function. Period jitter is simply timing jitter over an interval of one period ($\tau = T_0$):

$$\sigma_{PER}^2 = \sigma_{TIE}^2(T_0) \quad (1.27)$$

Cycle-to-cycle jitter represents the error between adjacent periods. It is equal to the difference in the TIE at $\tau = 2T_0$ and $\tau = T_0$. The variance for cycle-to-cycle jitter can be calculated as

$$\sigma_{CTC}^2 = 4\sigma_{PER}^2 - \sigma_{TIE}^2(2T_0). \quad (1.28)$$

1.6.5 Jitter decomposition

Generally, jitter is composed of two major components, one that is possible to predict and one that is random [8]. The predictable component of jitter is called deterministic jitter. The random component of jitter is called random jitter. Random jitter comes from the random phase noise, while deterministic jitter comes from the noise-like process. Random jitter (RJ) is characterized by a Gaussian (normal probability) distribution and assumed to be unbounded. As a result, it generally affects long-term device stability. Because peak-to-peak measurements take a long time to achieve statistical significance, random jitter is usually measured as a RMS value.

Deterministic jitter (DJ) has a non-Gaussian probability density function (PDF) and is characterized by its bounded peak-to-peak (pk-pk) amplitude. Deterministic jitter is expressed in units of time, pk-pk. The following are examples of deterministic jitter:

- Periodic Jitter (PJ) or sinusoidal-e.g., caused by power supply feedthrough;
- Data Dependent Jitter (DDJ)-e.g., from channel dispersion of filtering;
- Duty Cycle Distortion (DCD)-e.g., from asymmetric rise/fall times;
- Sub-harmonic(s) of the oscillator-e.g., from straight-multiplication oscillator designs;
- Uncorrelated periodic jitter-e.g., from crosstalk by other signals (e.g. Electro-Magnetic Interference (EMI));
- Correlated periodic jitter.

Total jitter (TJ) is the combination of all independent jitter components:

$$Total\ jitter(TJ) = random\ jitter(RJ) + deterministic\ jitter(DJ) \quad (1.29)$$

Jitter decomposition is shown in the Figure [1.14](#).

1.6.6 Jitter measuring

Jitter measuring was well studied mainly by developers of high-speed communication links [59],[60],[61],[62]. Generally, jitter is possible to measure from outside or inside of FPGA. Both techniques have their pros and cons. When measuring outside, a special high-speed oscilloscope with jitter measurement feature is usually necessary. The jitter is possible to display as histogram with resolution of several picoseconds. It is also possible to show jitter evolution in time, when one wants to see time evolution of jitter. The mathematical treatment is shown in Jitter Fundamentals section. Practically, jitter is composed of RJ and DJ and therefore it is quite difficult to estimate standard deviation of RJ, because histogram will not fit into Gaussian

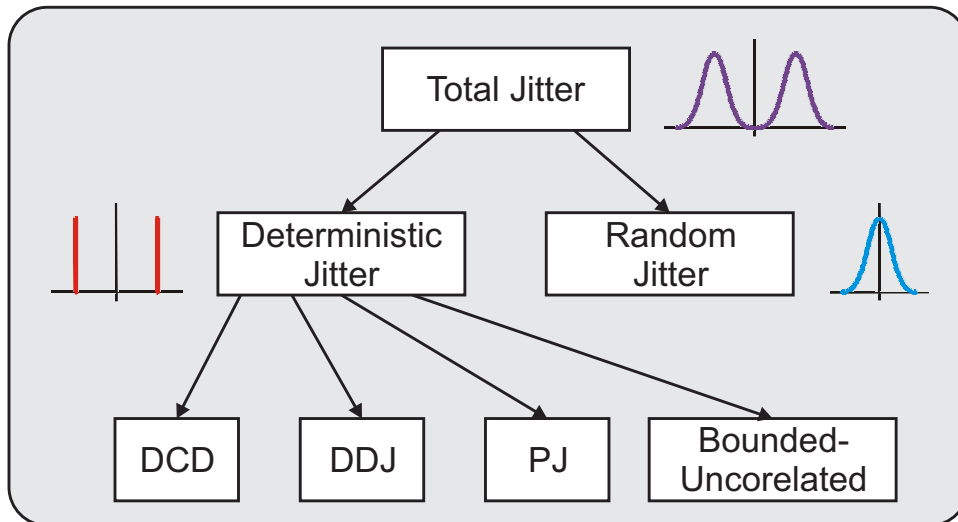


Figure 1.14: Jitter decomposition into random jitter and deterministic components such as: periodic jitter (PJ), Data Dependent Jitter (DDJ), Duty Cycle Distortion (DCD) and unbounded-uncorrelated component.

distribution. One of possible way is (Patent Pending) TailFitTM algorithm [15] which observes "Ltail" and "Rtail" (left and right ends of histogram, which always have Gaussian characteristic, when random composite is included). The screen shot of this algorithm at work is shown in Figure 1.15.

External methods have one serious disadvantage discussed in [63]. Measured signal gains much more jitter by FPGA input/output circuitry, when travelling from its source to the input of the oscilloscope. Therefore internal methods are necessary for gaining more reliable results. The method was published in [63] and is shown in Figure 1.16. The time-base generator is used to generate measurement interval (signal ena), which is used to enable ring oscillator output. The clock generated in ring oscillator is used to increment an n -bit counter, which is stopped at the end of measurement interval. The output value of the counter is proportional to the ring oscillator frequency.

1.6.7 Jitter in Ring Oscillators

Ring Oscillators (RO) are one of the most favorite noisy clock source in FPGA TRNG designs. RO is usually used also in PLL circuitry as VCO in ASICs [64]. PLLs are next favorite noisy clock source in TRNG designs. Therefore this section deals with RO features.

The jitter generated in RO clocks as a source of randomness was studied in [65]. There was also shown a simple physical model of jitter sources to show that the random jitter accumulates slower than the global and manipulable deterministic jitter. Authors also emphasize that most of today's TRNG based on ROs do not

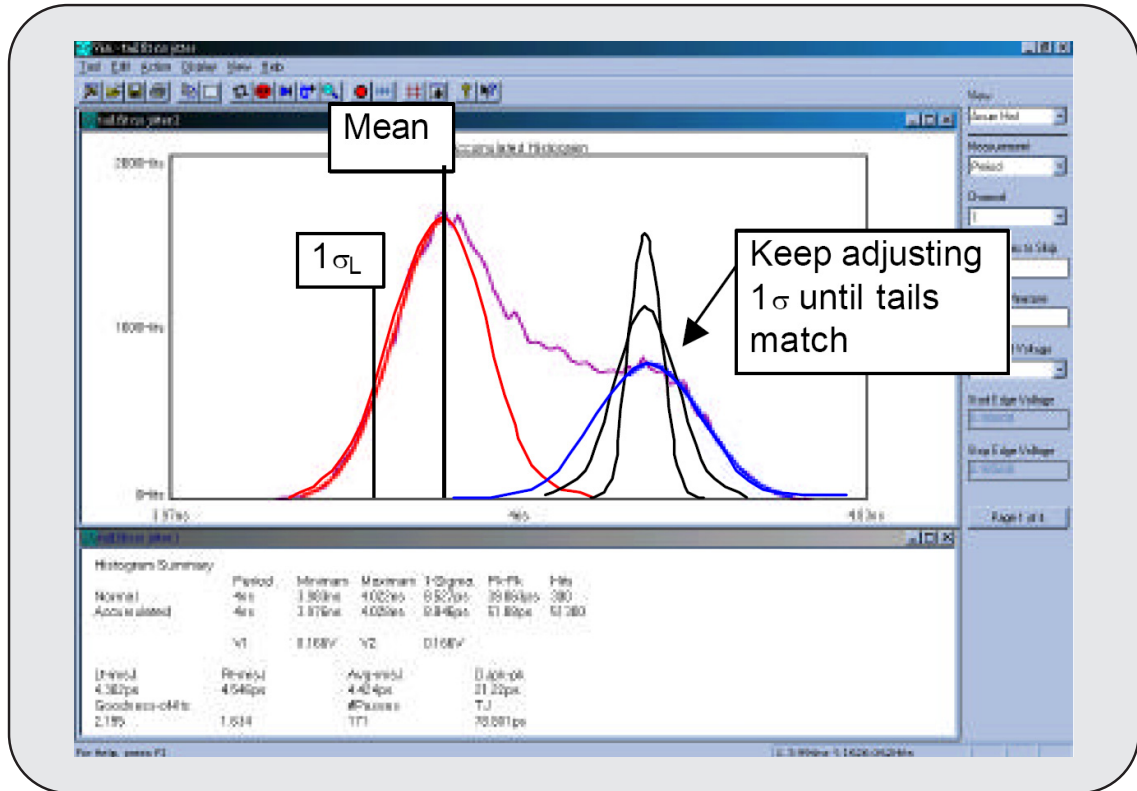


Figure 1.15: Tail FitTM Algorithm for measuring of random jitter [15].

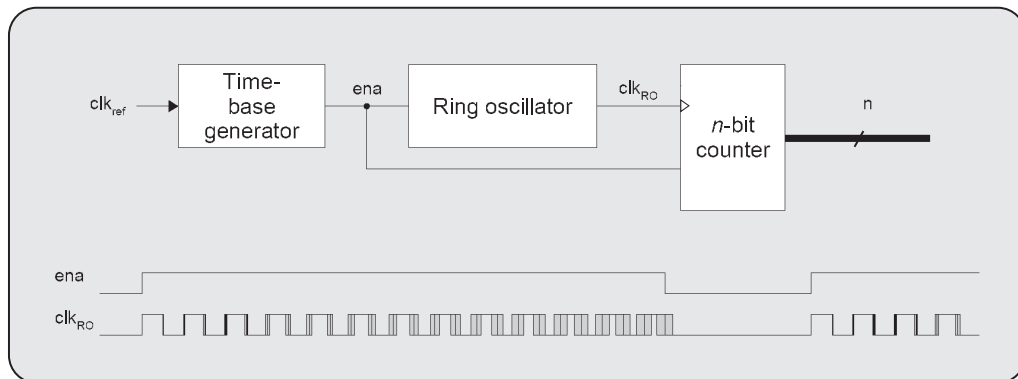


Figure 1.16: Internal measurement method of the jitter that is suitable for FPGA according [63].

have any countermeasures against possible attack which utilizes such deterministic jitter. In order to prevent this, authors propose simple but efficient countermeasure. The method was validated using the behavioral VHDL model and was shown to be efficient in hardware too. Native frequency (or jitter) of ROs are highly sensible to level of power supply and temperature changes. Such perturbation has impact to the

whole FPGA in the same way. The result is, that risk can be significantly reduced when the sampling clock in TRNG is generated by RO, since it depends on the global jitter sources in the same way as RO used to generate randomness. Accumulation of deterministic and random jitter from their experiments on Altera NiosII [66] evaluation board is shown in Figure 1.17. Their advice is also to implement all ROs in the same way (in order to have similar frequencies) and located close to each other (to be affected by the global jitter source in the same way). However, as was shown in [67] if ROs are implemented in the strict same way and as close as possible, they tend to match each other completely.

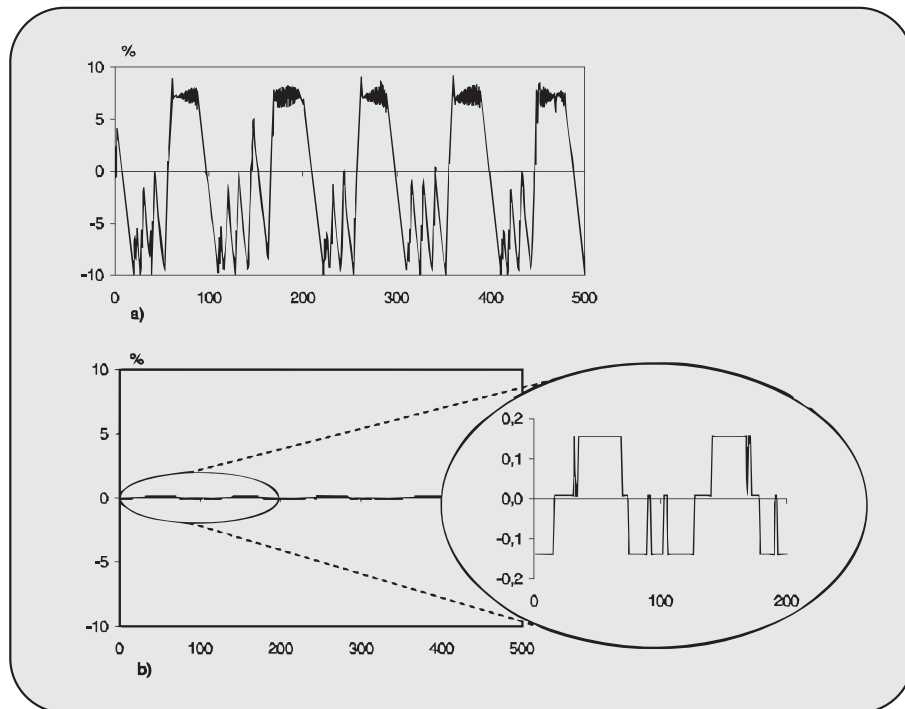


Figure 1.17: Result of the internal measurement method from [65].

Other very deep studies of jitter and noise models of ROs (implemented mainly in the ASICs) are in [68],[64],[69], [70].

2 Testing and evaluation of TRNG

As was mentioned above, common cryptosystems employ key that must be generated in a random fashion. Many cryptographic protocols also require random or pseudorandom inputs at various points, e.g. for auxiliary quantities used in generating digital signatures, or for generating challenges in authentication protocols. It is obvious, that strength of the cryptosystems often depends on the quality of randomness and therefore it is necessary to investigate it. Generally, a random bit stream generated by such generator has to pass various tests in order to use it for data security system in order to detect abnormalities coming from either poorly designed generator or an attack. Bit stream can be evaluated internally or externally. Whereas internal and online evaluation is preferred, it is not possible implement it in many cases. It was designed several batteries of test for this purpose by both individuals and institutions. We can mention basic test suite proposed by Menezes et. al. in [71] next, the Diehard battery of tests developed by Prof. Georges Marsaglia [72]. National institutions propose various requirements and tests for RNG as well and we can mention (U.S.) National Institute of Standards and Technology (NIST) and their documents [73], [13], [74], [14], [18], or (German) Bundesamt für Sicherheit in der Informationstechnik (BSI) and their documents [75] and [12].

It is interesting, that statistical tests on random numbers have begun to appear in the middle of 20-th century, e.g. in [76] is mention the frequency test and the poker test, which are used up to now.

2.1 Basic Statistical Tests

Menezes et.al. describes five basic tests for random sequences in [71]. There are five statistical tests that are commonly used for determining whether the binary sequence s possesses some specific characteristics that a truly random sequence would be likely to exhibit. It is emphasized again that the outcome of each test is not definite, but rather probabilistic. If a sequence passes all five tests, there is no guarantee that it was indeed produced by a random bit generator. Let $s = s_0, s_1, s_2, \dots, s_{n-1}$ be a binary sequence of length n . Recommended length of n is much more than 10,000 bits.

- Frequency test(monobit test)

The purpose of this test is to determine whether the number of '0' and '1' in s are approximately the same, as would be expected for a random sequence. Let n_0, n_1 denote the number of '0' and '1' in s , respectively. The statistic used is

$$X_1 = \frac{(n_0 - n_1)^2}{n} \quad (2.1)$$

which approximately follows a χ^2 distribution with 1 degree of freedom if $n \geq 10$.

- Serial test (two-bit test)

The purpose of this test is to determine whether the number of occurrences of 00, 01, 10, and 11 as subsequences of s are approximately the same, as would be expected for a random sequence. Let n_0, n_1 denote the number of '0' and '1' in s , respectively, and let $n_{00}, n_{01}, n_{10}, n_{11}$ denote the number of occurrences of 00, 01, 10, 11 in s , respectively. Note that $n_{00} + n_{01} + n_{10} + n_{11} = (n - 1)$ since the subsequences are allowed to overlap. The statistic used is

$$X_2 = \frac{4}{n-1}(n_{00}^2 + n_{01}^2 + n_{10}^2 + n_{11}^2) - \frac{2}{n}(n_0^2 + n_1^2) + 1 \quad (2.2)$$

which approximately follows a χ^2 distribution with 2 degree of freedom if $n \geq 21$.

- Poker test

Let m be a positive integer such that $\lfloor \frac{n}{m} \rfloor \geq 5 \cdot (2^m)$, and let $k = \lfloor \frac{n}{m} \rfloor$. Divide the sequence s into k non-overlapping parts each of length m , and let n_i be the number of occurrences of the i^{th} type of sequence of length m , $1 \leq i \leq 2m$. The poker test determines whether the sequences of length m each appear approximately the same number of times in s , as would be expected for a random sequence. The statistic used is

$$X_3 = \frac{2^m}{k} \left(\sum_{i=1}^{2^m} n_i^2 \right) - k \quad (2.3)$$

which approximately follows a χ^2 distribution with $2m - 1$ degrees of freedom. Note that the poker test is a generalization of the frequency test: setting $m = 1$ in the poker test yields the frequency test.

- Runs test

The purpose of the runs test is to determine whether the number of runs (of either zeros or ones) of various lengths in the sequence s is as expected for a random sequence. The expected number of gaps (or blocks) of length i in a random sequence of length n is $e_i = (n - i + 3) = 2^{i+2}$. Let k be equal to the largest integer i for which $e_i \geq 5$. Let B_i, G_i be the number of blocks and gaps, respectively, of length i in s for each i , $1 \leq i \leq k$. The statistic used is

$$X_4 = \sum_{i=1}^k \frac{(B_i - e_i)^2}{e_i} + \sum_{i=1}^k \frac{(G_i - e_i)^2}{e_i} \quad (2.4)$$

which approximately follows a χ^2 distribution with $2k - 2$ degrees of freedom.

- Autocorrelation test

The purpose of this test is to check for correlations between the sequence s and (non-cyclic) shifted versions of it. Let d be a fixed integer, $1 \leq d \leq \lfloor n/2 \rfloor$. The number of bits in s not equal to their d -shifts is $A(d) = \sum_{i=0}^{n-d-1} s_i \oplus s_{i+d}$, where \oplus denotes the XOR operator. The statistic used is

$$X_5 = 2(A(d) - \frac{n-d}{2})/\sqrt{n-d} \quad (2.5)$$

which approximately follows an $N(0, 1)$ distribution if $n-d \geq 10$. Since small values of $A(d)$ are as unexpected as large values of $A(d)$, a two-sided test should be used.

2.2 Diehard battery of tests

The Diehard battery of tests was developed in 1996 by Prof. Georges Marsaglia from the Florida State University for testing randomness of sequences of numbers [77]. It was supposed to give a better way of analysis in comparison to original FIPS statistical tests. It consists of 15 different, independent statistical tests. This battery of tests is usually considered to be "the most powerful general test of randomness": many of the claimed-good generators would not pass some of these tests. As the NIST test suite, results of these tests are P - values and are supposed to be uniformly distributed. Unlike the NIST test suite, the test is considered to be successful when the P - value is in range where $[0 + \frac{\alpha}{2}, 1 - \frac{\alpha}{2}]$ is the level of significance of the test. For example, with a level of significance of 5%, P - value are expected to be in [0.025, 0.975]. Note that if the P - value is not in this range, it means that the null hypothesis for randomness is rejected even if the sequence is truly random. These tests are:

- Birthday spacings: Choose random points on a large interval. The spacings between the points should be asymptotically Poisson distributed. The name is based on the birthday paradox.
- Overlapping permutations: Analyze sequences of five consecutive random numbers. The 120 possible orderings should occur with statistically equal probability
- Ranks of matrices (three kinds of rank matrices tests): Select some number of bits from some number of random numbers to form a matrix over 0,1, then determine the rank of the matrix. Count the ranks.
- Monkey tests: Treat sequences of some number of bits as "words". Count the overlapping words in a stream. The number of "words" that don't appear should follow a known distribution. The name is based on the infinite monkey theorem.

- Count the 1s (two kinds of tests): Count the 1 bits in each of either successive or chosen bytes. Convert the counts to "letters", and count the occurrences of five-letter "words"
- Parking lot test: Randomly place unit circles in a 100 x 100 square. If the circle overlaps an existing one, try again. After 12,000 tries, the number of successfully "parked" circles should follow a certain normal distribution.
- Minimum distance test: Randomly place 8,000 points in a 10,000 x 10,000 square, then find the minimum distance between the pairs. The square of this distance should be exponentially distributed with a certain mean.
- Random spheres test: Randomly choose 4,000 points in a cube of edge 1,000. Center a sphere on each point, whose radius is the minimum distance to another point. The smallest sphere's volume should be exponentially distributed with a certain mean.
- The squeeze test: Multiply 231 by random floats on $[0,1)$ until you reach 1. Repeat this 100,000 times. The number of floats needed to reach 1 should follow a certain distribution.
- Overlapping sums test: Generate a long sequence of random floats on $[0,1)$. Add sequences of 100 consecutive floats. The sums should be normally distributed with characteristic mean and sigma.
- Runs test: Generate a long sequence of random floats on $[0,1)$. Count ascending and descending runs. The counts should follow a certain distribution.
- The craps test: Play 200,000 games of craps, counting the wins and the number of throws per game. Each count should follow a certain distribution.

A complete description of these 15 tests and software to perform them is available on the Diehard CDROM [78]. A major drawback of these tests is that to run correctly, the suite requires at least 80 millions bits (10-12 Megabytes). In [79], Marsaglia and Tsang proposed to reduce the number of test in Diehard battery of test of randomness to only three tests. They claim that if a random number generator passes these three tests then it is likely to pass the tests in Diehard. Another advantage is that this "distilled version" is easier to apply because unlike the 12 megabytes files required for Diehard, only 32-bits random integers are needed to perform the tests.

These three tests are:

- The gcd test, based on Euclid's algorithm for computing the gcd of two random 32-bits integers. In particular distribution of the gcd and k the number of steps needed to compute the gcd of two random 32-bits integers are studied in this test.

- The Gorilla test, a stronger version of Monkey test presented in Diehard.
- The Birthday Spacing test, a stronger version of iterated birthday spacing test presented in Diehard.

Even if some applications often involve random integers with hundreds of bits, this "distilled version" seems to be an interesting way of study and could be applied for longer sequences (48, 64 bits and so on). Marsaglia and Tsang conclude that analogous tests for larger integers may be worth considering.

2.3 Tests and requirements of the NIST

Founded in 1901, NIST is a non-regulatory federal agency within the U.S. Department of Commerce. NIST's mission is to promote U.S. innovation and industrial competitiveness by advancing measurement science, standards, and technology in ways that enhance economic security and improve our quality of life.

2.3.1 Federal Information Processing Standard Publication FIPS 140

This standard specifies the security requirements that are to be satisfied by a cryptographic module. The standard provides four increasing, qualitative levels of security intended to cover a wide range of potential applications and environments. The security requirements cover areas related to the secure design and implementation of a cryptographic module. These areas include basic design and documentation, module interfaces, authorized roles and services, physical security, software security, operating system security, key management, cryptographic algorithms, Electro-Magnetic Interference/Electromagnetic Compatibility (EMI/EMC), and self-testing. This document describe basic testing of random numbers. The final version of the first publication was introduced in 1994 [73], the second [13] in 2001 and third [74] which is in development now. The comparison of [73] and [13] is in [80]. We can mention changed required intervals or threshold when particular test passes as an example of one of the most important changes. Four statistical tests were even removed from [13].

Here is description of the tests that are included:

- T1: The Monobit Test
 1. Count the number of ones in the 20,000 bit stream. Denote this quantity by X .
 2. The test is passed if $9,725 < X < 10,275$
- T2: The Poker Test
 1. Divide the 20,000 bit stream into 5,000 contiguous 4 bit segments. Count and store the number of occurrences of each of the 16 possible 4 bit values. Denote $f(i)$ as the number of each 4 bit value i where $0 \leq i \leq 15$

2. Evaluate following:

$$X = (16/5000) \cdot \left(\sum_{i=0}^{15} [f(i)]^2 \right) - 5000 \quad (2.6)$$

3. The test is passed if $2.16 < X < 46.17$

- T3: The Runs Test

1. A run is defined as a maximal sequence of consecutive bits of either all ones or all zeros, which is part of the 20,000 bit sample stream. The incidences of runs (for both consecutive zeros and consecutive ones) of all lengths (≥ 1) in the sample stream should be counted and stored.
2. The test is passed if the number of runs that occur (of lengths 1 through 6) is each within the corresponding interval specified in Table 2. This must hold for both the zeros and ones; that is, all 12 counts must lie in the specified interval. For the purpose of this test, runs of greater than 6 are considered to be of length 6.

- T4: The Long Run Test

1. A long run is defined to be a run of length 26 or more (of either zeros or ones).
2. On the sample of 20,000 bits, the test is passed if there are NO long runs.

2.3.2 NIST 800-22

Document [14] discusses the randomness testing of random number and pseudo-random number generators that may be used for many purposes including cryptographic, modeling and simulation applications. The focus of this document is on those applications where randomness is required for cryptographic purposes. A set

Table 2: FIPS 140 - the Runs Test tresholds, according [13]

Length of Run	Required Interval
1	2,343 - 2,657
2	1,135 - 1,365
3	542 - 708
4	251 - 373
5	111 - 201
6+	111 - 201

of statistical tests for randomness is described in this document. The NIST believes that these procedures are useful in detecting deviations of a binary sequence from randomness. However, a tester should note that apparent deviations from randomness may be due to either a poorly designed generator or to anomalies that appear in the binary sequence that is tested (i.e., a certain number of failures is expected in random sequences produced by a particular generator). It is up to the tester to determine the correct interpretation of the test results.

The NIST Test Suite is a statistical package consisting of 15 tests today. The original proposal [14] consists of 16 tests, but two of them were wrong. Authors in [81] deal with mistakes in the Discrete Fourier Transform (Spectral) Test and the Lempel-Ziv Compression Test and also correct them. However, the Lempel-Ziv Compression Test was removed finally. Tests in [14] were developed to test the randomness of (arbitrarily long) binary sequences produced by either hardware or software based cryptographic random or pseudorandom number generators. These tests focus on a variety of different types of non-randomness that could exist in a sequence. Some tests are decomposable into a variety of subtests. The final 15 tests are:

- The Frequency (Monobit) Test,
- Frequency Test within a Block,
- The Runs Test,
- Test for the Longest-Run-of-Ones in a Block,
- The Binary Matrix Rank Test,
- The Discrete Fourier Transform (Spectral) Test,
- The Overlapping Template Matching Test,
- Maurer's "Universal Statistical" Test,
- The Linear Complexity Test,
- The Serial Test,
- The Approximate Entropy Test,
- The Cumulative Sums (Cusums) Test,
- The Random Excursions Test, and
- The Random Excursions Variant Test.

2.4 Tests and requirements of the BSI

Evaluation methodologies AIS 20 [75] and AIS 31 [12] (abbreviated from Application Notes and Interpretation of the Scheme) were introduced by German BSI. The former represents methodologies for PRNG while the latter represents methodologies for TRNG. It is necessary to note that AIS 31 is the only methodology aimed for TRNG.

2.4.1 AIS 20

AIS 20 represents evaluation criteria for deterministic RNG. The basic idea is that the suitability of deterministic RNG should be assessed with reference to the cryptographic applications in which they are used.

Different cryptographic applications also have different requirements regarding the random numbers which are necessary, and in any case deterministically generated random numbers can only behave like "truly" random numbers with respect to certain criteria. This suggests that the suitability of deterministic random number generators should be assessed according to the intended applications. Four downward-compatible functionality classes (K1, K2, K3, K4) are defined and discussed by AIS 20. The practical application of the criteria is demonstrated in the document.

Class K1

Table 3: Classification of various PRNGs according to class according to AIS 20

Class	Example (may be dependent on the choice of suitable parameters)	Minimum E level/ Strength of mechanism
K1	E.1 Counter E.2 Linear congruential generator E.3 Linear shift register E.4 Recursive call of a block Cipher E.5 Counter with hash function E.6 RSA generator	E.2 / low, medium E.3 / high
K2	E.2 - E.6	E.2 / low, medium E.3 / high
K3	E.4 - E.6	E.3 / medium, high
K4	E.6	E.3 / medium, high

There should be a high probability that random vectors formed from random numbers are mutually different. The statistical properties of these vectors are unimportant.

Applications of K1 class PRNG:

- Challenge-Response protocols (e.g. for use in a smart card-terminal authentication process).

Class K2

The random numbers generated possess similar statistical properties to random numbers which have been generated by an ideal random number generator.

Applications of K2 class PRNG:

- Stream ciphers which are controlled through a shift register bundle whose initial values are derived from secret long-term keys and a session key which is transmitted in plaintext at the beginning of the communication.

Class K3

It is practically impossible for an adversary to work out or guess the numbers which precede or follow a random number subsequence $r_i, r_{i+1}, \dots, r_{ri+j}$ or to work out or guess an internal state. The adversary's assumed attack potential depends on the strength of mechanism.

Applications of K3 class PRNG:

- generation of pairs of signature keys
- generation of DSS signatures
- generation of session keys for symmetric cryptographic mechanisms
- pseudorandom padding bits
- zero-knowledge proofs

Class K4

It is practically impossible for an adversary to work out or guess predecessor random numbers or predecessor internal states from knowledge of the internal state $s[i]$. The adversary's assumed attack potential depends on the strength of mechanism.

Applications of K4 class PRNG:

- generation of pairs of signature keys
- generation of DSS signatures
- generation of session keys for symmetric cryptographic mechanisms
- pseudorandom padding bits

Tests (for classes K2-K4):

Tests T1 - T4 together with their designation and critical values are taken from [73]. Bits b_1, \dots, b_{20000} refers to a bit sequence of length 20,000.

- T1: monobit test
- T2: poker test
- T3: run test
- T4: long run test
- T5: autocorrelation test

For $\tau \in \{1, \dots, 5000\}$, $Z_\tau := \sum_{j=1}^{5000} (b_j \oplus b_{j+\tau})$. The sequence b_1, \dots, b_{20000} passes the autocorrelation test (with shift τ), if $2326 < Z_\tau < 2674$. (Note that the subsequence $b_{10001}, \dots, b_{20000}$ does not enter into the test statistic.)

2.4.2 AIS 31

AIS 31 describes evaluation criteria for TRNG and is counterpart to the mathematical basis of AIS 20. AIS 31 also defines online statistical tests for TRNG which are applied during effective operation on the digitized noise signal sequence generated by the TRNG or to internal random numbers with the aim of verifying that the TRNG is behaving correctly. A conspicuous statistical feature detected by an online test leads to a noise alarm which in turn leads to the TRNG being stopped. There is defined total failure if the digitized noise signal sequence is constant.

Assessment of a TRNG is essentially based on statistical tests. On the basis of different potential attack scenarios, various applications can place different requirements on the properties of the external and internal random numbers. In order to take this circumstance into account AIS 31 defines two functionality classes P1 and P2. P1 property requires the internal random numbers to be statistically inconspicuous. The P2 specific requirements should guarantee, that they are practically impossible to determine even if the predecessors or successors are known. Online test must also itself recognize total failure or any interference that occurs in the noise source and may need to be able to resist systematic manipulation attempts.

Class P1

A sequence of random vectors formed from the internal random numbers is most likely pairwise different. Internal random number sequence and their projections to individual bits pass certain statistical tests. If the strength of mechanisms or functions is "medium" or "high", total failure of the noise source should be detected when the TRNG is switched on and during operation. The statistical properties of the internal random numbers are tested during operation. If the strength of mechanisms or functions is "high", the statistical properties of the internal random numbers should not be influenced by external conditions (temperature, climate, ageing).

Applications of P1 class TRNG:

- Challenge-response protocols
- Openly transmitted, non-constant initialization vectors
- Seed generation for PRNGs of the classes K1 and K2

Class P2

The statistical behavior of the digitized noise signal sequence is inconspicuous. If the strengths of mechanism or functions is "medium" or "high", the functionality of the physical noise source is tested when the TRNG is switched on. Total failure of the physical noise source is detected when the TRNG is switched on or during operations. The TRNG tests the statistical properties of the digitized noise signals during operation at least, when the tests are triggered externally. If the strength of mechanisms or functions is "high", the execution of the online tests must be triggered independently.

Applications of P2 class TRNG:

- Generation of signature key pairs
- Generation of DSS signatures
- Generation of session keys for symmetric encryption mechanisms
- Random padding bits
- Zero-knowledge proofs
- Generation of seeds for PRNG in classes K3 and K4

AIS 31 Tests

The following lists the statistical tests needed to verify the P1-specific properties as well as the P2-specific properties. First six tests run on the output of TRNG (including postprocessing) while tests T6-T8 investigate raw random bits from digitized noise source.

Tests T1 - T4 are taken from [13], test T5 is the same as in AIS 20.

- T0: disjointness test

The sequence $w_1, \dots, w_{2^{16}} \in \{0, 1\}^{48}$ passes the disjointness test if the subsequent members are pairwise different. This test needs $2^{16} \cdot 48$ bits.

- T1: monobit test
- T2: poker test
- T3: run test
- T4: long run test
- T5: autocorrelation test T1 - T5 need 20,000 bits.
- T6: uniform distribution test

The sequence $w_1, \dots, w_n \in \{0, 1\}^k$ passes the uniform distribution with parameters (k, n, a) if:

$$\frac{1}{n} \cdot |\{j \leq n \mid w_j = x\}| \in [2^{-k} - a, 2^{-k} + a] \quad (2.7)$$

for all $x \in \{0, 1\}^k$

Recommended number of sequences w_1, \dots, w_n is 100,000, when $k = 1$.

- T7: comparative test for multinomial distributions

For each $i \in \{1, \dots, h\}$ let the n -element sample w_{i1}, \dots, w_{in} assume values from the set $\{0, 1, \dots, s - 1\}$. According to the null hypothesis, the multinomial distributions on which the individual samples are based are identical. Furthermore, for $t \in \{0, \dots, s - 1\}$ let $f_i[t] := |\{j : w_{ij} = t\}|$, and let $p_t := (f_1[t] + \dots + f_h[t]) / (hn)$ be the relative frequency for the occurrence of t determined from the total of all samples. Under the null hypothesis, the test variable $\sum_{i=1, \dots, h} \sum_{t=0, \dots, s-1} (f_i[t] - np_t)^2 / np_t$ is approximately χ^2 -distributed with $(h - 1)(s - 1)$ degrees of freedom.

Subsequences has to contain at least 100,000 elements.

- T8: entropy test

The bit sequence $b_1, \dots, b_{(Q+K)L}$ is segmented into non-overlapping outputs words w_1, \dots, w_{Q+k} of length L . A_n is the distance from w_n to its predecessor with the same value, and

$$A_n = \begin{cases} n & \text{if no } i < n \text{ exist with } w_n = w_{n-i} \\ \min\{i | i \geq 1, w_n = w_{n-i}\} & \text{in all other cases} \end{cases}$$

Test variable $f : \{0, 1\}^{(Q+K)L} \rightarrow R$ is determined for the Coron test by:

$$f_c(\bar{s}) = \frac{1}{K} \sum_{n=Q+1}^{Q+K} g(A_n) \quad (2.8)$$

where

$$g(i) = \frac{1}{\log(2)} \sum_{k=1}^{i-1} \frac{1}{k}. \quad (2.9)$$

Recommended number of bits is determine by $L = 8$, $Q = 2,560$ and $K = 256,000$.

2.5 Oline Tests

The great contribution to online testing of random numbers inside embedded chip give authors in [82], [83] and [84]. If the reader is interested at testing procedures he should be refer to the named documents. The inner testability means that the generator should enable to evaluate the entropy of the source of randomness before post-processing. This functionality is required in recent TRNG evaluation procedures [83]. However, when randomness extraction and post-processing are joined in the same process, the un-processed random signal is not available. Even if this signal is available, it is sometimes composed of a pseudorandom pattern combined with a truly random bit-stream. The pseudo-random pattern makes statistical evaluation of the signal more difficult.

Requirements for online tests were formulated in [82] and they are:

- An online test has to detect a total breakdown of the noise source very soon.
- An online test should detect non-tolerable statistical weaknesses of the digital noise signal.
- The probability for a noise alarm should be small if the deviation of the statistical properties of the digital noise signal from that of ideal sequences is tolerable.
- An online test should run fast and require only a few lines of code and little memory or hardware resources.

3 Implementation Platform Issues

We will analyze target FPGA platforms in this chapter for implementation of TRNG. There are several FPGA vendors - Altera [85], Xilinx [86], Actel [87], Lattice [88], etc. Each of them has FPGA sub-families of different performance. There are many significant differences between them, that could influence TRNG designing. Particularly, Altera and Xilinx consist of logic cells which configuration is held in Static Random Access Memory (SRAM), which has to be configured after power-on from an external Flash memory. On the other hand Actel is representative of configurable logic cell based on the Flash fabric. There are completely different structure of logic cells - Altera uses Logic Elements (LE) [89], Xilinx uses Configurable Logic Blocks (CLB) [90] and Actel uses VersaTiles [41]. Each of them could have impact to TRNGs constructed by logic gates only. Furthermore, some TRNG uses PLLs as source of randomness and each vendor uses different clock management - Altera and Actel are faithful to traditional PLL [91],[41], while Xilinx has chosen more innovative way represented by Delay Locked Loops (DLL)[90].

3.1 Altera FPGAs

There are three main product lines of FPGAs in current Altera's portfolio with following families:

- High-End FPGAs
 - Stratix IV [92] (E and GX),
 - Stratix III [93] (L and E),
 - Stratix II [94] (and GX [95]) and
 - Stratix I [96] (and GX [97]).
- Midrange FPGAs
 - Arria (GX) [98].
- Low-Cost FPGAs
 - Cyclone III [99],
 - Cyclone II [100] and
 - Cyclone I [101].

3.2 Xilinx FPGAs

There are three main product lines of FPGAs in current Xilinx's portfolio with following families:

- High-End FPGAs
 - Vitrex-5 (LX, LXT, SXT, FXT, TXT) [102] and
 - Virtex-4 (LX, SX, FX) [103].
- Low-Cost FPGAs
 - Spartan-3 A, AN, A DSP [104].

3.3 Actel FPGAs

There are three main product lines of FPGAs in current Xilinx's portfolio with following families:

- Mixed Signal FPGA
 - Fusion [41].
- Low-Power FPGAs
 - IGLOO [105] and
 - ProASIC3 [106].

3.4 Logic Cells Comparison of the Various FPGA Producers

Architectures of logic cell are shown in the Figure 3.1, Figure 3.2 and 3.3 for Altera, Xilinx and Actel, respectively. Both, Altera and Xilinx use the Look-Up Table (LUT) for implementation of logic function with 4 inputs. "Hardware" D-flip-flop (DFF) is the next component that they have in common. In some cases [107] is possible to use LUT as DFF of slightly different properties. Representatives were chosen from Low-cost families Cyclone III (Altera) and Spartan (Xilinx). Performance families (Stratix, Virtex) have more complex architectures, but LUT and DFF still take the part of. Actel's VersaTile is completely different case, because there are not any LUT or distinctive DFF. Either three-input logic function or DFF is possible to construct using basic logic of the VersaTile. But not at once in the same VersaTile. This difference can have impact to overall timings or behavior during metastable state. However, topics in terms of how such differences have impact to particular architectures of TRNG lack in the literature. Synthesizing TRNG's architecture often requires nonstandard approaches because synthesizer usually optimizes the design completely, that means, it removes whole TRNG away! The effort of authors in [35] is possible to mention as example. Actel provides **INVD** (inverter) and **BUFD** (buffer) parts in its library, which will not be removed by the synthesis what seems to be an easy way of TRNG synthesizing.

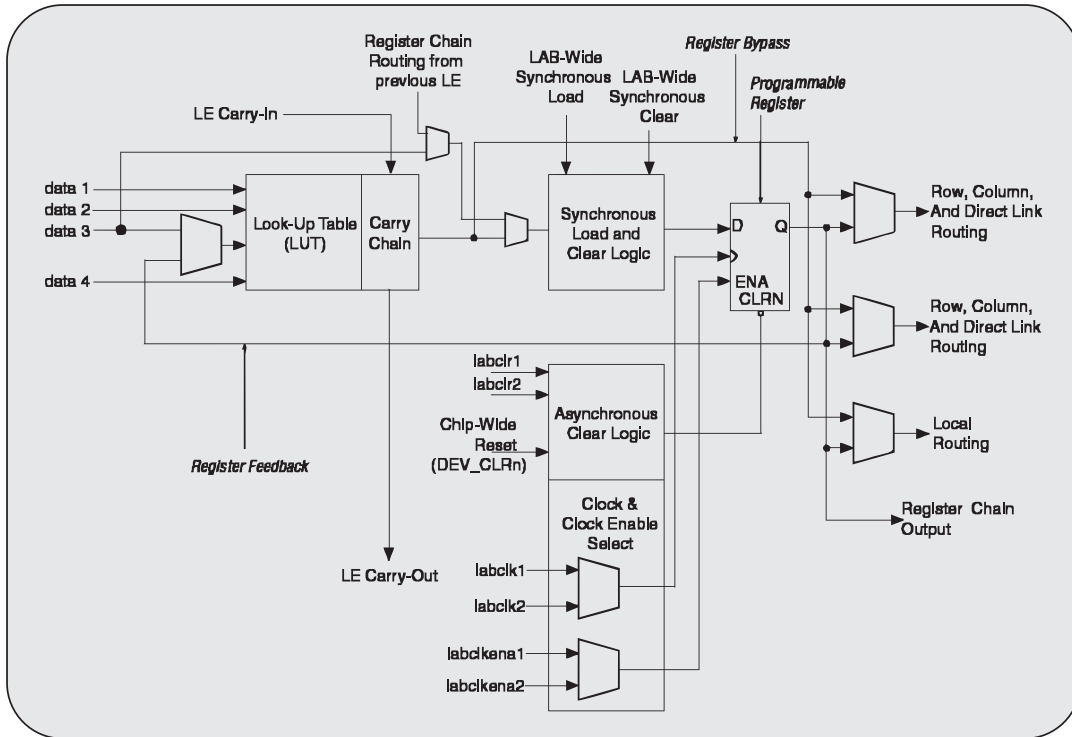


Figure 3.1: The Logic Element of Altera's Cyclone III FPGA with LUT and dedicated DFF [89].

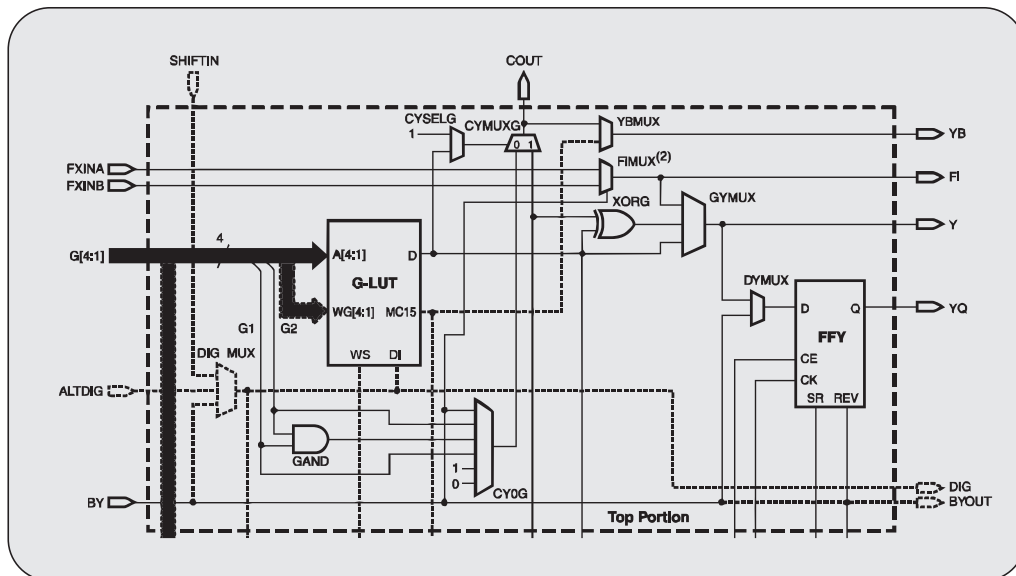


Figure 3.2: The part of Configurable Logic Block of Xilinx's Spartan FPGA with LUT and dedicated DFF [90].

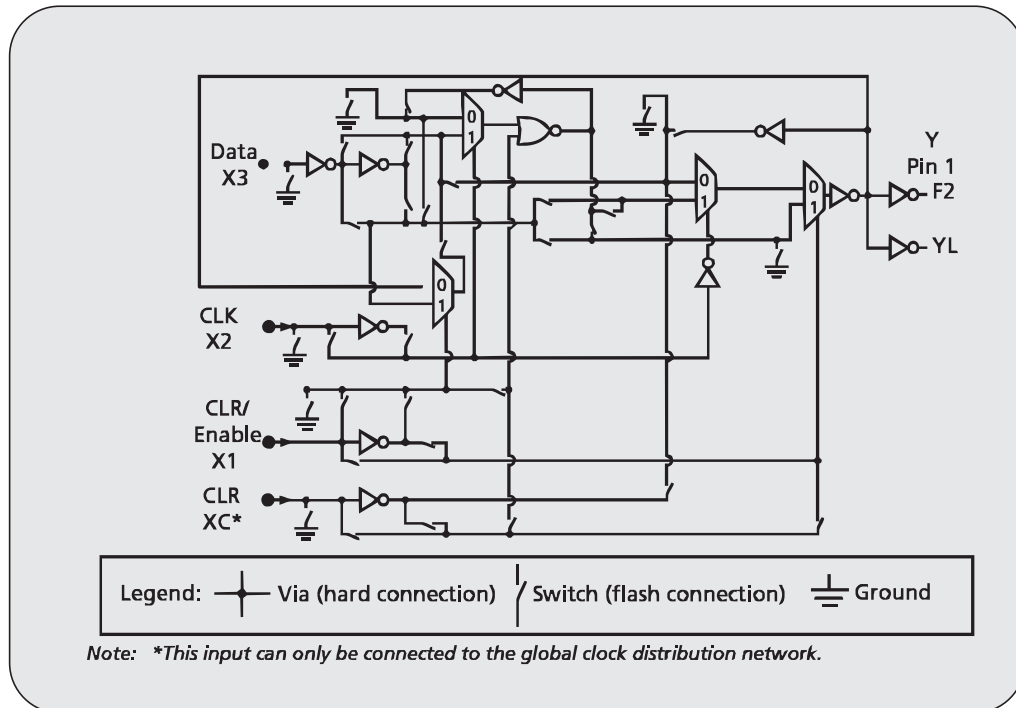


Figure 3.3: The VersaTile of Actel's Fusion FPGA without LUT and dedicated DFF; there are standard gates and switchers instead [41].

3.5 Comparison of the PLLs and the DLLs of Various FPGA Manufacturers

Today's FPGAs commonly use sophisticated timing management in order to synchronize whole logic inside. Each of three most dominant vendors has chosen different way of synthesizing required clock frequency. The TRNG proposed in [1] use PLL's jitter as the source of randomness and so we briefly compare benefits of the each vendor. There was also attempt to construct the TRNG using DLLs [108]. Altera and Actel supports PLLs while Xilinx DLLs.

PLLs have been used since the 1940's in analog implementations. Figure 3.4 depicts the main principle of the PLL. Phase detector (in the simplest way realized by the XOR gate) compares the input reference frequency F_{IN} and output frequency divided by M factor. The output of detector filtered by low-pass filter create the control voltage of Voltage Controlled Oscillator (VCO). Strong feedback ensures the constant output frequency. The most vulnerable part of this loop is VCO, which requires very clean power supply. Any deterministic changes of power supply voltage would appear in the histogram of the jitter of output signal. There are listed reasons of unlocking the loop of PLL which are posted on Altera Web [109]:

- Jitter on PLL input clock is out of specification

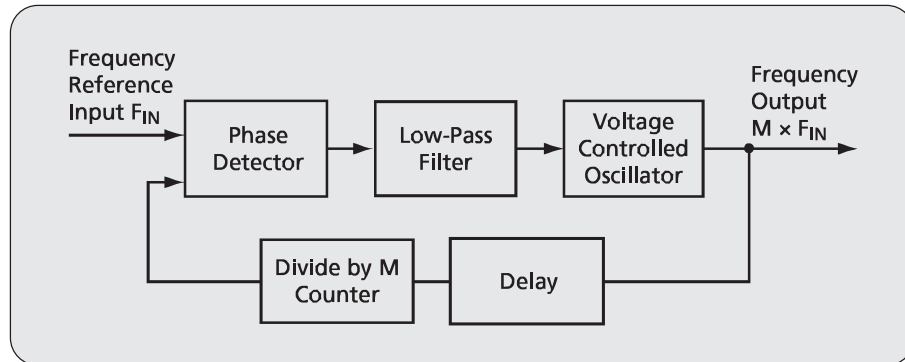


Figure 3.4: The general PLL functional block diagram.

- Simultaneous Switching Noise (SSN)
- Power supply noise
- Input clock stops/glitches or there is a sudden phase change
- Stratix or Cyclone PLLs lose lock at low temperatures (less than -20C)
- end others...

Those troubles Xilinx likes to emphasize in order to magnify DLL’s benefits.

Altera and Actel claims that jitter of their PLLs is around 100 ps. The architectures of clock management of Altera and Actel are shown in Figure 3.5 and Figure 3.6 respectively.

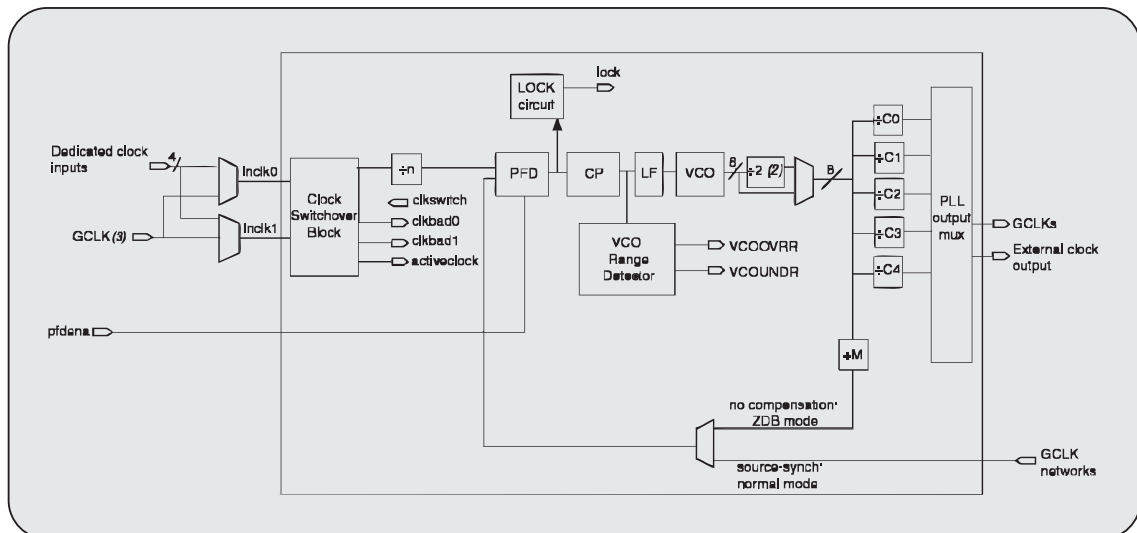


Figure 3.5: PLL of Altera’s FPGAs (Cyclone III) [91].

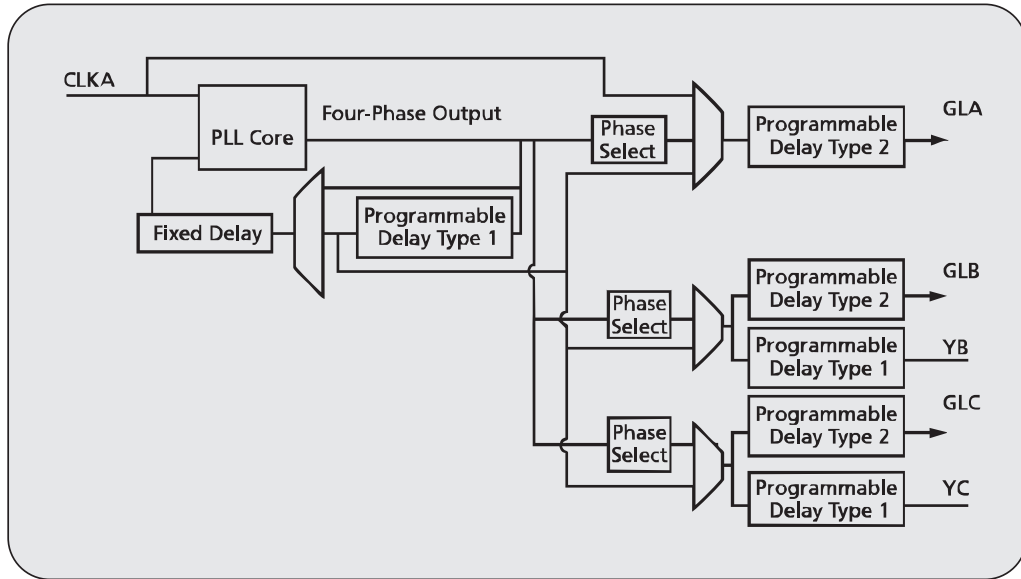


Figure 3.6: PLL of Actel's FPGAs (Fusion) [41].

On the other hand, DLLs are much less sensitive to the power supply voltage noise. Main principle of Digital Clock Managers (DCM) is depicted in Figure 3.7. It is necessary to mention, that DLL can only create clocks of various phases or to twice the input frequency. If another input/output ratio is needed, the designer has to employ Digital Frequency Synthesizer (DFS) (Figure 3.8)

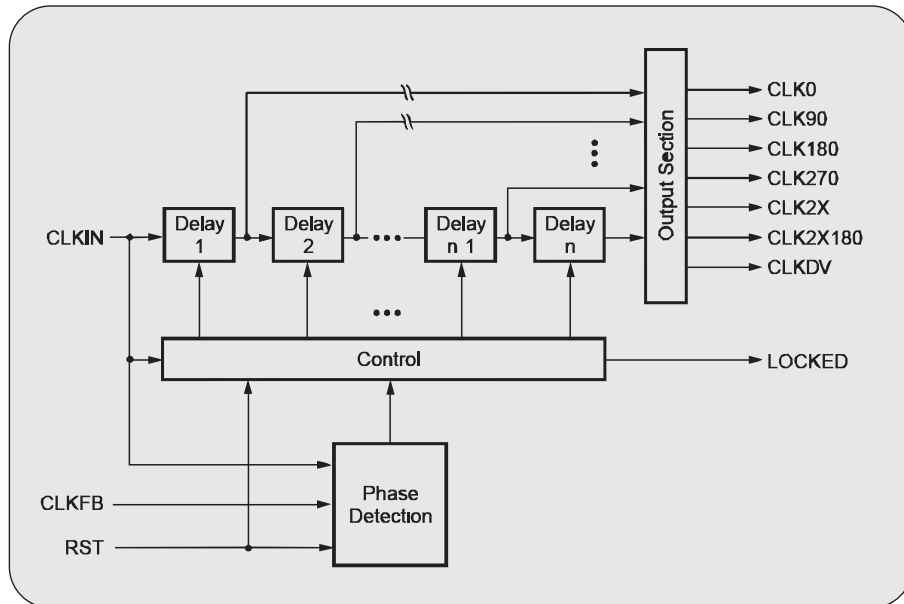


Figure 3.7: The principle of the DLL [90].

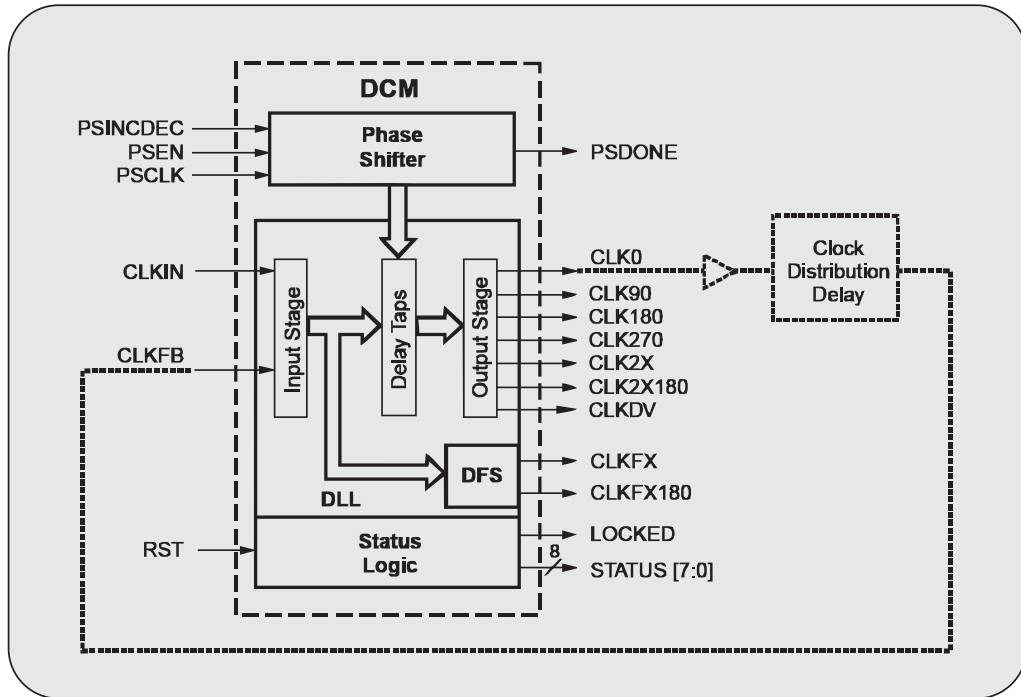


Figure 3.8: DCM block in the Xilinx's FPGAs [90].

If the question arises which concerning amount of output jitter, Xilinx provides nice jitter calculator [110]. The RMS jitter varies from hundreds of ps to units of ns depending on input/output ratio and input frequency.

This amount of jitter is greater comparing to Altera or Actel, and so Altera take the advantage of this fact and uses it in order to defend PLL qualities. Figure 3.9 [111] shown amount of jitter in PLL's (3.9B) and DLL's (3.9C) output when certain jitter is appearing in input signal (3.9A).

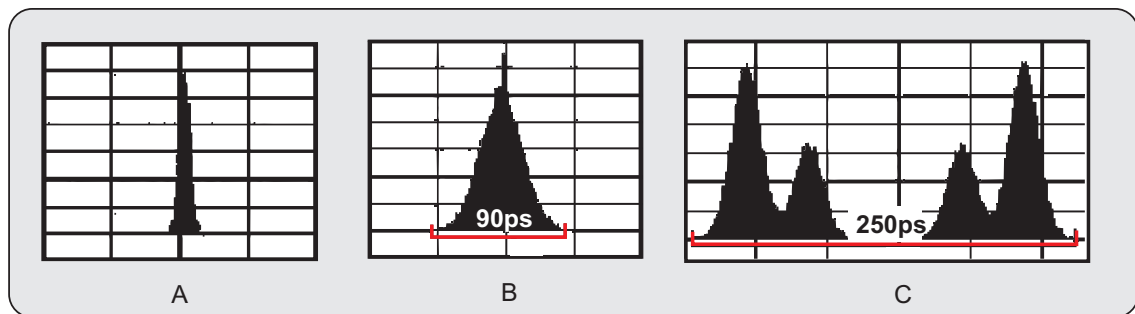


Figure 3.9: Jitter histogram of the reference crystal oscillator(A), Jitter histogram at the output of Altera's PLL (B), Jitter histogram at the output of Xilinx's DLL (C) [111].

3.5.1 Comparison of DFF Metastability of Various FPGA Families

Dependence of $MTFB$ function on the time when DFF is in the metastable state are shown in Figure 3.10, Figure 3.11 and Figure 3.12 respectively. In the reader is wondered about the measurement technique he has be refer to [36], [38] and [37]. DFFs are commonly used as synchronizers or samplers and are often used in TRNG designs and thus, the deeper study is necessary. Metastability also seems to be possible randomness source inside FPGA. The parameters are slightly comparable, and the development on FPGA technology improves it over the time.

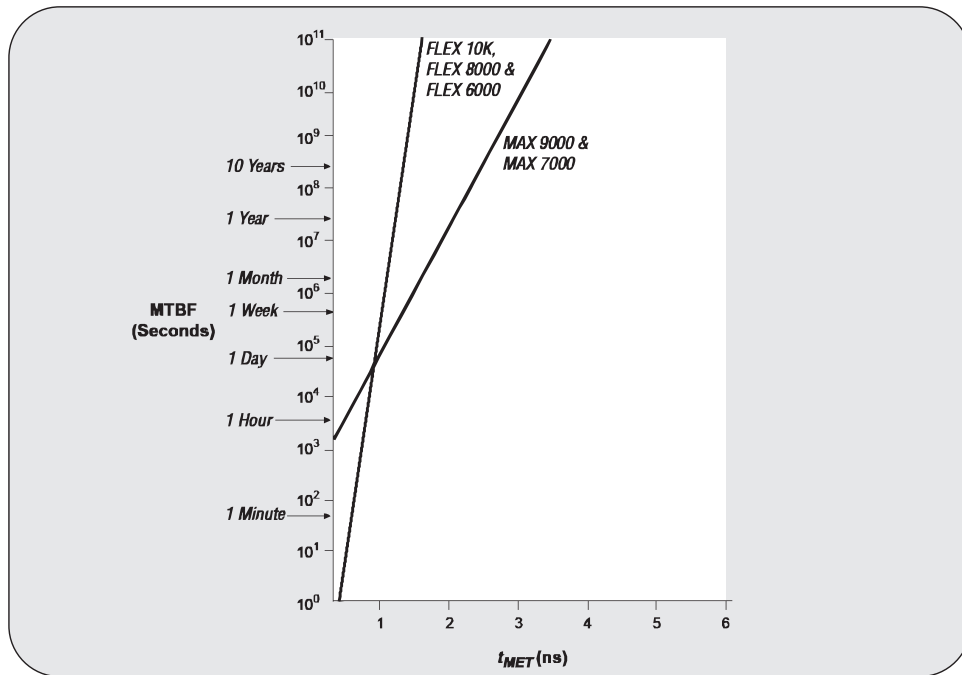


Figure 3.10: Metastability of Altera's DFF in FLEX FPGAs and MAX CPLDs [36].

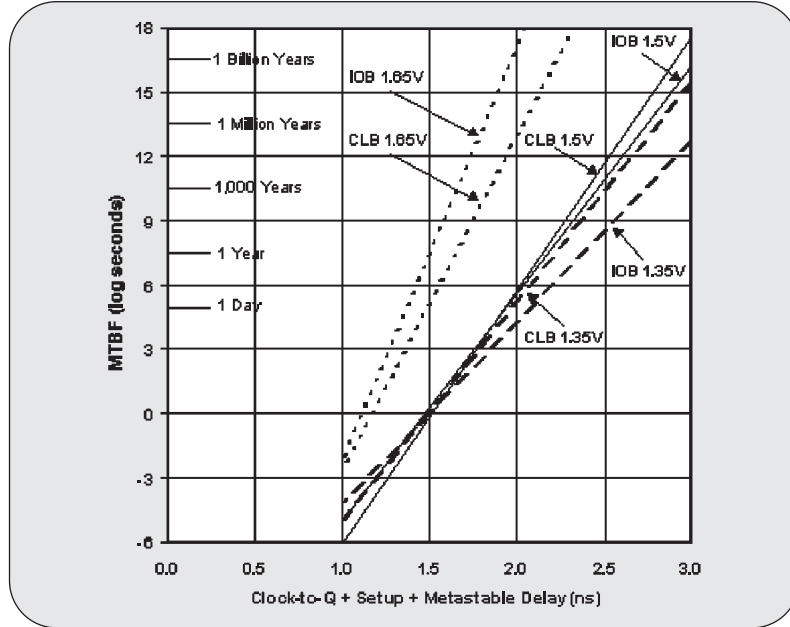


Figure 3.11: Metastability of Xilinx's DFF in Virtex FPGAs [38].

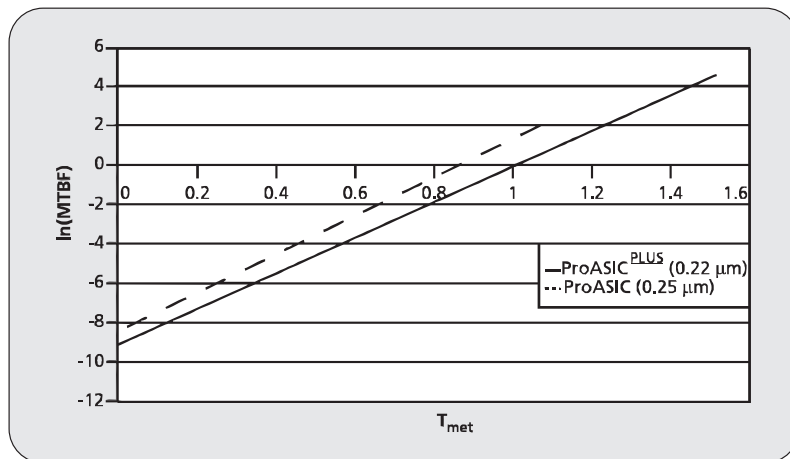


Figure 3.12: Metastability of Actel's DFF in ProASIC FPGAs [37].

4 Practical Implementations of TRNG

This section deals with TRNG designs mainly for FPGA which have appeared in scientific literature. Principles are based on both, jitter of noisy oscillator and metastability.

4.1 TRNGs based on the noisy clock sampling

This subgroup employs the jitter as the effect of semiconductor noise randomness. The use of this phenomena to generate truly random numbers is not new. The RAND Corporation used this phenomena to generate a table of a million random digits which it published in 1955 [76]. Thirty years after, Fairfield, Mortenson and Coulthard proposed in [112] first integrated circuit which was able to generate true random bit-stream utilizing the same phenomena. Nowadays, the jitter is used most of all in comparison to other effects.

4.1.1 Sunnar's type TRNGs

Sunnar et al. proposes "A Provably Secure True Random Number Generator with Built-In Tolerance to Active Attacks" in [26]. The basic principle is depicted in the Figure 4.1.

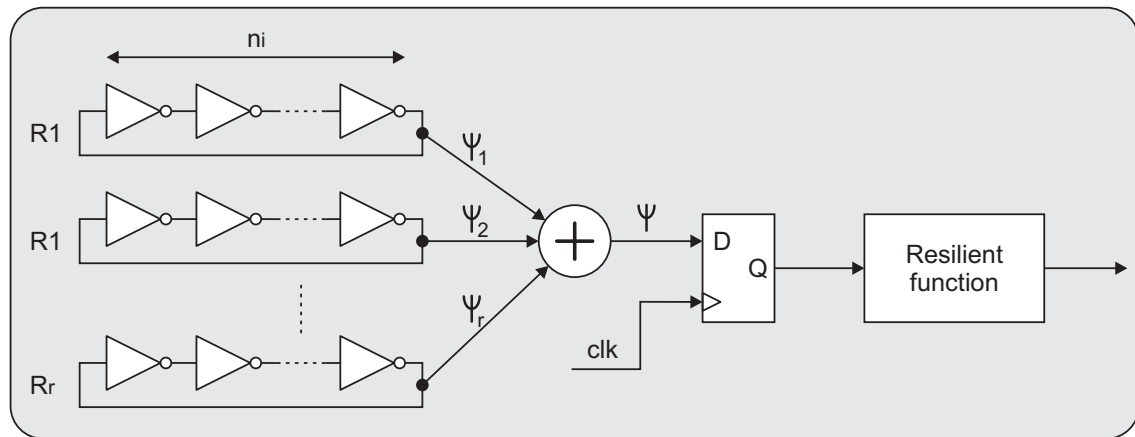


Figure 4.1: Principle of Sunnar's method; where outputs of huge number ROs are XORed together, sampled by DFF and finally post-processed by resilient function [26].

There are r distinct oscillator rings arranged on a chip. The i -th ring is composed by n_i inverters, where parameters n_i and r have to be computed as optimal. The analog output signal Ψ of the circuit is the XOR of the outputs Ψ_i of the individual rings $R_1 \dots R_r$. The XOR function is typically implemented as a binary XOR-tree. The output Ψ is converted into a digital signal by sampling it at a regular clock frequency f_{clk} . Digital signal is post-processed by the resilient function.

However, from a practical viewpoint the paper left much unexplored. In paper [113] authors focus on the practical aspects of the ring designs. In particular, they consider transistor level effects such as phase interlock, narrow signal rejection, transmission line attenuation. They showed results that changing operational conditions such as power supply voltage, or the operating temperature may affect the output. They also proposed design modifications which significantly improves its robustness against attacks. They stay some constrains in order to implement the TRNG into FPGA:

- Sampling Frequency: 100 *MHz*,
- Output bit rate: 80 *Mbps*,
- Logic reserved for TRNG: 1000 LUT,
- Amount of power reserved for TRNG: 0.3 *W*.

Besides meeting these constrains, the output bit stream of the TRNG should have a statistical quality for range of operating conditions - power supply and temperature. They also observed the usage of various resilient functions as post-processing. The one of result using less strong resilient function with 80 *Mbps* rate is in the Table 4:

Stronger resilient function improves results of both NIST and Diehard tests. The rate was reduced to 66.67 *Mbps*.

Original Sunnar's paper proposes to use 114 rings of 13 inverters. In the paper "FPGA Vendor Agnostic TRNG", aka Leuven TRNG [114] authors analyze Sunnar's proposal with the result of practical implementation in Xilinx FPGA. They used resilient function based on cyclic codes, which can be implemented in very efficient matter. They proposed to use other ring amount of less inverters according to Table 5. Authors claim that statistical parameters was evaluated by Diehard and NIST test suites with good results. The speed is in *Mbps* range.

It is obvious that design consumes quite a lot of logic cell, for comparison, minimal known AES implementation [115] takes less cells than minimal conception of this TRNG. More over, this TRNG was theoretically attacked by Dichtl in [2]. Dichtl claims that the security proof is "irrelevant" since the underlying assumptions "can never be satisfied". The main remarks were:

Table 4: Results of NIST800-22 and Diehard test suites for original Sunnar's design for various length of RO.

Ring length	3	7	11	15	19	23	27
NIST 800-22	FAIL	PASS	FAIL	FAIL	FAIL	PASS	FAIL
DIEHARD	FAIL	PASS	FAIL	FAIL	PASS	PASS	FAIL

- Based on a completely unrealistic assumption on jitter, that a RO has a perfect built-in clock and that jitter occurs only around the transitions of this clock,
- 210 ROs are implicitly assumed to be statistically independent. His experiments show they are not (coupling),
- No chip can compute the XOR at the speed required. For the Leuven FPGA design, the XOR of 210 ROs would have an average frequency of 69.9 GHz (70 transitions per gate delay),
- Even if the XOR could be computed, it could not be sampled, because the setup- and hold-times are violated (On average, 23.8 transitions in 0.17 ns , while the signal must be constant). Sunnar answers to this remarks in [116].

Dichtl et. al. in [117] go further, they process SPICE simulations of original Sunnar’s design. They concern namely to simulate XOR of the many RO outputs, what is impossible to compute as it would result in too high frequency to be processed with current technology. According results authors claim that original Sunnar’s design is not well suited for practical implementations.

4.1.2 TRNGs based on two rationally related clocks

The basic principle behind the TRNG shown in Figure 4.2 is to extract the randomness from the jitter of the clock signals synthesized in the embedded analog PLLs [1].

The jitter is detected by the sampling of a reference signal CLJ using a rationally related (clock) signal CLK synthesized in the on-chip analog PLLs with frequencies

$$F_{CLJ} = \frac{M_{CLJ}}{D_{CLJ}} F_{OSC} \quad (4.1)$$

$$F_{CLK} = \frac{M_{CLK}}{D_{CLK}} F_{OSC} \quad (4.2)$$

where F_{OSC} is a reference clock signal and parameters $K_M = M_{CLJ} \cdot D_{CLK}$, $K_D = M_{CLK} \cdot D_{CLJ}$ are related to the PLL structures. The signal CLJ is sampled

Table 5: Results of NIST800-22 and Diehard test suites for Leuven remake of Sunnar’s design for various length and number of ROs.

Noise Source	r	i	# slices
Sunnar Reference Design	114	13	1664
Minimal	110	3	565
Robust	210	3	973

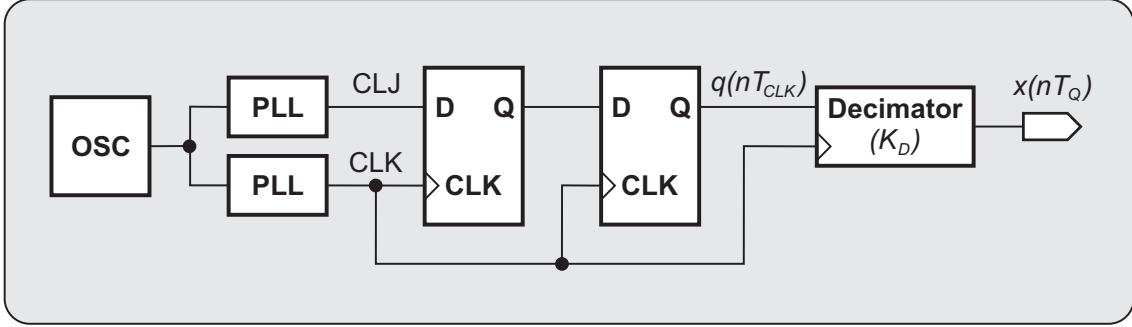


Figure 4.2: Principle of TRNG based in two rationally related clocks proposed by Fischer & Drutarovský in [1].

into the first D flip-flop using a clock signal with frequency F_{CLK} . There are F_D rising edges of CLK signal and $2K_M$ (rising and falling) edges of a CLJ waveform during the time period

$$T_Q = \frac{1}{R} = K_D T_{CLK} = K_M T_{CLJ} \quad (4.3)$$

where R is the bit-rate of the output TRNG sequence. Because there is a probability that the first D flip-flop could become metastable, the second D flip-flop is cascaded in order to decrease influence of metastability on the following logic behavior. Authors reach speed up to almost 100kbps and described generator passes NIST800-22 tests successfully. The length of tested sequence was 1 *Gbit*.

Authors in [118] implement TRNG of this principle into Altera's Stratix FPGA. Principle was evaluated in Actel's Fusion FPGA as well [119].

Authors in [120] presents a simple stochastic model of this TRNG. The existence of such model is a necessary condition in the security certification process. The proposed model can be used to test, in real time, the proper behavior of the generator and thus guarantee its robustness against cryptographic attacks. The model is validated on real data.

4.1.3 Ring Oscillator remake of previous TRNG

Authors in [121] extend a technique that uses on-chip jitter and PLLs to a much larger class of FPGAs that do not contain PLLs (e. g. Xilinx). Their design (Figure 4.3) uses only the CLBs common to all FPGAs and has self-testing capability. Using the intrinsic jitter contained in digital circuits they produce random bits of up to 0.5 *Mbps* with good statistical characteristics accordingly authors.

The design consists of two independent and identically configured ring oscillators, a sampling circuit, and control logic. The two ring oscillators each supply a stream of pulses to the sampler unit. The frequency of two clock signals is chosen to be close but not identical. It is necessary to process Place and Route manually. It is important that frequency of the two ring oscillators does not wander apart due to

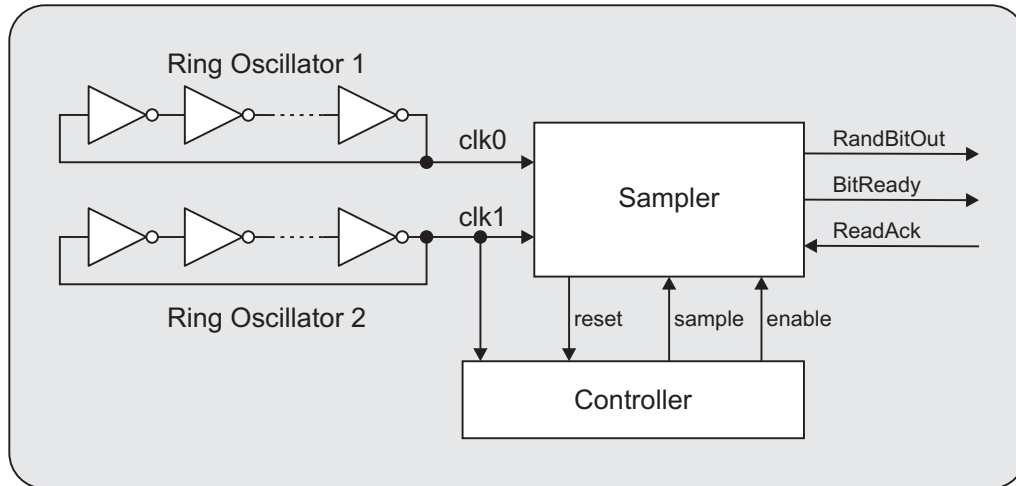


Figure 4.3: Principle of the TRNG based on two identical ROs proposed in [121] as response to [1].

temperature differences on the chip. For this reason they found that it is important to place the two ring oscillators close to each other. The sampler unit uses one clock signal to sample the other clock signal. The stream of samples consists of a run of ones and gap of zeros. The length of this gap is counted mod 2 and output as random bit. Authors produced 1 Gbits of random data in order to test it using NIST800-22. The generator passed the tests.

4.1.4 How to use DLLs instead of PLLs?

After good response to generator proposed in [1], which was designed with usage of two PLL's (it means the TRNG is suitable for Altera and Actel FPGAs), some considerations of DLL usage arises. If one would like to use Xilinx FPGA for certain cryptographic system where is demand of TRNG, he can employ DLL's as well as was proposed in [108]. The scheme is shown in Figure 4.4.

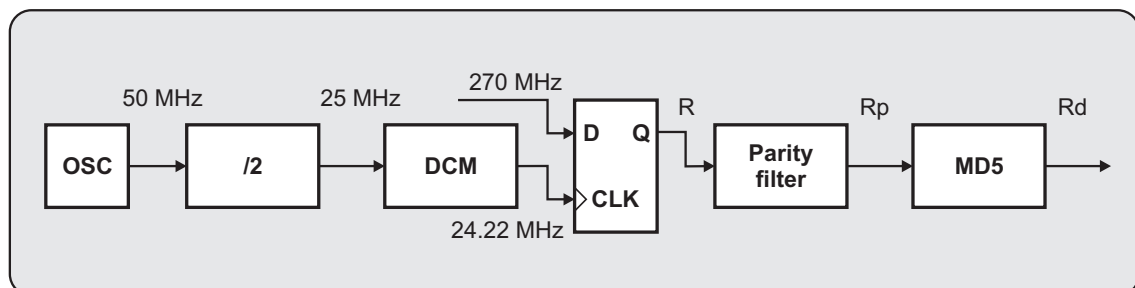


Figure 4.4: Principle of the TRNG based on DLL proposed in [108] as response to [1].

The common idea is not new, there is only sampler which samples low speed jitter clock by high speed clock. Parity filter was used in order to compress sequence and increase entropy per bit. MD5 hash function improves statistical parameters of output sequence. Authors process NIST800-22 test suite over sequences of 10^6 bit each. They process Diehard test suite as well. From their results, it was found that only the bit stream which was generated by 8-tap parity filter and MD5 fixing function of compression rate 1 and the other one which was generated by 2-tap parity filter and MD5 fixing function of compression rate of 2 can pass both test suites. They achieved output speed up to 6 Mbps. However, idea of randomness extracting and post-processing seems to be not very new and furthermore, this kind of post-processing can mask low entropy source.

4.2 TRNGs based on LFSRs and CASRs architectures

4.2.1 Tkacik's TRNG

The 32-bit hardware random number generator [122] is based on LFSR, and Cellular Automata Shift Register (CASR). Figure 4.5 shows a simplified block diagram of the generator.

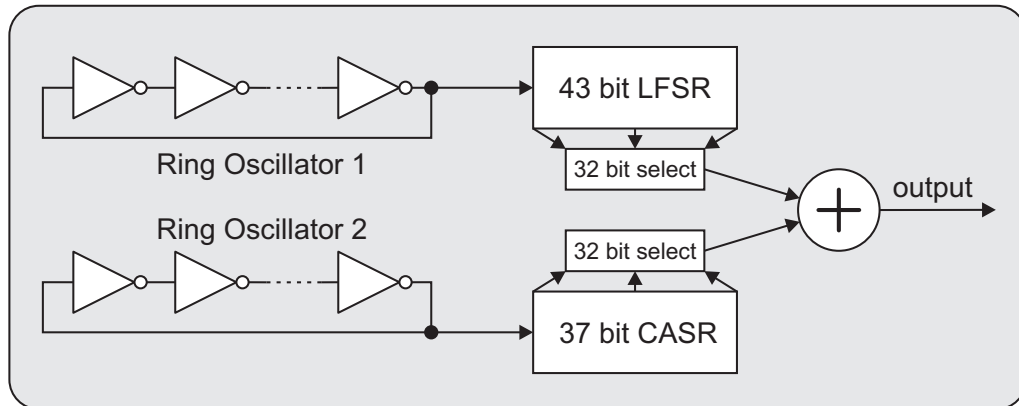


Figure 4.5: Principle of Tkacik's TRNG; Two ROs clock the LFSR and CASR and consequently the 32 bits digest of each is XORed together, what results into 32 bit random number [122].

Each shift register is clocked by an independent ring oscillator, and the output is sampled only when a new number is requested. The LFSR has 43 bits, and a characteristic polynomial of $X^{43} + X^{41} + X^{20} + X + 1$. This is a primitive polynomial and gives a cycle length $2^{43} - 1$. The hardware random number generator uses 37-bit CASR. The CASR has a maximal length of $2^{37} - 1$. To generate a random number, 32 bits of the LFSR and CASR are selected and permuted, and XORed together. Because the cycle lengths of the two state machines are relatively prime, the cycle length of the combined generator is close to 2^{80} . Patterns generated of both state machines are in the Figure 4.6

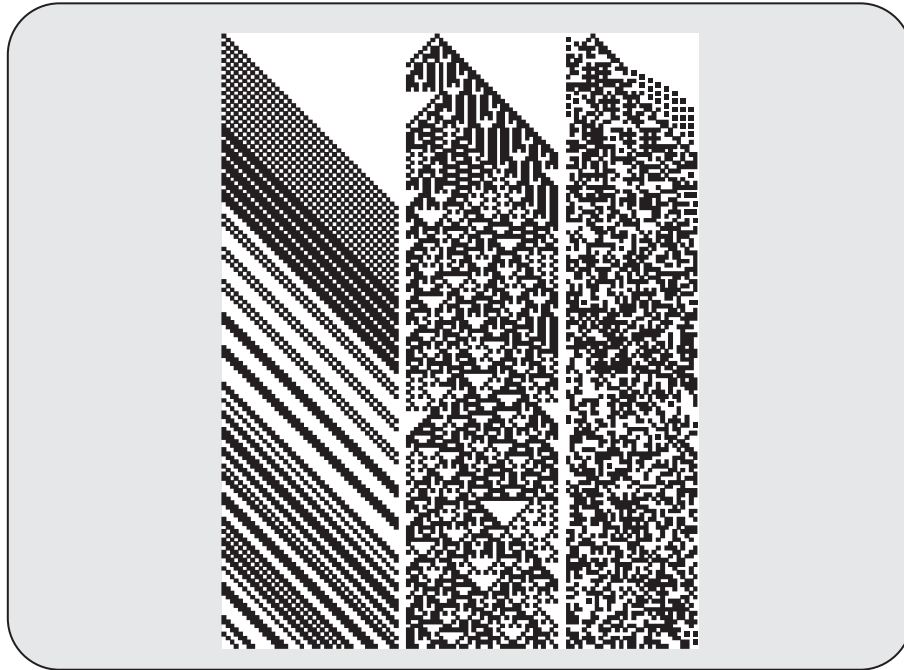


Figure 4.6: Plots of LFSR output, CASR output, LFSR and CASR XORed together [122].

The quality of the output was evaluated by Diehard tests by author. Both the LFSR and CASR fail the test miserably. For the combination of the LFSR and CASR test result show, that there still flaws in the generator, but it is significantly improved over either individually. With variable time sampling, the hardware random number generator passes these tests. Dichtl in [7] show that there could be serious security flaws and one could in theory predict the output of this generator. The flaws are presented only at a theoretical level without practical realization of proposed attack.

In fact, this generator is probably supported and used by Motorola in their chips. Several years ago, the semiconductor division was detached from Motorola and was named Freescale Semiconductor. They offer wide variety of semiconductor products (microcontrollers, microprocessors, digital signal processors, radio-frequency chips). There is also chips with cryptographic engine inside, e. g. MCF51JM128 [123]. This chip also contains RNG, and it seems it works with Tkacik's principle: *"The RNGA (Random Number Generator Accelerator) module is a digital integrated circuit capable of generating 32-bit random numbers. It is designed to comply with FIPS-140 standards for randomness and non-determinism. The random bits are generated by clocking shift registers with clocks derived from ring oscillators. The configuration of the shift registers ensures statistically good data (i.e. data that looks random). The oscillators with their unknown frequencies provide the required entropy needed to create random data."* But they also add:

”There is no known cryptographic proof showing that this is a secure method of generating random data. In fact, there may be an attack against the random number generator described in this document if its output is used directly in a cryptographic application (the attack is based on the linearity of the internal shift registers). In light of this, it is highly recommended that the random data produced by this module be used as an input seed to a NIST approved (based on DES or SHA-1) or cryptographically secure (RSA Generator or BBS Generator) random number generation algorithm. It is also recommended that other sources of entropy be used along with the RNGA to generate the seed to the pseudorandom algorithm. The more random sources combined to create the seed the better.”

4.2.2 Golic’s TRNG

Another idea of generating the random sequences was introduced in [124]. The method is based on asynchronous logic circuits with feedback. In particular, a concrete technique using so-called Galois and Fibonacci ring oscillators is developed and analyzed both theoretically and experimentally. The generated random data may have a very high speed and higher and more robust entropy rate. In paper is also proposed a method of random data postprocessing based on self-clock-controlled linear feedback register. This post-processing can provide both randomness extraction and computationally secure speed increase of input random data. It is suitable for FPGA implementation and is supported by in-depth mathematical analysis of the underlying principle. The conception is shown in the Figure 4.7

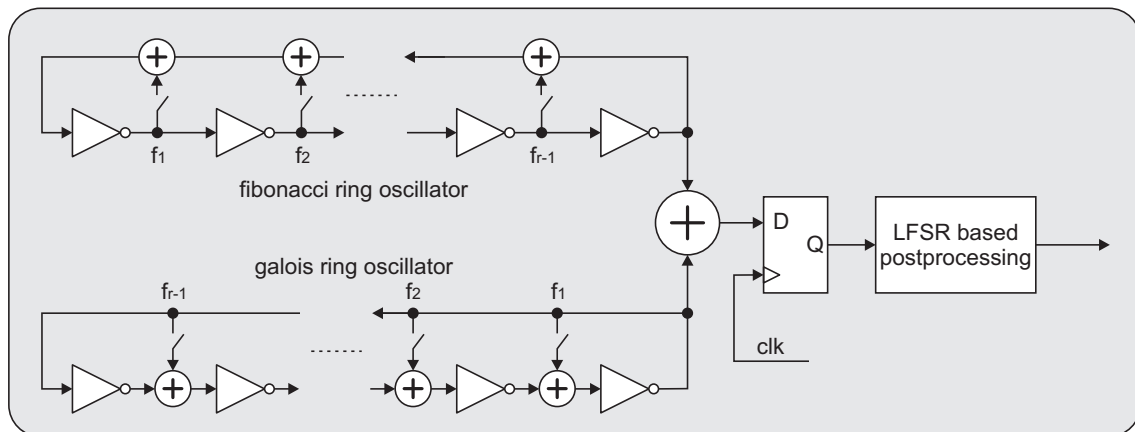


Figure 4.7: Principle of the Golic’s TRNG; Outputs of the unique Fibonacci and Galois ROs are XORed together, sampled by DFF and finally post-processed by LFSR [125].

In [2] was shown development the XOR output up to 80 ns time after restarting into the same state of both ring oscillators. Although the beginning of development was the same, it gained random manner fast.

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.

4.3 TRNG based on open delay chain

Paper [107] presents a method and an architecture allowing higher bit rates. This requirement becomes stringent for secure communications applications such as the cryptographic Quantum Key Distribution protocols (QKD) which could require a few hundred of *Mbps* for the key generation. The proposed architecture is very simple as it is based on an open loop structure, and it can be designed with no specific component such as PLLs.

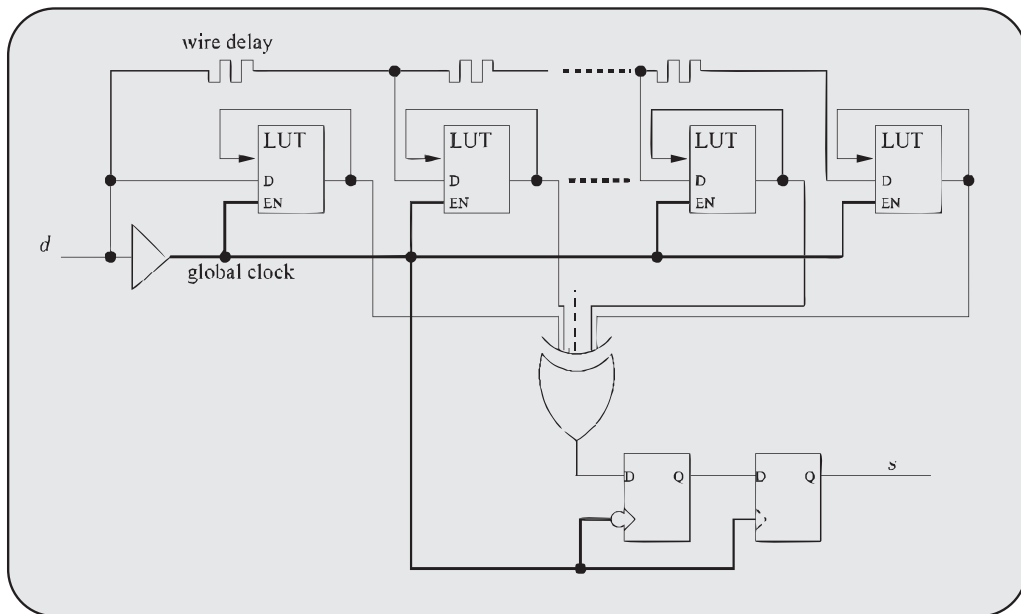


Figure 4.8: Principle of the TRNG based on open delay chain; Dedicated DFF are replaced by LUTs with DFF functionality [107].

In order to carry out feasibility tests on the open-loop based TRNG, the structure of the adjustable data delay is made up of n instances forming a chain. Every delay element output d_i is sampled by its own DFF. The n DFFs use the same common clock generated by a global buffer and routed by a specific rail. The DFF outputs are then XORed in order to capture at least one change in one of DFF. Finally the output is resynchronized by a DFF to remove the potential metastable state.

The traditional DFFs (situated in delay chain) are replaced by a latch made by a looped LUTs, as the DFF has been hardened to reduce the metastability. The latch build from the LUT has got a much greater resolving time. Consequently it is necessary to resynchronize the TRNG output by a DFF with inverted global clock as the latch output signal is memorized only half time. The delay chain consists in routing wires in order to improve the sampling resolution. The wire delay value is about a few ps in current FPGA chips. The routing has to be carefully constrained so that only the wire length influences the delay.

Figure 4.8 shows the structure of the delay chain. Sequences of the length 1 *Gbit* produced by this TRNG passes NIST800-22 test suite.

4.4 Stateless TRNG

In this section we describe an TRNGs based on a stateless (memoryless) noise source and a stateless post-processing algorithm. Since the noise source is assumed memoryless, the generated symbols are independent and, since the post-processor is also memoryless, the internal random numbers are independent too. Therefore, the entropy limit can be verified directly after the post-processor, controlling that the assumed compression ratio in the post-processor is well chosen with respect to the available entropy per bit from the source.

4.5 Generator of Bucci et al.

The principle of random number generator proposed by Bucci et al. in [11] and is presented in Figure 4.9. The generator exploits the relative jitter between two gated ring oscillators sharing the same delay elements. Four delay elements are connected by means of a symmetrical switch to obtain two ring oscillators whose mean period is identical. The cross-coupled oscillators start synchronously and their relative jitter increase when they continue running. A detector allows to detect a relative jitter greater than a given threshold (an inverter propagation delay). According to the sign of the detected jitter, a random bit is generated and both oscillators are stopped and started again to begin a new cycle. With this approach, the relative jitter depends on the available noise in the circuit and so the data rate of generator is not constant. The generator has been implemented in Complementary Metal Oxide Semiconductor (CMOS) standard cells from Infineon Technologies. After about 20 complete periods, the relative jitter is sufficient for a generation of a new bit. Nevertheless, it is more difficult to implement this generator in FPGA because the routing of the cross-coupled oscillators has to be perfectly symmetrical.

Although this generator is aimed for ASICs, the idea of "stateless" generator can be well adapted to FPGAs as well.

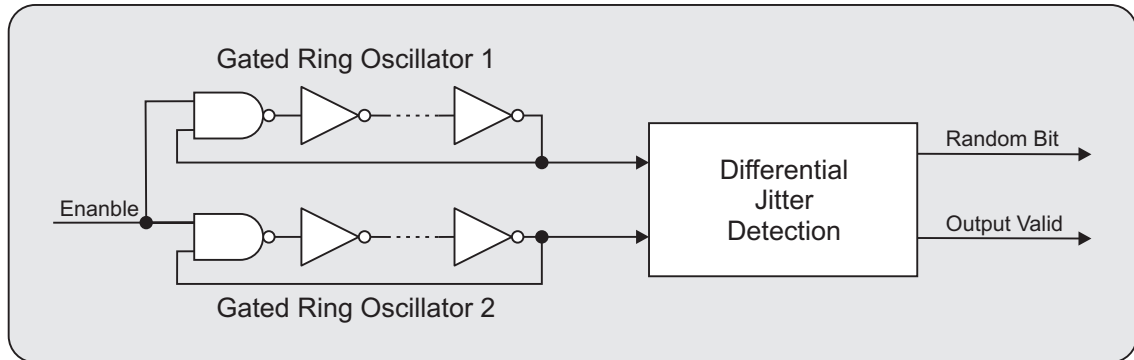


Figure 4.9: Principle of the stateless TRNG; After extracting single random bit, the whole system is restarted and thus possible correlation could be canceled [11].

4.6 TRNG based on meta-stability

TRNG concepts based on metastability appear in literature much more rarely in comparison to noisy clock sampling or other concepts. Almost each of them are dedicated for ASICs. There was found only the one design which was evaluated in FPGA as well, but with not satisfactory results [35].

One of the first approaches was shown in [126]. Authors used the 74LS74 DFF in configuration show in Figure 4.10

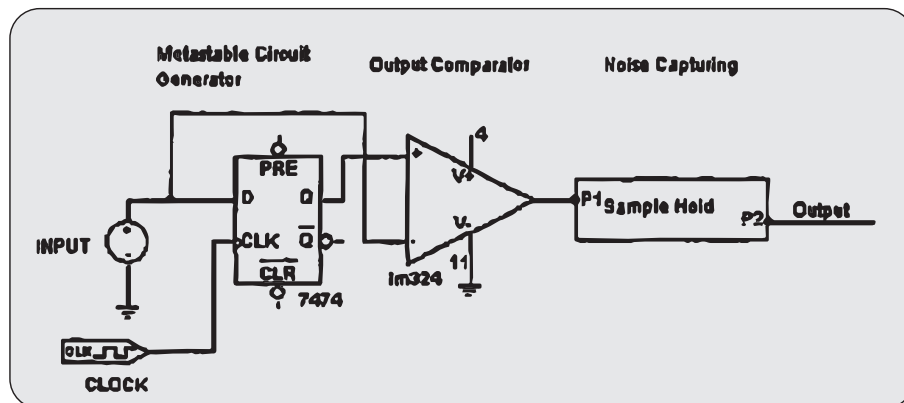


Figure 4.10: Principle of the TRNG based on metastability, which utilize simple integrated circuits such as 74LS74 (DFF) and LM324 (Operational Amplifier).

The idea is that the smaller the $MTBF$ the more likely the DFF output will become metastable, and thus a random bit stream can be observed.

The first stage, the metastable circuit generator stage, generates a metastable output using the input signal jitter, DFF electronic noise, and a system clock rate that minimizes the $MTBF$. Specific parameter values will be given in the next section. A pulse signal is then applied to the input and, under normal operation; the DFF output should appear as a sampled version of the input. However, the DFF

is operating under metastable conditions so the expected output becomes metastable and produces oscillatory behavior.

The metastable output is sent through the output comparator stage. The comparator determines an active '0' or an active '1' state for the output samples based on the noise in the signal. To obtain a true measure of electronic noise, the input signal is used as a reference for the comparator. The noise capturing stage contains a sample-and-hold circuit that allows the noise to be captured into a data stream and used for target applications.

Authors state that the average time between failures is approximately 11 femto seconds. The likelihood that the DFF will enter into a metastable state, as desired, is very high. The performance of the DFF circuit was evaluated using a digital oscilloscope and no deeper study was done.

Design and implementation of a TRNG based on digital circuit artifact was proposed in [40]. The design utilizes a pair of oscillators that are permitted to freerun. At some point, the free-running oscillators are coerced to matching states via a bistable device. While authors believe that the circuit utilizes the metastability artifact, this conjecture has not been proved. However, they have experimental results from a similar circuit composed of discrete components, which does show metastability. In the same paper is presented the discussion of the two possible causes of randomness, metastability and oscillator drift and jitter. Their design (Figure 4.11) is based on bi-stable memory element. The random sequence generator is a pair of cross-connected inverters where the multiplexers are used to switch between oscillatory and bistable phases.

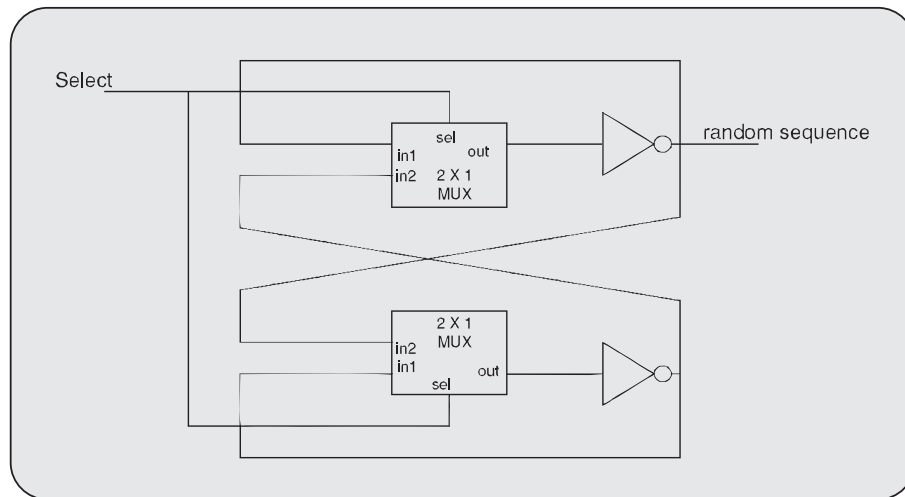


Figure 4.11: The metastable circuitry used in principle of TRNG proposed in [40].

The system of random bits acquisition is shown in Figure 4.12. The "select" signal is used to drive the multiplexers that choose between acquisition and oscillation phase of the random number generator.

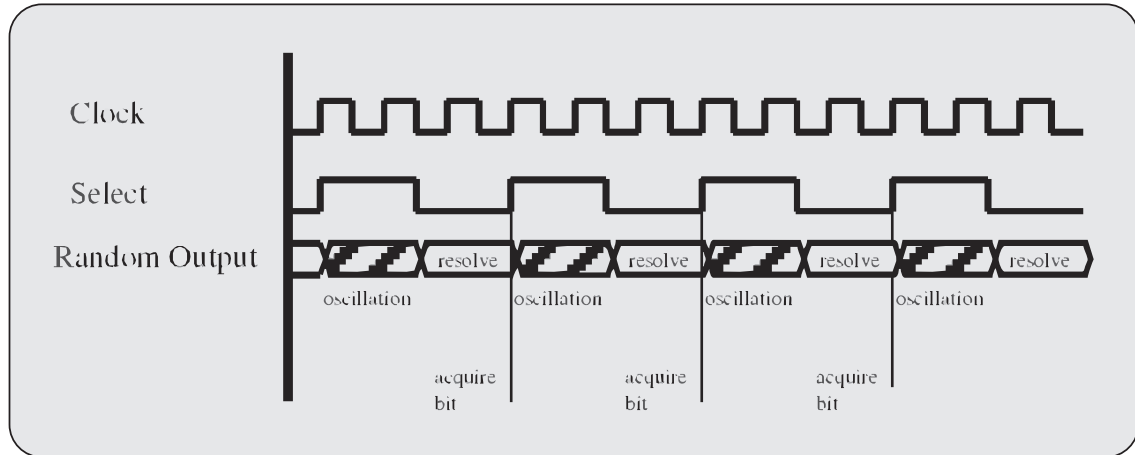


Figure 4.12: Acquiring states of TRNG based on metastability proposed in [40]

However, as author confess, variations within the manufacturing tolerances and unpredictable environmental changes (temperature, supply voltage etc.) will alter the behavior of the randomness circuit. The circuit may fail in an obvious way such as producing all '0' or all '1' output or possibly a heavily biased sequence. The simplest solution to this problem is to lay out many, slightly different versions of the circuit, on the chip. Under different environmental conditions some of the versions will work randomly while others will produce a biased output. If the differences in the circuits are small and there are a large number of varieties of the randomness circuit, with high probability at least one variety will always produce random results. The randomness of all of the different varieties is collected by simply XOR'ing all of the outputs of the random circuits on the chip. If there is at least one truly random output sequence the final result will still be random. Since each random source is small, only a few hundred standard logic gates will provide very high quality random numbers in real world conditions. Their TRNG passes Diehard test suite, but not for all testing conditions.

Authors in [127] present a fully integrable generator based on metastability and thermal noise as sources of randomness, and demonstrate results from a fabricated circuit. The technique employs a Physically Unclonable Function (PUF) circuit which has an exponential number of delay path configurations determined by a challenge input. The observation of the results of a PUF reveals that for certain challenges, the setup and hold time violation of an arbiter (D-latch) leads to unpredictable responses as the arbiter enters into a metastable condition. The responses from these challenges can be used to generate a random bit stream.

The block diagram in Figure 4.13 depicts the structure of a PUF random number generator (PUFRNG) circuit. The circuit accepts an n bit challenge $c_0, c_1, c_2, \dots, c_{n-1}$ to form two delay paths in $2n$ different configurations. To generate a response bit, two delay paths are excited simultaneously to allow the transitions to race against each other. The arbiter block at the end of the delay paths determines

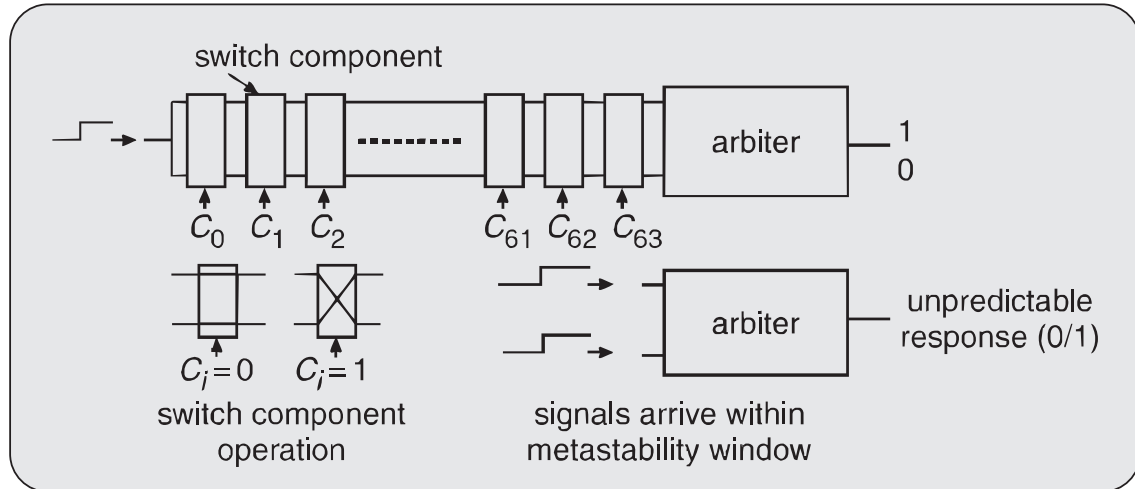


Figure 4.13: Principle of the TRNG based on PUF and metastability [127].

which rising edge arrives first and sets its output to '0' or '1'. When two transitions violate the setup and hold times of the arbiter, the arbiter becomes metastable and generates random responses. The switch components are implemented using a pair of two-to-one multiplexers. The total power consumption of a PUFTRNG circuit is about 130mW in their implementation. However, input and output functions of the generator are responsible for most of the power consumption and the power consumption of the generator core is relatively small.

The generator needs postprocessing due to bias. Post-process method involves parsing the bits in pairs, and then transforming them according to the scheme outlined in [71]. This method resulted in a typical reduction in the original PUFTRNG bit stream in the range of 65 to 75%. The PUFTRNG was used to obtain 4.5 million random bits after post-processing of the output sequence for testing. Sequence passed NIST test suite, however they did not provide any detailed methodology of evaluating.

In [128] is presented a metastability-based TRNG that achieves high entropy and passes NIST randomness tests. The generator grades the probability of randomness regardless of the output bit value by measuring the metastable resolution time. The system determines the original random noise level at the time of metastability and tunes itself to achieve a high probability of randomness. Dynamic control enables the system to respond to deterministic noise and a qualifier module grades the individual metastable events to produce a high-entropy random bit-stream. The grading module allows the user to trade off output bit-rate with the quality of the bit-stream. A fully integrated true random number generator was fabricated in the CMOS technology with an area of 0.145mm^2 .

The block diagram of the TRNG is shown in Figure 4.14. The system is divided into three major components:

- The random bit generator includes the metastable system, completion detector

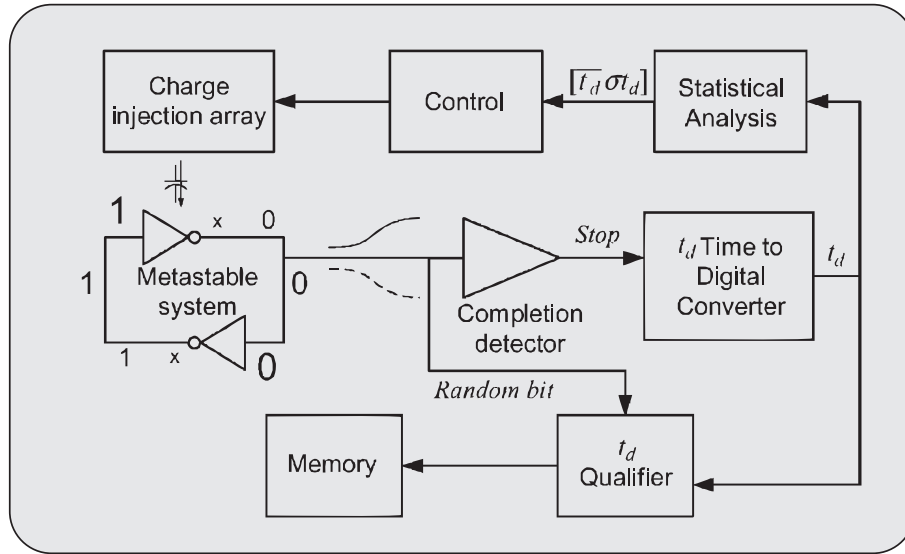


Figure 4.14: Principle of TRNG based on measuring the metastable resolution time [128].

and a Time-to-Digital Converter (TDC). It operates each cycle and produces the random output bit as well as its associated digital value of t_d .

- its control system consists of a module that extracts the statistical information for a set of t_d measurement samples, 128 for this TRNG, and the control module that determines the feedback value to tune the metastable latch.
- A grading module compares the generated output bit t_d with a predefined t_d threshold and stores the bit in memory if the generated t_d is greater than the threshold.

The metastability latch, shown in Figure 4.15, is the key component of the TRNG. This latch is based on two cross-coupled inverters that have been sized for equal rise and fall delays, resulting in an equal metastable voltage for both inverters. The latch sizing and maximum load are constrained such that the bandwidth of the region of operation lies where thermal noise dominates over flicker noise. However, the bandwidth of the region of operation must not be over constrained to avoid difficulty in meeting the resulting resolution time for the TDC. The latch has switches (M5-M9) that are used to bias and turn it on/off.

Authors in [35] proposed TRNG, based entirely on digital components. The design has been implemented using a fast random number generation method, which is dependent on a new type of ring oscillator with the ability to be set in metastable mode. Proposed method leverages the metastability phenomenon in digital circuits and applies it to a ring oscillator. The new entropy employment method allows an increase in the TRNG throughput by significantly reducing the required entropy

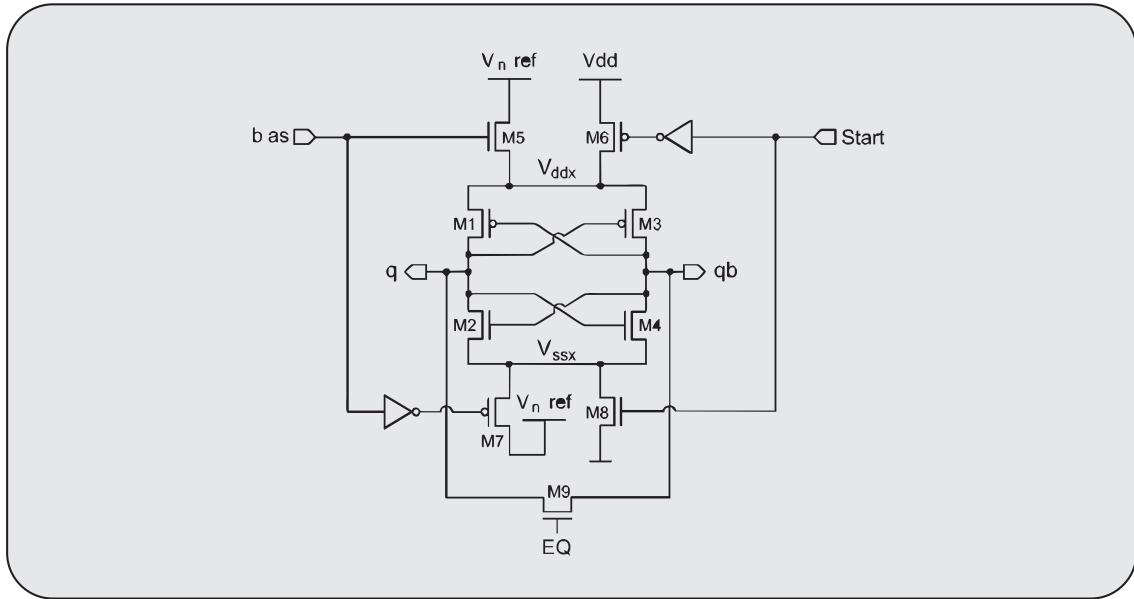


Figure 4.15: The metastable system of the method of generating random bits proposed in [128].

accumulating time. The plots of jitter histogram of traditional RO and Meta RO is in the Figure 4.17. Samples obtained from simulation of TRNG design have been evaluated using AIS 31 and FIPS 140-1/2 statistical tests. The results of these tests have proven the high quality of generated data. Investigated in FPGA technology, phase distribution highlighted the advantages of the proposed method over traditional architectures.

In 4.16, the generic scheme of metastability employment based on a CMOS inverter is shown. If the inverter is connected into the loop by a switch, the output voltage converges to metastability level and stays there as long as required (see 4.16b). Due to inherited thermal noise, the output voltage stochastically fluctuates around the metastable level. When a ring oscillator is composed of such schemes, after disconnecting the feedback loop, the initial state of the ring oscillator is completely defined by the entropy from stochastic fluctuations of each inverter

The proposed method operates as follows (see Figure 4.18). First, the Control Clock Generator switches the system into MS mode by sending the corresponding signals to the switching Components to disconnect each inverter from the others and connect it into a loop (this helps to apply the metastability point to the input of every inverter after a while). Since each inverter is disconnected from the other and the threshold point voltage is applied to its input, they form a set of independent noise sources.

After a while, the system is switched into the generation mode, where inverters are reconnected to each other to form a traditional RO. Since in the previous MS mode the value of each inverter output was defined by random noise, the momentary

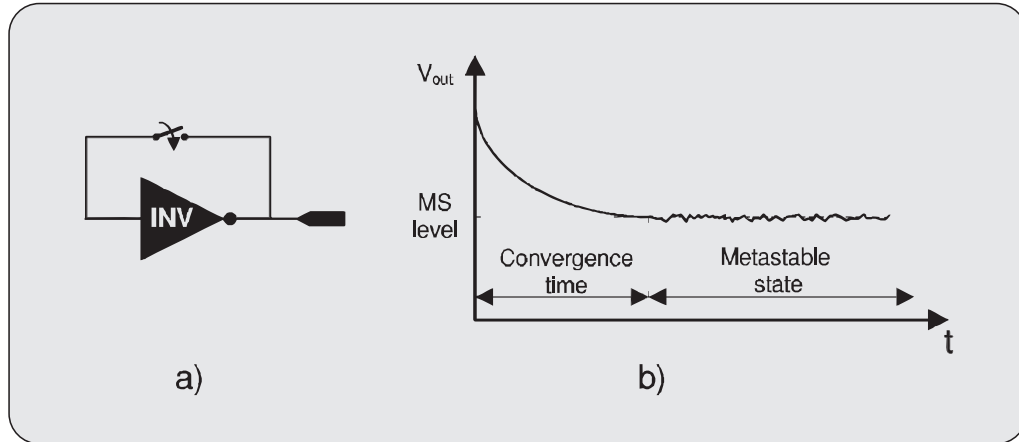


Figure 4.16: Method of getting an inverter into the metastable state by making the short circuit of its input and output [35].

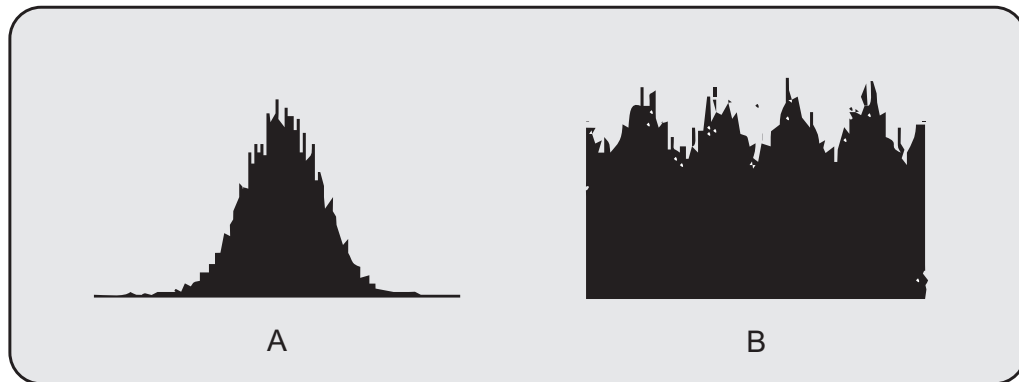


Figure 4.17: Comparison of the accumulated jitter of classic RO(A) and metastable RO(B) [35].

voltages inside the RO are also random, causing high entropy. After sampling a random bit, the TRNG system again is switched to MS mode to collect a new random value. Since for whole process it is required to wait just several periods of RO oscillation, the total TRNG throughput can be increased significantly compared to traditional jitter employment architectures.

For FIPS 140-1/2 and AIS 31 tests, evaluation was made completely over 1 *Gbits* of data samples. The preliminary investigation in Xilinx XC2V30005 showed that the statistical properties of the samples vary during the generation. The reason for such instability can be explained by the fact that the FPGA design is sensitive to temperature fluctuations and voltage supply noise. Improving FPGA operation and environment conditions (using a stable power supply source and installing a cooler over the FPGA) allowed us to obtain more satisfactory results.

The FPGA design has no correlation problem (Table 6), i.e., in 1 *Gbit* of total

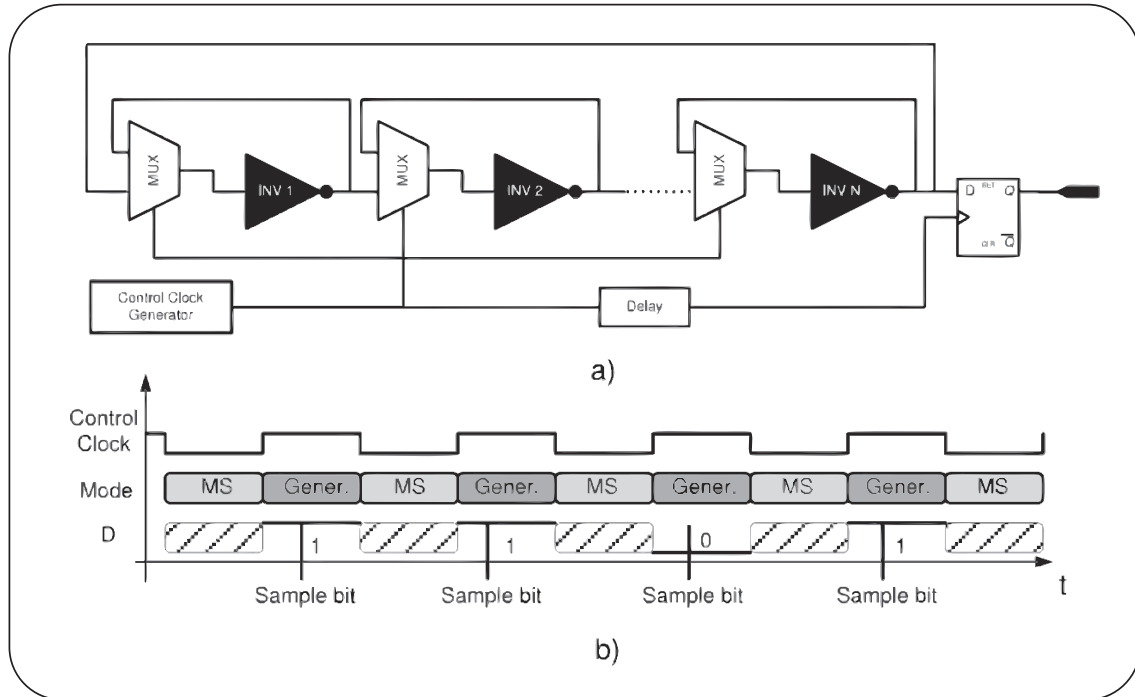


Figure 4.18: The block diagram of metastable RO TRNG (A), acquiring states of metastable RO TRNG (B) [35].

data there was no single failure in either the AIS 31 T5 Autocorrelation test or the NIST STS Spectrum test⁶. There are still some bias weaknesses, however, which could be fixed by post-processing (where applicable). The FPGA design successfully passes FIPS 140-1/2 and AIS 31 Class P1, but problems may arise with AIS 31 Class 2. Taking into account the fact that FPGA implementation is not very stable compared to ASIC, authors expect that real ASIC implementation will have no such weaknesses.

Table 6: Test results of metastable RO TRNG [35].

Test Suite	Test	No Postprocess	With Postprocess
FIPS 140-1/2	T1-T4	68%	100%
AIS 31 Class P1	T0-T4	68%	100%
AIS 31 Class P1	T5	100%	Not Needed
AIS 31 Class P2	T0-T4	68%	100%
AIS 31 Class P2	T5	100%	Not Needed
AIS 31 Class P2	T6-T8	88%	Not Needed
NIST 800-22	Spectrum Test	100%	Not Needed

4.7 The Quantum TRNG

The quantum TRNG [129] was developed in collaboration of UP optics laboratory and physical department of Czech Academy of Science. This unique TRNG is claimed as the highest class cryptographic TRNG in the World. It extracts randomness direct from quantum physics, where the most processes and equations are based on the probability.

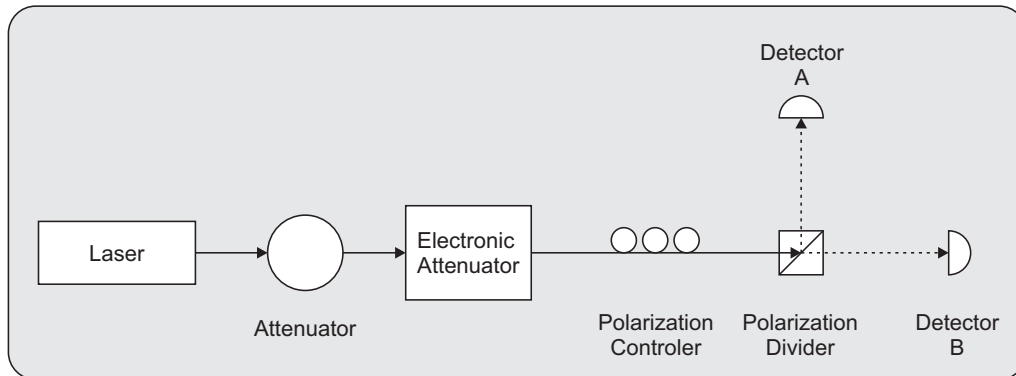


Figure 4.19: Principle of the TRNG based on a quantum mechanics [129].

Schematic diagram of the optical part of the experiment is shown in Figure 4.19. The semiconductor laser is used as source of light pulses. Its working wavelength is 830 nm . The duration of a pulse is possible to choose from interval 400 ps - 4 ns with repetitive frequency 100 Hz to 1 MHz . Each pulse contains approx. 108 photons. Pulses are attenuated in mechanical way firstly, and after, the mean amount of photons is set by the electronic attenuator in order to have 1.38 photons for pulse. Beam divider with selectable dividing ratio consists of two items - polarization controller and polarization divider. It is possible to set arbitrary dividing ratio by polarization controller. The outputs of beam divider are captured by two avalanche-effect photodiodes. It is necessary to note, that it is completely random if the photon will travel to detector A or B. Signals from detectors are processed by electronics means. There was chosen two methods for post-processing: XOR and Von Neuman corrector. The speed of generator was 22.1 kbps when using XOR corrector. Von Neuman corrector causes better bias of random bits.

4.8 TRNGs for FPGAs comparison

Fischer et al. made one of the first attempt of complex summarizing the TRNGs properties in [5]. They have focused in most famous TRNGs described in this section. They have focused in following parameters in order to evaluate differences between them:

- Post-processing method (PP):

-
- 0: complex post-processing is needed (e.g. resilient function)
 - 1: simple post-processing is sufficient (e.g. XOR corrector)
 - 2: post-processing is not necessary
 - Resource usage (RU)
 - 0: huge resources
 - 1: negligible resources
 - Design automation (DA)
 - 0: manual intervention needed (P/R)
 - 1: automatic design possible
 - Regularity of the output speed (RS)
 - 0: output bit-rate varies with time
 - 1: constant output bit-rate
 - Output bit-rate (BR)
 - 0: output bit-rate up to 1 *Mbits*
 - 1: output bit-rate 1 to 10 *Mbit/s*
 - 2: output bit-rate more than 10 *Mbits*
 - Power consumption (PC)
 - 0: negligible
 - 1: relatively high
 - Inner testability (IT)
 - 0: inner tests infeasible
 - 1: inner tests feasible
 - 2: absolute inner tests feasible
 - Existence of a mathematical model (MM)
 - 0: model does not exist and probably infeasible
 - 1: model does not exist, but probably feasible
 - 2: model exists
 - Security, robustness, resistance against attacks (RO)

Method	Observed Parameters										Result
	BR	DA	RO	PC	RU	IT	MM	FE	RS	PP	
Sunnar	1	1	1	0	0	1	0	2	0	0	6
Bucci	1	0	0	1	1	2	1	0	0	2	8
F&D PLL	1	1	1	1	0	2	2	1	1	1	11
Tkacik	1	1	0	1	1	0	1	2	1	2	10
Golic	1	1	0	1	1	1	1	2	0	1	9
Danger	2	0	0	1	1	1	1	1	1	1	9

Table 7: Evaluation of TRNG principles using following parameters: Output bit rate (BR), Design automation (DA), Robustness, resistance against attacks (RO), Power consumption (PC), Ressource usage (RU), Inner testability (IT), Existence of a mathematical model (MM), Feasibility in logic devices or FPGAs (FE), Regularity of the output speed (RS), Post-processing method (PP)

- 0: no security issues processed
- 1: some security issues solved
- 2: provable secure
- Feasibility in logic devices or FPGAs (FE)
 - 0: generator not feasible or difficult to implement in FPGAs
 - 1: generator feasible in some FPGAs
 - 2: generator feasible in all FPGAs

Table 4.8 gives an overview of TRNG parameters and their evaluation according [5].

5 Conclusion and Discussion

TRNGs in FPGAs play crucial role for cryptographic purposes. There was proposed several, but not many (comparing to ASIC based TRNGs) designs with satisfactory functionality. One of the greatest challenges in TRNG designing is to find more effective randomness extracting from FPGA's resources with well described mathematical model. Generally it is not an easy task, because the developer has limited possibilities of logic synthesis inside FPGA chip. Secure and efficient post-processing of raw random data is the next issue for research. Randomness assessment both internal and external is next serious challenge as well. The fact, that TRNG field provides wide area for the next research and inventions, underlines high number of recent scientific articles and patents as well. It is possible to name patents such as: US 6,807,553; US 7,284,024; US 7,426,527; US 7,233,965; US 2006/0069706; US 7,117,233 and many others. According facts listed above it is certainly worth to target next research towards TRNG area and to make strong effort in order to enhance this irreplaceable components of each cryptographic system.

Potential Topics of the PhD. Thesis

According to studied materials and experiments done with TRNGs, following questions and potential topics of the Ph.D. thesis arise:

- Design TRNG architectures optimized in quality, speed and/or size,
- Design VHDL codes for TRNG architectures that are independent on the target device,
- Optimize TRNG designs in order not to emit any information useful for side channel analysis,
- Study of correlation of RO used for TRNG in FPGAs,
- Study of effective randomness extraction inside FPGAs,
- Study of effective realization of corrector inside FPGAs,
- Find economic embedded mechanism that uncovers TRNG output of poor quality,
- Observe where true randomness exactly comes from and what is the maximum randomness volume inside FPGAs, and
- Behavior study and comparison of dedicated DFF and LUT based DFF in the logic cells in various FPGA families due to the metastability.

References

- [1] V. Fischer and M. Drutarovský, “True random number generator embedded in reconfigurable hardware,” in *Cryptographic Hardware and Embedded Systems - CHES 2002, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, ser. LNCS, vol. 2523. Springer, 2002, pp. 415–430.
- [2] M. Dichtl and J. Golic, “High-speed true number generation with logic gates only,” in *Cryptographic Hardware and Embedded Systems - CHES 2007, Vienna, Austria, September 10-13, 2007, Proceedings*, ser. LNCS, vol. 4727. Springer, 2007, pp. 45–61.
- [3] H. Bock, M. Bucci, and R. Luzzi, “An offset-compensated oscillator-based random bit source for security applications,” in *Cryptographic Hardware and Embedded Systems - CHES’04, Cambridge, MA, USA, August 11-13, 2004, Proceedings*.
- [4] B. Barak, R. Shaltiel, and E. Tromer, “True random number generators secure in a changing environment,” in *Cryptographic Hardware and Embedded Systems - CHES 2003, Cologne, Germany, September 8-10, 2003, Proceedings*, ser. LNCS, vol. 2779. Springer, 2003, pp. 166–180.
- [5] V. Fischer, A. Aubert, B. Valtchanov, and N. Bochard, “True random number generators in configurable logic devices,” September 2008, processing.
- [6] I. Goldberg and I. Wagner, “Randomness and the netscape browser,” *Dr. Dobb’s Journal*, pp. 66–70, 1996.
- [7] M. Dichtl, “How to predict the output of a hardware random number generator,” in *Cryptographic Hardware and Embedded Systems - CHES 2003, 5th International Workshop, Cologne, Germany, September 8-10, 2003, Proceedings*, ser. LNCS, vol. 2779. Springer, 2003, pp. 181–188.
- [8] R. Cedra, “Impact of ultralow phase noise oscillators on system performance,” *RF Design*, vol. 28–34, 2006.
- [9] P. Davies, “Flexible security,” Thales e-Security, White Paper - Cryptography & Interoperability, August 2003.
- [10] D. V. Maher and R. J. Rance, “Random number generators founded on signal and information theory,” in *Cryptographic Hardware and Embedded Systems - CHES’99, Worcester, MA, USA, August 12-13, 1999, Proceedings*, ser. LNCS, vol. 1717. Springer, 1999, pp. 219–230.
- [11] M. Bucci and R. Luzzi, “Design of testable random bit generators,” in *Cryptographic Hardware and Embedded Systems - CHES 2005, Edinburgh, UK, August 29 - September 1, 2005, Proceedings*, ser. LNCS, vol. 3659. Springer, 2005, pp. 147–156.
- [12] W. Killmann and W. Schindler, “Functionality classes and evaluation methodology for true (physical) random number generators,” <http://www.bsi.bund.de/zertifiz/zert/interpr/trngk31e.pdf>, 09 2001, version 3.1.
- [13] “Federal information processing standards publication fips pub 140-2,” U.S. Department of Commerce / National Institute of Standards and Technology, Tech. Rep., May 2001.
- [14] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, and S. Vo, “A Statistical Test Suite For Random And Pseudorandom Number Generators For Cryptographic Applications,” NIST Special Publication 800-22, May 2001.
- [15] WAVECREST, “Understanding jitter,” WAVECREST Corporation, Tech. Rep., 2001, online. Available at: <http://www.wavecrest.com/>.

- [16] S. W. Wedge, "Predicting random jitter," *IEEE Circuits and Devices Magazine*, pp. 31–38, november/december 2006.
- [17] P. Lacharme, "Post-processing functions for a biased physical random number generator," in *Fast Software Encryption workshop - FSE 2008*.
- [18] E. Barker and J. Kelsey, "Nist special publication 800-90: Recommendation for random number generation using deterministic random bit generators," National Institute of Standards and Technology (NIST), Computer Security Division Information Technology Laboratory, March 2007.
- [19] R. B. Davies, "Exclusive or (xor) and hardware random number generators," <http://webnz.com/robert/>, February 2002.
- [20] J. von Neumann, "Various techniques used in connection with random digits," *J. Research Nat. Bur. Stand., Appl. Math. Series*, vol. 12, pp. 36–38, 1951.
- [21] P. Elias, "The efficient construction of an unbiased random sequence," *The Annals of Mathematical Statistics*, vol. 43, no. 3, pp. 865–870, 1972.
- [22] M. Dichtl, "Bad and good ways of post-processing biased physical random numbers," in *Fast Software Encryption workshop - FSE 2007*.
- [23] Y. Peres, "Iterating von neumann's procedure for extracting random bits," *The Annals of Statistics*, vol. 20, no. 1, pp. 590–597, 1992.
- [24] A. Juels, M. Jakobsson, E. Shriver, and B. Hillyer, "How to turn loaded dice into fair coins," *Information Theory, IEEE Transactions on*, vol. 46, no. 3, pp. 911–921, 2000.
- [25] S. Markovski, G. D., and L. Kocarev, "Unbiased random sequences from quasigroup string transformations," in *Fast Software Encryption workshop - FSE 2005*.
- [26] 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. 56, no. 1, pp. 109–119, January 2007.
- [27] C. Colbourn, J. Dinitz, and D. Stinson, "Applications of combinatorial designs to communications, cryptography and networking," in *Surveys in Combinatorics*. 1999 British Combinatorial Conference, 1999, pp. 37–100.
- [28] D. Stinson and K. Gopalakrishnan, *Applications of Designs to Cryptography, In: CRC Handbook of Combinatorial Designs*, colbourn c.d and dinitz, j.h. ed. CRC press, 1996.
- [29] B. Chor, O. Goldreich, J. Hastad, J. Freidmann, S. Rudich, and R. Smolensky, "The bit extraction problem or t-resilient functions," in *Proc. 26th IEEE Symposium on Foundations of Computer Sciences*, 1985, pp. 396–407, online. <http://citeseer.ist.psu.edu/chor85bit.html>.
- [30] D. Stinson and J. Massey, "An infinite class of counterexamples to a conjecture concerning non linear resilient functions," *Journal of cryptology*, vol. 8, no. 5, p. 167173, 1995, online. <http://citeseer.ist.psu.edu/629195.html>.
- [31] B. Jun and P. Kocher, "The Intel random number generator," 1999, cryptography Research, Inc. White Paper prepared for Intel Corporation.
- [32] T. Schmitt-Manderbach, F. M. Weier, H., R. Ursin, F. Tiefenbacher, T. Scheidl, J. Perdigues, Z. Sodnik, C. Kurtsiefer, J. G. Rarity, A. Zeilinger, and H. Weinfurter, "Experimental demonstration of free-space decoy-state quantum key distribution over 144 km," *Physical Review Letters*, January 2007.
- [33] C. H. Bennett, F. Bessette, G. Brassard, L. Salvail, and J. Smolin, "Experimental quantum cryptography," September 91.

- [34] L. M. Reyneri, D. D. Corso, and B. Sacco, "Oscillatory metastability in homogeneous and inhomogeneous flip-flops," *Journal of Solid-State Circuits*, vol. 25, no. 1, pp. 254–264, February 1990.
- [35] I. Vasylytsov, E. Hambardzumyan, Y. S. Kim, and B. Karpinskyy, "Fast digital trng based on metastable ring oscillator," in *Cryptographic Hardware and Embedded Systems - CHES 2008, Washington, DC, USA, August 10-13, 2008, Proceedings*, ser. LNCS, vol. 5154. Springer, 2008, pp. 164–180.
- [36] Altera, "Metastability in altera devices," Altera, Tech. Rep., March 1999, online. <http://www.altera.com/literature/an/an042.pdf>.
- [37] Actel, "Metastability characterization report for actel flash fpgas," Actel, Tech. Rep., January 2008, online. http://www.actel.com/documents/Flash_Metastability_HBs.pdf.
- [38] P. Alfke, "Metastable recovery in virtex-ii pro fpgas," Xilinx, Tech. Rep., February 2005, online. http://www.xilinx.com/support/documentation/application_notes/xapp094.pdf.
- [39] B. Rogina and B. Vojnovic, "etastability evaluation method by propagation delay distribution measurement," in *Proceedings of the Fourth Asian Test Symposium*, November 1995, pp. 40–44.
- [40] 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 *Cryptographic Hardware and Embedded Systems - CHES 2003, Cologne, Germany, September 8-10, 2003, Proceedings*, ser. LNCS, vol. 2779. Springer, 2003, pp. 152–165.
- [41] Actel, *Actel Fusion Handbook*, November 2008, online. http://www.actel.com/documents/Fusion_HB.pdf.
- [42] S. Callegari, R. Rovatti, and G. Setti, "First direct implementation of a true random source on programmable hardware," *International Journal of Circuit Theory and Applications*, pp. 1–16, 2005.
- [43] T. Addabbo, M. Alioto, A. Fort, S. Rocchi, and V. Vignoli, "A feedback strategy to improve the entropy of a chaos-based random bit generator."
- [44] A. Gerosa, R. Bernardini, and S. Pietri, "A fully integrated chaotic system for the generation of truly random numbers," *IEEE Transactions on Circuits and Systems-I: Fundamental Theory and Applications*, vol. 49, no. 7, pp. 993–1000, July 2002.
- [45] S. Ozoguz and S. Ergun, "A non-autonomous ic chaotic oscillator and its application for random bit generation," in *Proceedings of the 2005 European Conference on Circuit Theory and Design*, September 2005, pp. 165–168.
- [46] S. Gregori and A. Cabrini, "Cmos discrete-time chaotic circuit for low-power embedded cryptosystems," in *48th Midwest Symposium on Circuits and Systems*, August 2005, pp. 1498–1501.
- [47] T. Zhou, M. Yu, and Y. Ye, "A pipelined switched-current chaotic system for the high-speed truly random number generation in crypto processor," in *19th International Conference on VLSI Design*, January 2006, p. 6.
- [48] F. Pareschi, G. Setti, and R. Rovatti, "A fast chaos-based true random number generator for cryptographic applications," in *Solid-State Circuits Conference, 2006. ESSCIRC 2006. Proceedings of the 32nd European*, September 2006, pp. 130–133.
- [49] V. von Kaenel and T. Takayanagi, "Dual true random number generators for cryptographic applications embedded on a 200 million device dual cpu soc," in *Custom Integrated Circuits Conference, 2007. CICC apos;07. IEEE*, September 2007, pp. 269–272.

- [50] S. M. Ulam and J. von Neumann, "On combination of stochastic and deterministic processes." *Bulletin of the American mathematical Society*, 53, 1120,, 1947.
- [51] "Butterfly effect," Wikipedia, online. http://en.wikipedia.org/wiki/Butterfly_effect.
- [52] T. Stojanovski and L. Kocarev, "Chaos-based random number generators-part i: analysis[*cryptography*]," *Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions on*, vol. 48, no. 3, pp. 281–288, March 2001.
- [53] S. Ergiin and S. Ozoguz, "Truly random number generators based on a double-scroll attractor," in *49th IEEE International Midwest Symposium on Circuits and Systems, 2006. MWSCAS '06.*, August 2006, pp. 322–326.
- [54] M. Drutarovsky and P. Galajda, "A robust chaos-based true random number generator embedded in reconfigurable switched-capacitor hardware," in *Radioelektronika, 2007. 17th International Conference*, April, pp. 1–6.
- [55] S. Callegari, R. Rovatti, and G. Setti, "Embeddable adc-based true random number generator for cryptographic applications exploiting nonlinear signal processing and chaos," *IEEE Transactions on Signal Processing*, vol. 53, no. 2, pp. 793–805, February 2005.
- [56] Y. Mao and W. Cao, L. Liu, "Design and fpga implementation of a pseudo-random bit sequence generator using spatiotemporal chaos," in *International Conference on Communications, Circuits and Systems Proceedings*, June 2006, pp. 2114–2118.
- [57] "Definitions and terminology for synchronization networks - series g: Transmission systems and media digital transmission systems - digital networks - design objectives for digital networks," International Telecommunication Union/ITU Telcommunication Sector, Tech. Rep., August 1996.
- [58] E. Ferre-Pikal, J. Vig, J. Camparo, L. Cutler, L. Maleki, W. Riley, S. Stein, C. Thomas, F. Walls, and J. White, "Draft revision of ieee std 1139-1988 standard definitions of physical quantities for fundamental, frequency and time metrology-random instabilities," in *Frequency Control Symposium, 1997., Proceedings of the 1997 IEEE International*, May 1997, pp. 338–357.
- [59] D. Hong, C. Dryden, G. Saksena, and M. Panis, "An efficient random jitter measurement technique using fast comparator sampling."
- [60] J. Wilstrup, "A method of serial data jitter analysis using one-shot time interval measurements," February 1998.
- [61] S. Sunter and A. Roy, "On-chip digital jitter measurement, from megahertz to gigahertz," *IEEE Design and Test of Computers*, vol. 314–321, July/August 2004.
- [62] T. Yamaguchi, M. Soma, D. Halter, R. Raina, J. Nissen, and M. Ishida, "A method for measuring the cycle-to-cycle period jitter of high-frequency clock signals," in *IEEE VLSI Test Symposium*, 2001, pp. 102–110.
- [63] B. Valtchanov, A. Aubert, F. Bernard, and V. Fischer, "Modeling and observing the jitter in ring oscillators implemented in fpgas," in *The 11-th IEEE Workshop on Design and Diagnostics of Electronic Circuits and Systems (DDECS)*, Bratislava, April 2008, pp. 158–163.
- [64] B. H. Leung, "A novel model on phase noise of ring oscillator based on last passage time," *IEEE Transactions on Circuits And Systems*, vol. 51, no. 3, pp. 471–482, March 2004.
- [65] V. Fischer, F. Bernard, N. Bochard, and M. Varchola, "Enhancing security of ring oscillator-based rng implemented in fpga," in *Field-Programable Logic and Applications (FPL)*, September 2008, pp. 245–250.

- [66] “Altera development kits offer,” Tech. Rep., online. http://www.altera.com/products/devkits/kit-dev_platforms.jsp.
- [67] M. Varchola, “Design and evaluation of optimized fpga ip-cores for cryptography,” in *Computer Architectures & Diagnostics (Pocitacove Architektury & Diagnostika - PAD)*, September 2008, pp. 113–118.
- [68] Q. Dou and J. A. Abraham, “Jitter decomposition in ring oscillators,” in *Asia and South Pacific Design Automation Conference ASP-DAC*, 2006, pp. 285–290.
- [69] T. Pialis and K. Phang, “Analysis of timing jitter in ring oscillators due to power supply noise,” in *Circuits and Systems, 2003. ISCAS apos;03. Proceedings of the 2003 International Symposium on*, 25–28 May 2003, pp. I-685–I-688.
- [70] A. Hajimiri, S. Limotyrakis, and T. Lee, “Jitter and phase noise in ring oscillators,” *IEEE Journal of Solid-State Circuits*, vol. 34, no. 6, pp. 790–804, June 1999.
- [71] J. Menezes, P. Oorschot, and S. Vanstone, *Handbook of Applied Cryptography*. New York: CRC Press, 1997, online. <http://www.cacr.math.uwaterloo.ca/hac/>.
- [72] G. Marsaglia, “The marsaglia random number cdrom with the diehard battery of tests of randomness, supercomputer computations research institute and department of statistics, florida state university,” <http://www.csis.hku.hk/diehard/>, 1996.
- [73] “Federal information processing standards publication fips pub 140-1,” U.S. Department of Commerce / National Institute of Standards and Technology, Tech. Rep., January 1994.
- [74] “Federal information processing standards publication fips pub 140-3 (draft),” U.S. Department of Commerce / National Institute of Standards and Technology, Tech. Rep., July 2007.
- [75] B. für Sicherheit in der Informationstechnik (BSI), “Functionality classes and evaluation methodology for deterministic random number generators,” <http://www.bsi.de/zertifiz/zert/interpr/ais20e.pdf>, 02 1999, version 1.
- [76] R. Corporation, *A Million Random Digits with 100,000 Normal Deviates*, 1955.
- [77] G. Marsaglia, “Diehard: Battery of tests of randomness,” Online. Available at: <http://stat.fsu.edu/pub/diehard/>, 1996.
- [78] —, “The marsaglia random number cdrom with the diehard battery of tests of randomness,” Online. Available at: <http://www.csis.hku.hk/diehard/>, 1996.
- [79] G. Marsaglia and W. Tsang, “Some difficult-to-pass tests of randomness,” *Journal of statistical software*, vol. 7, 2002, issue 3.
- [80] R. Snouffer, A. Lee, and A. Oldehoeft, “Nist special publication 800-29: A comparison of the security requirements for cryptographic modules in fips 140-1 and fips 140-2,” National Institute of Standards and Technology (NIST), Tech. Rep., June 2001.
- [81] S. J. Kim, K. Umeno, and A. Hasegawa, “Corrections of the nist statistical test suite for randomness,” *Joho Shori Gakkai Shinpojiumu Ronbunshu*, vol. 2004, no. 12, pp. 337–342, 2004.
- [82] W. Schindler, “Efficient online tests for true random number generators,” in *Cryptographic Hardware and Embedded Systems - CHES 2001, Paris, France, May 14-16, 2001, Proceedings*, ser. LNCS, vol. 2162. Springer, 2001, pp. 103–117.
- [83] W. Schindler and W. Killmann, “Evaluation criteria for true (physical) random number generators used in cryptographic applications,” in *Cryptographic Hardware and Embedded Systems - CHES 2002, 4th International Workshop, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, ser. LNCS, vol. 2523. Springer, 2003, pp. 431–449.

- [84] W. Killmann and W. Schindler, "A design for a physical rng with robust entropy estimators," in *Cryptographic Hardware and Embedded Systems - CHES 2008, Washington, DC, USA, August 10-13, 2008*, ser. LNCS, vol. 5154. Springer, 2008, pp. 146–163.
- [85] "Altera website," online. <http://www.altera.com/>.
- [86] "Xilinx website," online. <http://www.xilinx.com/>.
- [87] "Actel website," online. <http://www.actel.com/>.
- [88] "Lattice website," online. <http://www.latticesemi.com/>.
- [89] "Logic elements and logic array blocks in cyclone iii devices," Altera, Tech. Rep., online. http://www.altera.com/literature/hb/cyc3/cyc3_ciii51002.pdf.
- [90] *Spartan-3 FPGA Family Data Sheet*, Xilinx, online. http://www.xilinx.com/support/documentation/data_sheets/ds099.pdf.
- [91] "Clock networks and plls in cyclone iii devices," Altera, Tech. Rep., online. http://www.altera.com/literature/hb/cyc3/cyc3_ciii51006.pdf.
- [92] *Stratix IV Device Handbook*, Altera, December 2008, online. http://www.altera.com/literature/hb/stratix-iv/stratix4_handbook.pdf.
- [93] *Stratix III Device Handbook*, Altera, October 2008, online. http://www.altera.com/literature/hb/stx3/stratix3_handbook.pdf.
- [94] *Stratix II Device Handbook*, Altera, May 2007, online. http://www.altera.com/literature/hb/stx2/stratix2_handbook.pdf.
- [95] *Stratix II GX Device Handbook*, Altera, May 2007, online. http://www.altera.com/literature/hb/stx2gx/stxiigx_handbook.pdf.
- [96] *Stratix Device Handbook*, Altera, January 2005, online. http://www.altera.com/literature/hb/stx/stratix_handbook.pdf.
- [97] *Stratix GX Device Handbook*, Altera, March 2005, online. http://www.altera.com/literature/hb/sgx/sgx_handbook.pdf.
- [98] *Arria GX Device Handbook*, Altera, May 2008, online. http://www.altera.com/literature/hb/agx/arriagx_handbook.pdf.
- [99] *Cyclone III Device Handbook*, Altera, October 2008, online. http://www.altera.com/literature/hb/cyc3/cyclone3_handbook.pdf.
- [100] *Cyclone II Device Handbook*, Altera, February 2007, online. http://www.altera.com/literature/hb/cyc2/cyc2_cii5v1.pdf.
- [101] *Cyclone Device Handbook*, Altera, May 2008, online. http://www.altera.com/literature/hb/cyc/cyc_c5v1.pdf.
- [102] *Virtex-5 Family Overview*, Xilinx, September 2008, online. http://www.xilinx.com/support/documentation/data_sheets/ds100.pdf.
- [103] *Virtex-4 Family Overview*, Xilinx, September 2008, online. http://www.xilinx.com/support/documentation/data_sheets/ds112.pdf.
- [104] *Extended Spartan-3A Family Overview*, Xilinx, July 2008, online. http://www.xilinx.com/support/documentation/data_sheets/ds706.pdf.
- [105] *IGLOO Handbook*, Actel, August 2008, online. http://www.actel.com/documents/PA3L_HB.pdf.
- [106] *ProASIC3 Handbook*, Actel, October 2008, online. http://www.actel.com/documents/IGLOO_HB.pdf.

- [107] J. L. Danger, S. Guilley, P. Hoogvorst, and M. Amara, "Very high speed trng in fpga's," in *DATE'07*, 2007.
- [108] S. H. M. Kwok and E. Y. Lam, "Fpga-based high-speed true random number generator for cryptographic applications," in *TENCON 2006. 2006 IEEE Region 10 Conference*, November 2006, pp. 1–4.
- [109] Altera, "Possible causes for pll loss of lock," online. http://www.altera.com/support/devices/pll_clock/basics/pll-loss-lock.html?GSA_pos=4&WT.oss_r=1&WT.oss=PLL%20lock.
- [110] "Xilinx dll jitter calculator," Xilinx, Tech. Rep., online. http://www.xilinx.com/support/documentation/data_sheets/s3a_jitter_calc.zip.
- [111] "Jitter comparison analysis: Apex 20ke pll vs. virtex-e dll," Altera, Tech. Rep., January 2001.
- [112] R. Fairfield, R. Mortenson, and K. Coulthart, "An lsi random number generator," in *Proceedings of CRYPTO 84 on Advances in cryptology*, 1984, pp. 203–230.
- [113] S. Yoo, B. Sunar, D. Karakoyunlu, and B. Birand, "A robust and practical random number generator," August 2007.
- [114] D. Schellekens, B. Preneel, and I. Verbauwhede, "Fpga vendor agnostic true random number generator," in *Proc. 16th International Conference Field Programmable Logic and Applications (FPL)*. IEEE, 2006, 6 pages.
- [115] P. Chodowicz and K. Gaj, "Very compact fpga implementation of the aes algorithm," in *Cryptographic Hardware and Embedded Systems - CHES 2003, Cologne, Germany, September 8-10, 2003, Proceedings*, ser. LNCS, vol. 2779. Springer, 2003, pp. 319–333.
- [116] B. Sunar, "Response to dichtl's criticism," March 2008.
- [117] M. Dichtl, B. Meyer, and H. Seuschek, "Spice simulation of a 'r
- [118] V. Fischer, M. Drutarovský, M. Simka, and N. Bochard, "High performance true random number generator in altera stratix fplds," in *Field-Programmable Logic and Applications (FPL)*, ser. LNCS, vol. 3203. Springer, September 2004, pp. 555–564.
- [119] M. Varchola, M. Drutarovsky, R. Fouquet, and V. Fischer, "Hardware platform for testing performance of trngs embedded in actel fusion fpga," in *Radioelektronika, 2008 18th International Conference*, May 2008, pp. 145–148.
- [120] M. Simka, M. Drutarovsky, V. Fischer, and J. Fayolle, "Model of a true random number generator aimed at cryptographic applications," in *International Symposium on Circuits and Systems (ISCAS)*, 2006, pp. 5619–5622.
- [121] P. Kohlbrenner and K. Gaj, "An embedded true random number generator for fpgas," in *Proceedings of the ACM/SIGDA 12th International Symposium on Field Programmable Gate Arrays, FPGA 2004*, ser. ACM, R. Tessier and H. Schmit, Eds. Monterey, California, USA, 2004, pp. 71–78.
- [122] T. E. Tkacik, "A hardware random number generator," in *Cryptographic Hardware and Embedded Systems - CHES 2002, Redwood Shores, CA, USA, August 13-15, 2002, Revised Papers*, ser. LNCS, vol. 2523. Springer, 2003, pp. 450–453.
- [123] *MCF51JM128 ColdFire Integrated Microcontroller Reference Manual*, Freescale Semiconductor, January 2008.
- [124] J. Golic, "New paradigms for digital generation and postprocessing of random data," *Cryptology ePrint Archive*, Report 2004/254, 2004.

-
- [125] —, “New methods for digital generation and postprocessing of random data,” *IEEE Transactions on Computer*, vol. 55, no. 10, pp. 1217–1229, October 2006.
- [126] S. Walker and S. Foo, “Evaluating metastability in electronic circuits for random number generation,” in *VLSI, 2001. Proceedings. IEEE Computer Society Workshop on*, May 2001, pp. 99–101.
- [127] D. C. Ranasinghe, D. Lim, S. Devadas, D. Abbott, and P. Cole, “Random number from metastability and thermal noise,” *Electronics Letters*, vol. 41, no. 16, 2005.
- [128] C. Tokunaga, D. Blaauw, and T. Mudge, “True random number generator with a metastability-based quality control,” *Solid-State Circuits, IEEE Journal of*, pp. 78–85, January 2008.
- [129] J. Hrubý, “Kvantový Šumátor,” *Crypto-World*, no. 11, pp. 12–17, 2008, online. <http://www.cryptoworld.info/>.