ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

ARCES – ADVANCED RESEARCH CENTER ON ELECTRONIC SYSTEMS
FOR INFORMATION AND COMMUNICATION TECHNOLOGIES E. DE CASTRO

CHAOS-BASED RANDOM NUMBER GENERATORS: MONOLITHIC IMPLEMENTATION, TESTING AND APPLICATIONS

Fabio Pareschi

TUTORS

Professor
Gianluca Setti

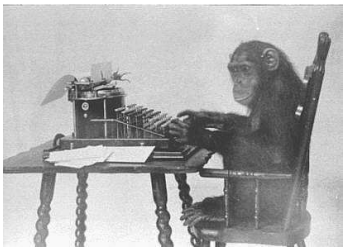
Professor
Riccardo Rovatti

COORDINATOR

Professor
Riccardo Rovatti

PHD. THESIS
January, 2004 – December, 2006

PHD PROGRAM IN INFORMATION TECHNOLOGY
CYCLE XIX – ING-INF/01



"Concevons qu'on ait dressé un million de singes à frapper au hasard sur les touches d'une machine à écrire, [..] ces volumes se trouveraient renfermer la copie exacte des livres de toute nature et de toutes langues conservés dans les plus riches bibliothèques du monde."

Èmile Borel, *J. Phys. 1913*

Contents

Chapter 1

Introduction

THE WORD RANDOM is used typically to express lack of purpose, cause, order, or predictability. A random process is a repeating process in whose outcomes it is impossible to find a describable deterministic pattern. The term randomness is often used in statistics to signify well defined statistical properties, such as lack of bias or correlation. Saying that a variable is random means that the variable follows a given probability distribution; under these terms, random is different from arbitrary, because to say that a variable is arbitrary does not imply that there is such determinable probability distribution.

The existence of random processes have been known and have been used since ancient times. For example, the *divination* was the attempt of giving a supernatural interpretation of some random events. However in all times and cultures the prevalent association of random events was with gambling, as in the common perception examples of random events include dice, coin flipping, or the shuffling of playing cards.

Actually, nowadays, legalized gambling still represents a very important economic aspect of modern society. For this reason, in every state where gambling is legalized, plenty of stringent regulations exist regarding the devices used to ensure the unpredictability of the outcome results. This is true both for physical devices like dice, cards decks, etc; employed in gambling facilities like Casinos, as well as for all electronic devices used for gambling [66].

The usage of random in electronic devices is strictly connected to the implementation of electronic random number generators. By definition, Random Number Generators (RNGs) are a class of devices whose aim is to generate a sequence of perfectly *independent and identically distributed* (IID) symbols, with the property that, when restarted, they do not reproduce every time the same sequence (*non repeatability*). They find several application in many engineer-

ing tasks, not limited to gambling devices. For example in Information and Communication Technology (ICT) they are used widely in device testing and simulation [80]; sequences of (pseudo) random numbers are often used to generate a white spectrum; or they can be used to generate a simulated network traffic with certain statistical properties in order to perform an off-line test of a network device. Random numbers are also used in computer simulation *(Monte-Carlo simulations)* to include all non-idealities (e.g. noise, device mismatches, particle interactions) present in real systems.

However even though in device testing the requirements on the used generators are usually not so tight, since their only purpose is to introduce some deviations from the ideal and perfectly deterministic behavior of a system, thus mimicking a real system (roughly speaking, their purpose is to simulate a system noise), an application where random numbers generators are fundamental and where true unpredictability is a main issue is computer security. Particularly in the last years, the increasing demand of electronic financial transactions, as well as the storing and the managing of personal, sensible data over an open and insecure public network as Internet, as well as the expanding use of wireless communication, has led to define new high-security standards for data encryption [43].

Data encryption is defined as the process of converting ordinary information (*plaintext*) into something unintelligible (*ciphertext*); decryption is the reverse process, i.e. moving from unintelligible ciphertext to plaintext. A cipher is a pair of algorithms which perform this encryption and the reversing decryption. The detailed operation of a cipher is controlled both by the algorithm and, in each instance, by a key. A cryptographic key is a secret parameter (that means that is known only to the communicants) for the cipher algorithm. Keys are fundamental in modern cryptography as ciphers without keys, though very common in the early history of cryptography, are trivially breakable and so rather less than useful.

In other words, modern cryptography has transferred all the unintelligibility from the cipher to the key. Many standard algorithms exist; the more commonly used ciphers are the DES [67] and the AES [69], which have been designated cryptography standards by the US government (though DES's designation was finally withdrawn after the AES was adopted in 2001).

These ciphers are public and universally studied, as well as known not to present intrinsic flaws or weakness; the used key is a string of bits, usually ranging from hundreds to thousands of bits depending on the type of cipher used. However, like a common phrase often used by information security offi-

cers is that "a chain is only as strong as its weakest link", it is useless to develop and use a high-security encryption/decryption scheme like the above standard ciphers, if the generation and the distribution of the security key do not follow the same high-security standards.

The generation of a cryptographic key is the link between modern cryptography and random numbers. The key is secret; it has not to be exposed, but also it is extremely important that it cannot be guessed by anyone. For this reason the key is chosen randomly, i.e. essentially it is a strings of random bits. It is fundamental that the key is *truly* random, i.e. it must not happen, for example, that looking at the sequence of bits composing the key, some patterns or some regularities are found, that may help to guess the key from the knowledge of part of it. This would help in the effort to guess the key, lowering the security of the system. In conclusion, good cryptography requires good random numbers [47, 65].

Historically, the first way used to generate random numbers in ICT was represented by pseudorandom generators. They appeared in the early 50s together with the first electronic computing machine, thus substituting existing random number tables used by researchers [24].

A pseudorandom number generator is an algorithm that generates a sequence of numbers which approximates some of the properties of random numbers, while in fact, they are deterministically computed starting from an initialization vector, called *seed* [23, 32]. The seed, which actually represents the only uncertainty in this class of generator, is set at the system initialization; it is not uncommon to initialize a pseudorandom generator with a sort of true random vector, that can be taken, for example, from the lower bits of the system clock. Of course the process is not random at all, as John von Neumann, a pioneer in the field of modern computing and of theory of games, put it [23]: "Anyone who considers arithmetical methods of producing random digits is, of course, in a state of sin."

In fact, because all pseudorandom generators run on a deterministic algorithm, its output will inevitably have one property that a true random sequence can never have: periodicity. In fact, a periodic system is a repetitive system, and so completely predictable after the disclosing of the period. Such a generator is usually implemented on a finite state machine, i.e a machine whose model of behaviour can be described in terms of a limited number of *states*, which store information about the past evolution, and the possibility to change from one state to another (i.e. a *transition*) depending on some external or in-

ternal events. Since a pseudorandom generator is an *autonomous* system, its transitions do not depend on external events (of course, neglecting the necessary external clock signal), but only on the internal state of the system. Being this machine fully deterministic, and given sufficient time, it will revisit a previously visited internal state, after which it will repeat the prior sequence forever. Non-periodic algorithms can be designed, but internal numbers of states (i.e. memory requirements) would grow without limit during runs; even if this is theoretically possible, it is not practicable in real equipment. However, since the length of the maximum period typically doubles with each bit added to the computing precision of the machine used to implement the algorithm, it is easy to build pseudorandom generators with periods so long that no computer could ever complete a single period.

This, however, does not change the fact that a pseudorandom generator *is not* a random generator.

In opposition to these generators, many devices exist which are based on the direct observation of a physical process that exploits random-like features, like the flip of a coin or the roll of a dice. The are called *true* random number generators, and instead of being based on the observation of macroscopic phenomena like those described above, they are more commonly built upon a microscopic phenomenon.

There are several physical phenomena presenting random like features, ranging from the Brownian motion to quantum effects. All these phenomena arise from the basic physical processes at the molecular or lower level, and are usually referred as “noise”.

The history of noise is fascinating, and begins in the early days of the invention of the microscope, when came the discovery that a drop of pond water contains an incredible world of microscopic life, single-celled and multicelled organisms of incredible varieties. Most of the organisms moved about seemingly without rhyme or reason. Anyway, it was noticed that other things like pollen also had erratic movements, and yet they seemed very different than the obviously alive paramecium, amoebas, and such. Many thought that these erratic movements were possibly due to some primitive life force. This question is generally regarded as having been solved by the English botanist Robert Brown in 1827 [12]. He started with the usual pollen and then went through observing that an incredible number of materials that are obviously not alive has the same, identical, behavior; in this way he was able to exclude that the motion was due to pollen particles being “alive”, although the origin of the

motion was yet to be explained.

The origin of the motion was explained by Einstein, in one of his paper of the so-called *year of miracles* (1905). In that year there Einstein produces the three so called miracles: one of them is the proof of the existence of atoms based on the explanation of the Brownian motion [15, 18]; the other two were the introduction of the basic ideas of the quantum physics [14] and of the special theory of relativity [16, 17]. Einstein explained that the Brownian motion is caused by the irregular thermal movement of the molecules of the liquid. He also suggests that the relation could be used for the determination of the so called (but at the time still unnamed) Avogadro's number, thus establishing an extremely important connection with the theory of heat. It was the french physicist Jean Baptiste Perrin few years afterwards who did the experimental work to test Einstein's predictions and computed the Avogadro's number [19], though, it was Einstein who developed the statistical properties and got specific results.

Many phenomena that can be described in a very similar way to the Brownian motion was discovered in the following years with the introduction of the first electronic devices; the Johnson thermal noise over a resistor and the Shot noise in vacuum tubes are just few examples. Also, with the introduction of quantum mechanics, which is inherently probabilistic, new kind of noise arose. The radioactive nuclear decay of instable isotopes is a well known example of what could be referred as a quantum noise.

The basic idea of a true random generator is to embed a random like phenomenon and observe it. The generation of the uncertainty is left to the observed phenomenon, while the device itself acts only as an interface with the phenomenon and to convert the uncertainty in a useful form of randomness [49, 52, 81].

The meaning of "useful form of randomness" depends on the application. Since our scope is ICT, it means "random bits". Being the bit the basic information unit, used to express any other information, it is possible from a string of random bits of an adequate length to retrieve any possible random information. Note that, for this reason, we can freely confuse a random number generator with a random bit generator.

The generation of a random bit can be modeled as the result of the flips of an unbiased *fair* coin where the sides "head" and "tails" is associated to the bits "0"and "1". The unbiased fair coin is a coin where each flip having a probability of exactly $1/2$ of producing a "head" or a "tail"; furthermore,

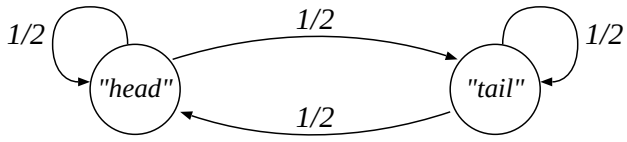


Figure 1.1: Markov chain associated to the unbiased "fair" coin toss.

the flips are independent of each other: the result of any previous coin flip does not affect future coin flips. In this way the value of the next element in the sequence cannot be predicted, regardless of how many elements have already been produced. This behavior can be described as the *Markov chain* of Figure 1.1. A Markov chain could get various different definition; to our purpose it is simply an autonomous finite-state machine whose transition from one state to another is regulated by a discrete-time stochastic process [57, 59, 77]. At every time step, the probability to remain in the same state or to jump to another one depends only by the current state. Briefly speaking, it is a finite state machine in which transitions happen randomly. The arrows in figure indicate the probability of of a transition between the two states, named "head" and "tail", and corresponding to the possible outcomes of the system.

For these reasons, all true random number generator can be schematized like a device capable of elaborating a noise-like process to obtain a behavior like the Markov chain in Figure 1.1. They can de described in terms of three fundamental components:

ENTROPY SOURCE: this is the physical process from whose observation random data are extracted. All the uncertainness generated depends on it; so it represents the core of the device. Mathematically, it is very common to refer to the quantity of uncertainty generated as *entropy* of the system; hence the name entropy source.

HARVESTING MECHANISM: data from the entropy source has to be *read* and to be *converted* in a sequence of bits; this process is called harvesting. It is fundamental that the harvesting mechanism does not interfere the physical process above, since this would result into a degradation of the statistical properties of the above process, and so in a reduced amount of available entropy.

POST-PROCESSING: although this component is not strictly necessary, it is often used to digitally process the sequence of random bits and strengthen the random number generator design. A post processing stage can

be used to mask imperfection in the entropy source; more often it is necessary to mask impairing in the observed process due to the harvesting mechanism. Many algorithms have been developed for extracting a random bit sequence out of almost any stream without any knowledge of its generation process. One example was given by von Neumann, to be used if a cheater has altered a coin to prefer one side over another (a *biased* coin) [23]:

1. Toss the coin twice.
2. If the results match, start over, forgetting both results.
3. If the results differ, use the first result, forgetting the second.

The reason this process produces a fair result is that the probability of getting heads and then tails must be the same as the probability of getting tails and then heads, as the coin is not changing its bias between flips. By excluding the events of two heads and two tails by repeating the procedure, the coin flipper is left with the only two remaining outcomes having equivalent probability. However, as in the example where for every game we have to toss the coin at least twice, in such algorithms the output data rate is usually slower than the input one.

One of the most famous random number generator, as one of the first low-cost true random number generator in ICT, was the Intel random number generator presented in 1999 [54] and integrated in Intel chipsets up to few years ago, when was retired due to some flaws discovered. The generator was based on the direct observation of the thermal noise upon a resistor; this noise was amplified and used to modulate a low-speed local oscillator. A second high-speed local oscillator was sampled by the low-speed one, thus providing random bits. After that, a von Nuemann post-processing is then used to improve the quality. However the performances of the Intel random number generator were not exceptional especially in terms of speed; it achieved a throughput of about 75 Kbit/s, while many generators nowadays can reach speeds ranging from Megabits to hundreds of Megabits per second.

In this dissertation a detailed description of two true random number generators is provided. Both generators have been designed in integrated CMOS technology, the first one in AMS 0.35 μm technology and the second in UMC 180 nm. They both rely on a *chaotic* system as entropy source.

The concept of chaos is sometimes misinterpreted as "disorder" or maybe even "random". Mathematically, a chaotic system is a system which is effec-

tively a deterministic system (since it is possible to write down all the evolution equations), and presents two unique characteristics: *(a)* the presence of irregular, aperiodic trajectories; *(b)* an extreme sensitivity to initial conditions. If the lack of periodicity is, as seen above, a necessary condition for unpredictability, it is the second characteristic the more interesting one. The sensitivity to initial conditions is often popularly referred to as the butterfly effect; the term "butterfly effect" itself is related to the work of Edward Lorenz, who in a 1963 paper for the New York Academy of Sciences noted that "One meteorologist remarked that if the theory were correct, one flap of a seagull's wings could change the course of weather forever". Later speeches and papers by Lorenz used the more poetic butterfly instead of the seagull, asking "Does the flap of a butterfly's wings in Brazil set off a tornado in Texas?".

The concept of the butterfly effect is often used in popular media, usually inaccurately. The basic concept is that, if we have two identical systems starting at two initial condition, which are apparently identical, but instead present a even very small difference, they may end up with two totally different evolutions [46, 60]. In this way, a long term prediction of a chaotic system is practically impossible, due to the assumption that when observing a real system, it is possible to measure the initial condition of the system only with a limited precision. The classical example of such a system is the weather.

At this point, one could argue that chaos-based random number generators and pseudorandom generators can be considered similar, since they are both based on a deterministic algorithm., Hence, two aspects make them different. First, being a chaotic circuit an *analog* machine, it is necessary to introduce a quantization of the state to generate random bits. Being the quantization a non-reversible operation, the internal state of the system cannot be retrieved by the only knowledge of the quantized values. Second, like any other analog electronic circuit, a chaotic circuit is influenced by the noise. Even neglecting the noise during operation, which continuously modifies the internal state and so the evolution, it would be enough to consider that the initial system condition is set by the noise at the system startup, thus regulating all the evolution of the system, to consider this generator like a true random number generator. From this point of view, a chaos-based random number generator is not different from a generator based on the direct observation of a noise-like phenomenon.

In other words, choosing the right chaotic circuit and quantization function, it is possible to study the evolution of the quantized states just like a *truly* probabilistic finite-state machine, i.e. a *Markov chain*, regardless of the deter-

ministic trajectories followed by the analog state [20, 25, 58]. This is exactly the situation of Figure 1.1.

Actually, the use of chaos, in particular of discrete-time chaotic circuit in the realization of random numbers generators has been known since many years [34, 44, 78, 79]. Ulam and von Neumann suggested the use of *logistic map* in 1947 [20], partly because it had a known algebraic distribution, and its iterated values could be easily transformed into any desired distribution. The advantages of dealing with a chaotic circuit is that it is theoretically possible to avoid any interference with its statistical properties. Considering at design time both the chaotic circuit and the harvester, all parasitics introduced by the harvester can be included and compensated in the chaotic circuit model. This means that there is no impairing of the chaotic circuit, and (theoretically) all the entropy is available to the output bit stream. Also, as already noticed by Ulam and von Neumann, the available entropy rate in a discrete time chaotic source is directly proportional to the circuit speed. The higher the working speed of the circuit, the higher the available entropy rate. The upper bound is set only by the technology limit, and it does not depend on the underlying physical process; this virtually allows to design a circuit generating any desired amount of entropy.

The approach followed in this dissertation presents a second advantage, i.e. that the used schematic is a slightly modification of a schematic already used in pipeline analog-to-digital converters (ADCs) based on 1.5 bit/stage cells [74]. This solution allow a vast re-use of design competences and macroblocks developed in this field and also ensure high embeddability in all mixed signal integrated circuits, as well as a very high working speed, up to several Megahertz. At this point it is necessary to recognize that the original idea of this approach was firstly proposed by Sergio Callegari in 2002 [38]. Here that idea was studied and developed from a circuit level point of view. The main original work of this dissertation is the hardware implementation of that basic idea.

The two implemented circuits are described accurately in Chapter 2; both prototypes have been fabricated within the mini@sic Europractice framework, thus allowing a reduction of fabricating costs. Many thanks are necessary for the staff of the two Europractice centers (Fraunhofer-IIS, Erlangen for AMS technology and IMEC vzw, Leuven for UMC technology) for the support and the help given in the design process.

Being prototypes whose purpose was only to test the analog core, no post-

processing stages have been integrated. However, several post-processing stages have been considered during the test phase; a description of them can be found in Chapter 3. Some of the post-processing functions considered are completely new; for them, an evaluation of performance is given in terms of the minimum complexity required to transform the output stream of the designed prototypes into a true random bit stream.

Finally, an evaluation of the performances of the designed prototypes is provided. A particular attention has been given to this section, since testing and validating a random number generator is a non-trivial task.

Many tests for randomness exist; they are extensively described in Chapter 4 along with an interpretation of their results. We made an intensive use of the SP 800-22 test suite, developed by the U.S. National Institute of Standard and Technology and first published in October 2000 [70]. This suite was preferred to other suites, in particular to Marsaglia DieHard suite, for two main reasons: *(a)* the NIST suite is homogeneous, i.e. all tests in the suite analyze a sequence of the same length; *(b)* for all tests in the suite a strong mathematical background is available. This does not happen, for example, in Marsaglia's suite, where the sequence length required could vary up to two orders of magnitude among different tests. All the above tests are statistical tests; in fact, quoting the NIST special publication, "Randomness is a probabilistic property; that is, the properties of a random sequence can be characterized and described in terms of probability."

The meaning of this is briefly described in the following. Consider a random number generator, and assume to have to test it. The only way to perform a test is to let the generator run, and look at a generated a sequence. Suppose to get "10110100101110110100101011…"; it would be difficult, at a first glance, to find a pattern in the generation of the bits, as difficult would be object about its randomness. Instead after the sequence "11111111111111111111111111…" everyone will recognize a pattern, and will consider the generator under test as completely non random. Actually, a statistical test works exactly in this way, except the fact that the output of a test is not an answer "it is random" or "it is not random", but a probability. In fact, it is not true that the sequence "11111111111111111111111111…" is not random; actually this sequence has the same probability to be generated by a random number generator as any other sequence; on the contrary, a generator that is not capable of generating this sequence is not to be consider random. The only thing that one could say looking at such a sequence is "well, *probably* this sequence is not random".

Starting from this, a statistical test gives the probability that, under some

assumptions that depends on the test, the sequence comes from a random number generator. The output of the test is called p-value which stands for *probability value* and, informally speaking, indicates the probability that has a perfect random generator to generate a sequence that is "less random" than the sequence under test.

The test used here is slightly more complex than the above described. Instead of a single p-value, a number of them are considered, following the last section of the NIST special publication, thus proving more reliable results. During the test phase, it was noted that this test is very sensible to the propagation of approximation errors introduced in the basic test, and can result in an always-failed test. This has been investigated for the simplest cases, and a proper solution was proposed and applied.

Results from tests on the two chaotic random number generators are presented in Chapter 5, as well as a comparison between the two proposed generators, and two high-end physical process based true-random generators. The first one is the VIA PadLock generator [42] integrated in a VIA C3 processor of an EPIA MX-II 10000 system; the second is an high end quantic generator developed by idQuantique [53], based on single photon reflection on a semi-transparent mirror. Results confirm that the approach proposed in this dissertation can effectively be used to design high-end random numbers generators, since in terms of quantity of randomness generated the designed chaotic prototypes outperforms the two commercial generators by one order of magnitude.

This dissertation ends with two chapters whose topic is slightly different from the main theme. The first one (Chapter 6) describes an application of random numbers, i.e. the reduction of Electromagnetic Interferences (EMI) using a random modulation of the clock. Two prototypes implementing a spread spectrum clock generator with EMI reduction were implemented, the first one running at quite slow speed, while the second one was designed to work at very high speed. The presentation of the design and of the measurements of the first prototype, respectively, won the best paper award both at the 16^{th} International Zurich Symposium on Electromagnetic Compatibility, 2005 and at 17^{th} European Conference on Circuit Theory and Design, 2005. This topic was studied in collaboration with Luca Antonio De Michele, ARCES - University of Bologna.

Chapter 7 describes another security related issue, that is the so called design of side channel attack resistant circuit, which is a main problem of all the embedded security integrated circuits. This last topic was studied during a study-abroad period in collaboration with Katholieke Universiteit Leuven,

Belgium.

The main innovative points developed by author during the described period are here briefly summarized, listed in chronological order.

- A theoretical noise robustness condition for PWAM chaotic maps has been elaborated. It is described in Appendix A, Section A.6, and it links the noise robustness only to topological properties of the chaotic maps. The condition found has a very general validity, since it does not make any assumption on the noise shape. See [1, 11].

- A prototype implementing an ADC-based RNG has been designed and tested. Its description can be found in Chapter 2, while results of testing are in Chapter 5. The performances of this prototype are very high, since, with a comparable quality, outperforms the two high-end commercial RNGs used in the comparison by one order of magnitude. This prototype was the first implementing the ADC-based RNG idea. For this topic, see [3, 8].

- Two prototypes of a spread spectrum clock generator for EMI reduction which use a random modulation are implemented. They are described in Chapter 6 and they use an ADC-based RNG as source of randomness. These prototype are the first implementing a frequency *binary* modulation, which is a modulation where the driving signal is a binary PAM signal, and allows a maximum EMI reduction with respect to all other known modulations. See [4, 7, 9, 10].

- A new, simple and effective post-processing scheme is proposed and analyzed. Its description can be found in Chapter 3, Section 3.6, while results on the effectiveness of this post-processing stage are reported in Chapter 5, Section 5.2. See [5]

- A bound on the application of second level NIST statistical tests for randomness has been found. This is described in Chapter 4, Section 4.4 and relates the approximation errors introduced into the reference distribution of the test, to a maximum error in the test outcome. For this topic, see [2].

Chapter 2

Hardware Implementation of a Chaos-Based RNG

THIS CHAPTER describes the implementation of the two designed RNG prototypes, starting from a brief overview of pipeline A/D converters, recognizing the similarity between a pipeline A/D converter stage and a chaotic map, and describing the two prototypes in detail, along with the guide-line used in the design. The theoretical demonstration that the proposed circuits can, in ideally conditions, generate true random bits is out of the purpose of these chapter, and can be find in Appendix A.

2.1 Pipeline A to D Converters

A pipeline A/D converter belongs to the category of successive approximation converters, in which the mapping between the analog input and the digital output is completed in more than one step, exploiting a binary search of the digital value closer to the input analog quantity. In particular, in pipeline converters this is done by performing a series of h coarse intermediate conversions over different hardware blocks at different time steps [74].

The typical structure of these converters is presented in Figure 2.1, depicting a h-stages converter which provides a representation of an input variable $v^{(in)}$ defined on an interval X into a l-bits numerical notation. Note that, even if this is supposed in this dissertation, it is not strictly necessary that all stages are identical; in particular the last stage, having nothing downhill, always presents a simpler structure and it is usually composed by a simple flash converter.

The i-th stage computes, usually with a small and fast flash converter, a

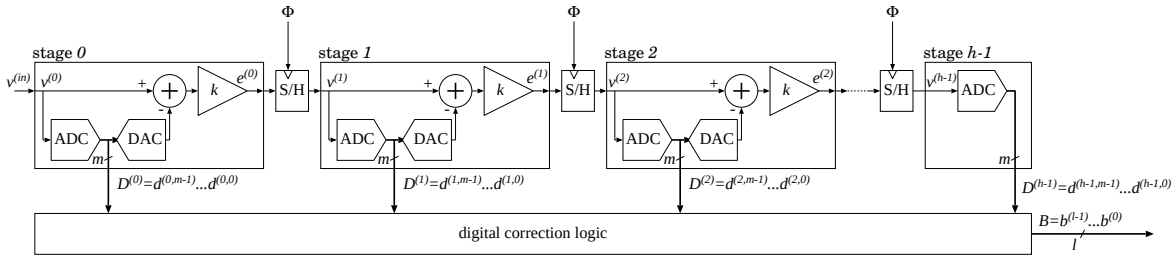


Figure 2.1: Basic structure of a pipeline ADC.

coarse m-bit representation $D^{(i)} = d^{(i,m-1)} \dots d^{(i,0)}$, of its input $v^{(i)}$ sampled at the time step n, and then calculates (and rescales) an analog *error conversion* $e^{(i)}$ to be passed at the time step $n+1$ to the following stage $(i+1)$-th as its input $v^{(i+1)}$.

In this design, only the first stage provides a direct conversion of the input $v^{(in)} \equiv v^{(0)}$; all other stages provide a representation of the intermediate conversion errors. Since the conversion error $e^{(i)}$ of the stage i is bounded in an interval smaller than X, it is sensible to rescale it before passing it to the next stage as $v^{(i+1)}$ in order to let every $v^{(i)}$ span the whole available range X. Note that this is a necessary condition for having identical stages; otherwise no additional information about the conversion can be retrieved from all stages beyond the first. Then, a *digital correction logic* processes the digital outputs of all the h stages in order to retrieve the l bits $b^{(l-1)} \dots b^{(0)}$ of the conversion, with $l \leq h \cdot m$.

It is easy to see that if $k = 2^m$ (e.g. $m = 2$ bits, $k = 4$), then the conversion is done exactly as in a SAR *(Successive Approximation Register)* converter, and the conversion word is obtained just by collecting in the right order all the intermediate conversion bits, with $l = h \cdot m$. However in the general case, $k < 2^m$ and the number of significative bits l in the conversion is smaller then the total number of computed bits $h \cdot m$; this means that there is a sort of *redundancy*. This redundancy, associated to a proper correction logic (hence, the name "digital correction logic") can be used to relax some constraints about the accuracy in the circuital implementation.

For this reason, the maximum number of stages in the pipeline (the higher the number of stages used, the higher the resolution of the converter) is not limited by the accuracy of the implementation but by the noise. In particular the noise introduced by the first stage, that passes through and is amplified by all stages, is the main factor in the determination of the maximum number of stages. In practical cases, the number of stages is limited to 8–10.

One major advantage of this approach is that the flow of information can be synchronized exactly as in a digital pipeline. Since the various stages are separated by *sample and hold* blocks (S/Hs), every stage is free to start operating on the next piece of data as soon as the following S/H has stored the rescaled conversion error. This permits to increase the throughput of the system up to the inverse of the latency of a *single* stage, which is much larger than the inverse of the time needed by the *whole* conversion, at the cost of an increasing complexity of the digital correction logic, which has to process data coming from different time instants.

One of the most used configuration for pipeline A/D converters is the so-called *one bit and a half* per stage [33, 41]. In this arrangements, supposing X the normalized interval $X = [-1, 1]$, the A/D conversion function $Q(x)$ employed at each stage is:

$$Q(x) = \begin{cases} -1, & \text{for} \quad x < -\frac{1}{2} \\ 0, & \text{for} \quad -\frac{1}{2} \leq x < \frac{1}{2} \\ +1, & \text{for} \quad x \geq \frac{1}{2} \end{cases}$$

Obviously, to represent this three-level quantization function, at least two bits are required. Usually the conversion is obtained by confronting $v^{(i)}$ with the two values $\pm 1/2$ by means of two comparators; it is common to take a thermometric coding for $D^{(i)}$, so that each $d^{(i,j)}$ is the output of a comparator:

$$D^{(i)} = d^{(i,1)}d^{(i,0)} = \begin{cases} 00, & \text{for} \quad v^{(i)} < -1/2 \\ 01, & \text{for} \quad -1/2 \leq v^{(i)} < 1/2 \\ 11, & \text{for} \quad v^{(i)} \geq 1/2 \end{cases} \tag{2.1}$$

Hence, $e^{(i)} = k\left(v^{(i)} - Q\left(v^{(i)}\right)\right)$, so if $v^{(i)}$ spans in $X = [-1, 1]$, then $e^{(i)}$ spans in $[-k/2, k/2]$. To take full advantage from this architecture, the rescaler has to be set with a gain equal to $k = 2$, so all the $v^{(i)}$ take values in the same range as $v^{(in)}$:

$$e(x) = \begin{cases} 2x + 2, & \text{for} \quad x < -\frac{1}{2} \\ 2x, & \text{for} \quad -\frac{1}{2} \leq x < \frac{1}{2} \\ 2x - 2, & \text{for} \quad x \geq \frac{1}{2} \end{cases} \tag{2.2}$$

The conversion function $Q(x)$ and the error function $e(x)$ are reported in Figure 2.2.

2.2 ADC-based Chaotic Map

A complete treatment about chaotic maps, PWAM maps and Markov chain can be found in Appendix A. In this chapter, it is enough the following

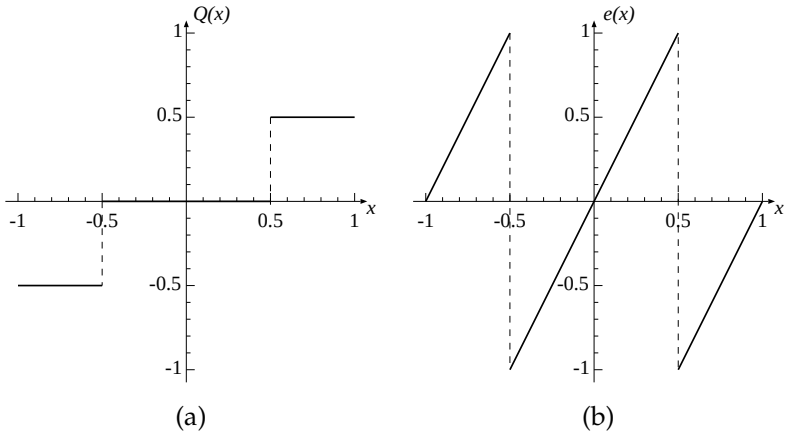


Figure 2.2: (a) Quantization function $Q\left(x\right)$; and (b) error function $e\left(x\right)$ of the 1.5 bits A/D converter.

REMARK 1. A chaotic map is defined as the a discrete-time autonomous system

$$x_{k+1} = M\left(x_k\right), \quad M : X \to X \tag{2.3}$$

starting from an arbitrary initial condition $x_0 \in X$, it generates the sequence

$$x_0, x_1, x_2, x_3, x_4, x_5, \ldots$$

that, given some properties on M, has all features of a chaotic sequence, i.e. aperiodicity, complexity, and strong dependence on initial condition. Additionally, a chaotic map is a Piece-Wise Affine Markov (PWAM) map when, informally speaking, a partition $\mathcal{X}$ of X exists such that *(a)* M is piece-wise affine; and *(b)* M is built upon the "grid" identified by the intervals of $\mathcal{X}$. The main property of a PWAM map is that the evolution of the system can be studied with a Markov chain: each interval of $\mathcal{X}$ represents a state in the Markov chain, and the jump from one interval to another in the map evolution corresponds to a state transition in the associated Markov chain.

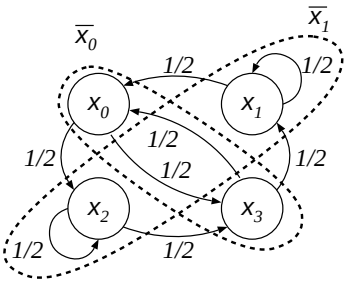
Actually, the error function $e\left(x\right)$ of Figure 2.2b fulfills all the requisites for being used in the implementation of a PWAM map, with $M\left(x\right) = e\left(x\right)$ [39], assuming a Markov partition $\mathcal{X} = \{X_0, X_1, X_2, X_3\}$ equal to

$$\mathcal{X} = \left\{ \left[-1, -\frac{1}{2}\right), \left[-\frac{1}{2}, 0\right), \left[0, \frac{1}{2}\right), \left[\frac{1}{2}, 1\right] \right\}.$$

The kneading matrix $\mathcal{K}$ and the four-state Markov chain associated to this map (referring to state x_0 if $x \in X_0$, x_1 if $x \in X_1$ and so on) are shown in Figure 2.3a and b, respectively.

The associated Markov chain is clearly not suitable for direct generation of identically distributed symbols; however, due to its particular structure it

$$\mathcal{K} = \begin{pmatrix} 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} & 0 & 0 \end{pmatrix}$$



(a) (b)

Figure 2.3: (a) Kneading matrix associated to the ADC-based chaotic map; and (b) associated Markov chain.

$d^{(i,1)}, d^{(i,0)}$	partition interval	state	macro-state
00	X_0	x_0	$\overline{x}_0$
01	X_1 or X_2	x_1 or x_2	$\overline{x}_1$
11	X_3	x_3	$\overline{x}_0$

Table 2.1: Markov interval for the ADC based map and corresponding associated states and macro-states.

is possible to *aggregate* the states of the graph two by two, as shown with the dotted lines of Figure 2.3b. If we introduce the two macro-states $\overline{x}_0$ and $\overline{x}_1$, respectively $\overline{x}_0$ corresponding to the system being either in x_0 or x_3, while $\overline{x}_1$ corresponding to the system being either in x_1 or x_2, the resulting diagram is identical to the ideal coin toss diagram.

Now, it is intuitive how a single 1.5 bits ADC stage can be used as a random bit generator; it is sufficient to directly close the output in a loop onto the input including a unity-delay block (that can be the S/H stage present inbetween every stage of the pipeline) to achieve the dynamic behavior

$$v^{(i)}\left(\left(k+1\right)T\right) = e\left(v^{(i)}\left(kT\right)\right)$$

that is the same behavior as (2.3), where the time steps $k, k+1, \ldots$ are substituted by the sampling instants $kT, \left(k+1\right)T, \ldots$. Also, in order to evaluate whether the system is in macro-state $\overline{x}_0$ or $\overline{x}_1$, it is enough to look at the digital outputs of the converter stage. In fact, the partition $\mathcal{X}$ is partially coincident with the quantization intervals of (2.1). For determining the macro-state, is it sufficient to take the exclusive-or between $d^{(i,1)}$ and $d^{(i,0)}$, as summarized by Table 2.1. The complete arrangement is illustrated in Figure 2.4.

Furthermore, a fundamental property of the ADC-based map is that it is a *robust* map, i.e. it is not affected by any problem discussed in Appendix A. Assuming that the map is linearly extended, i.e. that Equation (2.2) is extend

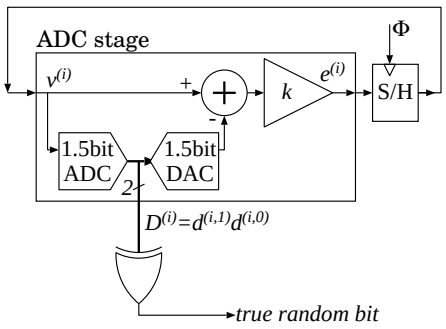


Figure 2.4: Complete arrangement for achieving a random bit generator from a 1,5 bit A/D stage.

$\forall x \in \mathbb{R}$, we have that

- it is not possible that the state could escape from the invariant set X, since the basin of attraction $B = [-2, 2]$ is sensible larger then $X = [-1, 1]$;
- no invariant sets other than the principal one exist or can arise due to map parameter variations;
- the map has an uniform invariant density, and the restriction of the map in $Y = [-3/2, 3/2]$ is periodic with $\Pi = 1$, Since $\mu(X) = 2\Pi$, the invariant density of the map is not affected by any noise bounded in $N = [-1/2, 1/2]$.

Hence, this map is an ideal candidate for a *practical* implementation of a chaotic source, since a good chaotic behavior is ensured of the circuit even in presence of non idealities. In addition, this is a very simple map, presenting a constant slope and only two breakpoints. Among all possible PWAM maps, only the Bernuolli map presents a simpler design, but it is well known not to be robust. Furthermore, designing a chaotic map based on already existing hardware, allows the implementation of a simple, reliable, chaotic map just simply reusing IP design blocks, or by transferring all the know-how from ADC technology ubiquitously used in mixed signal systems.

2.3 Description of Basic 1.5 bit Cell

For the implementation of the 1.5 bit A/D cell the classical switched capacitor implementation shown in Figure 2.5 has been adopted. While a single-ended configuration is shown for simplicity, the actual implementation is fully-differential. This stage operates on a two-phase clock. In a first phase (at the time step n, named "sample" phase), the input signal $v_n^{(i)}$ ranging in $X =$

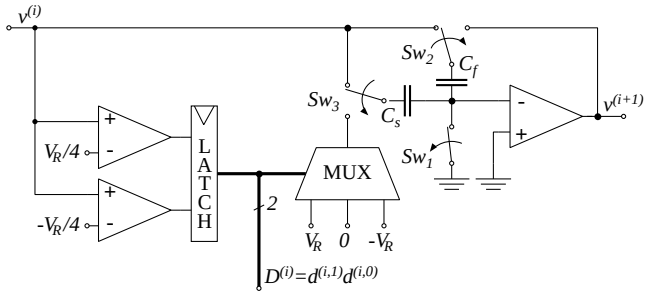


Figure 2.5: Standard 1.5 bits A/D switched capacitor converter stage used for the circuit implementation.

$[-V_R/2, V_R/2]$, is applied both to the coarse 1.5 bit ADC (a simple flash converter made of two comparators with thresholds $-V_R/4$ and $V_R/4$) and to the sampling capacitors C_s and C_f. The output of the ADC $D_n^{(i)} = d_n^{(i,1)} d_n^{(i,0)}$ is also latched at the end of the clock phase, while the analog error output $v_n^{(i+1)}$ is not significant.

During the second phase ("evaluating" phase, time step $n + 1/2$), C_f closes the negative feedback loop around the op-amp while C_s is switched to the output of the DAC (a simple three-inputs multiplexer). Due to Sw_1 that opens at the beginning of this time phase, the node connecting the switch Sw_1, the two capacitors and the inverting input of the operational amplifier (that is supposed ideal in this brief analysis) is now an isolated node; this means that the charge stored at this node (that is the total charge stored by the two capacitors) remains constant during this phase. The total charge at the beginning at the phase is

$$Q_s + Q_f = C_s V_s + C_f V_f = (C_s + C_f)\, v_n^{(i)} \tag{2.4}$$

while, due to the feedback, at the end of the transient it is:

$$Q_s + Q_f = C_s v_{n+\frac{1}{2}}^{(\mathrm{mux})} + C_f v_{n+\frac{1}{2}}^{(i+1)} \tag{2.5}$$

where $v^{(\mathrm{mux})}$ is the output voltage of the multiplexer. Imposing the conservation of the charge between (2.4) and (2.5), it is possible to write down the equation of the system:

$$v_{n+\frac{1}{2}}^{(i+1)} = \begin{cases} \left(1 + \dfrac{C_s}{C_f}\right) v_n^{(i)} - V_R, & v_n^{(i)} < -\dfrac{V_R}{4} \\ \left(1 + \dfrac{C_s}{C_f}\right) v_n^{(i)}, & -\dfrac{V_R}{4} < v_n^{(i)} < \dfrac{V_R}{4} \\ \left(1 + \dfrac{C_s}{C_f}\right) v_n^{(i)} + V_R, & v_n^{(i)} > \dfrac{V_R}{4} \end{cases} \tag{2.6}$$

Setting $C_s = C_f$ the resulting input/output characteristic is the desired one of Figure 2.2b, with the definition set equal to $X = [-V_R/2, V_R/2]$.

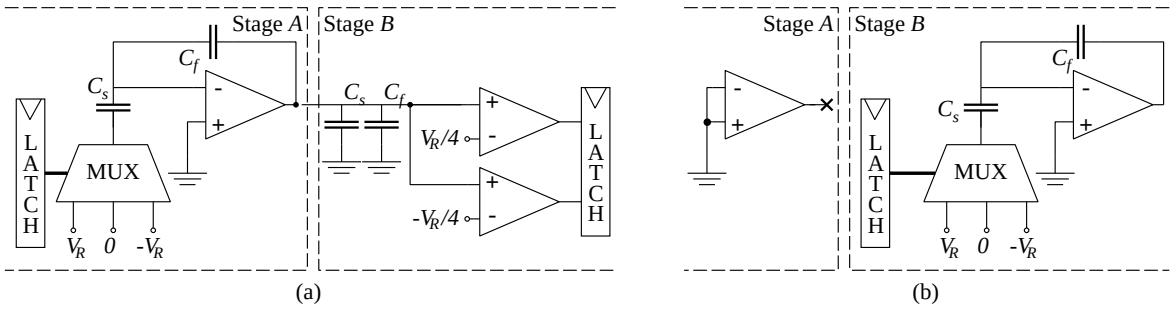


Figure 2.6: (a) In the first half time step, stage *A* provides a valid output and stage *B* is sampling it; while (b) during the second half time step the output of the first stage is no more valid, but is disconnect from stage *B*.

A first advantage of this structure is represented by the accuracy of this circuit, that is expected to be very high. As can be noticed from (2.6) the quality of the circuit relies only on the ratio of two, equal, capacitors and on the ratio of the reference voltages $\pm V_R$, $\pm V_R/4$. However, the values of the capacity of C_s and C_f, as well as the value of the reference voltage V_R, are not important; a change in the value of the capacitor affects only the transient time, while a change in the value of V_R implies only a scaling of the definition set X. Applying matching techniques in the design of the capacitors and of the voltage sources allows to get a very high accuracy.

Also it is possible to notice that this circuit introduces a delay equal to half time step. The input is sampled at (the end of) time step n, while the output is available at (the end of) time step $n + 1/2$. This is particular important, since it allows to directly connect the input of a stages to the output of the previous one without the interposition of any S/H. In fact, if the two stages work on the two different phases of the clock, when the first stage is computing the output the second one is sampling it; as soon as the the first stage goes into the sampling phase and its output is no more valid, the second one goes into the evaluating phase, in which the input has already been sampled and it is no more used. So, it is possible to avoid S/H stages in the pipeline by driving *alternatively* the cells in the pipeline with two opposite clocks. This arrangement is illustrated in Figure 2.6.

2.4 ADC-based Random Number Generator

Instead of being based on the basic model of Figure 2.4, the implemented RNGs are based on the schematic of Figure 2.7 [39]. The main reason is to take advantages of the half time step delay present in the basic cell, thus avoiding the

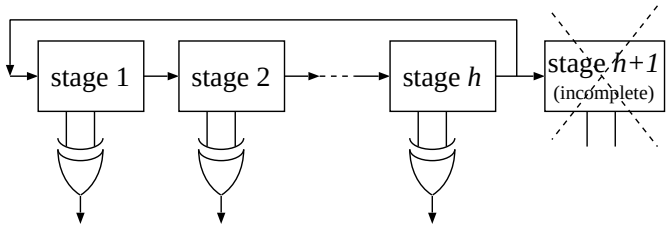


Figure 2.7: Schematic used for the implementation of the ADC-based random number generators.

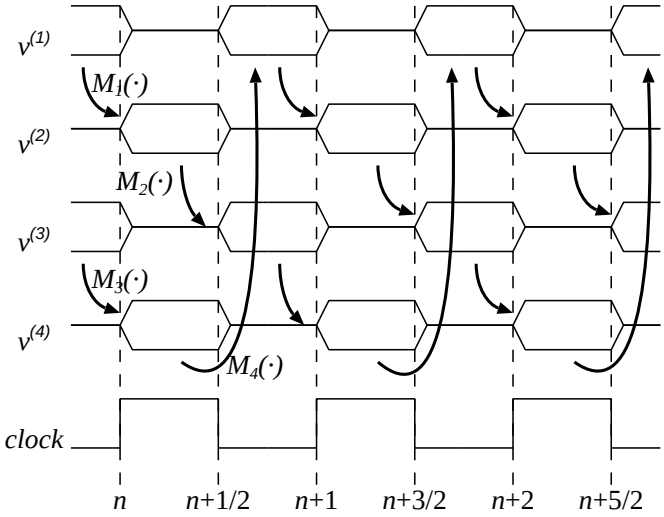


Figure 2.8: Example of evolution of the system of Figure 2.7 for $h = 4$.

S/Hs. The system is a closed pipeline composed of h identical stages, driven alternatively by the two phases of the clock, and can be described by the following model:

$$\begin{cases} v_{n+\frac{1}{2}}^{(1)} = 0 \\ v_{n+\frac{1}{2}}^{(2)} = M_1\left(v_n^{(1)}\right) \\ \cdots \\ v_{n+\frac{1}{2}}^{(h-1)} = 0 \\ v_{n+\frac{1}{2}}^{(h)} = M_{h-1}\left(v_n^{(h-1)}\right) \end{cases} \qquad \begin{cases} v_{n+1}^{(1)} = M_h\left(v_{n+\frac{1}{2}}^{(h)}\right) \\ v_{n+1}^{(2)} = 0 \\ \cdots \\ v_{n+1}^{(h-1)} = M_{h-2}\left(v_{n+\frac{1}{2}}^{(h-2)}\right) \\ v_{n+1}^{(h)} = 0 \end{cases}$$

This is discrete-time, h-dimension autonomous system, with a h dimensional state space. Despite its apparent complexity, it is very simple to analyze. An example of the system evolution, assuming $h = 4$, is depicted in Figure 2.8. The system is *interleaved*, in the sense that every of the h state of the system depends only by one state at the previous half-time step. Of course the number of stages after closing the pipeline has to be an *even* number.

Furthermore, it is very easy to understand that if $M_1 \equiv M_2 \equiv \ldots \equiv M_h \equiv M$ (i.e. the h stages of the A/D are identical) the behavior of the system is the same as h 1-D systems working in parallel. The only difference is that in the interleaved system the data stream is continuously shifted between all different h stages. Practically, the system is equivalent to h basic system of Figure 2.4, half providing output in the first phase of the clock, half in the other phase. Obviously the throughput of this system is equal to h bits per time step.

2.5 Design of the Basic Cell

We look now for some relationships between the operational amplifier's characteristics and the basic cell circuit (Figure 2.5) performances. In the following brief analysis all components (comparators, capacitors, switches) are considered ideal, and the operational amplifier is modeled as an ideal amplifier with a limited bandwidth, i.e. it is described by the low-pass first order transfer function:

$$A(j\omega) = \frac{A_0}{1 - j\frac{\omega}{\omega_c}}$$

where the open loop base-band gain A_0 is very large, and ω_c is the open loop bandwidth; we call GBW its gain-bandwidth product $GBW = A_0 \cdot \omega_c$.

The operational amplifier has two completely different configuration in the two phases. They are studied separately.

EVALUATING PHASE. In this phase (Figure 2.6a) C_s is connected to the multiplexer and C_f closes the feedback loop of the operational amplifier. They are both precharged at the output voltage $v_{n-1/2}^{(i-1)}$ at the previous stage in the (half) previous time step; the output changes to $v_n^{(i)}$ after a brief transient. Due to the simplified model, this transient has to be the same as any transient of the circuit in Figure 2.9a in response of an ideal step at its input:

$$v_{out}(t) = V_0 + (V_\infty - V_0)\left(1 - e^{-\frac{t}{\tau}}\right) \tag{2.7}$$

where V_0 and V_∞ are, respectively, the output voltage at the beginning and at the end of the transient (i.e. at time respectively, $t = 0$ and $t \to \infty$), and the time constant τ is the inverse of the bandwidth of the system. The step is supposed starting at time $t = 0$.

What is interesting in this analysis is the *relative error* $\varepsilon(t)$ in the response which is:

$$\varepsilon(t) = \frac{V_\infty - v_{out}(t)}{V_\infty - V_0} = e^{-\frac{t}{\tau}}$$

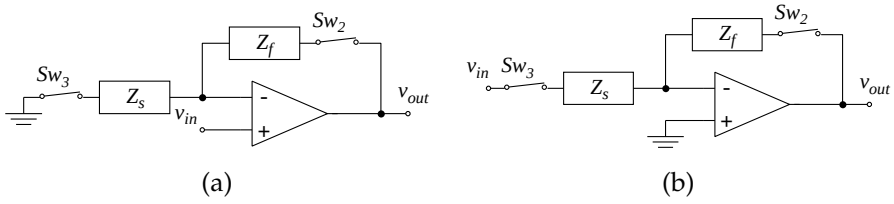


Figure 2.9: Equivalent circuits for transients of the basic cell during the evaluating phase.

Supposing to have a limited time T available for the transient, if the relative error has to bo smaller than $\overline{\varepsilon}$ at the end of the available time, the relation needed is:

$$\tau < -\frac{T}{\ln \overline{\varepsilon}}$$

For estimating τ it is enough notice that the circuit is a standard non inverting amplifier, whose frequency domain behavior is

$$v_{out} = A(j\omega)\left(V^+ - V^-\right) = A(j\omega)\left(v_{in} - v_{out}\frac{Z_s}{Z_s + Z_f}\right) \tag{2.8}$$

with V^- and V^+ are the voltage at the inverting and non-inverting input of the operational amplifier, and setting $\beta = \dfrac{Z_s}{Z_s + Z_f}$

$$V_{out} = \frac{A(j\omega)}{1 + \beta A(j\omega)} V_{in}$$

This comes in the standard form used in feedback system analysis; supposing $A_0 >> 1$, *(a)* the gain of the system is $1/\beta$; and *(b)* the gain-bandwidth product is constant, i.e. the bandwidth B of the system in closed loop in $B = \beta \cdot GBW$. In this case, $Z_s = 1/j\omega C_s$, $Z_f = 1/j\omega C_f$, so $\beta = 1/2$; the time constant in the transient is $\tau = 2/GBW$.

Actually one could argue that the same circuit can be seen as a unity gain inverting amplifier as in Figure 2.9b where, apparently, the gain is halved and the bandwidth doubled with respect to the previous configuration. However, a detailed analysis of the circuit shows that

$$v_{out} = A(j\omega)\left(V^+ - V^-\right) = -A(j\omega)\frac{1}{Z_s + Z_f}\left(Z_f v_{in} + Z_s v_{out}\right)$$

and, this time

$$v_{out} = -\frac{A(j\omega)}{1 + \beta A(j\omega)} v_{in}(1 - \beta)$$

So, from the feedback analysis point of view, the signal $v_{in}(1 - \beta) = v_{in}/2$ is amplified by a factor 2 with a time constant $\tau = 2/GBW$. The time response of the system is the same as in the previous case.

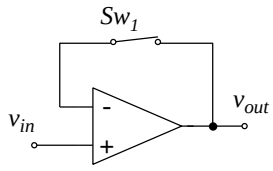


Figure 2.10: Equivalent circuits for transient of the basic cell during the sample phase.

Given this relation between τ and the operational amplifier GBW, the constraint is

$$GBW > -\frac{2\ln\overline{\varepsilon}}{T} \tag{2.9}$$

SAMPLE PHASE. Now (Figure 2.6b) the inverting input of the amplifier is grounded through Sw_1 (i.e. all the charge present at the node at the previous time step is removed through the switch) while the feedback loop through Sw_2 is open. Actually in order to *(a)* speed-up the process; and *(b)* avoid the open loop in the operational amplifier, an alternative schematic has been implemented, i.e. Sw_1 has not been connected between the inverting input of the operational amplifier and ground but between the inverting input and the output. The operational amplifiers works as in Figure 2.10 in a buffer configuration (unity gain configuration). Since the non-inverting input is grounded, and due to the negative feedback loop, also the output (and so the inverting input) will, after a brief transient, reach zero voltage; however in this case all the residual charge is removed actively through the switch by the amplifier. The time of the transient is not dependent on the R_{ON} of the switch (which is, in the closed feedback loop model, divided by the open loop gain A_0 of the amplifier) but only on the bandwidth of the amplifier. In this phase the behavior of the system is the same as the response to an ideal step of the circuit in Figure 2.10.

Following the same procedure as in the previous step, $\beta = 1$, so the time constant in the discharge is $\tau = 1/GBW$ and is smaller than in the previous case. This means that the critical case is the evaluating phase.

SLEW-RATE. Another aspect to take into account in the design of the circuit is the limited slew rate of the operational amplifier. The main reason why the slew-rate mode has to be avoided is that, during the slew rate, the behavior of the transient can be completely different from the first order transient; for example it could happen that many transistors in the operational amplifier are turned off, and turning them on again requires a certain amount of time. During this time the output is still rising, and could reach an unwanted overshoot, as illustrated in Figure 2.11. As in the figure, it is possible that another system

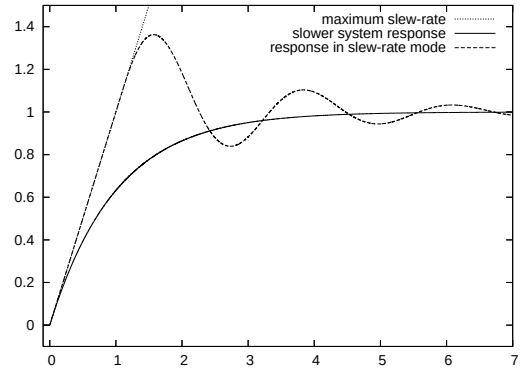


Figure 2.11: Considering two systems with the same slew-rate (dotted line), the slower system that does not enter in slew-rate mode (solid line) could result in a smaller settling time with respect to a faster system (dashed line) that enters in slew-rate mode.

with the same slew-rate, but that is slower in the response (that mean a smaller bandwidth), does not enter into the slew-rate mode thus resulting in a shorter settling time.

If ΔV is the output voltage step $V_\infty - V_0$ during the transient, the maximum variation in the output voltage can be obtained deriving (2.8)

$$\left|\frac{dV_{out}}{dt}\right| = \frac{|\Delta V|}{\tau} e^{-\frac{t}{\tau}}$$

and is maximum for $t = 0$

$$\max\left|\frac{dV_{out}}{dt}\right| = \frac{|\Delta V|}{\tau}$$

This quantity has to be smaller than the maximum slew-rate $S.R.$ allowed by the operational amplifier, and, referring to the evaluating phase, this results in the constraint:

$$GBW < \frac{2S.R.}{|\Delta V|} \tag{2.10}$$

Actually, since it has been derived by a worst-case analysis and with a very simplified model, (2.10) is too stringent. For example, it is false that the slew rate is to be avoided; what should be avoided is that no transistors in the amplifiers are turned off. This is of course a more relaxed constraint than requiring to avoid the slew rate. Also, the response of the system is actually not exactly as in (2.7) especially at the beginning of the transient, due to the non-ideality of the switches. For these reasons, (2.10) is substituted by

$$GBW < \alpha\frac{2S.R.}{|\Delta V|} \tag{2.11}$$

where α is a constant that is empirically computed with simulations to be equal to $\alpha = 2$.

Note that the two constraints we have found in this paragraph come from a very simplified model of the system, and must be paired with the support given by simulations, to lead to an optimized design.

2.6 Description of the 0.35 μm RNG prototype

The first prototype has been designed in 0.35 μm C35B3C1 technology; this technology, provided by AustriaMicroSystem AG, is a n-well CMOS technology with a minimum MOS width of 0.35 μm and a minimum resolution of 0.05 μm. The technology also provides a double polysilicon layer (with a poly-poly capacitor module and a high resistive polysilicon module) and three level of metalization. This technology requires a power supply voltage of 3.3 V.

Instead of the single-ended configuration showed up to now, a fully differential implementation has been chosen. This means that every signal is not simply the voltage across a wire, but it is represented by the difference between the voltages of two lines. More precisely, given a reference level V_{ref}, the signal $v^{(i)}$ is represented by the two voltages $V_{\mathrm{ref}} + v^{(i)}/2$ and $V_{\mathrm{ref}} - v^{(i)}/2$. Note the the differential voltage swing is double with respect to the voltage swing of a single line. Yet, the increment in the complexity of the circuit (every signal has to be routed as two different interconnection lines, and every component has to be doubled, including C_s and C_s) is balanced by the more robust circuit in terms of noise and perturbations tolerance.

The reference voltage has been set to $V_{\mathrm{ref}} = 1.2$ V and $V_R = 500$ mV; thus a biasing stage is needed to generate the five voltage levels V_{ref}, $V_{\mathrm{ref}} \pm V_R/2$ and $V_{\mathrm{ref}} \pm 2V_R$. From these voltages it is possible to generate the five differential voltages 0, $\pm V_R$ and $\pm 4V_R$ needed by the converter stage. Notice that in this sense, the complexity of the biasing stage is not increased with the introduction of the fully differential architecture. These voltages have been generated with a matched resistive ladder, biased with a constant current.

The nominal speed for this circuit has been set in $f_{\mathrm{nom}} = 5$ MHz, i.e. with a settling time for each transient phase equal to $T = 100$ ns. Imposing a relative error of $\varepsilon = 0.001$ in (2.9):

$$GBW > 22MHz$$

However handling (2.11) requires a more detailed circuit description. The capacitors C_s and C_f are implemented through a matched array of 4×4 smaller capacitors; they are parallely connected in group of four, reaching a value of $C = 2$ pF. A microphotograph showing in detail of the capacitor array can be

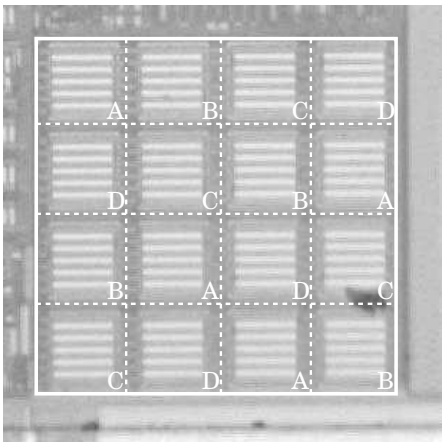


Figure 2.12: Detailed microphotograph of the 4×4 array of capacitors used for C_s and C_f.

Load capacitance *(considered)*:	5 pF
Compensation capacitance:	3 pF
Gain-Bandwith product:	30 MHz
Phase margin:	80°
Differential gain:	43 dB
Common mode gain:	-17 dB
Power consumption:	1.6 mW

Table 2.2: Electrical characteristic for the operational amplifier designed for the 0.35 μm circuit.

seen in Figure 2.12; the identification of the connected capacitors can be done looking at the letters A–D in the corner of each array cell. In the evaluating phase (Figure 2.6a) the load capacitance of the operational amplifiers is estimated in $C_L = 8$ pF, considering also the compensation capacitance; since the two final stages of the operational amplifiers are biased with a current $I = 200$ μA each, the maximum slew rate to each output node is $S.R. = I/C_L = 25$ V/μs. With this value, and considering that the maximum $|\Delta V|$ (for a single output line) is $\Delta V = V_R$

$$GBW < 32MHz$$

In the designed operational amplifier the GBW has been limited to 30 MHz. The operational amplifier characteristics are reported in Table 2.2.

This prototype was designed including two pipelines. They are identical in everything but the number of stages: the first one is composed by two stages and includes two analog buffers for providing the internal state of the stages, and it is intended for testing the correct behavior of the chaotic map. The sec-

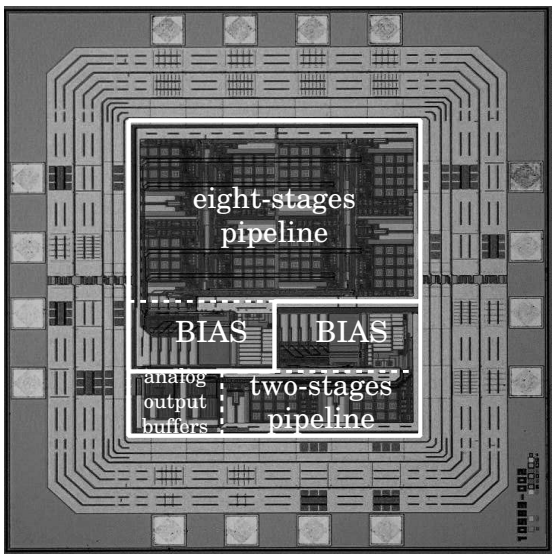


Figure 2.13: Microphotograph of the designed 0.35 μm prototype of the ADC-based RNG.

ond pipeline is composed by eight stages and its purpose is to work as a random bit generator. The two parts use two different biasing circuits to avoid interferences. The circuit works with an external nominal clock of frequency $f_{in} = 10$ MHz, that is halved, thus the internal maps work at a frequency of 5 MHz and a settling time $T = 100$ ns. This means that the circuit nominal output data rate is 40 Mbit/s for the eight-stages pipeline and 10 Mbit/s for the two stages pipeline.

However, the digital outputs (as well as the analog one in the two stages pipeline) are rearranged to provide a simpler interface. Instead of having a number of output pins equal to the number of stages, only one output pin has been connected to two stages. This also allows to synchronize all the output signals with the input clock f_{in}.

A microphotograph of the integrated circuit is shown in Figure 2.13 while the circuit characteristics are reported in Table 2.3.

2.7 Macromodel for 0.35 μm RNG prototype

To validate the design, a netlist extracted from layout and affected by parameter variations reproducing fabrication imperfections must be simulated and results matched against test for randomness. Due to the switched capacitor nature of the circuit, time-domain simulations are necessary. These simulations are extremely expensive in terms of computing power. With a state-of-the-art

Nominal working frequency:	5 MHz
Nominal data throughput:	5 Mbit/s per stage
(two-stages pipeline):	10 Mbit/s
(eight-stages pipeline):	40 Mbit/s
Area (with pads):	2.400 mm^2
	(1480 μm x 1620 μm)
Area (without pads):	0.752 mm^2
(two-stages pipeline):	0.234 mm^2
(eight-stages pipeline):	0.518 mm^2
Power supply voltage:	3.3 V
Power consumption:	56 mW
(two-stages pipeline):	27 mW
(eight-stages pipeline):	29 mW

Table 2.3: Circuit characteristics of the designed 0.35 μm prototype of the ADC-based RNG.

CPU and a commercial *spectre* simulator, a speed of about 600bit/hour (i.e. about 0.15bit/s) for the two stages-pipeline circuit was obtained. This is of course unacceptable, since statistical tests require millions of bits to run. For this reason an efficient macro-model capable of a throughput of several order of magnitude higher than the full circuit simulation has been investigated. The macro-model has been developed from the circuit implementing a two-stages pipeline, aiming to describe the single stage and so to simulate a pipeline with any number of stages.

Since the circuit is, ideally, 1D discrete-time and time independent, a 1D discrete-time and time independent model has been selected, i.e. the focus has been posed on modeling the profile of the implemented M and describe how its varies depend on implementation inaccuracies. Due to the discrete-time nature of the circuit, the M function can be analyzed only with a collection of (x_{k+1}, x_k) points obtained from simulations. This could have been done with a parametric simulation of a single time step with different initial values of x_k; however there is no guarantees that the computed initial solution is the actual one. So it was preferred to extract a set of the points (x_{k+1}, x_k) from a single, long, transient simulation, and down-sample the output stream with a sampling instant few ns before the front of the clock as shown in Figure 2.14, where only the differential values of the circuit outputs are drawn for simplicity. Since the points x_k are ideally uniformly distributed in X, from these points a good representation of M could be obtained. This simulation has also been performed with different values of the actual process parameters to obtain a

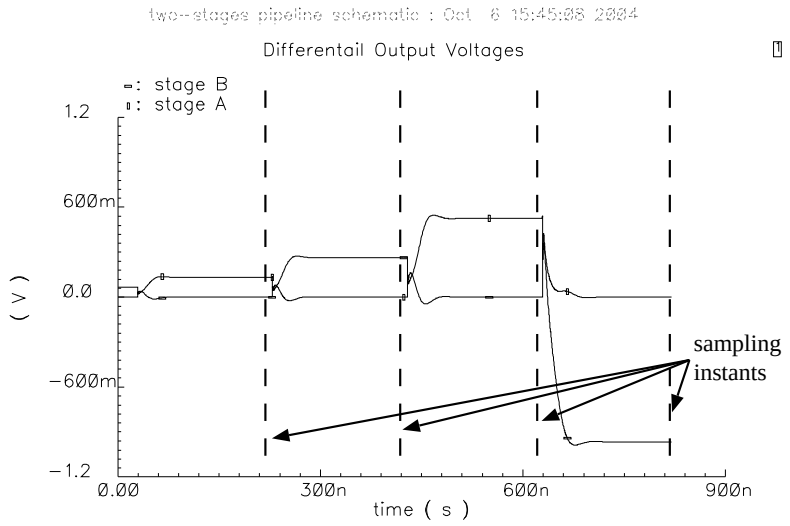


Figure 2.14: Short time transient analysis for the two-stages pipeline.

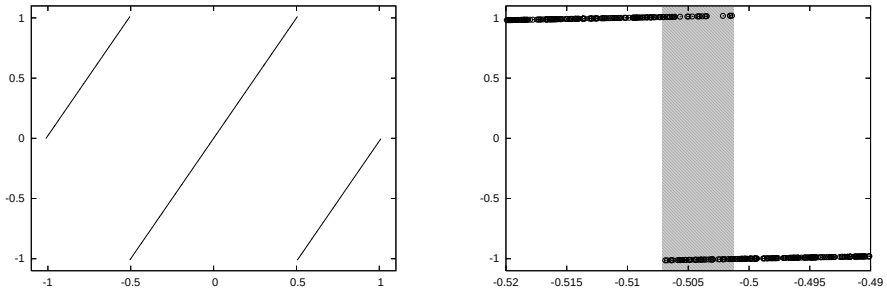
model which includes all implementation inaccuracies.

So, many Monte-Carlo runs of about 25×10^3 clock periods have been simulated for the two-stages pipeline circuit. From each of these runs, two sets of about 25×10^3 points (x_{k+1},x_k), one for every stage, have been extracted. From these sets, a version of function M is computed for every stage of every Monte-Carlo run; these functions have been analyzed to obtain a simple but realistic map description including an evaluation of the differences which may exist between two stages of two different pipelines or between two stages of the same pipeline.

The model used for the M function is a piece-wise linear model. The switched capacitor implementation ensures (Figure 2.15a) a very high linearity, and also a very good precision on the multiplying factor. Also the fully differential architecture ensures a high symmetry; so the M can be described by

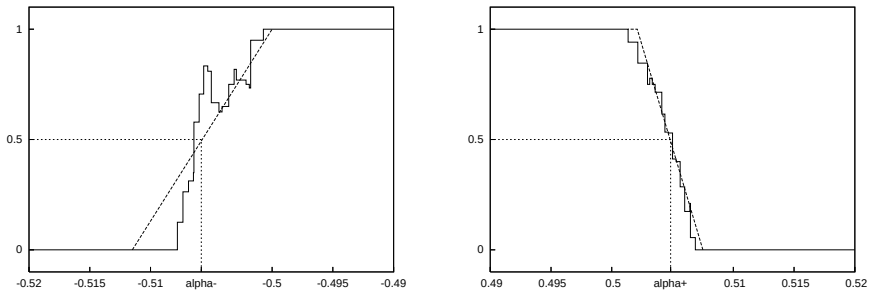
$$M(x) = \begin{cases} 2x + \beta & \text{if condition } \lambda_1(x) \text{ is true} \\ 2x & \text{if condition } \lambda_2(x) \text{ is true} \\ 2x - \beta & \text{if condition } \lambda_3(x) \text{ is true} \end{cases}$$

The determination of the three condition λ_1, λ_2 and λ_3 is non-trivial. Ideally, two breakpoints α^- and α^+ exist, with $\lambda_1(x) : x < \alpha^-$, $\lambda_2(x) : \alpha^- \leq x < \alpha^+$ and $\lambda_3(x) : x \geq \alpha^+$. Yet, the real behavior of the system can be seen in Figure 2.15b, which represents a zoom of Figure 2.15a around the ideal breakpoint



(a) (b)

Figure 2.15: (a) A collection of (x_{k+1},x_k) points for a single 1.5 bit ADC stage; and (b) zoom around breakpoint α^-

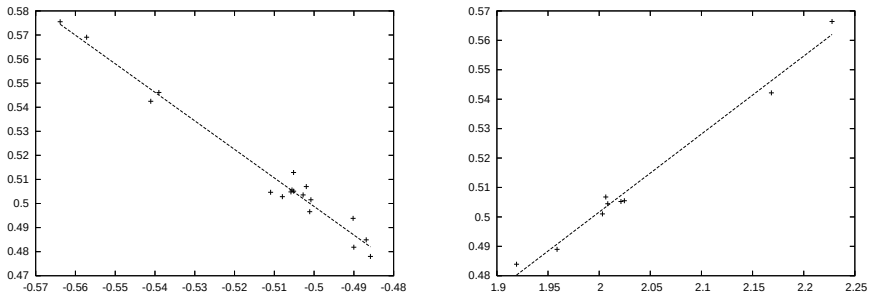


(a) (b)

Figure 2.16: (a) Density of points satisfying condition λ_2 (solid line) and linear approximation $p(x)$ (dotted line) around α^-; and (b) around α^+.

α^-. While at a certain distance from the breakpoint the behavior is fully deterministic, a point very close to the breakpoint could sometimes verify condition λ_1 and sometimes λ_2 (the gray area in the figure). This could be explained considering interferences (for examples spikes on the power supply voltage) coupling from the other parts of the circuit which may alter the behavior of the two comparators. Due to the static nature of the macro-model, these interferences cannot be modeled in any way but as noise perturbation. So a stochastic transition model has been implemented; in this model a probability function decides which linear piece of M is used.

The solid lines of Figure 2.16 show the density of points around the breakpoints verifying condition λ_2 in a Monte-Carlo run; the figure has been obtained with an histogram analysis. Assuming the system is ergodic, this function has been taken as the probability function $p(x)$ that condition λ_2 is verified for a point x. With this the macromodel have been set



(a) (b)

Figure 2.17: (a) Scatter plot for α^- vs α^+; and (b) for β vs α

$$M(x) = \begin{cases} 2x+\beta & x<0, \text{with probability } 1-p(x) \\ 2x & \text{with probability } p(x) \\ 2x-\beta & x>0, \text{with probability } 1-p(x) \end{cases}$$

Ideally $p(x) = \chi_{[\alpha^-,\alpha^+]}$, where χ is the classical indicator function of an interval. In this model, $p(x)$ has been considered a trapezoidal function (the dotted line in figure 2.16):

$$p(x) = \begin{cases} 0 & x < \alpha^- - \dfrac{1}{2s^-} \\ \dfrac{1}{2} + \left(x-\alpha^-\right)s^- & \alpha^- - \dfrac{1}{2s^-} \le x < \alpha^- + \dfrac{1}{2s^-} \\ 1 & \alpha^- + \dfrac{1}{2s^-} \le x < \alpha^- - \dfrac{1}{2s^+} \\ \dfrac{1}{2} - \left(x-\alpha^+\right)s^+ & \alpha^+ - \dfrac{1}{2s^+} \le x < \alpha^+ + \dfrac{1}{2s^+} \\ 0 & x \ge \alpha^+ + \dfrac{1}{2s^+} \end{cases}$$

The breakpoints α^- and α^+ are the points where the (fitted) probability to be in one or another of the two linear pieces of M is equal. These parameters, as well as s^- and s^+, are computed through two separate linear regression, including all points around α^- (or α^+ respectively) that verify condition λ_2 in each Monte-Carlo run. All points far enough from the breakpoints to ensure a deterministic decision (i.e. when the density is 1 or 0) have not been considered. Even if some slightly differences can be observed in the value of α^- and α^+, they are strongly related (see Figure2.17a) so we can assume $\alpha = \alpha^+ = -\alpha^-$.

Numerical analysis shows that there can be large differences in these parameters for different pipeline implementations. However the differences between different stages in a single pipeline are very small. This reflects the fact

parameter(s)	mean value	standard deviation
β	2.0376	0.09806
$\Delta\alpha$	0	0.004551
s^+,s^-	124.7316	60.177

Table 2.4: Expected value and deviation for model parameters.

that the latter differences depend only on inaccuracies such as matching errors, which are typically limited.

For example, a variation up to $\pm 20\%$ from its nominal value can be observed in β in different Monte-Carlo runs since it depends on reference voltages which may strongly vary; however no sensible variation can be observed in β for the two stages of a single Monte-Carlo run, since in a single simulation (i.e. in a single pipeline) the reference voltages are the sames. So β in the macro-model is assumed to be a global parameter for the entire pipeline. From the analysis of β in all Monte-Carlo runs, its value is modeled as a normal distributed random variable with mean value and variation shown in Table 2.4. On the contrary, a fluctuation on the value of α can be observed even between the two stages of a single pipeline. The value of α in the macro-model so is computed for every stage of a pipeline as $\alpha = \alpha^0 + \Delta\alpha$ where α^0 is a global parameter for the entire pipeline, and $\Delta\alpha$ is computed for every stage as in Table 2.4. Actually, α^0 is not an independent parameter, and it is strongly related to β (see Figure 2.17b). In the model it is taken $\alpha^0 = \beta/4$, since this is the expected relation between these two parameters. However, no relations between the values of s^-, s^+ and the other parameters has been found; also since no link with the other parameters can be anticipated based on circuit design, they are taken as independent for each pipeline stage.

In the light of this, the proposed model has four parameters:

- β, which is assumed as a global random variable for the entire pipeline.
- $\Delta\alpha$, s^- and s^+, which are random variables computed for each stage of the pipeline.

All these parameters have been assumed to be normal distributed random variables.

2.8 Design of the RNG circuit in 180 nm technology

A circuit implementing an ADC-based RNG was also designed in UMC 180 nm technology as a RNG used in a spread spectrum clock generator see Chapter

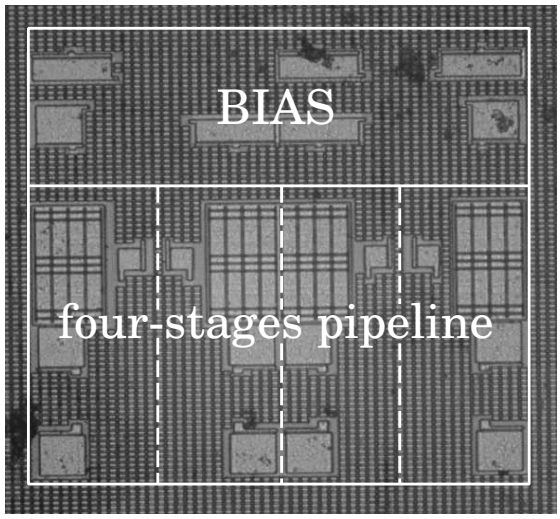


Figure 2.18: Layout of the designed ADC-based RNG.

6, Section 6.3. As the previous technology, this is a standard n-well CMOS technology, but with it is optimized for digital applications. It has a single polysilicon layer, up to six metal layers, and a metal/metal capacitor options between the two top metal layers. The core power supply voltage is reduced to 1.8 volts, but 3.3 volts transistors are available as I/O devices.

This prototype includes a four-stages pipeline, and it is the porting of the previous design into the new technology. Many attentions have been given to the circuit speed: its purpose was to give a throughput of about 47 Mbit/s, thus working with a settling time equal to $T = 42.5$ ns for each phase; however it was designed to reach much higher speeds.

The operational amplifier implemented has a differential GBW of about 200 MHz, and a phase margin of 67°. Its power consumption was estimated in about 2.7 mW. Due to the reduction in the power supply voltage, all reference voltage have been scaled, i.e. $V_{\text{ref}} = 800$ mV and $V_R = 150$ mV. The value of C_s and C_f was chosen in 500 fF.

The microphotograph of the area of the RNG is shown in Figure 2.18 while a summary of the performance can be found in Table 2.5. Note that in this circuit the output stage is slightly different from the first prototype. The input frequency f_{in} is divided by two before reaching the pipeline as in the 0.35 μm circuit; however, to keep a low bitrate at the output pins, thus simplifying the data acquisition, an output pin has been assigned to each stage. This is reflected into the fact that the outputs are no more synchronized with the input clock, but they change every two rising fronts of the input clock.

Nominal working frequency:	12 MHz
(throughput):	48 Mbit/s
Maximum working frequency:	25 MHz
(throughput):	100 Mbit/s
Active area:	0.126 mm^2
	(350 μm x 360 μm)
Power supply voltage:	1.8 V
Power consumption:	22 mW

Table 2.5: Circuit characteristic for the UMC 180 nm porting of the ADC-based RNG.

2.9 Conclusion

In this chapter the architecture of common pipeline ADCs has been analyzed and reused for designing a robust chaotic circuit, thus implementing a chaos-based random number generator. Two prototypes of a RNG has been presented; the first one is designed in 0.35 μm CMOS technology to operate at a speed up to 40 Mbit/s, while the second one is designed in 180 nm CMOS technology and can operates at a much higher speed.

Chapter 3

How to Improve the Quality of a RNG

WHEN CONSIDERING a real implemented true random number generator it is always necessary to deal with implementation errors, parasitic components, or simply with the non-ideality of the used devices. As an example, theoretically the noise generated by the thermal agitation of the electrons inside an electrical conductor in equilibrium (often referred as *Johnson noise*, or *Nyquist noise*) is approximately white and Gaussian distributed, with a constant power spectral density (i.e. voltage variance) per hertz given by

$$\overline{v}_n^2 = 4k_B T R$$

where k_B is the Boltzmann's constant, T is the resistor's absolute temperature, and R is the resistor value. Such a white (and so infinite-bandwidth) noise signal is purely a theoretical construction. By having power at all frequencies, the total power of such a signal would be infinite. Practically it is observed that the white noise assumption is a very good approximation until few Gigahertz at room temperature.

In the time domain, a white Gaussian noise has a Gaussian amplitude distribution; furthermore, labeling it as "white" describes that the noise is also uncorrelated. This theoretically means that is enough to sample at the desired frequency such a noise to have uncorrelated samples that, being the process Gaussian, also independent.

However, since the white noise model is just an approximation, it is necessary to set an upper bound in the sampling frequency to ensure that the correlation between samples is limited. This problem is extremely relevant when

we consider that we are trying to *measure* this noise, i.e. we connect the noise process to an harvester.

The observed power spectral density is now far from being white, depending on the bandwidth of the measure circuit. For this reason, samples at the harvester output can present a strong correlation, thus being no more independent. Similar phenomena happen in all physical based RNGs.

For this reason, all physical RNGs are completed with some post-processing functions (also called *entropy distillation processes*) to improve the quality of the output stream. The use of a distillation process is needed to overcome the production of an auto-correlated sequence of numbers (e.g., the occurrence of long strings of zeros or ones in a bit stream) [23, 36, 82].

Correlation is defined as a coefficient that indicates the strength and direction of a linear relationship between two random variables. This coefficient correlation cannot exceed 1 in absolute value. Mathematically the correlation $\rho_{X,Y}$ between two random variables X and Y is defined as:

$$\rho_{X,Y} = \frac{E\left[\left(X - E\left[X\right]\right)\left(Y - E\left[Y\right]\right)\right]}{\sqrt{E\left[X^2\right] - E^2\left[X\right]}\sqrt{E\left[Y^2\right] - E^2\left[Y\right]}}$$

where E is the expected value.

In a random sequence it is more appropriate to talk in terms of autocorrelation, which is a function describing the correlation between the process at different points in time:

$$R\left(t,s\right) = \frac{E\left[\left(X_t - \mu\right)\left(X_s - \mu\right)\right]}{\sigma^2}$$

with μ and σ the mean and the variance of the random process. The correlation is 1 in the case of an direct linear relationship, -1 in the case of a inverse linear relationship, and some value in between in all other cases, indicating the degree of linear dependence between the variables. The closer the coefficient is to either -1 or 1, the stronger the correlation between the variables.

If the variables are independent then the correlation is 0, but in general the converse is not true because the correlation coefficient detects only linear dependencies between two variables. Even if it is common to speak in terms of correlation instead of independence because of its simple definition and of the possibility to estimate the autocorrelation function in a sequence, this tool cannot be used to measure the "randomness" of the sequence.

Introducing a mathematical tool able to measure how "good" or how "bad" a random sequence is, as well as to explain the concept of "strong" or "weak" post-processing, is a necessary step before any further talk.

3.1 Information Theoretic Entropy

The classical concept of entropy of a system comes from thermodynamic theory and can be described qualitatively as a measure of disorder of a system; with the development of statistical thermodynamics and quantum theory, entropy changes have been described in terms of the mixing or "spreading" of the total energy of each constituent of a system over its particular quantized energy levels.

The concept of entropy in information theory was introduced by Claude E. Shannon in 1948, to describe how much information there is in a signal or event [21]. In case of a random number generator, the "information" associated to a generated sequence of symbols can be interpreted as the quantity of *independent* symbols in the sequence, exactly like in a communication system where all redundant data (for example, all the parity bits in a string) contain no additional information, since they are deterministically computed from all other data. Even if this is out of the scope of this dissertation, it is possible to prove that there are close parallels between the thermodynamic entropy of a physical system in the statistical thermodynamics established by Boltzmann and Gibbs in the 1870s and the information-theoretic entropy of Shannon developed in the 1940s.

The Shannon entropy (measured in bits) of a discrete random process is defined in terms of the discrete random variable X, with possible states (or outcomes) $x_0, \ldots, x_{n-1}$ as:

$$H(X) = -\sum_{i=0}^{k-1} \pi(x_i) \log_2 \pi(x_i) \tag{3.1}$$

where $\pi(x_i)$ is the probability of the i^{th} outcome $X = x_i$.

Actually, when considering a RNG, the random variable X is the n bits generated sequence. The possible outcomes are $k = 2^n$, each (ideally) with the same probability $p_i = 2^{-n}$. In this way, the entropy associated to the perfect random bit generator is measured in n bits, and this is the maximum possible value. Since this measure is dependent on the output sequence length, it is more practical to consider the *entropy per bit* (or normalized entropy, usually indicated as h), that is $h = H(X)/n$. The perfect RNG have an entropy per bit equal to $h = 1$ bit; for any other generator h is a real number between 0 and 1.

Even if, from a theoretical point of view, the entropy is the only tool capable of fully characterize a RNG, it does not find any practically usage in the RNG testing due to the impossibility to have a realistic estimation. The reason is very simple. An estimation of entropy can be obtained very easily; many

algorithms exist, the simplest one is just to take (3.1) and compute the entropy by substituting the probability $\pi(x_i)$ of a symbol with its observed frequency $\nu(x_i)$ in the outcomes of the generator during a period of time. One can refer to the n order entropy H_n as the entropy of the generator when considered as a system generating strings of n bits

$$H_n = \lim_{k\to\infty} -\sum_{i=0}^{k-1} \nu(x_i) \log_2 \nu(x_i)$$

However this is not a correct estimator of the system entropy. The problem is that, when substituting the probability $\pi(x_i)$ with the observed frequency $\nu(x_i)$, it was implicitly supposed that *(a)* the system is ergodic; and *(b)* the symbols generated are independent. None of these two assumptions are ensured. This estimation is correct only considering very large n; in this way dependencies among bits are reflected into non-uniform distribution of sequences:

$$h = \lim_{n\to\infty} \frac{H_n}{n}$$

Of course this estimation is not feasible in real systems, since any approximation would require extremely long sequences. In practical cases, in order to test randomness some statistical tests are used; this will be discussed later in Chapter 4. In this chapter the entropy is taken into account only from a theoretical point of view, and it is considered the main figure of merit of a RNG. More precisely, the figure of merit considered is entropy per time unit (sometimes called the *entropy rate*) that has the physical meaning of the quantity of independent bits that a RNG could produce per time unit. This normalization is necessary since it is always possible to increase the entropy per bit of a generator by applying a post-processing stage, it is not possible in any way to increase the entropy per time unit.

3.2 Increasing the Entropy of a Generator

Suppose to have a non-perfect RNG, that in a given time produces N bits, of which only M are discovered to be independent (i.e. the RNG has an entropy per bit equal to $h = M/N$). Now suppose to have a numerical algorithm capable of separate these independent M bits from the stream; this algorithm would be the perfect post-processing stage. The combination non-perfect RNG + perfect post-processing stage works as a perfect RNG; in this sense the quality of the generator has increased. What is not changed applying the post-processing is the entropy per time unit: in the given time, both generators produce M independent bits. Of course, there is the constraint $M < N$, thus meaning that

data at the output data rate of the post-processing is lower than the input one; such an algorithm is called *compressive*. The process of reducing the number of bits in the stream is called *decimation*.

Understanding this concept is fundamental when applying a post-processing to a random stream. At this point, it may help summarizing what a post-processing can and what it cannot do.

- A post-processing can increase the entropy per bit of a RNG, introducing a proper decimation. In this way, a post-processing function is a numerical algorithm used, given a set of values, to "distillate" the independent values from the dependent ones, thus creating a perfect random stream.

- A post processing cannot increase the entropy per time unit of a RNG. All post-processing functions that try to increase the bit-rate of a generator, or simply that does not introduce decimation, do not increase the quality of a random stream in the sense of increasing the entropy per bit. Yet, they exist and may produce good results according to statistical tests; however they work in the same way as a pseudorandom generator that is continuously seeded by a true random generator. In fact, we can remark that the entropy of a pseudorandomly generated stream is just the entropy of the seed.

Of course the perfect post-processing exists only theoretically; in real cases, finding a good post-processing function that increases the entropy per bit and that does not reduce entropy per time unit is an hard task. Note that a simple decimation is not an effective way to increase the quality of a random stream; decimating a string by a factor n could reduce also the entropy per time in the output stream, while the effect over the entropy per bit is uncertain, since if some relationship among bits more distant then n exists, this process is uneffective.

Usually post-processing functions consider a long sequence, and "spread" autocorrelation and the other order moments among all the sequence. In a correlated bit stream the correlation coefficient of two bits next to each other is usually much greater than the correlation coefficient of two very far bits, thus meaning that the relationship between two bits next to each other is much stronger than the relationship between two very far bits. In the same way it is usually simpler for any (human or not) observer notice some dependence between two close bits than between two far bits. A post-processing that only reshapes the autocorrelation function in order to get a more flat profile could get better results to statistical tests than a simple decimation function. In this

way, the stream appears much more random, or, more precisely, it is more difficult to find a relation among bits.

Yet, in order to speak in terms of "more random" and "less random", as well as of "good" and "bad" random sequence, one should only consider the entropy: the higher the entropy, the better the sequence. In case of a post-processing working only on the autocorrelation reshaping, one should not speak in terms of "increasing" of the randomness; however ironically this kind of post-processing usually gives sequences that appear much more random to statistical tests, than sequences given by other post-processing that works only on decimation, trying to effectively increment the entropy.

In the following some post-processing functions used in the testing phase are presented. Actually all post-processing functions can be divided into two main categories depending on their strategy

- algebraic: they take an input block of N bits and elaborate it, giving an output block of $M \leq N$ without any memory of the previous blocks.
- dynamic: they are essentially based upon a digital filter (i.e. a FIR o IIR), and every bit is processed depending on the previous history of the input (or output) sequence.

3.3 Von Neumann Post-processing

This algorithm was introduced by John von Neumann in 1951 to remove biasing in random sequences [23]. The technique divides the input sequence into non-overlapping groups of two bits, and converts the bit pair 01 into the output 0, the bit pair 10 into the output 1, and discards bit pairs 00 and 11. The working principle is very easy. If the bits in the sequence are independent, but not with the same probability , i.e. the probability of "1" is p and the probability of "0" is q, with $p \neq q$, then the tho symbols "01" and "10" have the same probability $p \cdot q$, while the symbols "11" and "00", with different probabilities p^2 and q^2, are discarded. Also in this way long sequences of 0s and 1s (that can be present in some class of generators, for example in continuous-time chaotic circuits that observe the chaotic trajectory hopping between two different attractors [55] are eliminated.

This algorithm does introduce decimation; however the decimation rate cannot be computed a priori. When processing a near-to-random stream where all four couplets are equally distributed, the decimation rate is expected to be 1 over 4; however if the sequence is far from random and in particular if long

sequences of 1s or 0s exist (and this is the typical application, since it is the case where we need a post-processing), the number of bits at the output of this post-processor could be sensibly smaller than the number of bits at its input.

3.4 Parity Based Post-processing

The input stream in divided into contiguous non-overlapping N-bits strings. For each string, the output is computed as the parity bit; then the string is discarded.

Note that computing the parity bit in a string is equivalent to compute the N-bit exclusive-or of the string; for this reason this post-processing is also known as xor post-processing. We refer to it as xor-N, thus indicating also the depth of the post-processing.

Note that this is an effective way to both reshape correlation and increase the entropy of a stream. However the payload in terms of data reduction is very heavy, since it results in a decimation factor equal to $1 : N$

3.5 Hash Function Based Post-processing

A hash function is an algorithm for turning data (usually a message or a file) into a number of fixed length. These functions provide a way of creating a small digital ”fingerprint” from any kind of data. The function chops and mixes (i.e., substitutes or transposes) the data to create the fingerprint, often called a hash value.

Hash functions find several applications in cryptography such as authentication and message integrity checks. The main property of a good hash function is that yields few hash *collisions*, i.e. it is extremely difficult, given a message with a certain hash value, to find another different message with the same hash value.

Due to their *chopping and mixing* properties, they can be effectively used as post-processing functions. The two most-commonly used hash functions are MD5 [75] and SHA-1 [48]; they work respectively with hash value of, respectively, 128 and 160 bits, while the input message can be of any size. So, it is possible to get a string of M bits from the input stream, and get a string of $N = 128$ or $N = 160$ bits at the output. In this way, they are completely algebraic functions, since there is no memory of the previous hashed strings. The decimation rate depends on M and N; theoretically it is also possible to “expand” a message, if $M < N$.

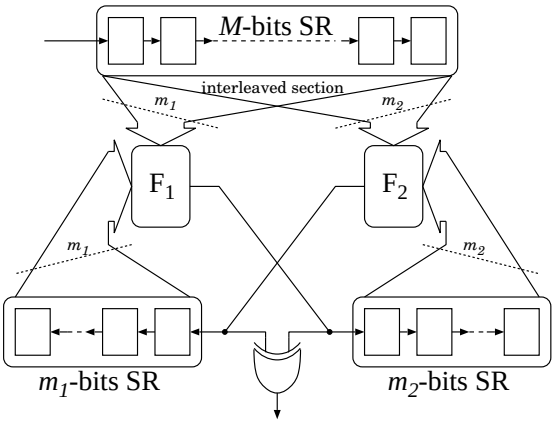


Figure 3.1: Block scheme of the finite-state machine used for the non linear shift register based post-processing.

The main problem of these algorithm is that they are computationally very heavy. As an example, the SHA-1 works with an internal state space of 160 bits, i.e. five word states of 32 bits. In a single stage few basic operations are performed on every word states are performed, including additions, bit rotations, and a non-linear function that varies, depending on the stage. The full algorithm consists of 80 stages; this means that these basic operations are iterated 80 times.

3.6 IIR Based Post-processing

In signal processing, a linear infinite-impulse response filter is defined as

$$y[n] = \sum_{i=0}^{P} b_i x[n-i] - \sum_{j=1}^{Q} a_j y[n-j] \tag{3.2}$$

When applied to a sequence of bits, this is to be interpreted in modulo 2 arithmetic, i.e. the addition is an XOR function and the multiplication is an AND function. These schemes operate at no throughput loss, and every output bit is computed as a linear combination of the current and the previous $P-1$ input bits and the previous $Q-1$ output bits.

Even if many differences from (3.2) could exist, many post processing work in a similar way. In particular, the applied filter does not necessary need to be linear. Two variants are considered in this chapter.

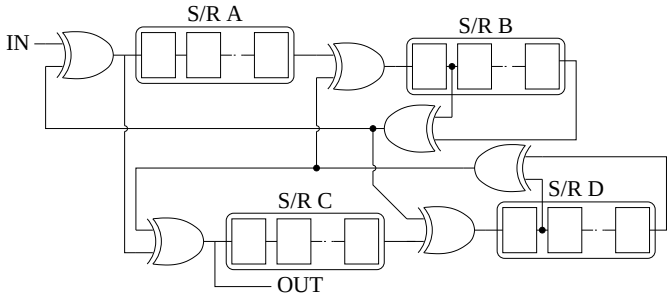


Figure 3.2: Block scheme of the finite-state machine used for the quadri-shift register based post-processing.

3.6.1 NLSR

This scheme was firstly proposed in [73]; its block diagram of the first one can be found in Figure 3.1 and it consists of three shift registers (SR) of length respectively equal to $M = m1 + m2$, $m1$ and $m2$ bits, and two combinatorial logics L1 and L2. Both logic L1 and L2 are based on elementary bit-processing operations from the standard Secure- Hashing Algorithm (SHA). The version used in this dissertation , the value of $m1$ and $m2$ is set respectively equal to 11 and 13. This schematic will be referred as a non-linear shift register (NLSR) based post-processing. For a more detailed discussion about this scheme, see [73].

3.6.2 QSR

The basic structure of the second scheme is depicted in Figure 3.2, and it as firstly presented by author in [5]. Strictly speaking, the working principle of the proposed stage is to XOR the bits coming from the RNG with bits coming from the delayed bit-stream memorized into a linear shift register. This architecture is very similar to a liner-feedback shift register used as a pseudo-random number generator. In order to increase the effectiveness of the process, this basic operation is repeated few times; also many XORs have been inserted between the different shift registers to increase the complexity of the stage. This post-processing is composed by four shift registers A, B, C, D, of lengths a, b, c and d respectively, which can be chosen arbitrarily; note that, for their particular structure, the shift-register B and D has to be composed at least by two stages, while A and C can be constituted by a single flip-flop. This schematic will be referred as a quadri-shift register (QSR) based post-processing.

3.7 Conclusion

In this chapter a brief analysis of some post-processing functions is provided, ranging from well known historical algorithms to new proposed ones. Additionally, a method for evaluating the performance of a post-processing stage is given, as well as an intuitive explanation of the reason why a post-processing function is necessary and how a post-processing function can increase the quality of a random stream.

Chapter 4

Statistical Tests for Randomness

IF DESIGNING a good RNG is a non trivial task, being able to test and validate is perhaps more complex. In the previous chapter the concept of *entropy* has been introduced, as well as how to use it in order to measure the performance of a RNG. It was also concluded that the estimation of entropy cannot be used to perform accurate tests on a random number generator.

Here the concept of statistical tests is introduced. This concept is associated to one of the most important property of a good RNG, that is unpredictability. Briefly speaking, a system is unpredictable if, from the direct observation of its outputs, it is not possible to retrieve any information about the future system evolution. Even if this is a very clear and easy definition, it is impossible to deal directly with it. For example, we can observe a sequence of numbers, no matter how long it is (in theory also an infinite-length sequence), looking for some patterns and, if a model is identified, we can try to predict the following number with a probability greater than the probability given by pure luck. In this case the examined generator is discovered not to be unpredictable. If any model has not being identified, it does not necessary mean that no pattern is present; just that we failed in the attempt of finding it. Roughly speaking, one can check the predictability (i.e. the non-randomness) of a generator, but he will never be able to find a proof that a generator is really unpredictable (that means it is random).

Actually, this property is known as forward unpredictability. In a true random system should also not be feasible to determine the past evolution from the knowledge of any generated outputs (i.e., backward unpredictability is also

required).

The purpose of a statistical test is to identify patterns that let guess some information regarding the following number. As an example, the most simple statistical test one can implement is a test that checks if the symbol produced by the generator under test are equally distributed; in case they are not, it is evident that expecting the most probable symbol gives an advantage in the determination of the successive generated numbers.

Starting from this very simple example, it is possible to address the main problem that can be found during the test phase. When analyzing a generator, what one is looking at is just a generated sequence; if the generator can be identified as a stochastic process, what is available to the test is just a realization. So, like every realization, a sequence has to be interpreted only in a probabilistic way. Everybody, looking at a sequence of all 0s, says "well, this sequence is not random at all". However for a perfect RNG this sequence has statistically the same probability of any other sequence; on the contrary, a RNG that is not capable of generating this sequence is not to be consider random. The only thing that one should do looking at a sequence of all 0s is recognize that the assumption of randomness implies that in a bit sequence the number of 1s is expected to be approximately equal to the number of 0s, and this is clearly not verified.

More detailed, a statistical test can be considered composed by the following two steps:

1. An initial assumption derived from the hypothesis of randomness of the sequence has to be made; for example one could recognize that among all possible randomly generated sequences, the sequences where the number of "0" and the number of "1" are approximately balanced are the majority; instead sequences where the number of "0" is very different from the number of "1" are not common.

2. The sequence is checked under the terms of the initial assumption; referring to the previous example, the numbers of "0" and the number of "1" are counted. This two numbers are expected to be similar; if not, either one of the two following cases is verified: *(a)* the sequence comes from a random generator, and it is one of the non common sequences where the difference between the number of "0" and the number of "1" is large; or *(b)* the sequence comes from a non-random generator. If one supposes not to be in the improbable case *(a)*, then the conclusion is that the sequence is non random. However, there is a (typically very small)

probability that the sequence is random; this probability is the tolerance of the test (usually referred as the level of significance of the test).

Under these terms, when analyzing a single sequence, the only thing that is possible is to give a probability that the sequence comes from a random generator. In fact, as the famous NIST publication SP 800-22 says, "Randomness is a probabilistic property; that is, the properties of a random sequence can be characterized and described in terms of probability." This concept is at the base of all statistical tests.

4.1 P-value Based Tests

P-value stands for *Probability value*. The p-value is the typical output of a statistical test for randomness, and intuitively speaking indicates the probability that the sequence under test is generated by a random generator. Formally, a p-value indicates "the probability that an ideal random generator produces a sequence that, in terms of the statistical feature examined by the test, is less typical than the sequence under test"; strictly speaking, the probability that the examined sequence has to be *more random* than a truly random sequence. The concept of *more random* is of course not rigorous; what is intended can be understood looking in detail how the statistical tests work.

Every test analyzes the input sequence, looking for a particular statistical feature, and expressing it as numerical quantity $s_0 \in S$. Typically this quantity s_0 is a vector or, in simpler cases, a scalar value. In the above example, s_0 can be the difference between the numbers of 0s and the numbers of 1s in a sequence. Then, s_0 is compared to the one s derived for a sequence composed of truly random bits. Clearly, since perfect random sequences can be characterized only in terms of probability, s is a random variable with mean value $E[s] = \overline{s}$ and probability density function $f_s : S \to \mathbb{R}^+$. If the norm $\|\cdot\| : S \to \mathbb{R}^+$ is defined, then $\|s - \overline{s}\|$ is a new random variable defined in $\mathbb{R}^+$ that indicates how far is the observed property from the expected one, whose cumulative distribution function $F_{\|\cdot\|} : \mathbb{R}^+ \to [0, 1]$ can be computed from f_s.

Then the p-value p is computed as $p = 1 - F_{\|\cdot\|}(\|s_0 - \overline{s}\|)$. In this way:

- $p = 1$, if $s_0 = \overline{s}$;
- $p \to 0$, if $\|s_0 - \overline{s}\| \to \infty$;
- for a perfect random generator, p is a random variable that is uniformly distributed in $[0, 1]$.

In other terms, p indicates how similar is the statistical feature observed in the sequence under test to the expected one.

What is intended in the above definition of p when saying that a sequence is *more random* than another one is, simply, that a sequence has a *p-value* greater than another sequence. Mathematically, being X a randomly generated sequence, and X_0 the sequence under test, the above definition of p-value can be expressed as

$$\text{Prob}\left\{p\left(X\right) < p\left(X_0\right)\right\} = p\left(X_0\right)$$

A first, intuitive, way to interpret a statistical test is to consider the test passed if p is high, and consider the test failed if p is low. In fact, this is the standard way to interpret a test, under the following assumptions commonly used in inferential statistic.

Let H_0 be the *null hypothesis*

$$H_0: \quad \text{“the sequence under test comes from a perfect random generator”}$$

Given $\alpha \in [0, 1]$, H_0 is rejected (i.e. the test is considered *failed*) if $p < \alpha$, while H_0 is accepted if $p \geq \alpha$. Of course, this interpretation not *exact*, since two errors can be committed:

- reject H_0 when the sequence is generated by a perfect random generator *(Type I error)*
- accept H_0 when the sequence is generated by a generator that is non random *(Type II error)*.

As far as the Type I error is concerned, its probability can be computed since a perfectly random sequences is completely characterized. Since p is uniformly distributed for a perfect RNG, the probability of a Type I error is simply α. For this reason, α is also called *level of significance*.

However, it is impossible to have a characterization *a priori* of the Type II error. To be considered *good*, a test should look at statistical features that are sensitive to the presence of pattern or regularities typical of non-ideal generator. In this way, the computed p-value drops to zero whenever a pattern is recognized.

4.2 NIST SP 800-22 Test Suite

The SP 800-22 is a publication (titled “A statistical test suite for random and pseudorandom number generator for cryptographic applications”) from the

U.S. National Institute of Standard and Technology describing a suite of 16 different statistical tests [70]. The original publication was dated October 2000, last update May 15, 2001.

Each test in the suite is applied to the same sequence of n bit (the NIST suggests $n = 10^6$) and gives a p-value. Actually, few tests generate two (the *Cumulative Sum* and the *Serial* tests) or more (*Non-Overlapping Template Matching*, *Random Excursion* and *Random Excursion Variant*) p-values; however, being strictly related, it is very common considering only one of them.

All test in the suite are well known tests and, for all of them, an exhaustive mathematical treatment is available. Additionally, the C/C++ source code of all tests in the suite is public available at the NIST website, and is regularly updated, thus confirming the interest about this topic [71]. At the time of this dissertation the latest version available is the 1.8, March 2005. In this new version some known problems with few tests have been fixed: the reference distribution in the *Spectral* test has been modified (see, for example, [50]) while the *Lempel-Ziv* test has been removed.

Actually, the SP 800-22 suite is not the only suite of statistical tests published by NIST. The Federal Information Processing Standard Publication 140-2 (shortly FIPS 140-2), was written as U.S. government computer security standard used to accredit cryptographic modules [68]. The title is "Security Requirements for Cryptographic Modules". First publication was in May 25, 2001 and was last updated December 3, 2002, supercedeing the old publication FIPS 140-1, January 11, 1994. This publication included in the first releases (as the superceded one) a set of four ON/OFF statistical tests (*monobit*, *poker*, *runs* and *long runs* tests) to be run as power-up self tests. These tests are not p-value based tests and they are considered passed if the observed statistical feature is within a predetermined interval. However, they are very easy to pass and so not very significative in the analysis of a random number generator. They are also discontinued by NIST, and have been eliminated in the last release of the publication: the requirement of the four above test is substituted by the need to perform a more generic *cryptographic algorithm test*, a *software/firmware integrity test* and a *critical functions test* (see section 4.9 of the NIST publication). For these reasons, these tests are not considered here.

In the following, a brief description of all test in the SP 800-22 test suite is reported. The mathematical notation is kept as close as possible to the original notation used in the NISP publication. Each test considers the sequence of symbols $X_i = \{-1, +1\}$, $i = 1 \dots n$, since dealing with a zero-average variable is simpler than dealing with the sequence of bits $\varepsilon_i = \{0, 1\}$, $i = 1 \dots n$. Of

course, the relation between the two different notations is simply $X_i = 2\varepsilon_i - 1$.

4.2.1 Frequency Test

Given the input sequence $X_i = \{+1, -1\}$, $i = 1 \dots n$, the balance between symbols -1 and symbols $+1$ is given by

$$S_n = \sum_{i=1}^{n} X_i$$

S_n is a zero average random variable which is binomial distributed. If n is large, the binomial distribution can be approximated with normal distribution, with in this case mean $\mu = 0$ and variance $\sigma^2 = n$. Being S_n is normal, then $|S_n|$ is half normal (i.e. $f_{|S_n|}(x) = 2f_{S_n}(x)\,,\ x \geq 0$). It is very easy to see that

$$p = 1 - F_{|S_n|}(|S_n|) = 1 - (2F_{S_n}(|S_n|) - 1) = \text{erfc}\left(\frac{|S_n|}{\sqrt{2n}}\right)$$

where erfc $(\cdot)$ is the complementary error function.

4.2.2 Block Frequency Test

Divide the input sequence $X_i = \{+1, -1\}$, $i = 1 \dots n$, into N contiguous non-overlapping strings of M symbols, with $n = N \cdot M$. For each i-th string, compute π_i, $i = 1, \dots, N$ the observed frequency of symbols $+1$ among the string. The expected value of each π_i is $1/2$; the distance of the sequence of π_i from the expected sequence is computed as:

$$\chi^2 = 4\,M \sum_{i=1}^{N} \left(\pi_i - \frac{1}{2}\right)^2$$

π_i is a random variable with a binomial distribution, which is approximated to a normal distribution since M is large. Then χ^2, that is the sum of N normal variable, is distributed according to a *chi-square distribution* with N degree of freedom. The p-value is computed through its cumulative density function, which is known to be obtained from the regularized incomplete gamma function

$$F_{\chi^2}(x) = \frac{\gamma(k/2, x/2)}{\Gamma(k/2)}$$

where k is the number of degree of freedom; in this case $k = M$.

NIST suggests to consider a string length equal to $M = 128$ bits.

4.2.3 Cumulative Sums Test

Given the input sequence $X_i = \{+1, -1\}$, $i = 1 \ldots n$, consider all the intermediate sums S_k, $k = 1, \ldots, n$ defined as

$$S_k = \sum_{i=1}^{k} X_i$$

The variable S_k can be described as a *random walk* process. Let also be

$$z = \max_{1 \le k \le n} |S_k|$$

the maximum excursion from zero of the random walk. The cumulative distribution of z, for large n, can be approximated with

$$\begin{aligned} F_z(z) = & \sum_{k=\left(\frac{-n}{z}+1\right)/4}^{k=\left(\frac{n}{z}-1\right)/4} \left[\Phi\left(\frac{(4k+1)\, z}{\sqrt{n}} \right) - \Phi\left(\frac{(4k-1)\, z}{\sqrt{n}} \right) \right] \\ & + \sum_{k=\left(\frac{-n}{z}-3\right)/4}^{k=\left(\frac{n}{z}-1\right)/4} \left[\Phi\left(\frac{(4k+3)\, z}{\sqrt{n}} \right) - \Phi\left(\frac{(4k+1)\, z}{\sqrt{n}} \right) \right] \end{aligned}$$

where $\Phi(x)$ is the normal cumulative distribution function. This test is repeated twice: the first time considering k from 1 to n *(forward test)* and a second time with k form n down to 1 *(backward test)*. Hence, two p-values are computed.

4.2.4 Runs Test

Given the input sequence $X_i = \{+1, -1\}$, $i = 1 \ldots n$, this test computes the total number v_n of runs in the sequence. A run consist in an uninterrupted sequence of identical symbols, and bounded by two the opposite symbols. Mathematically

$$r_k = \begin{cases} 0 & \text{if } \mathrm{X_k} = \mathrm{X_{k+1}} \\ 1 & \text{otherwise} \end{cases}$$

$$v_n = 1 + \sum_{k=1}^{n-1} r_k$$

given the proportion π of symbols $+1$ in the input sequence, v_n is approximated for large n with a normal distributed variable, with mean $\mu = 2n\pi\,(1-\pi)$ and variance $\sigma^2 = 2\pi^2\,(1-\pi)^2$.

4.2.5 Longest Run of Ones Test

Divide the input sequence $X_i = \{+1, -1\}$, $i = 1 \dots n$, into N contiguous non-overlapping strings of M bits, with $n = N \cdot M$. For each string, compute the length of the longest block composed only by symbols $+1$; among the N computed maximum lengths, count the hits v_i of each length i, with $\sum v_i = N$, and compare it to the expected value $N\pi_i$. Only the $K + 1$ most common lengths have to be considered: for example, for $n > 750,000$, it is $M = 10^4$ and $K = 6$; the 7 lengths considered are $i \leq 10$, $i = 11$, $i = 12$, $i = 13$, $i = 14$, $i = 15$, and $i \geq 16$. The distance of the distribution of the v_i from the expected distribution is computed with a chi-square goodness of fit test:

$$\chi^2 = \sum_{i=0}^{K} \frac{(v_i - N\pi_i)^2}{N\pi_i}$$

χ^2 is distributed according to a chi-square distribution with K degrees of freedom. The expected frequencies π_i and the number of degree of freedom K are precomputed and tabulated according to the three values of M, $M = 8$, $M = 128$ and $M = 10^4$, select according to the length of the sequence n.

4.2.6 Binary Matrix Rank Test

Divide the input sequence $X_i = \{+1, -1\}$, $i = 1 \dots n$, into N contiguous non-overlapping strings of $M \cdot Q$ bits, with $n = N \cdot M \cdot Q$. With each string build a binary $M \times Q$ matrix and compute its rank r, $0 \leq r \leq \min(M, Q)$. The rank is defined as the maximum number of lines or columns which are linearly independent. Note that this has to be computed using the binary algebra defined only on the symbols $\{-1, +1\}$. The probability that such a matrix has rank r is given by

$$p_r = 2^{r(Q+M-r)-MQ} \prod_{i=0}^{r-1} \frac{\left(1 - 2^{i-Q}\right)\left(1 - 2^{i-M}\right)}{\left(1 - 2^{i-r}\right)}$$

The distance of the observed frequency from the expected probability is measured with a χ^2 goodness of fit test with K degrees of freedom. NIST fixed $M = Q = 32$ and $K = 2$.

4.2.7 Spectral Test

Given the input sequence $X_i = \{+1, -1\}$, $i = 1 \dots n$, its Discrete Fourier Transform (DFT) is computed. A threshold value T is computed such that in the unilateral frequency spectrum, the 95% of the bin should have an amplitude smaller than T. The effective number of bins N_1 having an amplitude

smaller than the threshold value T is normal distributed, with mean $\mu = N_0 = \sqrt{-\ln(0.05) \cdot n}$, and variance $\sigma^2 = 0.95 \cdot 0.05 \cdot n/4$. [1]

4.2.8 Non-Overlapping Template Matching Test

The input sequence $X_i = \{+1, -1\}$, $i = 1 \dots n$, is divided into N contiguous non-overlapping strings of M symbols, with $n = N \cdot M$. Then, given a template sequence B of length m bits, W_i is the number of times the template B is present as non-overlapping string in the i-th string, with $i = 1 \dots N$. Under the assumption of randomness, such a number is normal with mean and variance

$$\mu = \frac{M - m + 1}{2^m} \quad \sigma^2 = M \left(\frac{1}{2^m} - \frac{2m - 1}{2^{2m}} \right)$$

From these N normal variables, the new random variable

$$\chi^2 = \sum_{j=1}^{N} \frac{(W_j - \mu)}{\sigma^2}$$

has a chi-square distribution with N degrees of freedom. The value of N has been fixed by NIST at the value $N = 8$.

This test produces a p-value for each used template; for the suggested value $m = 9$, there are 148 templates in the NIST template file.

4.2.9 Overlapping Template Matching Test

As in the previous test, a template B of m symbols are researched into the input sequence divided into N strings; however this time two hits of the template B can partially overlap themselves. After examining the N block, v_i is computed as the number of blocks where the template B is present exactly i times, with $\sum v_i = N$. v_i is poisson-distributed, with

$$\lambda = \frac{M - m + 1}{2^m}$$

The distance between the observed and the theoretical distribution is computed with a chi-square goodness of fit test with K degrees of freedom.

In the test, the value of M is fixed at $M = 1032$, while $K = 5$. NIST suggests $m = 9$; in the test, only the template composed by m symbols "+1" is used, so only one p-value is computed.

[1] Actually, these values come in the last release of the NIST code, following the suggestion in [50]. In the original NIST publication, these value were $\mu = \sqrt{3 \cdot n}$ and $\sigma^2 = 0.95 \cdot 0.05 \cdot n/2$

4.2.10 Universal Test

Divide the input sequence $X_i = \{+1, -1\}$, $i = 1 \dots n$, into $Q + K$ contiguous non-overlapping strings of L symbols, with $n = (Q + K) \cdot L$. Of these strings, Q are used for the initialization and K for the test itself. A table T_j with $j = 1, \dots, 2^L$ is created and preset with $T_j = 0, \forall j$; each entry T_j of the table is associated to one of all the 2^L possible different strings of length L.

Then for $i = 1, \dots, Q + K$, consider the i-th string among all strings in which the input sequence has been divided; consider also the entry of the table T_j associated to the considered string. During the test initialization, i.e. for $i = 1, \dots, Q$, update the table entry with the value $T_j = i$; during the test, i.e. $i = Q + 1, \dots, Q + K$, starting from $f_Q = 0$, build the sequence

$$f_i = f_{i-1} + \frac{1}{K} \log_2(i - T_j)$$

and then update the table entry $T_j = i$. The random variable f_{Q+K} is assumed normal for large values of K, with mean and variance empirically precalculated and depending on the value of L.

The values suggested by NIST are $L = 7$ and $Q = 1280$.

4.2.11 Approximated Entropy Test

Given the input sequence $X_i = \{+1, -1\}$, $i = 1 \dots n$, n overlapping strings of m bits are created (appending the first $m-1$ symbols to the end of the sequence when necessary). Then, the m order entropy is computed

$$\varphi^{(m)} = \sum_{i=1}^{2^m} \pi_i \log \pi_i$$

where π_i are the observed frequency of all the 2^m possible blocks of m bits.

The quantity $\varphi^{(m)} - \varphi^{(m+1)}$ is called *approximated entropy*; the random variable

$$\chi^2 = 2n \left(\ln 2 - \left(\varphi^{(m)} - \varphi^{(m+1)} \right) \right)$$

is, for large n, chi-square distributed with 2^m degree of freedom.

The value suggested by NIST for m is $m = 10$.

4.2.12 Random Excursion Test

Starting from the input sequence $X_i = \{+1, -1\}$, $i = 1 \dots n$, consider the random walk

$$S_k = \sum_{i=1}^{n} X_i, \; k = 1, \dots, n$$

Defining a zero crossing a point for which $S_k = 0$, and, for convenience, $S_0 = S_{n+1} = 0$, in the sequence S_k, $k = 0, \ldots, n+1$ there are $J+1$ zero crossings, which separates J *cycles*, i.e. strings of S_k separated by two successive zero crossing. Then, given an integer x, let be v_i, $i = 0, \ldots, J$ is the number of cycles in which x appears exactly i times, with $\sum v_i = J$. The sequence of the v_i is compared to the expected values (precomputed and tabulated) with a chi square goodness of fit test with $K = 5$ degrees of freedom. This test is performed only if there are a minimum number of cycles, and is repeated for $x \in \{-4, -3, -2, -1, 1, 2, 3, 4\}$. Eight p-values are so generated.

4.2.13 Random Excursion Variant Test

From the same random walk S_k as in the previous test, compute $\xi(x)$ equal to the number of times a given integer x occurs in the walk, i.e. $S_k = x$. The limit distribution of $\xi(x)$ is a normal distribution with mean and variance

$$\mu = J \quad \sigma^2 = J(4|x| - 2)$$

This test is repeated 18 times for $x \in \{-9, \ldots, -1, 1, \ldots, 9\}$, and, like the previous on, it is performed only if a minimum number of cycles exist. Note that, due to this limitation, this test and the previous one are not always performed even in case of a true random sequence. It can be shown that both tests are performed, with the limitation introduced by NIST, only on 63% of perfectly random sequences [56].

4.2.14 Serial Test

Given the input sequence $X_i = \{+1, -1\}$, $i = 1 \ldots n$, n overlapping m bits strings are created (appending the first $m-1$ symbols to the end of the sequence when necessary). Let be v_i, $i = 1, \ldots, 2^m$ the observed frequencies of all possible sequences of m symbols; then compute

$$\Psi_m^2 = \frac{2^m}{n} \sum_{i=1}^{2^m} \nu_i^2 - n$$

From the two quantities

$$\nabla \Psi_m^2 = \Psi_m^2 - \Psi_{m-1}^2$$
$$\nabla^2 \Psi_m^2 = \Psi_m^2 - 2\Psi_{m-1}^2 + \Psi_{m-2}^2$$

which are distributed according to a chi square distribution with, respectively, $m-1$ and $m-2$ degrees of freedom, two p-values are computed. The value suggested by NIST for m is $m = 16$.

4.2.15 Linear Complexity Test

The input sequence $X_i = \{+1, -1\}$, $i = 1 \dots n$, is divided into N contiguous non-overlapping strings of M bits, with $n = N \cdot M$ For each string, its linear complexity L_i , $i = 1, \dots, N$, is calculated with the Berlekamp-Massey algorithm [65]. Given the sequence $s^{(n)} = (X_1, \dots, X_n)$, its linear complexity $L\left(s^{(n)}\right)$ is defined as the length of the shortest linear feedback shift-register (LFSR) that generates $s^{(n)}$ as its first n terms for some initial state. The asymptotic distribution of L_i for large M is non trivial; the NIST test computes

$$T_i = (-1)^M (L_i - \mu) + \frac{2}{9}$$

where μ is the theoretical mean

$$\mu = \frac{M}{2} + \frac{9 + (-1)^{M+1}}{36} - \frac{\frac{M}{3} + \frac{2}{9}}{2^M}$$

The N values T_i, $i = 1, \dots, N$, are divided into 6 bins; the frequency of each bins is compared the the expected theoretical one with a chi-square goodness of fit test.

NIST suggests the value $M = 500$.

4.3 DieHard Test Suite

The Diehard tests suite is a battery of statistical tests developed by George Marsaglia at the Florida State University over several years and first published in 1995 on the CD-ROM *The Marsaglia Random Number CDROM including the Diehard Battery of Tests of Randomness* [62]. A new version of the tests battery is under development at the University of Hong-Kong [63], as remembered by Marsaglia himself:

> *"Supported by a grant from the National Science Foundation, 1000 copies of that CDROM were distributed in 1995 to math and science departments and to interested researchers worldwide. The 1000 copies were soon exhausted, but the CDROM has since been frequently accessed via the Internet. A new version of the CDROM is being prepared, and this file accompanies C source code for new versions of the Diehard tests, as well as several new difficult-to-pass tests."*

The battery includes various tests; many of them comes from popular culture or paradoxes, like the *Birthday Spacing Test*, referring to the Birthday Paradox [26, 28], that regards the fact that in a group of people the probability that

at least two of them will have the same birthday is much greater that the one which one could expect. Another test is the *Monkey Test* (changed into the *Gorilla Test* in the newer version of the battery), referring to the topos of the monkey striking randomly on a typewriter machine keys, started by the French mathematician Èmile Borel in 1913, and known in the anglophone world as "If a million monkeys were given a million typewriters, eventually one of them might produce the complete works of Shakespeare". However, in Borel's intentions, these monkeys would produce the whole content of the French National Library (literally, the "greatest library in the world").

Though implemented and used, no test results obtained with this battery are reported in this dissertation. The reason is twofold. First, all of the implemented tests are written to test the independency of 32 bits integer numbers and do not consider the string as a sequence of bits like the NIST suite; most of the tests, for example, turn a sequence of 32 bit value into a sequence of floating in $[0, 1]$; or maybe, if a tests require an number of bits $n < 32$, it is repeated many times, first considering bits from 0 to $n - 1$, then form 1 to n, and so on. This also ends in generating many p-values (globally, the number of them generated by the suite is larger than 200) which are, of course, not independent, and also in the requirements of many bits (the new gorilla test requires about 67M integers in 32 bits notation, that is nearly half of the data contained in a CD-ROM). The second problem, is that all tests require a fixed number of bits, which is different from test to test. Strictly speaking, the battery is not composed by a set of homogeneous tests: each test analyzes a sequence of different length, and gives a different numbers of p-values.

For these reasons only the NIST suite is considered in this dissertation; DieHard tests has also been performed though in a non-systematic way, always confirming the quantitative and qualitative conclusion of the NIST suite.

4.4 Second Level Tests

The usual way to test a random number generator is to generate a sequence of n bits and analyze it with a chosen test suite as described in Section 4.1. Given a level of significance, the sequence is considered random if all tests in the suite produce P-values greater then the level of significance, always remembering the possibility to commit a Type I or Type II error. To limit the probability of a Type I error, the value of α is usually kept low; a typical value for α is the value suggested by NIST, i.e. $\alpha = 0.01$.

The weakness of this approach is that it is well known that some pseudoran-

dom generators can very easily pass all known statistical tests. Let consider a periodic (and thus, *non random*) generator, whose period is the simple sequence "101100", and the above described frequency test. The generated sequence

$$101100101100101100101100101100101100\ldots$$

will always pass the frequency test, since the number of 1s and of 0s in the period is well balanced, independently on the sequence length. However this can be effectively used to discover the generator as non random. A perfect random generator produces sequences that do not always pass the frequency test, but that have a probability α of a failure. The above periodic generator has a failure probability equal to zero. According to this intuitive idea, it is possible to have a more reliable analysis, i.e. a lower probability of a Type II error, when considering a number N of sequences instead of a single sequences. Such a test involves several results from basic test; for this reason it is addressed as *second level test*.

At this point one could argue that a comparison between a basic test and a second level test is not fair, since while a basic test is performed on a sequence of n bits, a second level test involves $n \cdot N$ bits, i.e. a much greater number of bits. However it is easy to notice that, referring to the above periodic generator, even considering a longer sequence does not help in the effort to discover the non-randomness of the generator with a single basic test, since the test is always passed with $p \simeq 1$ independently of the number of bits.

Of course, a second level test is still a statistical test, characterized by a null hypothesis H_0 that usually is "data is random", and by a probability of Type I error and of Type II error. Before introducing some possible second level tests, it may help remember the following

REMARK 1. Let $X_1, X_2, \ldots, X_n$ a succession of independent random variables, that share the same probability distribution, with mean $\mu = E[X_i]$ and variance $\sigma^2 = E\left[X_i^2\right] - \mu^2$. The central limit theorem [31] asserts that, under some convergence conditions that are always satisfied under the above assumptions, the cumulative $F_n(\cdot)$ distribution of the random variable

$$S_n = \frac{1}{\sigma\sqrt{n}} \sum_{k=1}^{n} (X_k - \mu)$$

converges to the normalized cumulative normal distribution $\Phi(\cdot)$ (i.e. a cumulative normal distribution with $\mu = 0$ and $\sigma^2 = 1$) as n approaches $+\infty$. Also, the convergence rate is quantified by the Berry-Esseén inequality, that affirms

that a positive constant C exists such that

$$|F_n(x) - \Phi(x)| \leq \frac{CE\left[|X_i|^3\right]}{\sigma^3\sqrt{n}}$$

Calculated values of the constant C have decreased markedly over the years, from 7.59 (Esséen's original bound) to 0.7975 in 1972 (by P. van Beeck [27]). The best current bound is 0.7655 (by I. S. Shiganov in 1986 [29]).

NIST suggests, in Section 4 of its publication, to follow two strategies in order to implement a second level test. The first one is based on the observation that only a ratio equal to $1 - \alpha = 0.99$ of sequences generated by an ideal RNG should pass a basic test. So a number N of basic tests are performed, and (independently for each test in the suite) the ratio of sequences with $p > \alpha$ is computed and compared to the expected one. The deviation one may expect from this number can simply be computed considering the following random process

$$X_i = \begin{cases} 1, & \text{with probability } p \\ 0, & \text{with probability } q = 1 - p \end{cases}$$

and indicating $\mu_i = E[X_i] = p$ and $\sigma_i^2 = E\left[X_i^2\right] - \mu_i^2 = pq$, the central limit theorem states that variable

$$S_N = \frac{1}{N}\sum_{i=1}^{N} X_i$$

for a sufficient large N, is normal distributed with mean $\mu = p$ and $\sigma^2 = pq/N$. Passing a basic test can be schematized as the outcome $X_i = 1$, with $p = 1 - \alpha$; the ratio of sequences passing the basic test is S_N, i.e it is a normal distributed variable with $\mu = 1 - \alpha$ and $\sigma^2 = \alpha(1 - \alpha)/N$. From this, it is possible to compute a reference interval, such that the probability to be out of this interval is equal to a predetermined level of significance. In this way, the test can be considered passed if the observed ratio of passed test lies in the reference interval; the probability of a Type I error is the significance level. It is common to adopt the three-σ criterion in the calculation of the reference interval, i.e. the interval is bounded by $1 - \alpha \pm 3\sqrt{\alpha(1 - \alpha)/N}$; the probability that a perfect generator has not to pass this test is about 1% and is aligned with the probability of Type I error in a basic test.

The second approach proposed by NIST consists in checking if the obtained p-values are uniformly distributed in the interval $[0, 1]$. To this purpose, any goodness-of-fit test can be used; in the NIST publication a chi-square test over $k = 10$ bins is considered [13, 30]; here it is also considered a Kolmogorov-Smirnov test [22, 30]. Both tests consider a set of values, compare their distribution with a reference one (in this case, the uniform distribution in $[0, 1]$) and

compute a p-value, that has to be interpreted exactly as in Section 4.1: i.e. $p = 1$ means that the two distribution are identical, while we get $p = 0$ if they cannot be considered similar. In this case, H_0 corresponds to "the two distributions match"; again, H_0 is rejected if $p < \alpha'$, and accepted if $p \geq \alpha'$. The NIST publication considers a value $\alpha' = 0.0001$; this value seems however too small, and in this dissertation the value $\alpha' = 0.01$ is used. In this way, the Type I error probability is the same as the above three-σ criterion and of a basic test; the comparison between these three strategies can be done fairly .

The effectiveness of this approach can be shown by an example. Two pseudorandom generator have been considered: the 32 bits version of the KISS [64], which is a very simple but effective generator, and the BBS generator (also known as $x^2 \bmod n$) that is a computationally very heavy pseudorandom generator that has proven to be cryptographically secure (i.e. it passes al polynomial-time tests)[35]. The C source code of these two algorithms can be found in Appendix B For both generators a first level test on a single sequence, and a second level test , checking the distribution of $N = 5,000$ and $N = 10,000$ p-values obtained from the same number of different sequences, have been considered. Both the second level tests described above are performed; with the three-σ criterion the reference interval is , in first case, the interval $[0.9858, 0.9942]$, and in the second the interval $[0.9870, 0.9930]$, while in the χ^2 test for the uniformity test 16 bins have been taken into account. For the Random Excursion and the Random Excursion Variant, due to the smaller number of p-values generated, this interval is larger (see the description of these tests in Section 4.2)

Results are shown in Table 4.1 where, depending on the test type, a p-value or the ratio of p-values passing the standard test have been reported. All p-values smaller than the level of significance, as well as the ratio out of the reference interval, have been stressed in bold. It is easy to see that both generators pass all first level tests; however (apart from the *Spectral* test; this will be discussed in the following) only the BBS generator passes the second level tests. The proposed uniformity second level test is able to recognize the non-randomicity of the KISS generator, while a simple first level test fails in this attempt. Furthermore, the test performed on $N = 10,000$ p-values has shown to be much more sensitive with respect to the test performed on only $N = 5,000$ p-values. The example shows that the uniformity second level test is the most reliable one.

So, one could expect that increasing the number N of basic tests, the reliability of a second level uniformity test is improved. Regrettably, this is not al-

SP800-22 test	single test	2nd level test on 5,000 p-values			2nd level test on 10,000 p-values		
		$\pm 3\sigma$	chi-square	KS	$\pm 3\sigma$	chi-square	KS
Frequency	0.713479	0.991800	0.012855	0.010541	0.991500	**0.000037**	**0.000001**
Block Frequency	0.129962	0.990000	**0.003909**	**0.000106**	0.987400	**0.001542**	**0.000011**
Cumulative Sums	0.833869	0.992600	**0.006522**	**0.000508**	0.992000	**0.001617**	**0.000001**
Runs	0.768154	0.992400	0.572271	0.158968	0.991800	0.611109	0.158852
Longest Run of Ones	0.736930	0.988800	0.402453	0.341598	0.989800	0.664904	0.101711
Matrix Rank	0.224896	0.991400	0.533445	0.328789	0.988400	0.740669	0.312901
Spectral (DFT)	0.060580	0.987600	0.217923	0.040959	**0.986200**	**0.000117**	**0.000022**
NOT Matching	0.085400	0.991400	0.133840	0.401997	0.990500	0.752961	0.174745
OT Matching	0.105840	0.990200	0.238932	0.091968	0.987900	0.020062	**0.001076**
Universal	0.711080	0.988800	0.768687	0.154318	0.989100	0.018867	**0.000247**
Approx. Entropy	0.029426	0.988800	0.216497	0.081427	0.990500	0.429767	0.390263
Random Excursion	0.692131	0.991125	0.028218	0.093570	0.990648	0.815752	0.737741
Random Exc. Variant	0.280164	0.983835	0.206242	0.284070	0.987843	0.297997	0.489387
Serial	0.870041	0.988800	0.888194	0.626380	0.990300	0.043204	0.016218
Linear Complexity	0.998535	0.991600	0.832451	0.330536	0.988500	0.661848	0.209730

(a)

SP800-22 test	single test	2nd level test on 5,000 p-values			2nd level test on 10,000 p-values		
		$\pm 3\sigma$	chi-square	KS	$\pm 3\sigma$	chi-square	KS
Frequency	0.783016	0.989800	0.524521	0.754246	0.989000	0.497291	0.901241
Block Frequency	0.214954	0.992200	0.307848	0.393913	0.990600	0.425844	0.721963
Cumulative Sums	0.790206	0.989800	0.833942	0.866127	0.990800	0.563001	0.400115
Runs	0.719370	0.990400	0.927969	0.871212	0.991600	0.157584	0.351341
Longest Run of Ones	0.991280	0.990200	0.934397	0.942403	0.990800	0.858312	0.691410
Matrix Rank	0.027857	0.988000	0.698415	0.541068	0.988400	0.527570	0.493364
Spectral (DFT)	0.152641	0.987400	0.198573	0.068148	**0.986400**	**0.005823**	**0.001789**
NOT Matching	0.392848	0.987400	0.104923	0.214377	0.990600	0.682907	0.287446
OT Matching	0.358323	0.989400	0.914791	0.422329	0.988200	0.507020	0.150185
Universal	0.505726	0.988400	0.045007	0.323068	0.987700	0.283450	0.021532
Approx. Entropy	0.730140	0.989000	0.090770	0.050686	0.989600	0.116893	0.030692
Random Excursion	0.715979	0.988186	0.878736	0.463362	0.988949	0.450995	0.274106
Random Exc. Variant	0.288537	0.990421	0.995270	0.972168	0.989270	0.462840	0.537302
Serial	0.520702	0.988800	0.951673	0.821649	0.989800	0.834313	0.316490
Linear Complexity	0.814581	0.987800	0.846359	0.989176	0.988600	0.969684	0.887421

(b)

Table 4.1: (a) Results of randomness test for the KISS pseudorandom generator; and (b) for the BBS pseudorandom generator.

	BBS		VIA PadLock		Quantis		ADC based RNG	
SP800-22 test	chi-square	KS	chi-square	KS	chi-square	KS	chi-square	KS
Frequency	**0.003718**	0.548951	0.011355	0.014578	0.013303	0.194091	0.080238	0.236429
Block Frequency	0.255425	0.359622	0.418624	0.159548	0.127159	0.280038	0.735449	0.618841
Cumulative Sums	0.800947	0.166991	0.995862	0.789873	0.043241	0.550849	0.209272	0.663020
Runs	0.256956	0.369664	0.874245	0.148838	0.181876	0.035706	0.229527	0.354083
Longest Run of Ones	**0.007613**	0.015715	**0.000212**	**0.001203**	0.016752	0.019724	0.023731	0.068167
Matrix Rank	**0.000000**	**0.000006**	**0.000000**	**0.000054**	**0.000000**	**0.000015**	**0.000000**	**0.001051**
Spectral (DFT)	**0.000000**	**0.000000**	**0.000000**	**0.000000**	**0.000000**	**0.000000**	**0.000000**	**0.000000**
NOT Matching	0.828875	0.984424	0.491052	0.317815	0.379925	0.174608	0.744362	0.286197
OT Matching	**0.000000**	**0.000000**	**0.000000**	**0.000000**	**0.000000**	**0.000000**	**0.000000**	**0.000000**
Universal	**0.000000**	**0.000000**	**0.000000**	**0.000000**	**0.000000**	**0.000000**	**0.000000**	**0.000000**
Approx. Entropy	**0.006100**	**0.000272**	**0.009827**	**0.000055**	0.044520	**0.000079**	**0.000157**	**0.000041**
Random Excursion	**0.000053**	**0.000002**	**0.004256**	0.024853	0.123659	0.333511	0.036014	0.059785
Random Exc. Variant	**0.000006**	**0.000328**	**0.000000**	**0.000021**	**0.000000**	**0.000197**	**0.000267**	**0.000333**
Serial	0.897125	0.537821	0.753251	0.693649	0.681278	0.182290	0.933284	0.679118
Linear Complexity	0.326050	0.224488	0.569766	0.221725	0.824617	0.461214	0.131702	0.639498

Table 4.2: Results of uniformity second-level randomness test for different RNGs, with $N = 150,000$ sequences.

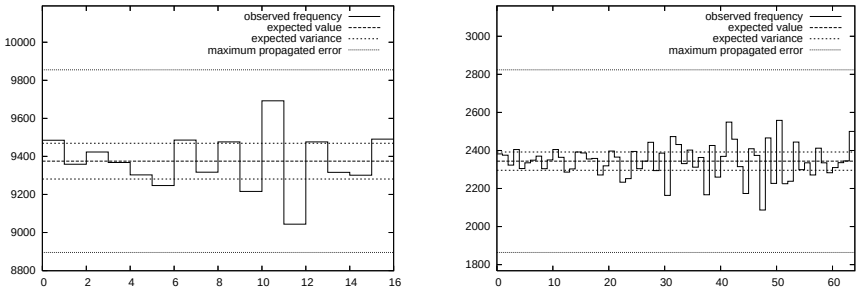
ways true. In fact, as reported by the example of Table 4.2, when $N = 150,000$, results are far from the desired ones, since nearly a half of the tests fails.

This does not happen only for the *(non random)* BBS, but also for the considered true random generators. The test was also repeated on three high-end physical process based true random generators: the VIA PadLock generator [42], the Quantis generator developed by idQuantique [53] (they are both described in detailed in Appendix B) and the eight stages 0.35 μm implementation of the ADC based RNG described in Chapter 2. All three physical generators have been considered with a very strong additional post-processing, composed by the QSR filter described in Chapter 3, Section 3.6, followed by a SHA function with a decimation rate equal to 20/32; this in order to hide all possible imperfections and be sure to analyze a stream as close as possible to a sequence of independent and balanced bits.

This flaw is due to the fact that all reference distributions used in the basic tests are just asymptotic distribution for very large value of some parameter, usually n. This reflects in errors in the p-value computation, and thus in errors in the p-value distribution.

In order to identify the problem, focus on the simplest *Frequency Test*. This test is not a particularly critical one; however the obtained p-values are, especially in the chi-square test, very near to the significance level for all generator, i.e. all the observed distributions of p-values are quite far from being uniform.

Figure 4.1 shows the observed distribution of the p-values for this test ap-



(a) $p = 0.003718$ (b) $p = 0.000000$

Figure 4.1: Comparison between expected deviation and measured deviation in the distribution of $N = 150,000$ p-values in the interval $[0, 1]$ for the Frequency Test in the cases: (a) $n = 10^6$, k=16; (b) $n = 10^6$, k=64.

plied to the BBS generator, in the case $k = 16$ bins and $k = 64$ bins. Considering the theoretical standard deviation in the distribution of N independent, uniformly distributed values over k bins, the number of values found in a bin as can be modeled as the sum of a N binary random variable with $p = 1/k$; from Remark 1 this is, for large N, normal distributed, with $\mu = N/k$ and $\sigma = \sqrt{N(k-1)}/k$. In the cases of Figure, it is respectively $\sigma \simeq 94$ and $\sigma \simeq 48$. The observed deviation is far from this value, and also this error seems to be not dependent on the number k of bins.

This deviation can be identified with an error propagated from the computation of the p-value in the basic test, and due to the introduced approximations with the central limit theorem.

In the test, the random variable S_n was computed as

$$S_n = \sum_{i=1}^{n} X_i$$

with S_n is assumed normal with $\mu = 0$ and $\sigma^2 = n$.

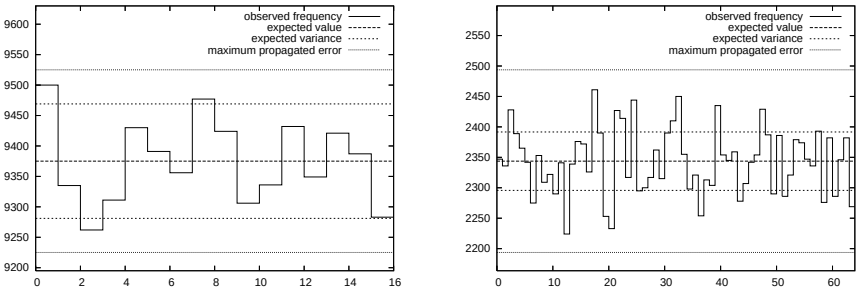
Instead of S_n, one could consider the variable

$$Z_n = \frac{1}{\sqrt{n}} S_n = \frac{1}{\sqrt{n}} \sum_{i=1}^{n} X_i$$

This comes exactly in the form of the Remark 1, since for each X_i, it is $\mu = 0$ and $\sigma^2 = 1$. The cumulative distribution $F_Z(\cdot)$ of Z_n for large n can be confused with the normalized cumulative normal distribution, i.e. $F_Z(\cdot) \simeq \Phi(\cdot)$. Considering the normalized variable Z_n instead of S_n, the p-value can be computed simply by:

$$p = 2F_Z(|Z_n|) \simeq 2\Phi(|Z_n|)$$

Now it is possible to notice that, since there is a proportional relation between the p-value and the normalized cumulative normal distribution, Berry and



(a) $p = 0.959299$ (b) $p = 0.098405$

Figure 4.2: Comparison between expected deviation and measured deviation in the distribution of $N = 150,000$ p-values in the interval $[0, 1]$ for the Frequency Test in the cases: (a) $n = 10^7$, $k = 16$; (b) $n = 10^7$, $k = 64$.

Esseèn inequality limits also the error ε on the p-value computation

$$\varepsilon = \sup_x |2F_Z(x) - 2\Phi(x)| = 2\frac{CE\left[|X_i|^3\right]}{\sigma^3\sqrt{n}} = 2\frac{C}{\sqrt{n}}$$

since $\sigma = 1$ and the third order moment is $E\left[|\xi_i|^3\right] = 1$. If $n = 10^6$, then $\varepsilon = 1.6 \cdot 10^{-3}$.

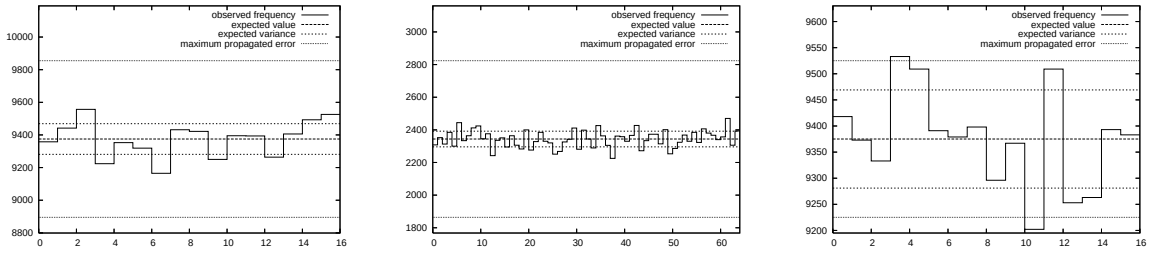
Assuming this bound on the error in the computation of a p-value, it is possible to bound also the maximum error in the distribution of N p-values in k bins. A P-value that should belong to a bin can be found into the nearby one only if its distance from the border of the two bins is less than ε. If we have N p-values uniformly distributed in $[0, 1]$ the number of P-values that can be found in the wrong bin is εN. Note that according to this, the propagated error would be effectively independent of the numbers of bins, as observed in Figure 4.1.

Since all bins (but the first and the last), have two neighbors, the maximum error Δ in the number of P-values in a bin is $\Delta = 2N\varepsilon$. In the case of Figure 4.1, $N = 150,000$, so $\Delta \simeq 480$. This value is compatible with what we can observe in both plot.

If the analysis were correct, increasing n one would see this propagated error decreasing as $1/\sqrt{n}$. To get an experimental verification, the second level uniformity test has been repeated with $n = 10^7$ bits; in this case $\Delta \simeq 150$. The obtained distribution for the Frequency Test is shown in Figure 4.2; as expected the error is bounded in an interval about three times smaller than in the previous case.

The analysis applied to the Frequency Test can be applied *as is* to the Runs Test. In this test

$$r_k = \begin{cases} 0 & \text{if } X_k = X_{k+1} \\ 1 & \text{otherwise} \end{cases}$$



(a) $p = 0.256956$ (b) $p = 0.226864$ (c) $p = 0.613857$

Figure 4.3: Comparison between expected deviation and measured deviation in the distribution of $N = 150,000$ p-values in the interval $[0, 1]$ for the Runs Test in the cases: (a) $n = 10^6$, k=16; (b) $n = 10^6$, k=64; (c) $n = 10^7$, k=16.

i.e. under the assumption of randomness, r_k is a random variable that can assume with the same probability the two values 0 and 1. Finding a proper variable change, thus reporting also this test in the form in which one could apply the Berry and Esseèn inequality, is an easy task. The error bound is exactly the same as in the previous case.

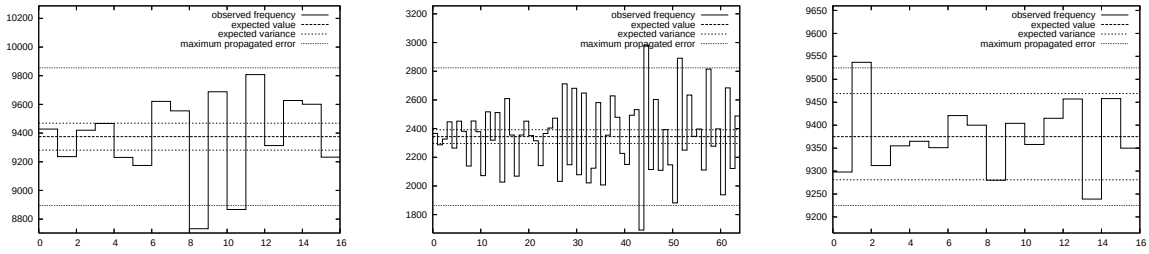
The observed error for the Runs test is reported in Figure 4.3 and, even if the case does not seem particularly unlucky (the second level uniformity test is still passed even for $k = 64$) it is possible to notice the error has the same behavior as in the previous case.

All other tests in the suite use a different, more complex, reference distribution, for which such a relation is hard to write down. Many tests use a chi-square distribution; a chi-square distribution can be obtained starting from K normal random variables $X_1, X_2, \ldots, X_K$ as

$$Q = \sum_{i=1}^{K} \frac{(X_i - \mu_i)^2}{\sigma_i^2} \tag{4.1}$$

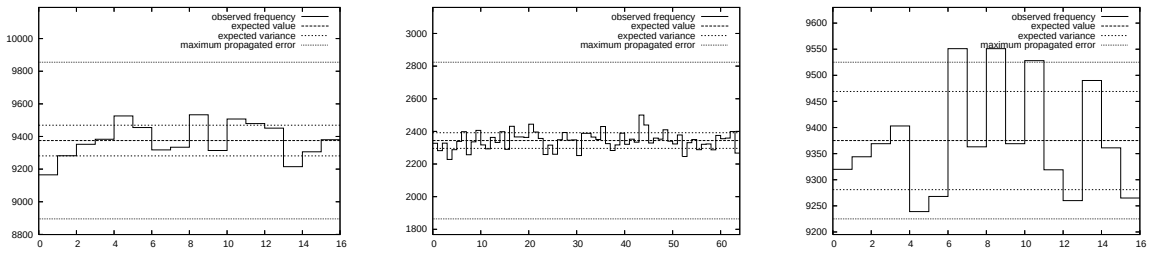
Q is said to have a chi-square distribution with K degree of freedom. In the Pearson chi-square goodness of fitness test [13] the construction is slightly different, but the distribution is exactly the same.

In any case, a closed form for the propagated error for the chi-square distribution has not been found; intuitively, one can think that in tests where the degrees of freedom K is constant (for example, the Matrix Rank Test) increasing the number of bits n is reflected in educing the error introduced by the approximation of all X_i variables with normal variables and thus in a smaller propagated error. On the other hand, in test when increasing n is reflected in increasing K, the error introduced by the approximation of all X_i variables with normal variables is constant, and from this point of view, no reduction of the propagated error is expected.



(a) $p = 0.000000$ (b) $p = 0.000000$ (c) $p = 0.913093$

Figure 4.4: Comparison between expected deviation and measured deviation in the distribution of $N = 150,000$ p-values in the interval $[0, 1]$ for the Matrix Test in the cases: (a) $n = 10^6$, k=16; (b) $n = 10^6$, k=64; (c) $n = 10^7$, k=16.



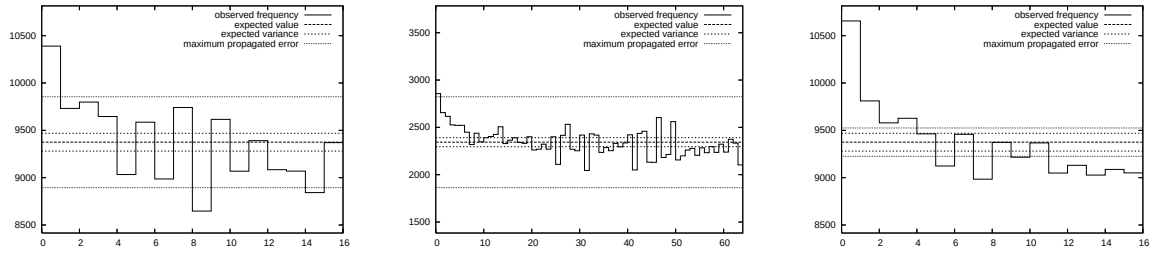
(a) $p = 0.255425$ (b) $p = 0.134417$ (c) $p = 0.366536$

Figure 4.5: Comparison between expected deviation and measured deviation in the distribution of $N = 150,000$ p-values in the interval $[0, 1]$ for the Block Frequency Test in the cases: (a) $n = 10^6$, k=16; (b) $n = 10^6$, k=64; (c) $n = 10^7$, k=16.

However these cases require more investigation. The observed errors for the Matrix Rank Test and for the Block Frequency Test are reported in Figure 4.4 and Figure 4.5. To allow a comparison with the two previous cases, the maximum propagated error for the Frequency and Runs Tests is indicated also in this case.

The case of the Frequency Test must be considered apart. Like the Frequency Test and the Runs Test, the Frequency Test uses a binomial distribution and considers a normal asymptotic distribution. However the observed error can be seen in Figure 4.6. It is evident that errors different from the simple asymptotic approximation make the distribution of p-values not uniform, as low-value p-values are much more common then what expected. Investigating the error propagation in this case does not help in the effort of increasing the test reliability.

In conclusion, the propagated error is typically dependent on n, that is the number of bits in the analyzed sequences; for a reliable second-level test, this error should be smaller, or at least, approximately equal to the random vari-



(a) $p = 0.000000$ (b) $p = 0.000000$ (c) $p = 0.000000$

Figure 4.6: Comparison between expected deviation and measured deviation in the distribution of $N = 150,000$ p-values in the interval $[0, 1]$ for the Spectral in the cases: (a) $n = 10^6$, k=16; (b) $n = 10^6$, k=64; (c) $n = 10^7$, k=16.

ance, which depends on the number of analyzed sequences N. In this case, the propagated error can be confused with a random probabilistic error, and does not affect the results of the test. Based on the analysis on the frequency test and with $n = 10^6$ as suggested by NIST, and $k = 16$, with the relations found above the number of sequences N used in the second-level test should be limited to $N \leq 20,000$.

4.5 Conclusion

In this chapter the concept of statistical tests for randomness has been introduced, and an overview on the most used tests is provided. Additionally, a more reliable test method called *second level test*, already proposed by the NIST, has been analyzed in detail. In particular, an estimation of the maximum error introduced by approximations in the basic test, and of the effect of the propagation of this error to a second level test, has been made in simplest cases. Starting from this estimation, an upper bound limit on the applicability of a second level test has been found.

Chapter 5

Test Results

IN THIS LAST chapter about random numbers, some of the results of statistical tests on the designed prototypes are presented. Both prototypes have been intensively tested, and results compared with other two commercial solutions: the RNG included in the PadLock core of the new microprocessors produced by VIA Technologies, Inc; and the quantic RNG developed by the swiss idQuantique SA. Both these RNGs are described in Appendix B.

Of course only the most significative results are here reported; the standard procedure used for the test is the procedure described in Chapter 4, Section 4.4, i.e. a test on the uniform distribution of 10,000 p-values obtained from the same number of generated sequences.

5.1 Estimated Entropy of the ADC-based RNG

As a first, preliminary test, an evaluation of entropy as described in Chapter 3 was performed. Even if this test is not a rigorous way to test a RNG, it can be very useful to compare the RNG results at different working speeds.

For this test, a sequence of 64 Mbits has been acquired at various speed; the entropy of the string has been estimated in the simplest possible way: the sequence was divided into contiguous non overlapping strings of k bits, and the frequency ν_i of each possible 2^k string have been computed. According to the notation used Section 3.1, the k order entropy has been computed and normalized as:

$$h = \frac{-\sum_{i=0}^{2^k-1} \nu_i \log_2 \nu_i}{k}$$

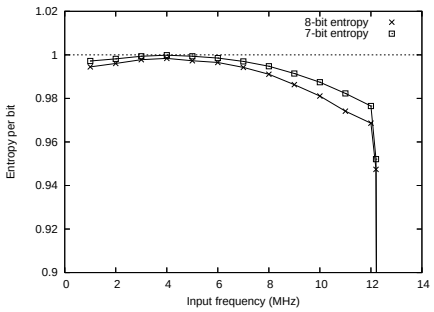


Figure 5.1: Estimated 7-bits and 8-bits entropy for the 0.35 μm two-stages pipeline at different frequencies.

Results for $k = 7$ and $k = 8$ for the two-stages pipeline of the 0.35 μm prototype are reported in Figure 5.1.

Two main aspects can be noticed. The first one is that the two entropy curves are slightly, but constantly, different. The 7 bit entropy is always higher then the 8 bit, thus indicating that in the stream there is a higher correlation between groups of 8 bits with respect to groups of 7 bits. This could be explained with the internal parallelism. The pipeline can be schematized as two different circuits working in parallel; any imperfection in the realization of the pipeline is transferred in equivalent imperfections in the the parallel circuits model. Briefly speaking, the pipeline (exactly as the parallel circuit) elaborates two autonomous and independent streams of bit, every one of them however presents an autocorrelation due to the above imperfections. This will be evident analyzing the sequence of bits two by two (yet, if in a group of two bits each of them comes from a different stream), and consequently, also in group of four, six, or eight bits. The other interesting aspect is that, as expected the entropy decrease at high frequency, but it decrease at low frequencies too. This is effectively unexpected and not easy to explain. Yet, switched capacitors circuits (like any dynamical circuits) have problems at very low speed, but this is not the case, since performances are expected to become lower at frequencies orders of magnitude smaller than the observed ones. However, it can be interesting notice the presence of a peak in the performance of the circuit around 3-4 MHz.

Results for the eight-stages pipeline are reported in Figure 5.2. Actually, being the same circuit as above, one could expect a very similar behavior. It can be noticed a larger difference in the two curves, and also that the presence in the peak is evident only in the 8 bits entropy. This peak is however exactly in the same position as in the two-stages pipeline.

For what concerns the 180 nm RNG, its behavior can be observed in Figure

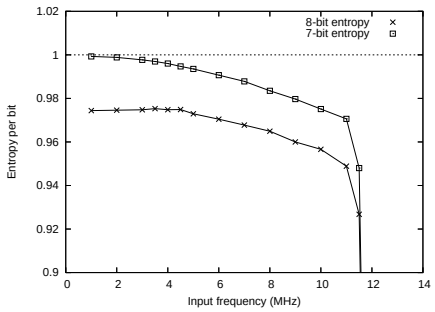


Figure 5.2: Estimated 7-bits and 8-bits entropy for the 0.35 μm eight-stages pipeline at different frequencies.

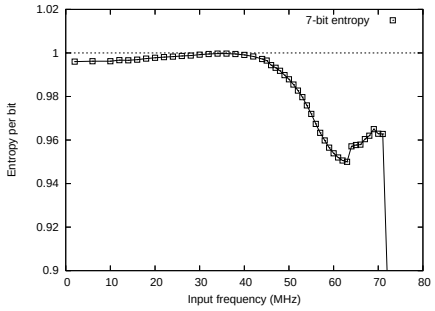


Figure 5.3: Estimated 7-bits entropy for the 180 nm four-stages pipeline at different frequencies.

5.3. As already noted, the circuit was intended to work with a throughput of about 47 Mbit/s, i.e. an input frequency of about $f_{in} = 24$ MHz, but designed to work a much higher frequencies. Indeed, the circuit has a peak in the performances around 35 MHz (i.e. with a throughput of 70 Mbit/s) and has a quite good behavior up to $f_{in} = 50$ MHz. For higher frequencies, the entropy is far from optimal.

5.2 Result of the QSR post-processing

The QSR post-processing proposed in 3.6 has a number of different parameters, which are the length a, b, c and d of the four shift register adopted, with $a, c \geq 1$ and $b, d \geq 2$.

It is obvious that the performances of the stage depends on the parameters; it is not obvious that the more complex the stage (i.e. the higher the total memory of the circuit, that is the total number of flip flops present), the higher the performances. In order to test it, this post-processing stage has been applied to the eight-stages 0.35 μm prototype, and results have been compared with results obtained from a xor post-processing. Furthermore, the test was

repeated on data acquired from prototype running at various frequencies, and so generating different quality random stream, according to the Figure 5.2.

In Table 5.1 results for processing data coming from the prototype of the RNG running at the nominal speed of 10 MHz are presented; while in Table 5.2 are reported results from the prototype overclocked at 12 MHz. A number of 10,000 sequences of length equal to 1 Mbits (after the decimation introduced by the post-processing) have been analyzed, and a chi-square goodness of fit test have been performed over 16 bins.

In term of post-processing, the following parameters are used:

- **Xor based post-processing**: this post-processing has been considered with depth equal to 4, 8 and 16 bit, i.e. with a decimation rate equal to 1:4, 1:8 and 1:16.
- **Shift-register based post-processing**: 5 possible architectures are considered, with different values of the four parameters a, b, c and d, corresponding to a complexity of the stage ranging from 6 to 14 flip-flops.

In both case it is possible to notice that increasing the complexity of the system is reflected in a better yield of the tests (i. e. in a better quality of the random stream). In the first case increasing the complexity means increasing the length of the string of which we compute the parity bit; this implies an increment of the decimation rate, i.e. a reduction of the speed of the circuit. While a depth of 4 is enough for passing all tests when the circuit is working at the nominal speed, the depth has to be increased up to 16 for passing all tests at the overclocked speed of 12 MHz.

In the second case, increasing the complexity means increasing the number of flip-flops, i.e. increasing the hardware cost (both area and power consumption). However, there is no payload in terms of decimation. For passing all tests (neglecting the Frequency Test) at the nominal speed, a minimum complexity of 10 FFs is required. For the circuit overclocked at 12 MHz, the minimum required complexity is 12 FFs.

5.3 SP 800-22 Test Results

Results from testing the eight-stages pipeline of the 0.35 μm prototype with different post-processing stages are shown in Table 5.3 and Table 5.4, for, respectively, the optimal working speed $f_{in} = 3$ MHz and the nominal working speed $f_{in} = 10$ MHz. The standard post-processing functions described in Chapter 3 have been used, including the the two IIR filters (referred as NLSR

and QSR) described in Section 3.6. For the QSR post-processing, a complexity of 10 FFs have been considered. Additionally, the SHA hash function has been taken into account, with a conversion 20 to 20 bytes (i.e. no decimation) and 32 to 20 bytes.

It is possible to notice that without any post-processing, many tests are not passed; also the von Neumann post-processing seems inadequate fore this generator. If the xor-2 is effective for many tests, especially at low speed, an xor-4 post processing, an IIR filter or the SHA function is necessary in both cases to pass all tests.

Results from testing of the 180 nm prototype, instead, are shown in Table 5.5 and Table 5.6, for, respectively, the optimal working speed $f_{in} = 35$ MHz and the maximum working speed $f_{in} = 50$ MHz.

According to tables results, this prototype produces a sensible worse stream than the previous one in terms of quality. Neglecting few tests where the obtained p-value is very near to the lever of significance, a depth of 8 for the xor post-processing is necessary for passing all tests at the lower speed, while this depth is increased to 12 at the higher speed. Tests are also not passed when considering the IIR filters of Section 3.6; this is true both for the NLSR and for the QSR, which has been considered with two different complexity, respectively equal to 10 FFs and 14 FFs.

Instead, Table 5.7 and Table 5.8 report results of testing for the two commercial RNGs considered: the VIA PadLock generator and the Quantis generator described in Appendix B. Data from first generator were generated at the speed of about 1 Mbit/s, while the Quantis generated data at a speed of 4 Mbit/s.

Unexpectedly, it seems that both generators cannot pass the second level uniformity test without any post-processing. Even if the quality appears very good, since nearly all tests are passed, few basic tests like the Block Frequency Test and the Runs Test are not passed. A p-value equal to $p = 0.000000$ ensures that this does not represent a Type I error. More unexpectedly, neither with a xor-2 post-processing these generator can pass the second level test. In this case, the critical tests are the Frequency Test and the Cumulative Sums Test. Also this time a p-value equal to $p = 0.000000$ clearly indicates that this is not the case of a Type I error.

For what said in Chapter 3 regarding the post processing, a comparison of the test results can be done only considering the entropy per second generated by the different RNGs. Also, since all IIR filters do not increase the entropy of the stream, but just to make harder for a test to discover the non randomness of a stream, here only the different xor post-processings are considered.

The eight-stages pipeline of the 0.35 μm running at $f_{in} = 10$ MHz, generates random data at a throughput of 40 Mbit/s; however an xor-4 post processing is needed in order to consider this stream a true random stream. This ends in an entropy per second equal to 10 Mbit/s.

Conversely, the 180 nm prototype require much more decimation, that is 1 to 8 when running at low speed, and 1 to 12 when running at the higher speed. In both cases, the entropy can be estimated in about 8 Mbit/s, that is lower than the first prototype.

For what concerns the two commercial generators, like the 0.35 μm prototype, they need an xor-4 post processing to eliminate all non-idealities. This ends in an entropy equal to 1 Mbit/s for the Quantis generator, and 250 kbit/s for the PadLock generator.

As a conclusion, it possible to affirm that the 0.35 μm prototype outperforms all other generators. Also, the two commercial generators result in an entropy per time unit that is one order of magnitude smaller than the entropy generated by the 0.35 μm ADC-based chaotic generator.

5.4 Conclusion

This chapter provides testing results for the two RNG prototypes presented in Chapter 2, as well as a comparison with two other commercial random generators. In particular, the prototype designed in AMS 0.35 μm CMOS technology presents both a high quality signal, and a high working speed; in terms of entropy per second, the designed prototype outperforms the two commercial generators by one order of magnitude.

	chi-square goodness of fit test on the uniform distribution of 10,000 p-values over 16 bins							
	for different XOR depths			*for different lengths a-b-c-d of the four shift registers*				
SP800-22 test	xor-4	xor-8	xor-16	1-2-1-2 (6)	2-3-1-2 (8)	3-4-1-2 (10)	4-5-1-2 (12)	5-6-1-2 (14)
Frequency	0.174634	0.083777	0.319791	0.114550	0.633237	0.450953	0.968870	0.352940
Block Frequency	0.087331	0.428987	0.095986	0.538390	0.835267	0.164045	0.898429	0.833756
Cumulative Sums	0.769052	0.184878	0.185059	0.042983	0.584243	0.524990	0.248388	0.472781
Runs	0.580840	0.468378	0.152085	0.938443	0.570979	0.257944	0.674979	0.600383
Longest Run of Ones	0.728101	0.018813	0.537447	0.865959	0.974240	0.303152	0.163011	0.103492
Matrix Rank	0.711307	0.021717	0.229438	0.662084	0.246619	0.642014	0.208219	0.644383
Spectral (DFT)	0.054125	0.072093	**0.001466**	**0.000062**	0.741774	0.051472	0.040307	**0.001187**
NOT Matching	0.219398	0.506058	0.517269	0.364886	0.213503	0.182168	0.853623	0.924495
OT Matching	0.094617	0.046303	0.236040	0.034607	0.023928	0.406271	0.536505	0.062333
Universal	0.138028	0.881566	0.752089	0.069235	0.040840	0.020770	0.093492	0.039211
Approx. Entropy	0.584583	0.285668	0.804021	**0.000000**	**0.000000**	0.589894	0.187602	0.738676
Random Excursion	0.276340	0.603662	0.174795	0.612689	0.880022	0.091926	0.029166	0.046329
Random Exc. Variant	0.029264	0.065300	0.802775	0.455994	0.254350	0.081032	0.243570	0.375608
Serial	0.923711	0.316323	0.618969	**0.000165**	**0.000000**	0.756654	0.116980	0.737345
Linear Complexity	0.556731	0.962670	0.287252	0.591324	0.299723	0.230033	0.417402	0.134919

Table 5.1: Results of randomness test for the ADC-based RNG running at $f_{in} = 10$ MHz.

	chi-square goodness of fit test on the uniform distribution of 10,000 p-values over 16 bins							
	for different XOR depths			*for different lengths a-b-c-d of the four shift registers*				
SP800-22 test	xor-4	xor-8	xor-16	1-2-1-2 (6)	2-3-1-2 (8)	3-4-1-2 (10)	4-5-1-2 (12)	5-6-1-2 (14)
Frequency	**0.000000**	**0.000000**	0.725135	0.125548	0.760909	0.368514	0.023796	0.939296
Block Frequency	**0.000000**	0.320856	0.310649	0.164275	0.618969	0.289688	0.618255	0.454302
Cumulative Sums	**0.000000**	**0.000000**	0.189158	0.058652	0.213209	0.596568	0.412461	0.667721
Runs	**0.000000**	0.697706	0.361741	0.201256	0.306538	0.754919	0.846896	0.209883
Longest Run of Ones	**0.000000**	0.642083	0.034446	**0.000000**	**0.000000**	**0.002800**	0.299035	0.014745
Matrix Rank	0.599102	0.992495	0.500373	0.029229	0.416673	**0.001415**	0.695188	0.121608
Spectral (DFT)	**0.000000**	0.198106	0.461719	**0.000024**	0.021044	**0.000226**	**0.005268**	**0.000502**
NOT Matching	**0.000000**	0.855524	0.342511	0.604197	0.942154	0.328036	0.809387	0.998892
OT Matching	**0.000000**	0.515503	0.473496	**0.000000**	**0.000000**	0.310762	0.079343	0.014800
Universal	**0.000000**	0.281840	0.695455	0.366093	**0.000209**	0.120616	0.010764	0.058555
Approx. Entropy	**0.000000**	0.253310	0.180118	0.000000	**0.000000**	**0.000000**	0.191596	0.193551
Random Excursion	N/A	0.410255	0.225801	0.444707	0.743415	0.062891	0.326281	0.315667
Random Exc. Variant	N/A	0.198540	0.626131	0.345316	0.520963	0.930836	0.942527	0.128987
Serial	0.297749	0.344407	0.755371	**0.000000**	**0.000000**	**0.000000**	0.162782	0.536740
Linear Complexity	0.296882	0.487620	0.767979	0.606104	0.785783	0.292314	0.112330	0.126853

Table 5.2: Results of randomness test for the ADC-based RNG running at $f_{in} = 12$ MHz.

chi-square goodness of fit test on the uniform distribution of 10,000 p-values over 16 bins

SP800-22 test	none	neumann	xor-2	xor-4	xor-8	qsr (10)	nlsr	sha-20	sha-32
Frequency	**0.000000**	**0.000000**	**0.000000**	0.697263	0.944664	0.110314	0.999274	0.959239	0.872575
Block Frequency	**0.000000**	**0.000000**	**0.000001**	0.995665	0.558255	0.105967	0.678715	0.340257	0.642488
Cumulative Sums	**0.000000**	**0.000000**	**0.000000**	0.538861	0.637747	0.124714	0.826809	0.305856	0.984634
Runs	**0.000000**	**0.000000**	0.820313	0.952586	0.912023	0.118212	0.360680	0.365289	0.455420
Longest Run of 1s	0.162097	0.024229	0.247447	0.065305	0.806413	0.491758	0.337560	0.448281	0.987497
Matrix Rank	0.312591	0.296370	0.447836	0.029359	0.293545	0.092769	0.101922	0.010154	0.035679
Spectral (DFT)	**0.000000**	**0.000014**	0.064616	**0.005823**	**0.000299**	0.010077	**0.000004**	**0.007401**	0.010795
NOT Matching	**0.000000**	**0.000000**	0.646986	0.475509	0.771238	0.901436	0.876942	0.953487	0.767195
OT Matching	0.041005	0.643909	**0.000000**	0.747277	0.096880	0.012965	0.024316	0.545706	0.024338
Universal	**0.000000**	**0.000000**	0.446947	0.047678	0.027861	0.116718	0.094512	0.688017	0.140933
Approx. Entropy	**0.000000**	**0.000000**	0.063051	0.353335	0.882484	0.631812	0.468248	0.064405	0.102469
Random Excursion	0.678818	N/A	0.204341	0.383543	0.608684	0.167508	0.181886	0.742872	0.926788
Random Exc. Var.	0.432355	N/A	0.296772	0.723727	0.551584	0.164290	0.641419	0.333552	0.206864
Serial	**0.000000**	0.763127	0.346662	0.800813	0.501454	0.060917	0.368110	0.167294	0.339678
Linear Complexity	0.500065	0.520073	0.609441	0.488310	0.654782	0.504698	0.740669	0.127604	0.407123

Table 5.3: Results of randomness test for the 0.35 μm eight-stages pipeline running at $f_{in} = 3$ MHz.

chi-square goodness of fit test on the uniform distribution of 10,000 p-values over 16 bins

SP800-22 test	none	neumann	xor-2	xor-4	xor-8	qsr (10)	nlsr	sha-20	sha-32
Frequency	**0.000000**	0.000000	**0.000000**	0.174634	0.623253	0.450953	0.173703	0.174306	0.227661
Block Frequency	**0.000000**	0.000000	**0.000000**	0.087331	0.354323	0.164045	0.276117	0.064405	0.567990
Cumulative Sums	**0.000000**	0.000000	**0.000000**	0.769052	0.410749	0.524990	0.012204	0.597999	0.995600
Runs	**0.000000**	0.000000	**0.000000**	0.580840	0.138401	0.257944	0.056182	0.757521	0.143605
Longest Run of 1s	**0.000000**	0.000000	**0.000000**	0.728101	0.056885	0.303152	0.203693	0.141853	0.534386
Matrix Rank	0.182543	0.472554	0.077967	0.711307	**0.007076**	0.642014	0.503539	**0.001537**	0.023561
Spectral (DFT)	**0.000000**	0.000000	**0.000026**	0.054125	0.055902	0.051472	**0.000010**	0.083862	0.142779
NOT Matching	**0.000000**	0.000000	**0.000000**	0.219398	0.528744	0.182168	0.145479	0.366698	0.246820
OT Matching	**0.000000**	0.000000	**0.000000**	0.094617	0.038176	0.406271	0.094879	0.011077	0.430640
Universal	**0.000000**	0.000000	**0.000000**	0.138028	0.971074	0.020770	**0.007665**	0.026409	0.002475
Approx. Entropy	**0.000000**	0.000000	**0.000000**	0.584583	0.338329	0.589894	0.321094	0.493139	0.714427
Random Excursion	N/A	N/A	0.796975	0.276340	0.461638	0.091926	0.972120	0.494889	0.984633
Random Exc. Var.	N/A	N/A	0.628486	0.029264	0.194737	0.081032	**0.004918**	0.468106	0.303025
Serial	**0.000000**	0.000014	0.744200	0.923711	0.429767	0.756654	0.136403	0.376441	0.772086
Linear Complexity	0.469606	0.776520	0.166594	0.556731	0.965536	0.230033	0.887126	0.553988	0.762482

Table 5.4: Results of randomness test for the 0.35 μm eight-stages pipeline running at $f_{in} = 10$ MHz.

chi-square goodness of fit test on the uniform distribution of 10,000 p-values over 16 bins

SP800-22 test	xor-4	xor-8	xor-12	qsr (10)	qsr (14)	nlsr	sha-20	sha-32
Frequency	**0.000000**	**0.007434**	0.103019	0.274430	**0.000004**	**0.000323**	0.373585	0.020713
Block Frequency	**0.000000**	0.219205	0.626821	0.256977	**0.000830**	**0.003277**	0.724153	0.024492
Cumulative Sums	**0.000000**	0.027385	0.195390	**0.000800**	0.898429	**0.000081**	0.986653	0.025026
Runs	**0.000000**	**0.007018**	0.667955	0.014855	0.020903	0.086610	0.935175	0.518904
Longest Run of Ones	0.051992	0.020829	0.934953	**0.002120**	**0.000721**	**0.003781**	0.896543	0.366093
Matrix Rank	**0.000810**	**0.003676**	0.169174	**0.002400**	0.492449	**0.001695**	**0.002755**	0.708961
Spectral (DFT)	**0.000000**	0.056611	0.000802	**0.000000**	0.392776	**0.000426**	0.032040	**0.000149**
NOT Matching	**0.000000**	0.558185	0.506323	0.019217	0.267255	**0.000509**	0.267917	0.061369
OT Matching	**0.000000**	0.144320	0.141137	**0.000000**	0.006052	**0.000001**	0.333159	0.061470
Universal	0.067354	0.448111	0.027102	0.051300	0.249806	**0.003058**	0.013835	0.275272
Approx. Entropy	**0.000000**	0.011708	0.626821	**0.000000**	0.135314	0.024229	0.092697	**0.001583**
Random Excursion	0.017488	0.566293	0.102463	**0.003259**	0.024116	0.070335	0.861232	0.866288
Random Exc. Variant	**0.000418**	0.078020	0.277339	**0.000685**	0.084428	**0.000364**	0.802326	0.890521
Serial	0.034880	0.141890	0.637984	**0.001534**	0.349205	0.346662	0.550673	**0.000336**
Linear Complexity	**0.000072**	0.017582	0.855020	0.101532	0.154925	0.037326	0.854672	0.299570

Table 5.5: Results of randomness test for the 180 nm four-stages pipeline running at 35 MHz.

chi-square goodness of fit test on the uniform distribution of 10,000 p-values over 16 bins

SP800-22 test	xor-4	xor-8	xor-12	qsr (10)	qsr (14)	nlsr	sha-20	sha-32
Frequency	**0.000000**	**0.000000**	0.020728	0.039279	0.744640	0.044097	0.787771	0.262326
Block Frequency	**0.000000**	**0.000004**	0.756553	**0.000017**	**0.000000**	**0.000077**	0.479611	0.295839
Cumulative Sums	**0.000000**	**0.000000**	0.291778	0.279341	**0.000232**	0.125362	0.414821	0.080357
Runs	**0.000000**	**0.000002**	0.491969	0.017511	0.017414	**0.003536**	0.427368	0.325210
Longest Run of Ones	**0.000000**	**0.001108**	0.936169	**0.000030**	0.273085	0.350186	0.088461	0.892115
Matrix Rank	0.032550	**0.008522**	0.013867	**0.001001**	0.011761	0.148966	0.318491	0.136105
Spectral (DFT)	**0.000000**	**0.000000**	**0.000002**	**0.000012**	0.006484	**0.002832**	**0.000492**	0.013224
NOT Matching	**0.000000**	0.248231	0.270001	0.327658	0.132476	0.015023	0.506787	0.423672
OT Matching	**0.000000**	**0.002325**	0.022503	**0.003270**	**0.001485**	**0.000001**	0.488999	0.021895
Universal	**0.000062**	0.029359	0.135069	**0.000069**	**0.000000**	0.519839	0.235434	0.027442
Approx. Entropy	**0.000000**	**0.000000**	0.127165	**0.000022**	**0.000373**	0.206840	0.275104	0.022786
Random Excursion	**0.000003**	0.408774	0.083227	0.026350	**0.002775**	0.055225	0.835904	0.813223
Random Exc. Variant	**0.002415**	0.014956	0.371283	**0.000993**	0.153910	0.457200	0.296293	0.910346
Serial	0.133156	0.410108	0.442457	**0.000000**	0.049605	0.272080	0.239851	0.022438
Linear Complexity	0.486933	0.026670	0.724760	0.464859	0.191596	0.010734	0.841119	0.784875

Table 5.6: Results of randomness test for the 180 nm four-stages pipeline running at 50 MHz.

	chi-square goodness of fit test on the uniform distribution of 10,000 p-values over 16 bins			
SP800-22 test	none	xor-2	xor-4	Neumann
Frequency	0.510275	**0.000000**	0.582507	0.187985
Block Frequency	**0.002671**	0.166477	0.268746	0.790446
Cumulative Sums	0.383633	**0.000000**	0.728870	0.358885
Runs	**0.000000**	0.283278	0.418696	**0.004468**
Longest Run of Ones	0.078153	0.633950	0.231524	0.344908
Matrix Rank	0.213805	0.027344	0.144852	0.682907
Spectral (DFT)	**0.004870**	0.012112	0.092625	**0.000023**
NOT Matching	0.850278	0.650532	0.069516	0.925101
OT Matching	**0.003469**	0.146003	**0.001056**	0.394450
Universal	**0.006685**	0.638221	0.031536	0.603482
Approx. Entropy	0.499371	0.266429	0.317194	0.341997
Random Excursion	0.121354	0.091008	0.845354	0.764485
Random Exc. Variant	0.203023	0.828277	0.910116	0.198517
Serial	0.749906	0.509344	0.955692	0.415036
Linear Complexity	0.534621	0.851867	0.398652	0.067573

Table 5.7: Results of randomness test for the VIA PadLock generator.

	chi-square goodness of fit test on the uniform distribution of 10,000 p-values over 16 bins			
SP800-22 test	none	xor-2	xor-4	Neumann
Frequency	0.960217	**0.000000**	0.892264	0.770601
Block Frequency	**0.000001**	0.100371	0.133547	**0.000000**
Cumulative Sums	0.125084	**0.000000**	0.649114	0.323897
Runs	**0.000000**	0.874849	0.235737	**0.000000**
Longest Run of Ones	0.559916	0.489918	0.115586	0.961811
Matrix Rank	0.205467	**0.000848**	0.027442	0.021287
Spectral (DFT)	**0.000252**	0.029152	**0.000023**	0.033505
NOT Matching	0.324084	0.122515	0.090065	0.902427
OT Matching	0.043020	0.001064	0.004735	**0.002606**
Universal	0.294955	0.088047	0.344519	**0.008705**
Approx. Entropy	0.208496	0.822234	0.464182	0.859516
Random Excursion	0.150665	0.802855	0.823094	0.682399
Random Exc. Variant	0.166545	0.111642	**0.003758**	0.852521
Serial	0.474372	0.443621	0.073312	0.704157
Linear Complexity	0.621350	0.679880	0.394450	0.360281

Table 5.8: Results of randomness test for the Quantis generator.

Chapter 6

Application of RNG: EMI Reduction

THE DESIGN OF electromagnetic compatible (EMC) timing signals in integrated digital or mixed-signal circuits is of great practical concern. It is worth noticing that common solutions to increase system EMC, based on *a-posteriori* methodologies, such the adoption of filters, shielded cables and filtered connectors cannot be employed in integrated technology; hence, *design-time* solutions should be adopted [87], assuring that the implemented electronic equipment *generates* electromagnetic interference with power spectral density as flat as possible, so that its integral within any frequency range (and therefore in the bandwidth of any unintentional receiver) is as low as possible.

This point of view is perfectly coherent with FCC and CE regulations [86] that link compliance with the ability of fitting the interfering power spectrum within a prescribed *mask*. Regrettably, clock signals are most likely to fail such a compliance, due to their sharp edges and their periodic nature, which concentrate power at multiples of their frequency.

The key idea for reducing peak power density in clock signals is a frequency modulation, producing a clock signal with edges which are slightly *delayed* or *anticipated* to avoid perfect periodicity. Of course, it is assumed that the maximum frequency deviation is compatible with the devices depending on the clock for proper operation. It can be intuitively accepted that the efficiency of these methods critically depends on the statistical properties of the modulating signal.

In classical literature [85, 88, 90, 91] emphasis is on continuous-valued frequency modulation with large modulation indexes (*slow-modulation*) for which

an analytical estimation of the spectrum profile can be provided. However more recently it has been introduced a *binary fast random* modulation [92], showing a *better* flattening properties as long as it is operated at proper modulation indexes, derived by means of numerical optimization. In this modulation the modulating signal is a Pulse Amplitude Modulated (PAM) sequence that can assume only two values (typically, -1 and +1) each with probability $p = 1/2$. In this case the instantaneous output frequency can assume only the two values $f_0 - \Delta f$ and $f_0 + \Delta f$, where f_0 is the carrier frequency and Δf is the maximum frequency deviation. Since both f_0 and Δf are typically fixed by the application, the only degree of freedom is given by the frequency $f_m = 1/T$ of the PAM signal. More commonly, this degree of freedom is expressed through the modulation index $m = \Delta f / f_m$. This index is used to flatten the power spectrum in the desired interval. A semi-analytical optimization shows that the lowest peak on the fundamental tone is achieved by setting $m \simeq 0.318$ [83, 92]. Such value for m however is not optimized for higher harmonics, that still feature peaks. Yet the power content of these harmonics is much lower than that of the fundamental, and so are the corresponding peaks.

In this chapter two Spread-Spectrum Clock Generators (SSCGs) designed to implement a fast binary modulation are described. For both, the SSCG structure is based on a PLL with few modifications to achieve a binary frequency modulator; an ADC-based random number generator is used to generate the random driving signal.

6.1 Generation of Spread-Spectrum Clock Signals

Consider clock signal $s(t)$ as the result of a frequency modulation:

$$s(t) = \text{sgn}\left(\cos\left(2\pi f_0 t + 2\pi\Delta f \int_{-\infty}^{t} \xi(\tau)d\tau\right)\right)$$

where f_0 indicates the carrier frequency, Δf the frequency deviation and $\xi(t)$ the driving PAM signal:

$$\xi(t) = \sum_{k=-\infty}^{+\infty} x_k g\left(t - kT\right) \tag{6.1}$$

given that $g\left(t\right)$ is a unit pulse of duration T and that x_k are *random* values (being $\rho\left(x\right)$ their probability density function) constituting the modulating sequence, belonging to the interval $[-1, 1]$.

It is possible to prove [85] that the contribution of each harmonic in the power spectrum can be analytically described by its corresponding low pass

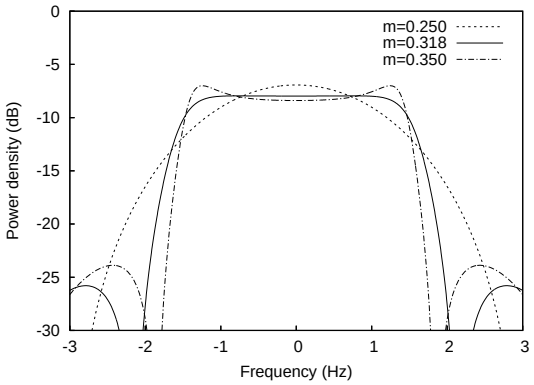


Figure 6.1: Normalized ($\Delta f = 1$) low-pass equivalent for PSD in the binary frequency modulation for value of the modulation index around m_{opt}

equivalent:

$$\Phi_{\tilde{S}\tilde{S}}(f) = E_x\left[\mathcal{K}_1(x,f)\right] + \text{Re}\left\{\frac{E_x^2\left[\mathcal{K}_2(x,f)\right]}{1 - E_x[\mathcal{K}_3(x,f)]}\right\}$$

where:

$$\mathcal{K}_1(x,f) = \frac{1}{2}T\,\text{sinc}^2(\pi T(f - \Delta f x))$$
$$\mathcal{K}_2(x,f) = j\frac{e^{-j2\pi T(f-\Delta f x)} - 1}{2\pi\sqrt{T}(f - \Delta f x)}$$
$$\mathcal{K}_3(x,f) = e^{-j2\pi T(f-\Delta f x)}$$

where T is the pulse width in (6.1) and Δf is the frequency deviation for the considered harmonic, which is proportional to the harmonic number (and equal to the modulation Δf only for the fundamental tone). In the particular case of binary modulation, it is:

$$\rho(x) = \frac{1}{2}\delta(x+1) + \frac{1}{2}\delta(x-1) \tag{6.2}$$

while statistical independence of $\{x_k\}$ implies:

$$E_x[f(x)] = \int f(x)\rho(x)dx \tag{6.3}$$

Given the exact expression for $\Phi_{\tilde{s}\tilde{s}}(f)$ and substituting (6.2) and (6.3), by means of numerical optimization it has been found that peaks in the PSD are minimized for the value of the *modulation index* $m = \Delta f\, T = m_{\text{opt}} \simeq 0.318$. Lower values of m cause the PSD to increase around 0, while higher values increase it around $f = \pm\Delta f$ (Figure 6.1).

Each harmonic is described by a different modulation index (m is proportional to the harmonic number), so this optimization can be achieved only on

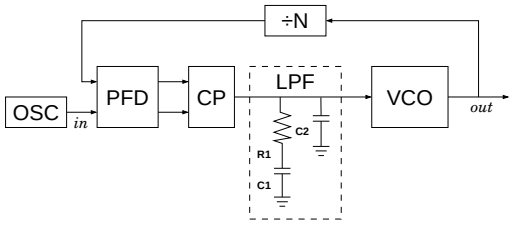


Figure 6.2: Block diagram of the PLL modified to achieve a frequency modulator.

one single harmonic. Since the power content of the fundamental tone is much higher than all other harmonics, and so are the corresponding peaks, best results in overall peak reduction are achieved when the modulation index is optimized for the fundamental tone, i.e. $m = m_{\text{opt}}$. Such a reduction is the best reduction with respect to all other known modulations [92].

6.2 Description of the 0.35 μm SSCG prototype

The first SSCG prototype here described has been implemented in 0.35μm AMS technology. The SSCG structure is based on a PLL with few modifications to achieve a frequency modulator. The PLL architecture is chosen to externally set the center-spread frequency, for example with a high-precision quartz oscillator. As driving signal, a PAM signal cominging from an ADC-based RNG composed by two stages has been used. The center-spread frequency has been set to the nominal value of $f_0 = 100$ MHz and the driving signal PAM frequency equal to $f_m = 10$ Mbit/s. The frequency deviation Δf is set to $\Delta f = 3.18$ MHz to achieve the optimal modulation index m value.

The block diagram of the modulator is shown in Figure 6.2: neglecting the driving signal, this scheme is the same of a conventional PLL-based clock generator. It includes a reference clock oscillator, a phase-frequency detector (PFD), a charge pump (CP), a second-order passive low-pass filter (LPF), a voltage-controlled oscillator (VCO) and a divider by N on the feedback path. Its purpose is to set the output frequency $f_{\text{out}} = N f_{\text{in}}$, where f_{in} is the frequency of the reference clock. The closed-loop transfer function $f_{\text{out}}(s)/f_{\text{in}}(s)$ has a *low-pass* nature, with cut-off frequency ω_n.

The conventional scheme is indeed modified with the addition of a driving signal between LPF output and VCO input. If we suppose that this signal is high frequency with respect to ω_n, we can notice that it drives the VCO as in an open-loop system, since it cannot pass through the loop composed by the divider, the PFD, the CP and the LPF, due to the low-pass nature of the loop.

This is evident considering the standard linearized PLL analysis [84] in the Laplace domain. The PFD and the CP can be modeled as a single component, which looks at the phase differences $\Delta\phi$ between the two inputs of the PDF, and gives a serie of high frequency pulses of intensity $\pm I_{\mathrm{pump}}$ and duty-cycle proportional to the phase difference; a phase difference of $\Delta\phi = 2\pi$ results in an average output current $\hat{I} = I_{\mathrm{pump}}$, while a phase difference of $\Delta\phi = -2\pi$ results in $\hat{I} = -I_{\mathrm{pump}}$. Analytically:

$$\hat{I} = \frac{I_{\mathrm{pump}}}{2\pi}\Delta\phi = K_1\Delta\phi$$

Obviously, the the phase difference is bounded in the interval $[-2\pi, 2\pi]$.

Since the LPF a cut-off frequency that is much lower (typically two or more order of magnitude) with respect to the frequency of the pulses coming from the CP (that is the frequency of the two input signals), only the average current $\hat{I}(s)$ can considered at its input:

$$\hat{I}(s) = K_1\Delta\phi(s)$$

Referring to Figure 6.2, the filter output voltage is

$$V_{\mathrm{filter}}(s) = \frac{1+sT_2}{sT_1(1+sT_3)}\hat{I}(s) = K_2\hat{I}(s)$$

with

$$\begin{aligned} T_1 &= \mathrm{C1} + \mathrm{C2} \\ T_2 &= \mathrm{R1}\,\mathrm{C1} \\ T_3 &= \mathrm{R1}\,\frac{\mathrm{C1}\,\mathrm{C2}}{\mathrm{C1}+\mathrm{C2}} \end{aligned}$$

then, neglecting the input signal $\xi(s)$ the VCO converts this voltage into an output frequency

$$f_{\mathrm{out}}(s) = K_{\mathrm{VCO}}\left(V_{\mathrm{filter}} + \xi(s)\right)$$

and, noticing that the phase $\phi(t)$ of a signal is just the integral of the instantaneous frequency $\omega(t)$ in the time domain

$$\phi_{\mathrm{out}}(s) = \frac{\omega_{\mathrm{out}}(s)}{s} = \frac{K_{\mathrm{VCO}}}{s}V_{\mathrm{filter}}(s) = K_3V_{\mathrm{filter}}(s)$$

So the open-loop transfer function $H_0(s)$ can be cast as

$$H_0(s) = K_1K_2K_3 = \frac{I_{\mathrm{pump}}}{2\pi}\frac{1+sT_2}{sT_1(1+sT_3)}\frac{K_{\mathrm{VCO}}}{s}$$

and, with the interposition of the driving signal $\xi(s)$, it is

$$\phi_{\mathrm{out}}(s) = H_0(s)\Delta\phi(s) + K_3\xi(s)$$

Now, in the closed-loop system, it is $\Delta\phi = \phi_{in} - \phi_{out}$; the closed-loop $\omega_{out}(s)/\omega_{in}(s)$ characteristic is

$$H_1(s) = \frac{\omega_{\text{out}}(s)}{\omega_{\text{in}}(s)} = \frac{\phi_{\text{out}}(s)}{\phi_{\text{in}}(s)} = \frac{H_0(s)}{1 + \dfrac{H_0(s)}{N}} \tag{6.4}$$

It is very common considering $\text{C2} \ll \text{C1}$, so $T_3 \simeq 0$ and

$$K_2 \approx \frac{1 + sT_2}{sT_1}$$

Under this assumption, (6.4) can be recast as

$$H_1(s) = N\frac{\omega_n^2(1 + sT_2)}{s^2 + 2\omega_n\zeta s + \omega_n^2}$$

with

$$\omega_n = \sqrt{\frac{I_{pump}K_{VCO}}{2\pi NT_1}}$$
$$\zeta = \frac{\omega_n T_2}{2}$$

This comes the standard form for analyzing a two poles transfer function; setting a dumping factor ζ with a value near to the unity, $H_1(s)$ is a transfer function with a low pass nature, presenting a double pole in ω_n and a zero in $1/T_2$. The transfer function cut-off frequency is ω_n, while the base-band gain is N, as expected.

However, when considering the transfer function between $\omega_{\text{out}}(s)$ and $\xi(s)$

$$H_2(s) = \frac{\omega_{\text{out}}(s)}{\xi(s)} = \frac{1}{s}\frac{\phi_{\text{out}}(s)}{\xi(s)} = s\frac{K_3}{1 + \dfrac{H_0(s)}{N}}$$

and applying the same simplification as above

$$H_2(s) = \frac{s^2/\omega_n^2}{s^2 + 2\omega_n\zeta s + \omega_n^2}$$

This transfer function presents a double zero in the origin, and a double pole at ω_n; this is a high-pass transfer function, with cut-off frequency equal to ω_n

The core block for the modulation is the adder, which is integrated into the VCO. The VCO is essentially composed of a seven-stage ring oscillator, which is followed by a wave-shaping buffer, in order to obtain proper values of logic levels and slew-rate for the output, and is controlled by an input stage, whose purpose is mainly to supply the correct operating current to the ring oscillator, and to decouple it from the other parts of the circuit.

Due to the discrete nature of this signal, a full analog adder is not necessary, thus simplifying the circuit. The additive function is performed by the input

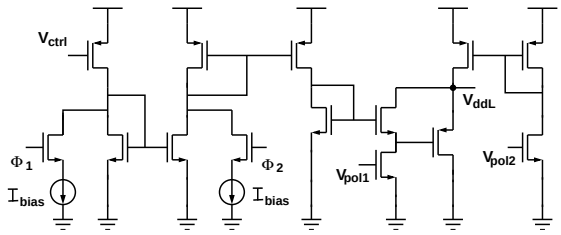


Figure 6.3: Modified input stage of the VCO

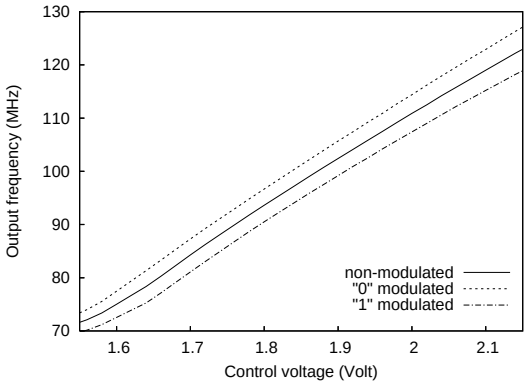


Figure 6.4: Voltage/Frequency characteristic of the VCO in non spread spectrum mode (solid line) and spread spectrum mode (dashed lines).

stage of the VCO (Figure 6.3), through the two pass-transistors driven by Φ_1 and Φ_2, along with the two current sources $I_{\rm bias}$. This circuit is designed to work with $\Phi_1 = \overline{\Phi_2} = \Phi$, where Φ is the signal coming from the random bit generator; however its behavior is more evident considering these two signals separately.

Supposing $\Phi_1 = \Phi_2 = 0$, the circuit acts as a linear voltage amplifier, where $V_{\rm ddL}$ is proportional to $V_{\rm ctrl}$; the obtained VCO $f_{\rm out}/V_{\rm ctrl}$ characteristic is represented by the solid line in Figure 6.4; the voltage/frequency ratio is set to:

$$K_{\rm VCO} = 518{\rm Mrad/s/V}$$

corresponding to $K_{\rm VCO} = 82.5{\rm MHz/V}$. When $\Phi_1 = 1$, the current $I_{\rm bias}$ is subtracted from the current mirror, thus shifting up the $f_{\rm out}/V_{\rm ctrl}$ characteristic. On the contrary, $\Phi_2 = 1$ adds $I_{\rm bias}$ to the current mirror and shifts down the characteristic. The two shifted characteristics are represented by dashed lines in Figure 6.4. The distance between the curves is approximately constant in the range of interest and represents the PLL Δf. Its value depends on $I_{\rm bias}$; furthermore there is an almost linear relationship between Δf and $I_{\rm bias}$, that

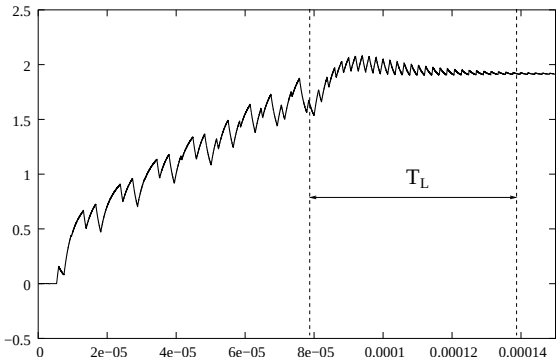


Figure 6.5: Simulation of the *pull-in* and *lock-in* process of the PLL.

is:

$$K_{\Delta f} = 1.106\,\text{Mrad/s}/\mu\text{A}$$

corresponding to $K_{\Delta f} = 0.176\,\text{MHz}/\mu\text{A}$.

In the project design, it has been set $I_{\text{pump}} = 400\mu A$ and $N = 64$. To ensure stability, the (external) filter has been designed with

$$\begin{aligned} C_1 &= 58\text{ nF} \\ C_2 &= 5.8\text{ nF} \\ R_1 &= 370\,\Omega \end{aligned}$$

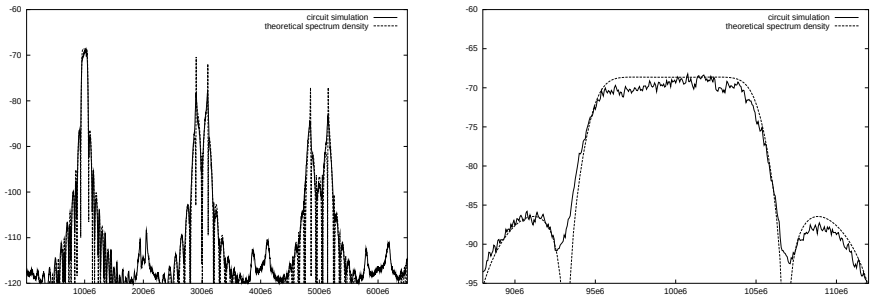
With these values, the closed-loop PLL bandwidth is equal to:

$$\begin{aligned} \omega_n &= \sqrt{\frac{I_{\text{pump}} K_{\text{VCO}}}{2\pi N C_1}} = 94.25\text{krad/s} \\ \zeta &\simeq 1 \end{aligned}$$

thus meaning a cut-off frequency of about 15 KHz, while the zero of $H_1(s)$ is at $1/T_2 = 46.6\text{krad/s}$, i.e. 7.4 KHz. Also, the PLL lock-in time can be estimated to $T_L = 2\pi/\omega_n \simeq 66\ \mu s$.

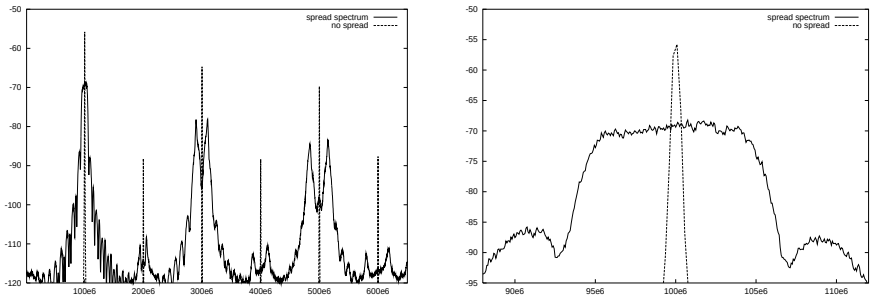
Figure 6.5 shows a simulation of the VCO control voltage during the pull-in process for the PLL without any driving signal. It is possible to notice that the PLL eventually reaches stability; also the time between entering the lock state (i.e. when major oscillations end) and reaching a complete settlement, is almost equal to the estimated lock-in time T_L.

The simulated power spectrum density of the output clock signal can be seen in Figure 6.6. The figures have been obtained from a 1.2 ms simulations and discarding the first 200 μs data, which is a sufficient time, according to the bandwidth of the PLL, to consider all circuit transitories extinguished. The



(a) (b)

Figure 6.6: Comparison between power spectrum density of the output clock obtained from the simulated circuit and the theoretical power spectrum density of the binary modulation, for (a) a wide set of harmonics; and (b) only for the fundamental tone.



(a) (b)

Figure 6.7: Comparison between power spectra density of the modulated and non modulated output clock for (a) a wide set of harmonics; and (b) only for the fundamental tone.

simulated spectrum is also compared with the theoretical one from Section 6.1. As can be seen, the simulated spectrum is very close to the theoretical one.

Figure 6.7 shows a comparison between the simulated power spectrum density of the output clock signal and the same spectrum obtained from the circuit without any driving signal, i.e. working as a standard PLL-based clock generator. The resolution bandwidth is set to 120 kHz, as indicated by CISPR regulations [89]. The comparison shows a peak reduction on the fundamental tone of about 13 dB.

All the simulation results are confirmed by measurements on the prototype. The chip microphotograph is shown in Figure 6.8 while Table 6.1 gives a performance summary of the integrated SSCG. The active area occupies 0.38×0.65 mm^2 and the total area including pads is 1.38×1.20 mm^2. The low-pass filter is off-chip. Figure 6.9 shows the measured spectrum of the 100 MHz output signal without any modulation (a) and modulated with the optimum index value $m = 0.318$ (b). The measured peak reduction is about 18 dB.

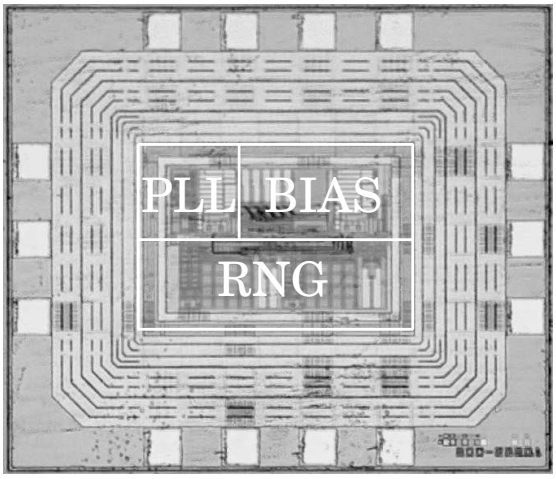
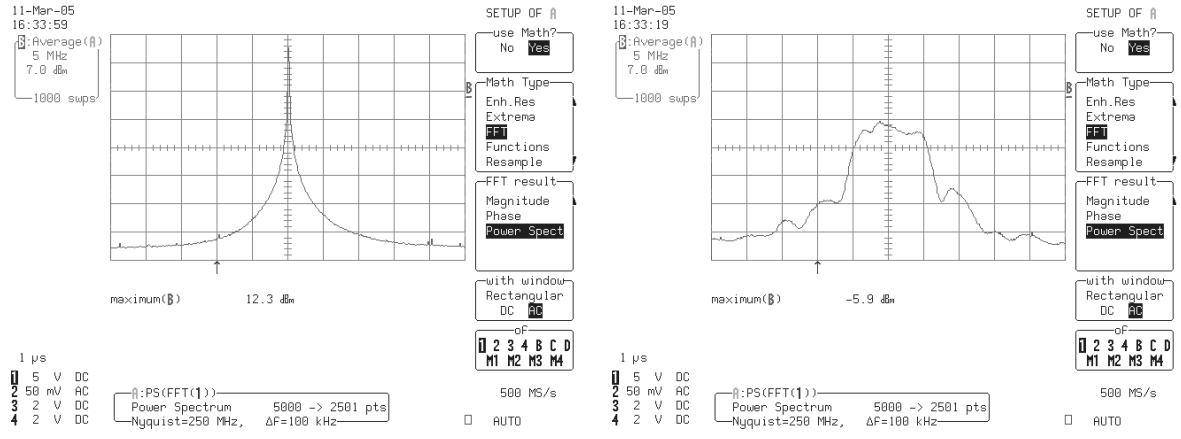


Figure 6.8: Microphotograph of the 0.35 μm SSCG prototype.

Output frequency	100 MHz
Modulation type	Binary Random
Modulation frequency	10 MHz
Frequency Deviation	3.18 MHz
Lock-range	63–108 MHz
Chip area	$1.38 \times 1.20\text{mm}^2$
Power consumption	20.5mW
Closed loop Bandwidth	15 KHz
	$C_1 = 58\text{nF}, C_2 = 5.8\text{nF}$
	$R_1 = 370\Omega$

Table 6.1: Performance summary of the 0.35 μm SSCG prototype



(a) (b)

Figure 6.9: (a) Measurements from the prototype in non spread spectrum mode; and (b) in spread spectrum mode.

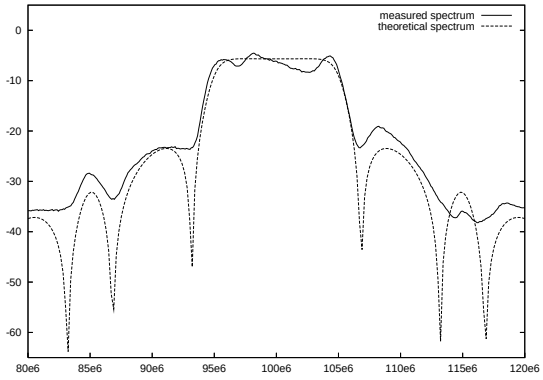


Figure 6.10: Comparison between the measured spectrum of Figure 6.9(b) and the theoretical one.

Figure 6.10 shows the comparison between the spectrum from Figure 6.9a and the theoretical one; the matching is very good, confirming the effectiveness of the proposed circuital approach.

6.3 Description of the 180 nm SSCG prototype

The above 0.35 μm integrated circuit has been completely redesigned in UMC 180 nm CMOS technology.

In this implementation, the center-spread frequency has been set to the nominal value of $f_0 = 3$ GHz and the frequency deviation Δf is set to the 0.5% of f_0, i.e. $\Delta f = 15$ MHz. This is a standard value for the frequency deviation, and it is used, for example, in the Serial ATA protocol [93]. Thus, to achieve the optimal modulation index m value, the random bit generator bit-rate is set to $f_m = 47.17$ Mbit/s. This random bit generator has been already described in Chapter 2, Section 2.8. The microphotograph of the circuit is shown in Figure 6.11.

The unmodulated and modulated VCO $f_{\rm out}/V_{\rm filter}$ characteristic for this implementation is represented in Fig. 6.12; the voltage/frequency ratio is set to:

$$K_{\rm VCO} = 248\,{\rm MHz/V}.$$

The power spectrum density of the output clock signal can be seen in Figure 6.13a. The simulated spectrum is also compared with the theoretical one. As in the previous prototype, the simulated spectrum is very close to the theoretical one.

Figure 6.13b shows a comparison between the simulated power spectrum density of the output clock signal and the same spectrum obtained from the

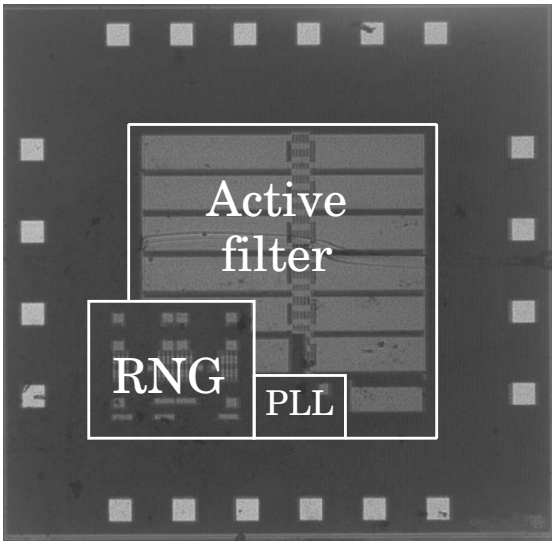


Figure 6.11: Microphotograph of the 180 nm SSCG prototype.

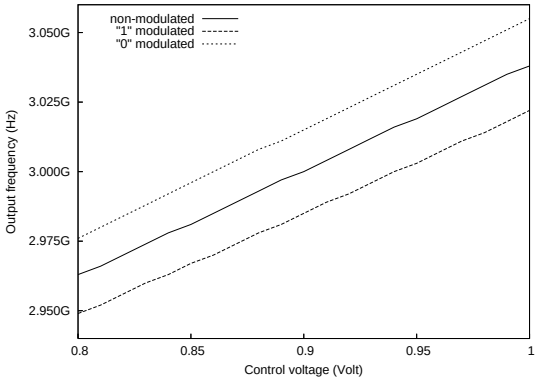
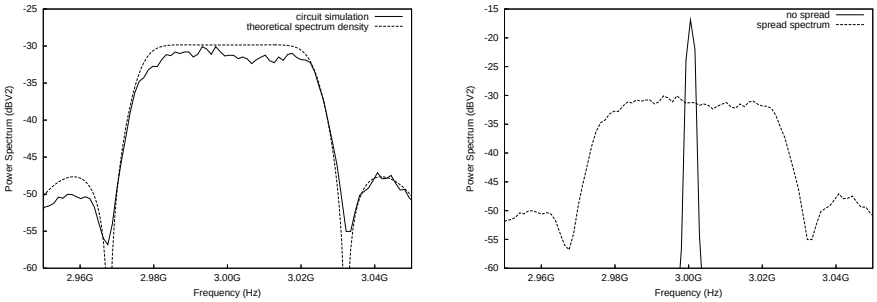


Figure 6.12: Voltage/Frequency characteristic of the VCO in non spread spectrum mode (solid line) and spread spectrum mode (dashed lines).

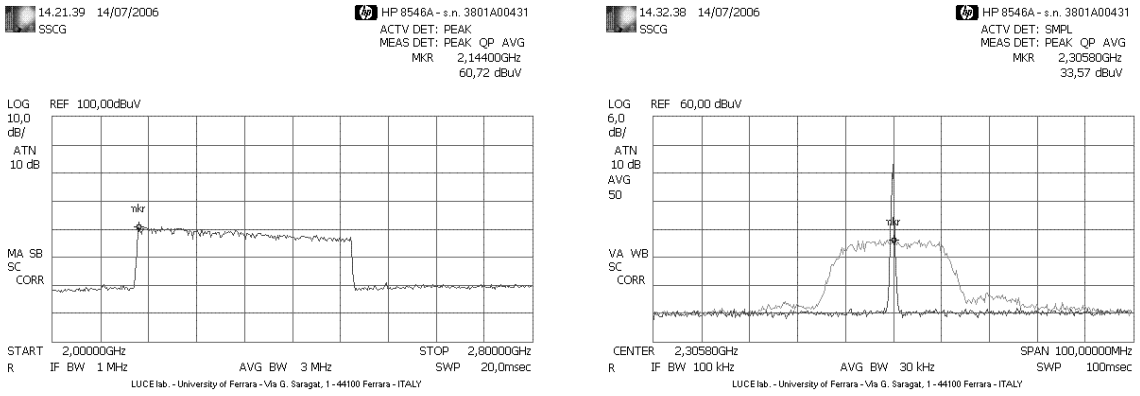
circuit without any driving signal, i.e. working as a standard PLL-based clock generator. The comparison shows a peak reduction on the fundamental tone of about 13 dB.

Regrettably, measurements from the prototypes indicate that the circuit works in a range of frequencies that is sensibly lower than expected. In fact, the lock range of the PLL (Figure 6.14a) goes from 2.2 GHz to 2.5 GHz, that is far from the 3 GHz expected. This is due probably to an understimation of the the parasitic effect in the VCO. However, as can be noticed from Figure 6.14b the binary modulation is properly applied, and the frequency spectrum is exactly the expected one. The peak reduction is measured in about 16 dB. A summary



(a) (b)

Figure 6.13: (a) Comparison between power spectrum density of the output clock obtained from the simulated circuit and the theoretical power spectrum density of the binary modulation; and (b) comparison between power spectra density of the modulated and non modulated output clock. The spectra are measured in dBV^2, with RBW = 120KHz.



(a) (b)

Figure 6.14: (a) Measured lock-range of the PLL; and (b) comparison between modulated and unmodulated power spectra.

of the prototype characteristic is reported in Table 6.2.

6.4 Conclusion

In this Chapter an application of the designed RNG described in Chapter 2 is presented, that is the ElectroMagnetic Interference reduction with the introduction of a spread spectrum clock. The spreading of the clock spectrum is achieved through a frequency modulation involving a random binary PAM signal as driving signal.

Two prototypes have been designed, the first one in CMOS 0.35 μm technology to operate at a clock of $f_0 = 100$ MHz, and the second one in CMOS 180

Output nominal frequency	3000 MHz
Lock range *(designed)*	2700-3150 MHz
(measured)	2200-2500 MHz
Modulation type	Binary Random
Maximum modulation frequency	100 MHz
Frequency Deviation	0.5%
Chip area	$1.48 \times 1.48\text{mm}^2$
(without pads)	$0.95 \times 0.95\text{mm}^2$
Power consumption	35.5mW
(PLL only)	13.5 mW
Closed loop Bandwidth	45 KHz

Table 6.2: Performance summary of the 180 nm SSCG prototype

nm technology to operate at a frequency $f_0 = 3$ GHz. Both prototype perform the requested modulation achieving the desired clock power spectrum; however for the 180 nm prototype a maximum working frequency lower than the expected one, and approximately equal to $f_0 = 2.5$ GHz, has been measured.

Chapter 7

Design of SCA Resistant Digital Programmable Hardware

The security IC is the emerging vulnerability in the security of an embedded application. They are an easy target for side-channel attacks (SCAs), which aim at finding the secret key of an encryption algorithm by monitoring characteristics such as the power consumption, the execution time, the electromagnetic radiation and other information that is leaked by the switching behavior of digital CMOS gates. Side-channel attacks (SCAs) are non-invasive and directed at observing the device in normal mode of operation [95, 96, 97, 100]. In general, SCAs do not require expensive equipment and are rather easy to set up. Even if measures are included to make the devices tamperproof, side-channel information can leak out. SCAs are a real threat for any device in which the security IC is easily observable, such as smart cards and embedded devices [98, 102].

Especially differential power analysis (DPA) is of great concern [97]. It is very effective in finding the secret key and can be mounted quickly with off-

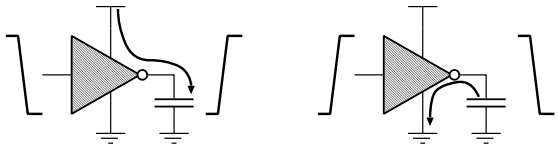


Figure 7.1: During a low-to-high output transition in CMOS logic there is a current request from the power supply, while during a high to low transition there is no such current request.

the-shelf devices. The attack is based on the fact that CMOS logic operations have power characteristics that depend on the input data. As in the example of Figure 7.1, a CMOS logic requires current from the power supply only during a low-to-high output transition, but not during a high-to-low transition [101].

Next to this, it relies on statistical analysis to extract the information from the power consumption that is correlated to the secret key [94, 97]. The attack can be mounted without precise knowledge of the architecture and implementation. It is only necessary to know which algorithm is being used and to have access to plaintext or ciphertext data. The only secure solutions to resist SCAs are hardware solutions, i.e. circuit-level solutions that aim at not creating any side-channel information, instead of concealing or decorrelating the side-channel information from the input data.

The idea is to create digital circuit styles that have a constant per cycle and so data-independent power consumption. After all, precisely the data dependent power consumption of traditional standard cells and logic (i.e., power consumption is dependent on the signal activity), is the fundamental reason that information is leaked through the power supply and power attacks are possible. A CMOS logic style, in order to be input-data independent, must fulfills two requirements: *(a)* the logic style has a single switching event per cycle and this independently of the input signals; and *(b)* the logic style charges a constant capacitance during that switching event.

Implementing a dynamic and differential logic (DDL) style meets the first requirement. It has a switching factor of 100%, since it alternates precharge and evaluation phases, in which the output is precharged to high and conditionally evaluated to low respectively. A differential logic style, on the other hand, holds two output signals with opposite polarity. As a result, the combination of dynamic and differential logic will evaluate exactly one of both precharged output nodes to low in order to generate a complementary output and this independently of the input value. During the subsequent precharge phase, the discharged node is charged and this independently of the input sequence. To fulfill the second requirement, simply the load at the two differential output nodes should be balanced.

Of course there is a heavy payload in terms of current consumption: a DDL circuit, with a switching factor of 100%, has a power consumption that is twofold the power consumption of a standard dynamic logic, and fourfold the power consumption of a standard static logic (when considering the same load capacity).

In this chapter the DDL logic is applied to the design of a programmable

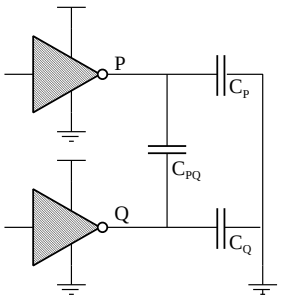


Figure 7.2: Model of the capacitive coupling between two transmission lines.

hardware, namely a FPGA. For the design, the UMC 130 nm CMOS technology has been considered. Any programmable logic hardware can substantially be dived in subcircuits belonging to two categories: logic and interconnections. They are analyzed separately, focusing first on interconnection circuits, and then to logic circuits, trying to make their power consumption data-independent.

7.1 Programmable Interconnections

The coupling between two transmission lines P and Q can be schematized as in Figure 7.2 with three capacitors, C_P between line P ang ground, C_Q between Q and ground, and a cross-coupling capacitor C_{PQ}. The two transmission lines P and Q can be:

- P and Q could be a differential line. In DDL style, both P and Q are precharged to high during the precharge phase, while only one of them is discharged to low during evaluating phase. First, it is possible to notice that that C_{PQ} has no influence, since it is discharged at every evaluation phase and and then recharged in precharge phase independently of the processed data. Instead, it is necessary that $C_P = C_Q$, otherwise it is possible to leak, sensing the charging current, which one of the two lines is being precharged.

- P and Q can be transmission lines coming from two different signals. It does not matter if they represent both represent the true signal, the inverted signal, or they are mixed. In this case, it is possible that in the evaluating phase, neither, both, of only one of them is discharged, and so recharged during the successive precharge phase. If one considers the role of C_{PQ}, it is precharged only if in the previous time period it was

BIAS	C_{DD}+C_{JD}	C_{SS}+C_{JS}	C_{DS}	C_{SD}	g_{DS}
$V_s = 0$, $V_d = 0$	2.9E-16 + 6.72E-16	2.9E-16 + 6.72E-16	-2.73E-20	-1.7E-20	1.01E-8
$V_s = 0$, $V_d = Vdd$	2.9E-16 + 4.23E-16	2.9E-16 + 6.72E-16	-4.2E-21	4.35E-23	4.38E-9
$V_s = Vdd$, $V_d = 0$	2.9E-16 + 6.72E-16	2.9E-16 + 4.23E-16	4.35E-23	-4.2E-21	4.38E-9
$V_s = Vdd$, $V_d = Vdd$	2.9E-16 + 4.23E-16	2.9E-16 + 4.23E-16	-3.38E-28	-1.12E-28	4.12E-23

Table 7.1: Parameters of a NMOS in the *OFF* state.

$P \neq Q$; to ensure a perfectly data independent current profile, it has to be $C_{PQ} = 0$.

Regarding C_P and C_Q, they are connected to different signal lines; for them the previous case has to be applied.

- As a last case, one of the two nodes can be a floating node. In this case each clock cycle C_{PQ} is partially charged and discharged, according to the charge partition ratio. In other terms, there is a memory effect. To ensure a constant current profile at every clock cycle, it has to be $C_{PQ} = 0$.

Briefly, the following guidelines can be summarized:

1. balance the capacitance between the two transmission lines of a signal and ground;
2. avoid directly coupling between two lines of two different signals;
3. avoid any floating node.

Programmable interconnections are realized substantially with pass–transistors; so the classical CMOS transfer gate has been taken into account.

Note that one could think to consider only NMOS pass transistors, since the critical phase is the discharge to ground during the evaluation phase. This solution presents many problems: for example in a chain of pass-transistors, many node can be floating; also it is note ensured that all intermediate nodes are precharged to vdd and additionally, if precharded, they are precharged to vdd minus a threshold voltage. Thus, every intermediate node of a pass-transistor chain should be independently precharged to vdd; however in this way there is no evident reduction in the complexity of the circuit.

Then, a brief model for a MOS transistor in the *OFF* state (i.e. $V_g = 0$ for a NMOS, $V_g = vdd$ for a PMOS) is considered. For the reference technology, and for a $W/L = 1\mu/120nm$ the obtained differential parameters for the NMOS are reported in Table 7.1. The capacitance effect between drain and source is 5 order of magnitude less than the total source (or drain) capacitance, i.e. the

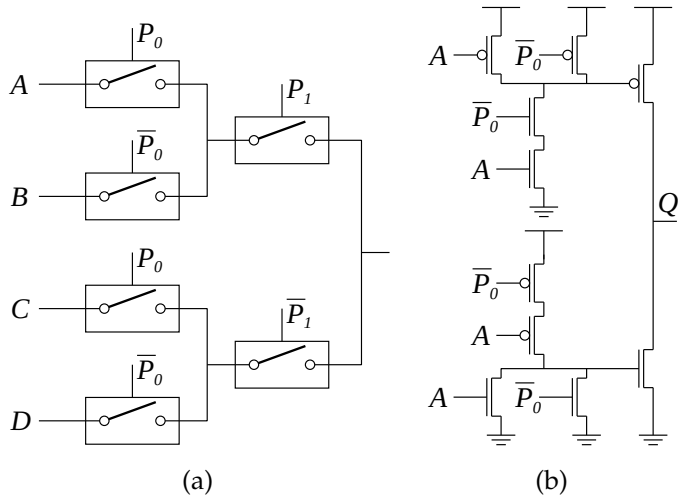


Figure 7.3: (a) Simple four-inputs multiplexer schematics; and (b) tri-state buffer.

junction capacitance and the channel capacitance; the R_{DS} is at least of the order of magnitude of hundreds of megaohm. So effectively a MOS in *OFF* state presents a very low coupling between source and drain and can be modeled as an open circuit, just with C_D and a C_S capacitances.

The following structures are being considered:

- **Multiplexer** Figure 7.3a depicts a four inputs multiplexer, where P_0 and P_1 are the two programming bits. Assuming that all inputs are low impedance, this structure does not present any floating node nor any parasitic caps between two different lines.

- **Tri-state buffer** The schematic considered is depicted in Figure 7.3b, where P_0 is the enable bit. This schematics has advantages in terms of speed and power consumption when driving a heavy capacitive line with respect to a t-gate structure.

7.2 Programmable Logic

In this section a Look-up Table (LUT) based programmable logic is proposed, following the implementation adopted in all Xilinx Spartan FPGA family, with only small variants from one release to another [104]. The basic computational unit is called slice and consists of a four inputs LUT and a Flip-Flop; it is reported in Figure 7.4a where both a static output (coming directly from LUT) and a dynamic output are provided.

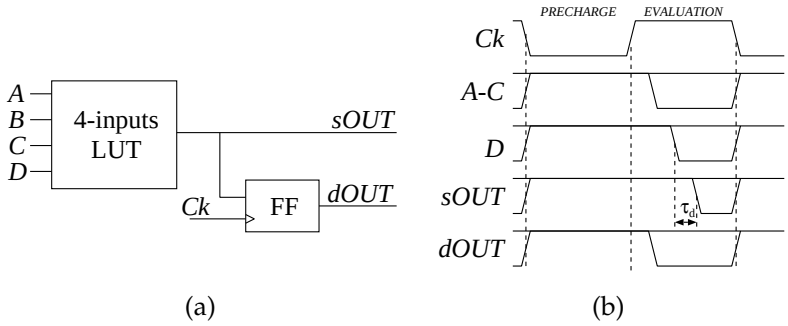


Figure 7.4: (a) Basic slice architecture; and (b) overview of temporization of the signals inside the slice.

A n-inputs LUT is simply a programmable device in which an output is associated to all of the 2^n possible input vectors. It can be considered simply as a 2^n bits memory, in which the logic function to implement has been stored. The static output is provided to connect more than one LUT in a chain, thus implementing logic functions with more than four variables; this however makes the design more complex. In fact in every dynamic logic, it is possible to take advantages of dynamic behavior interleaving circuit working at opposite phase of the clock. For example, it is possible to directly connect a p-type gate to a n-type gate, working with the same clock: due to their complementary when one gate is in the precharge phase, the other is in the evaluating phase. Or one could connect two n-type gates, running at the two opposite phases of the clock; this time, however, the interposition of a small inverter between the output of the first gate and the input of the second is required. In both solutions, every two gates a unity delay is present.

So, one can think to make the LUT and the FF working on the two different phases of the clock since they are connected in an interleaved way; however this architecture will prevent any LUT chain connection because all LUTs of the circuit work on the same clock phase.

An example of a temporization for this basic slice can be seen in Figure 7.4b. When clock is low, both FF and LUT are in precharge state. When clock rises, the output of the FF changes; and after a while (depending on the interconnection capacitance) all the inputs coming from the dynamic outputs of other LUTs (A to C in figure) also change. If we assume that D comes from the static output of a LUT, it may require much more time to asset. When all inputs are changed, the output of the LUT is elaborated in a time τ_D. After that, a time greater than the setup time of the flip-flop is necessary before the precharge phase in order to let the FF memorize the data from the LUT.

Both LUT and Flip Flop are designed following the sense amplifier based

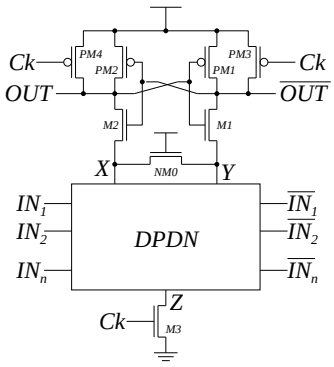


Figure 7.5: Basic architecture of the sense-amplifier based logic.

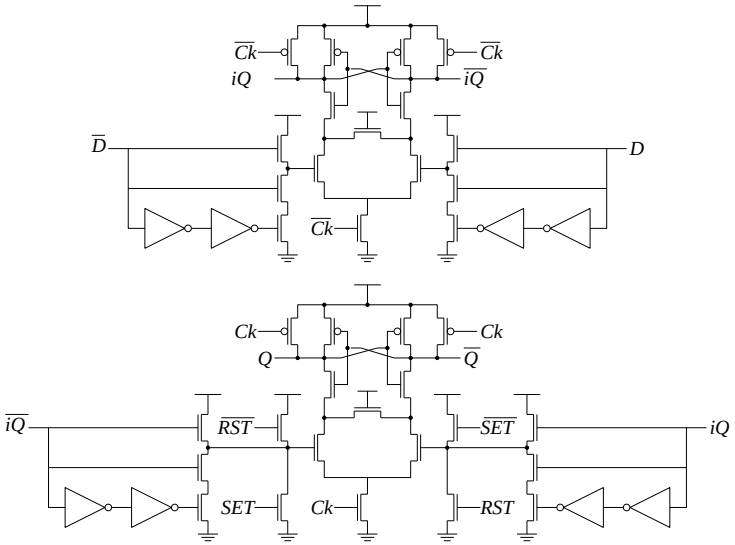


Figure 7.6: Implemented SABL Flip Flop schematic.

logic (SABL) suggested by K. Tiri [103] and based on the StrongArm 110 flip flop [99]. The flip flop is a standard D-type, composed by two SABL inverters working on the two different phases of the clock. The basic n-type SABL architecture is reported in Figure 7.5; it is essentially composed by a sense amplifier ($M1$,$M2$,$PM1$,$PM2$), a differential pull-down network (DPDN), and by a couple of additional MOSes that provide to precharge the circuit ($PM3$,$PM4$) and to enable the sense amplifier ($M3$). The purpose of the DPDN is to connect to node Z either the node X or Y depending on the inputs. When the sense amplifier is activated by $M3$, either OUT or $\overline{OUT}$ goes low depending on which node is discharged through the DPDN. The additional MOS $MN0$ serves to discharge both X and Y node after the circuit has take its decision whichever

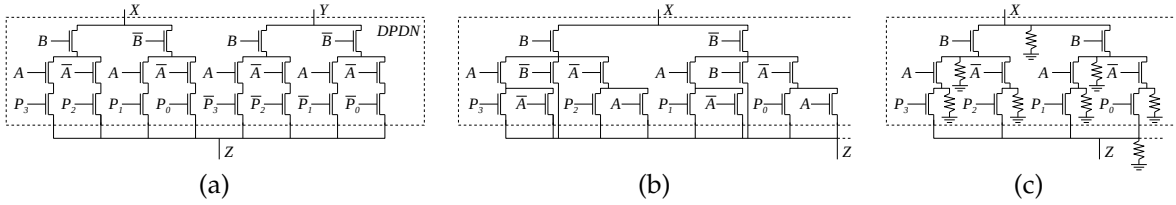


Figure 7.7: (a) Basic implementation of the DPDN; (b) fully connected implementation of the DPDN (particular); and (c) resistive discharged DPDN (particular).

branch was discharged by the DPDN, thus deleting any memory effect.

Since both FF and LUT are designed with n-type SABL logic, i.e. the discharge network is composed by NMOS transistors, the interposistion of static inverters between the output of the LUT and the input of the FF, as well as between the output of a slice and the input of another slice, is necessary.

The SABL Flip Flop designed (Figure 7.6) is composed by two inverters working at the two opposite phases of the clock, with a delay line on the input of both inverters. The delay line is necessary for correct operation, since the second inverter begins the evaluation phase exactly when the first inverter starts the precharge phase and charges its outputs to high; however the inputs of the second inverter must remain unchanged for a setup time in order to let the inverter memorize the data. The delay line is asymmetric; it has been designed to obtain a longer delay when the input is rising (it is being precharged) and a shorter delay when input is falling (it has been evaluated). This to ensure that *(a)* the input signal is stable for enough time before it goes into the precharge phase, thus correctly setting the output of following DDL stage; and *(b)* once the previous DDL stage has made a decision, this decision is transferred as soon as possible to the current stage. In particular, this is useful when considering the connection with the LUT, that has unknown temporization. The output from the LUT has to be transferred in the shortest time possible to the FF to not increase the setup time of the slice. In addition, a SET/RESET circuit for the FF has been added to the delay line; it is of course a synchronous SET/RESET due to the dynamic architecture used (i.e. it is not possible to force an output when it is in a precharge state). With the proposed circuit the delay for the rising/falling edge of the input signal are respectively, about 220ps and 45ps.

The proposed LUT design is a SABL circuit, whose DPDN is based on the multiplexer circuit (MUX) of Figure 7.7. Actually, only a two bits (i.e. four inputs) MUX is presented for simplicity, but the actual implementation is a four bits (i.e. sixteen inputs) MUX. P_0, P_1, etc are the inputs, while A and B the

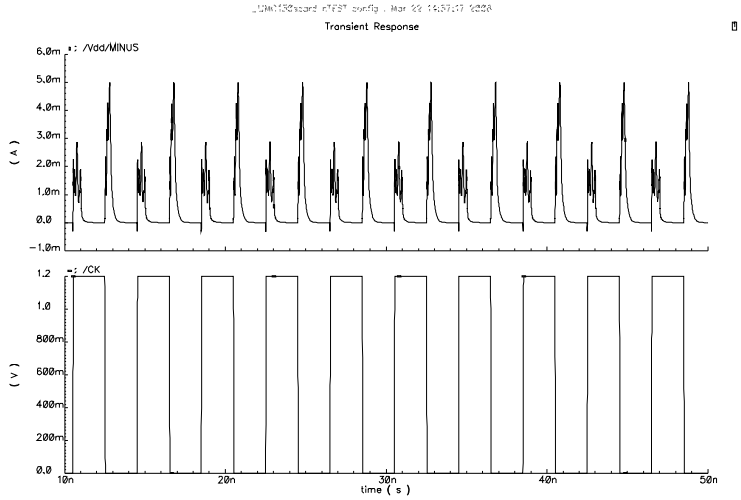


Figure 7.8: Typical current profile a the proposed slice.

programming bits. In the basic circuit design (Figure 7.7a), if we suppose that A and B are low, the active path is either the branch composed by the NMOS driven by $\overline{A}$, $\overline{B}$ and P_0, or the path driven by $\overline{A}$, $\overline{B}$ and $\overline{P_0}$. This means that, if P_0 is high, the node X is discharged, and OUT goes low; if $\overline{P_0}$ is high, the node Y is discharged and so $\overline{OUT}$.

However, this schematic is not fully connected, since many internal node are not discharged and may let arise a memory effect. A fully connected pull down network is necessary to discharge all internal nodes, so deleting all information about the internal state of the DPDN; this is reported in Figure 7.7b.

Another possibility is to statically connect every node to ground with a resistor (this means, in CMOS technology, a resistive MOS always in ON state), like in Figure 7.7c. Note that in the last example the MOS $NM0$ connected between nodes X and Y is no more necessary, since both nodes are already discharged through their own resistor.

The performances of the three proposed DPDNs have been analyzed with simulations. Even if the first one is not fully connected, it has been considered since it is the simplest one with the smallest parasitic. To test the DPDN, four slices have been considered and connected each other (the output of every slice is connected to all four slices) and the LUTs programmed to implement a four-bits counter. A typical profile of the power supply current is reported in Figure 7.8.

The current profile presents two peaks: the first, higher, at the beginning

DPDN	1^{st} peak	2^{nd} peak	average
Not connected	5.0mA $\pm$ 2.9uA	2.8mA $\pm$ 10uA	540uA
Fully connected	4.7mA $\pm$ 2.3uA	3.1mA $\pm$ 22uA	560uA
Resistive	5.1mA $\pm$ 0.4uA	2.9mA $\pm$ 0.6uA	590uA

Table 7.2: Comparison between the performance of the proposed slice with the three DPDNs of Figure 7.7.

of the precharge phase; the second, lower, in the evaluation phase. The speed of the clock has been set to 250 MHz (4 ns clock period). Data extracted from a 80 ns transient simulation, including (average) peak value, the maximum difference among peak values, and the average current, are reported in Table 7.2.

All solutions present some differences in the peak values, more relevant on the second highest peak (i.e. in the evaluation phase) which is the peak due to the buffers driving the input of the LUT. Other differences could be sometimes found on other peaks, but they are not considered because all other peaks would be hard to recognize in a real system. An explanation for for the observed differences is given in the following.

- one can address the floating nodes problem and so the memory effect for the peak differences in the first (not-connected) solution. However, due to the simplicity of the design, the parasitic capacitances are quite small, so the differences among peaks are very limited, measurable in few point per hundreds on the average current value and (the most important factor) few points per thousands on the peak value.

- even if the fully connected pull down network does not present floating nodes, it has affected by the same problem known as clock feedthrough in transfer gates. Whenever a MOS device is turned off, all the charge present in the MOS channel has to be eliminated; since all transistor in the pull down network are off in the precharge phase, that charge cannot be removed and affect the power supply current in the same way as if one neglects the problem of the floating nodes. In addition, the circuit average current is increased since a fully connect pull down network require about the double of transistors than a not connected network.

- if it is possible to afford the increment ot the average current, one could introduce a static discharge on every node of the pull down network. This is an effective way to get rid of the clock feedtrough problem, since now all charges can be dispersed through the resistive channel. However

DPDN	1^{st} peak	2^{nd} peak	average
Not connected	4.9mA $\pm$ 3.3 uA	4.1 mA $\pm$ 23uA	790uA

Table 7.3: Performance of the proposed slice when considering a more realistic system.

other second-order problems are still present, due to the (possible) asimmetricity of the programming bits. It is possible to consider for example the gate current: in a single slice the current of the eight (all four inputs and their complementary) inverters that drive the LUT inputs, a difference up to 0.5uA depending on the input combination can be measured. This variation is of the same order of magnitude as the overall variation.

7.3 A Realistic System

A more real system has been considered, including 4 slices, eight tri-state buffers connected to each output of the slice (i.e. both the dynamic output and the algebraic output), eight interconnection lines, and a four-inputs multiplexer to every slice input. Even if not particularly complex, this system contains all the macroblocks of a FPGA and thus can be considered representative for a real circuit. The interconnections have been programmed to connect all four outputs to every slice input, as in the previous example. The four, unused, connection lines have been connected to ground to avoid floating nets. The slices have been programmed to implement a four bit counter. Only the simplest DPDN for the LUT, i.e the not connected one, has been considered. Also, all parameters have been set as above for a comparison. What we can observe is reported in Table 7.3.

7.4 Open Problems

First of all, one can notice a problem that, in various ways, affects all dynamic logic. From the description of timing, in the precharge phase the outputs of the LUT is in the precharged state “11”. As the clock rises, the inputs are expected to change; as soon as the inputs change the LUT can take a decision and modify its output from “11” to “10” or “01”. However, if the change arrives too late, or does not arrive at all, the invalid state “11” is transmitted to the FF. Actually, the invalid state is not transmitted; just as the LUT also the flip flop does not see a change at its input, and cannot take a decision; for sake of simplicity, this process is addressed as the transmission of an invalid state to the following

stage. At the next clock cycle, the invalid state "11" of the FF is transmitted to all other LUTs to which it is connected, and so on; the invalid state will eventually reach all devices in the FPGA. If this is not generally a problem in a dynamic logic, this can be a real problem here since in a programmable logic it is not possible to compute a priori the maximum propagation time of a signal. To cope with this problem the only solution is to slow down the clock signal, and to add a check that halt the system as soon as an invalid state propagation is detected.

Another problem that can be addressed is the fact that actually, the model used for the power supply is an infinite-bandwidth model, when in the real world all power supply have a limited bandwidth. In other words, one has to consider all the distributed parasitic – inductors, capacitors and resistors – on the power supply network, that are however quite difficult to estimate.

Also, a programmable device needs, of course, to be programmed. In this study, all programming bits are considered coming from a bi-stable circuit (i.e. a static memory bit block) whose value was given by the simulation initial condition. It is possible, though unlikely, that some information can be transmitted to the programmable logic. This should be investigated.

Chapter 8

Final Conclusion

THE MAIN TOPIC of this dissertation is the presentation of a monolithic chaos-based Random Number Generator and of its testing results. However, this is not the only original contribute of this dissertation. All the original results presented are summarized here, divided into theoretical results and practical results. They are sorted in order of relevance, of course according only to author's point of view.

Theoretical Results

Second Level NIST Statistical Tests

The methodology to check for the uniform distribution on different p-values coming from different analyzed sequences in statistical tests for randomness, here addressed as second level test, has been investigated in order to improve the reliability of the test. This approach is not new, and suggested also by NIST in the last part ot if special publication. This approach is proved to increase the reliability of the test, since it is able to recognize the KISS generator as a pseudo-random generator.

This approach however presents a strong sensitivity on approximation error introduced in the reference distribution used in statistical tests for computing the p-value. Here for the simplest test, a mathematical theory has been elaborated for the explanation and the estimation of the maximum propagated error. In particular, it has been found that this propagated error is dependent on the number of bits n in the analyzed sequence; for a reliable second-level test, this error should be smaller, ot at least, approximately equal to the random variance that one cold expect looking at results coming from a number N

of sequences. In this case, the propagated error can be confused with a random probabilistic error, and does not affect the results of the test. Based on the analysis on few basic tests included in the NIST SP 800-22 publication, and with $n = 10^6$, the suggested number N of sequences to be used in the second-level test it is $N \leq 20,000$.

Noise Robustness of Chaotic Maps

The behavior of chaotic system perturbed by an external additive map independent noise has been investigated. As a prerequisite to tolerate the noise, it has to be noticed that the map has to be somehow extended in order to tolerate this noise. Then, a mathematical model of the extended system perturbed by the noise has been studied; a mathematical condition for noise robustness has been found when limited on chaotic maps with piecewise constant invariant density, so including all PWAM maps. This condition can be easily verified assuming that the extended map is the restriction of a periodic function, thus linking noise robustness with topological properties of the map; for simple maps it has also been shown that periodicity is also a necessary condition.

The given condition cannot be applied to any chaotic map, since includes constraints that sometimes are impossible to satisfy by any extension of the basic map; the condition can be used to investigate if, given a chaotic map, a noise robust extension exists, but also as a synthesis tool, to design a noise robust map from given statistical properties.

Practical Results

Design of a Monolithic Chaos-based RNG in 0.35 μm Technology

By exploiting the statistical approach to the study of non-linear dynamic circuits, more precisely of chaotic maps, it was recognized that the architecture used for common pipeline ADCs can be reused for designing a robust chaotic circuit appealing for the generation of random numbers. Following this approach a prototype of a RNG has been designed in 0.35 μm CMOS technology. The prototype is capable of generating up to 40 millions random bit per second. The prototype has been tested with the most advanced tests for randomness available considering few very simple post processing stages, and results compared with results of two high end commercial generators. The results of the

comparison indicate that the designed circuit can be considered a high quality RNG, outperforming by one order of magnitude the commercial generators considered, suitable for the most advanced security-related applications.

Design of a Monolithic Spread Spectrum Clock Generator for EMI Reduction

The design and measurements of a spread-spectrum clock generator implementing a random binary frequency modulator are presented. This modulation is known to present the maximum peak reduction in the power spectrum of the generated clock (i.e. the maximum electromagnetic interference reduction) with respect to all other known methods.

The spreading of the clock is driven by a random number generator implemented through a chaotic map. A first prototype of this circuit has been designed in 0.35 μm CMOS technology working at a center frequency $f_0 = 100$ MHz. Both simulations and measurements confirms that the behavior of the circuit is the expected one, as there is a very good matching between the theoretical power spectrum, the simulated power spectrum, and the measured one. The observed peak reduction in the power spectrum between the spread spectrum clock generated by the prototype and a classical clock can be measured in about 18 dB.

A second prototype has been designed in 180 nm CMOS technology, aiming to work at $f_0 = 3$ GHz. Regrettably, even if the circuit performs the expected modulation, and the matching between the measured power spectrum and the theoretical one is good, the maximum working frequency of the prototype is only about $f_0 = 2.5$ GHz.

Simple and Effective Post-Processing Stage for the Designed Chaos-based RNG

When taking into account the effects of implementation errors in the design of any RNG, a suitable post-processing function for their compensation and for assuring robust behavior has to be considered. So, a simple post-processing architecture that ensures a certain quality of the sequences generated by the ADC-based RNG has been investigated.

Yet, strictly speaking, this topic should not be considered *practical* since this post-processing has not been included in the prototype design; however since a theoretical analysis of the post-processing function has not been provided,

the practical category has been considered more adequate than the theoretical one.

The post-processing function is based on four shift-registers closed into a feedback loop with the interposition of some XOR gates; it does not introduces decimation, and it has been tested with the 0.35 μm RNG prototype running at 40 Mbit/sec and overclocked at 48 Mbit/sec. In both case, a minimum complexity of the post-processing required to pass a second level NIST statistical test for randomness has been found.

Appendix A

Introduction to Discrete-time Chaos Theory

THIS APPENDIX is intended to give a general overview of discrete-time chaos theory. In particular the attention is focused on two main aspects; the first one is under which aspects a chaotic circuit can be analyzed as a Markov chain. The second is the robustness of a dicrete time chaotic process.

A.1 Chaotic maps

Systems referred to as *chaotic maps* are 1-D discrete-time autonomous chaotic dynamical systems [46, 60, 72]. Consider the domain X and a nonlinear, non-invertible function $M : X \to X$. These system are dynamical since they have memory of the past; the domain X of M is called the *state space* of the transformation.

Starting from an initial point $x_0 \in X$, the successive states at times $1, 2, 3, \ldots$, are given by the trajectory $x_1, x_2, x_3, \ldots$, where

$$x_{k+1} = M(x_k) \tag{A.1}$$

for $k = 0, 1, 2, \ldots$.

In these systems, like in any chaotic systems, the observation of chaotic trajectories is difficult and brings very little information, because a slight change in the initial state give rise to a substantially different evolution of the system. Typically, few iterations are sufficient to make the new trajectory almost uncorrelated from the previous one so that any error in the knowledge of the state results in the impossibility to predict the position in the state space.

To cope with this problem, it is usual to consider a different approach, i.e. a probabilistic approach. However, before introducing the main operators used in the classical analysis, it is necessary to remind some definitions.

Denote with M^k the k-th iterate of M, so that for any set $Y \subseteq X$, $M^k(Y)$ is the set $\left\{y \in X \,|\, y = M^k(x) \wedge x \in Y\right\}$ and $M^{-k}(Y)$ is the set $\left\{x \in X \,|\, y = M^k(x) \wedge y \in Y\right\}$. Indicate with $\mathcal{A}$ a σ-algebra of subsets of X and with μ a measure on $\mathcal{A}$. The triple $(X, \mathcal{A}, \mu)$ is called a measure space. Moreover, if $\mu(X) = 1$, then $(X, \mathcal{A}, \mu)$ will be named a *probability space*. With regard to a measure space $(X, \mathcal{A}, \mu)$, a map $M : X \to X$ is said to be

1. *nonsingular*, if $\mu\left(M^{-1}(Y)\right) = 0$ for all sets $Y \in \mathcal{A}$ such that $\mu(Y) = 0$;
2. *measure-preserving*, if $\mu\left(M^{-1}(Y)\right) = \mu(Y)$ for any set $Y \in \mathcal{A}$.

A measure-preserving transformation is necessarily nonsingular.

A.2 The Perron-Frobenius Operator

Assume that the initial state x_0 is a random variable drawn according to a (probability) density $\rho_0 : X \to \mathbb{R}^+$. If the map is iterated, a new random variable $x_1 = M(x_0)$ is obtained and a link exists between ρ_0 and the (probability) density ρ_1 associated to it

$$\rho_1(x) = \frac{d}{dx} \int_{M^{-1}([0,x])} \rho_0(\xi)\, d\xi$$

Indicate with $\mathbb{L}_1$ the space of all the Lebesgue integrable functions $\phi : X \to \mathbb{C}$ and with $\|\cdot\|_1 = \int_X |\cdot|$ the associated $\mathbb{L}_1$ norm, and consider a measure space $(X, \mathcal{A}, \mu)$, where μ is the Lebesgue measure.

DEFINITION 1. If M is a nonsingular map, the unique operator $\mathbf{P} : \mathbb{L}_1 \to \mathbb{L}_1$ defined by

$$[\mathbf{P}\phi](x) = \frac{d}{dx} \int_{M^{-1}([0,x])} \phi(\xi)\, d\xi = \int_X \phi(\xi)\, \delta(M(\xi) - x)\, d\xi$$

is called the *Perron-Frobenius Operator* (PFO) corresponding to M.

The PFO $\mathbf{P}$ is a functional linear operator characterized by the following properties [60, 61]:

$$\mathbf{P} \text{ is positive, i.e. } \mathbf{P}\phi \geq 0, \text{ if } \phi \geq 0 \tag{A.2}$$

$$\mathbf{P} \text{ conserves } \|\cdot\|_1, \text{ i.e } \int_X |[\mathbf{P}\phi](x)|\, dx = \int_X |\phi(x)|\, dx \tag{A.3}$$

$$\text{The PFO corresponding to } M^k \text{ is } \mathbf{P}_k = \mathbf{P}^k \tag{A.4}$$

Properties (A.2) and (A.3) assure that the PFO maps density functions into density functions so that the restriction of the PFO to the set $\mathbb{D}(X)$ of probability densities defined on X can be considered. Property (A.4) assures that the PFO associated with the k-th iterate of the map is the k-th successive application of the PFO associated to M. If the initial condition x_0 of the map is drawn according to ρ_0, its state after k iterations is regulated by the density

$$\rho_k = \mathbf{P}\rho_{k-1} = \mathbf{P}^k\rho_0$$

For all the maps considered here, and for k large enough, the density $\rho_k = \mathbf{P}^k\rho_0$ converges to a final density $\overline{\rho}$ independently of ρ_0. Such a final density must be a *fixed point* of the PFO associated to it, i.e. $\mathbf{P}\overline{\rho} = \overline{\rho}$. This is usually expressed by saying that $\overline{\rho}$ is the *invariant density* of the map. Moreover, $\mathbf{P}\overline{\rho} = \overline{\rho}$ if and only if the measure $d\overline{\mu} = \overline{\rho}dx$ is invariant under M [60, 61] and $\overline{\mu}$ is referred as the *invariant measure* of the map.

A.3 Ergodic, Mixing and Exact Maps

The existence of a unique invariant density $\overline{\rho}$ is linked to particular properties of M.

DEFINITION 2. Consider a measure space $(X, \mathcal{A}, \mu)$. A nonsingular map $M : X \to X$ is said to be *ergodic* if every invariant set $Y \in \mathcal{A}$ is a trivial subset of X, i.e., if either $\mu(Y) = 0$ or $\mu(X \backslash Y) = 0$.

A set Y is an *invariant set* if $M(Y) = Y$. The above definition states that in an ergodic map, no invariant sets other than X can exist; X is also called *principal invariant set*. The following theorem links ergodicity with the existence of $\overline{\rho}$.

THEOREM 1. *Let $(X, \mathcal{A}, \mu)$ be a measure space, M a nonsingular transformation, and $\mathbf{P}$ the PFO associated to M. If M is ergodic, then there is at most one invariant density $\overline{\rho}$ for $\mathbf{P}$. Furthermore, if there is a unique invariant density $\overline{\rho}$ of $\mathbf{P}$ and if $\overline{\rho}(x) > 0$ almost everywhere, then M is ergodic.*

DEFINITION 3. A map $M : X \to X$ is called piecewise C^2 if it exists a sequence of points $0 = y_0 < y_1 < \ldots < y_n = 1$ in X such that for any $j = 1, \ldots, n$ the restriction of M to the open interval $]y_{j-1}, y_j[$ is a C^2 function which can be extended to the corresponding closed interval $[y_{j-1}, y_j]$ remaining of class C^2. M does not need be continuous at the points y_j.

DEFINITION 4. Consider a probability space $(X, \mathcal{A}, \mu)$ and a measure preserving transformation M. M is called *mixing* if for any $Y_1, Y_2 \in \mathcal{A}$ one has that $\lim_{n\to\infty} \mu(Y_1 \cap M^{-n}(Y_2)) = \mu(Y_1)\mu(Y_2)$.

For a mixing map $\mu\left(Y_1 \cap M^{-n}\left(Y_2\right)\right)/\mu\left(Y_2\right) \to \mu\left(Y_1\right)$ as $n \to \infty$. The first term is the probability (according to μ) that a typical system trajectory arrives in Y_1 after n time steps, given that it starts in Y_2 while the second term simply represents the probability that a typical trajectory is in Y_1. It follows that for large n the two events $\{x \in Y_2\}$ and $\{M^n\left(x\right) \in Y_1\}$ become statistically independent.

A mixing map is also ergodic but has a much more complicated behavior; in fact it is common to indicate as chaotic only those maps which are at least mixing. Additionally, for a mixing map, it can be proven that

$$\left\|\mathbf{P}^k\rho_0 - \overline{\rho}\right\|_{BV} \leq H\left\|\rho_0\right\|_{BV} r_{\mathrm{mix}}^k$$

for a suitable constant $H > 0$ [51] and a suitable defined bounded variation norm $\|\cdot\|_{BV}$ [60] so that if the initial condition of a mixing map is randomly chosen according to a bounded variation density ρ_0, then the state x_k distributes according to a density $\mathbf{P}^k\rho_0$ which converges to the invariant one at an exponential rate r_{mix} (called *rate of mixing*) in the bounded variation norm.

Finally, some noninvertible transformations may possess a stronger form of mixing, which is called exactness [60].

DEFINITION 5. Consider a probability space $(X, \mathcal{A}, \mu)$ and a measure preserving transformation $M : X \to X$. M is called *exact* if $\lim_{n\to\infty} \mu\left(M^n\left(Y\right)\right) = 1$ for any $Y \in \mathcal{A}$ with $\mu\left(Y\right) > 0$.

It can be proven that exact maps are also mixing; the chaotic maps that found practical applications usually are exact maps.

A.4 Markov Chains and PWAM Maps

In mathematics, a Markov chain, named after the russian mathematician Andreyevich Markov, is a discrete-time stochastic process with the Markov property, that is, briefly speaking, the property of a process to keep memory only of the last realization (i.e. memory-1 property).

For the scope of this chapter, it is enough to consider a finite Markov chain, that is a finite-state machine, where transition from one state to another, as well as the possibility to rest in the same state, is regulated by a stochastic process.

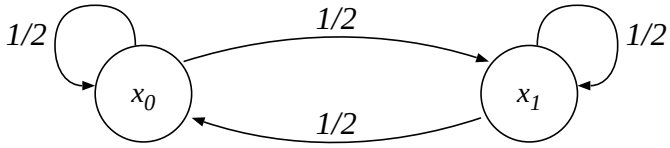


Figure A.1: Example of Markov chain.

The Markov property ensure that the conditional probability distribution of a state in the future can be deduced using only the current state; no additional information is given by the knowledge of the past evolution. In other words, the past states carry no information about future states.

A Markov chain is usually depicted as in Figure A.1, where two states x_0 and x_1 are present, and all the possible transitions between the states are indicated with an arrow, and by the probability associated to this transition. This is the Markov chain that describes the fair coin toss, where, for example, being in the state x_0 correspond to the outcome "head" of the toss, while the state x_1 is associated to the outcome "tail". Every time we toss the coin, i.e. every time the finite-state machine may change the state, the probability to stay in the same state, i.e. to get the same outcome of the previous coin toss, or the probability to change state, i.e. to have an outcome different from the previous one, are both equal to $p = 1/2$.

Note that every Markov chain can be expressed as a stochastic matrix which is a square matrix each of whose rows consists of nonnegative real numbers and sums to 1. Each state of the Markov chain is associated to a row of the matrix; each element of the row is the conditioned probability to be in the state associated to the row, and have a transition towards one of all the possible states. The stochastic matrix associated to the Markov chain of the example is

$$\boldsymbol{P} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}$$

The goal of this section is to introduce a particular class of chaotic maps whose behavior can be described as a Markov chain.

DEFINITION 6. A map $M : X \to X$ is said to be a Piece-Wise Affine Markov (PWAM) map with respect to a given partition $\mathcal{X}_n = \{X_1, X_2, \ldots, X_n\}$ if the intervals X_j are such that M is affine on each X_j and that for any couple of indices j and k, either $X_k \subseteq M(X_j)$ or $X_k \cap M(X_j) = \emptyset$.

Under the assumption of an exact PWAM map, if one limits himself to initial probability densities ρ_0 which are stepwise in $\mathcal{X}$, all subsequent probability

ρ_k densities are then compelled to be stepwise on the interval partition. Note that this analysis is not restrictive, since also $\overline{\rho}$ is compelled to be stepwise, and for ergodic maps the existence of a unique invariant density has been proved.

Consider a partition $\mathcal{X}_n$ and indicate as Σ_n the n-dimensional subspace of $\mathbb{L}_1$ generated by χ_{X_j}. Any function φ step-wise in $\mathcal{X}$ is belonging to Σ_n, and can be expressed as a n-dimensional vector $\boldsymbol{\varphi} = (\varphi_1, \ldots, \varphi_n)$ with respect to the basis $\{\chi_{X_j}, X_j \in \mathcal{X}_n\}$. Then, define the $x \times n$ matrix $\boldsymbol{\mathcal{K}}$ as

$$\boldsymbol{\mathcal{K}}_{j_1,j_2} = \frac{\mu\left(X_{j_2} \cap M^{-1}(X_{j_1})\right)}{\mu\left(X_{j_2}\right)}$$

This is often referred to as the *kneading matrix* since its entry in the j_1-th row and j_2-th column records the fraction of X_{j_2} that is mapped into X_{j_1}. It follows that $\boldsymbol{\mathcal{K}}$ is a stochastic matrix that can be used to express the evolution of density step-wise in $\mathcal{X}_n$ starting from an arbitrary $\boldsymbol{\varphi}_0$:

$$\boldsymbol{\varphi}_{k+1} = \boldsymbol{\mathcal{K}}\boldsymbol{\varphi}_k$$

In other words, $\boldsymbol{\mathcal{K}}$ can be interpreted as the restriction of **P** into the finite-dimensional subspace Σ_n. The invariant density $\overline{\boldsymbol{\varphi}}$ can be simply computed as $\overline{\boldsymbol{\varphi}} = \boldsymbol{\mathcal{K}}\overline{\boldsymbol{\varphi}}$, i.e. it is given by the right eigenvector $\mathbf{e} = (e_1, \ldots, e_n)$ of $\boldsymbol{\mathcal{K}}$ with unit eigenvalue.

Note that, supposing that the state x_k at the time step k is in X_j, and neglecting the exact knowledge of x_k, the j-th row of $\boldsymbol{\mathcal{K}}$, by definition, represents the probability that the following state x_{k+1} belongs to any intervals of $\mathcal{X}_n$. This means that the evolution of the map, when considering only the interval of $\mathcal{X})n$ where the state is, i.e. the evolution of the system in Σ_n, can be modeled by the Markov chain associated to the stochastic matrix $\boldsymbol{\mathcal{K}}$. Note also that this is not an approximation of the evolution of the map, but comes from the exact analysis of the restriction of the evolution of the system in Σ_n.

Finally, the method to determine the exactness of PWAM maps relies on the following theorem [76]

THEOREM 2. *If M is a PWAM map and its kneading matrix is* primitive, *i.e., an integer l exists such that $\overline{\boldsymbol{\mathcal{K}}}^l$ has no null entries, then M is exact.*

A.5 Robustness of a chaotic map

Robustness is the property of a chaotic circuit to maintain its statistical features in presence of small perturbations. Regrettably, even very little errors,

unavoidable in any analog circuit implementation, as well as noise perturbations during operation, may not only prevent the statistics of the signals generated by the map to align the desired ones, but even prevent the map from achieving a chaotic behavior [37, 40, 45]. It is important to stress that every implementation of a chaotic circuit must be an analog implementation. As a consequence, the existence of maps with good statistical robustness properties is of great practical concern.

In this paragraph two main issues about chaotic map robustness are considered. The first one is known as *escape from principal invariant set*.

Consider the normalized domain $X = [-1, 1]$ and the a *Bernoulli* map

$$M(x) = \begin{cases} 2x + 1 & x < 0 \\ 2x - 1 & x \geq 0 \end{cases} \tag{A.5}$$

Note that the Bernoulli map is the simplest possible PWAM map, since it is build on a partition $\mathcal{X}_n$ composed only by two intervals, and it has a constant slope.

Given the partition $\mathcal{X} = \{[-1, 0[, [0, 1]\}$ of X, the associated kneading matrix is

$$\boldsymbol{\kappa} = \begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & \frac{1}{2} \end{pmatrix}$$

with eigenvector $\mathbf{e} = (1/2, 1/2)$, so the invariant density $\overline{\rho}$ of M is uniform in X. Note that the Bernoulli map has the same stochastic matrix as the fair coin toss.

A real implemented system working only on a closed interval such as it is X is not feasible. Many reasons could drive the input data out of their nominal definition sets, for example an external perturbation or just a wrong initialization; a real system accepts and elaborates these data, even if in this case the system response is uncontrolled. A practical implementation of a real system should consider also the response of the system for these unexpected data.

Suppose to extend the Bernoulli the map to the whole $\mathbb{R}$ in the simplest possible way, i.e. (A.5) stands not only in X but in all $\mathbb{R}$. Then suppose that the state of the system, at time state k, is $x_k = -1 - \varepsilon$, $\varepsilon > 0$. Such a state is not in X and it is normally not reachable; however, this may happen due to a noise perturbation, or even for a small error in the implementation of the map.

The evolution of the map, at the time steps $k + 1, k + 2, k + 3, \ldots$ is

$$-1 - 2\varepsilon, -1 - 4\varepsilon, -1 - 8\varepsilon, \ldots$$

i.e. the orbit has "escaped" from the principal invariant set X and is heading to $-\infty$ at exponential rate.

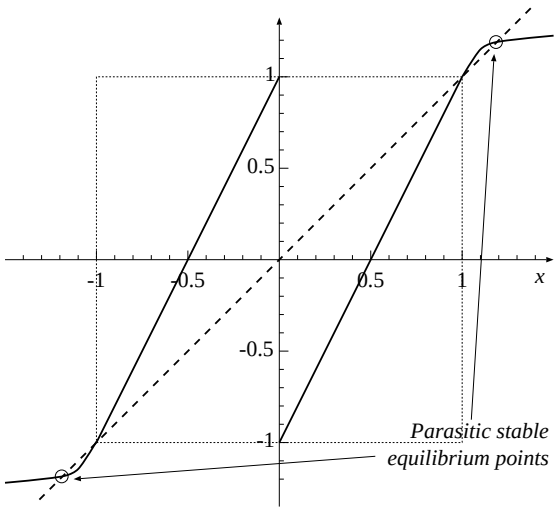


Figure A.2: Since saturation is unavoidable in any circuital implementation of a chaotic map, parasitic stable equilibrium points can exist.

A typical situation of a true implemented map is presented in Figure A.2. In that implementation, the map has been linearly extended out of X until saturation occurs. In this case the saturation creates two parasitic stable equilibrium points, which work as attractor every time the state escape from X.

To cope with this, a classical solution [40] is to add to the map suitably defined "hooks" i.e. to define the map even out from its definition set to reinject all the escaped states into X. In other words, it is necessary to create a *basin of attraction* B much greater than X. The basin of attraction is defined as the set of points that, after one or more iterations, are reinjected into X, i.e. for a sufficient large k it is $M^k(B) = X$. Mathematically

$$B = \left\{ x \,|\, \exists\, k \in \mathbb{N} : M^k(x) \in X \right\}$$

Another common problem is the loss of ergodicity, i.e. when due to implementation errors an invariant set $Y \subset X$ with $\mu(Y) \neq 0$ or $\mu(X \backslash Y) \neq 0$ exists. In this case Y is called *trap*, due to the fact that if the state reaches a point of Y, is trapped in Y forever. A trap is common to arise whenever a breakpoint of the map is close to a fixed point of M.

A.6 Noise Robustness in PWAM Maps

Let now assume that the disturbances present in a practical implementation of a chaotic map $M : X \to X$ with $X = [X^-, X^+] \subset \mathbb{R}$ can be modeled as a

discrete-time, time-independent and map-indipendent stochastic process with samples $\nu_k \in N = [N^-, N^+] \subset \mathbb{R}$, with, for sake of simplicity, $N^- < 0$ and $N^+ > 0$, which are drawn according to the probability density function ρ_ν.

Due the presence of such a superimposed noise, the map domain must be extended to $Y = [X^- + N^-, X^+ + N^+]$, so that the model of the evolution of the system, as described in (A.1), changes into

$$x_{k+1} = M_E(x_k + \nu_k) \ , \quad M_E : Y \to X \tag{A.6}$$

where the extended map M_E is such that $M_E(x) = M(x)$ for all $x \in X$. This assures that the unperturbed case (i.e. $\nu_k = 0 \ \forall k$) is consistent with the ideal evolution model.

REMARK 1. In (A.6) the codomain of M_E has been set in X; this is of course sufficient, but not strictly necessary, to ensure that, eventually, any escaped state is reinjected into X. However, for sake of simplicity, M_E has been considered as a function $Y \to X$.

REMARK 2. Instead of (A.6), one could have considered an additive map-independent noise as

$$x_{k+1} = M_E(x_k) + \nu_k \ , \quad M_E : Y \to X$$

It is easy to see that the two models are absolutely equivalent.

It is possible to introduce the companion dynamical system representing the statistical evolution of (A.6). If we indicate with $\mathbf{P}_E$ the PFO associated to M_E

$$[\mathbf{P}_E \phi](x) = \int_Y \phi(\xi)\, \delta(M_E(\xi) - x)\, d\xi \tag{A.7}$$

it is easy to prove that $\mathbf{P}$ is the restriction of $\mathbf{P}_E$ to densities whose support is X. Considering that the probability density function of the sum of two independent random variables is the convolution of their probability density functions, the companion system model of (A.6) is simply expressed as

$$\rho_{k+1} = \mathbf{P}_E(\rho_k * \rho_\nu)$$

where $*$ is the usual convolution operator.

However, due to the presence of the superimposed noise, a regular behavior of the system is not assured. In order to study system (A.6) through its companion dynamical system it is necessary to require that, like in the ideal case, for any choice of ρ_0 and for the considered ρ_ν, density ρ_k is convergent to a new density $\tilde{\rho}$, which is the the unique density function that solves

$$\tilde{\rho} = \mathbf{P}_E(\tilde{\rho} * \rho_\nu) \tag{A.8}$$

This assumption is necessary to study the noise perturbed system through its companion dynamical system and will be implicitly supposed in the following. Generally, $\tilde{\rho}$ is different from the invariant density $\overline{\rho}$ of the unperturbed system, depending on both M_E and ρ_ν. Obviously $\tilde{\rho}$ is equal to $\overline{\rho}$ in the unperturbed case, i.e. for $\rho_\nu = \delta(0)$.

DEFINITION 7. Let $M : X \to X$ be a chaotic map, and $M_E : Y \to X$ its extension in Y. The extended map M_E is noise robust if $\tilde{\rho}$ defined by (A.8) is equal to the unperturbed map invariant density $\overline{\rho}$, for any choice of ρ_ν.

Hence, it is possible to prove the robustness of an extended chaotic map M_E by simply verifying that

$$\overline{\rho} = \mathbf{P}_E\left(\overline{\rho} * \rho_\nu\right), \quad \forall \rho_\nu \tag{A.9}$$

LEMMA 1. *Let $M : X \to X$ be a chaotic map, and $M_E : Y \to X$ its extension in Y. Let also $\overline{\rho}$ be the unperturbed system invariant density and $\Delta\overline{\rho}^{\varepsilon}(x) = \overline{\rho}(x) - \overline{\rho}(x-\varepsilon)$ the difference between shifted and non-shifted invariant density. The extended map M_E is noise robust if and only if $\boldsymbol{P}_E \Delta\overline{\rho}^{\varepsilon} = 0 \, \forall \varepsilon \in N$.*

Proof. For the necessary condition, it is enough to consider the linearity of $\boldsymbol{P}_E$ and a constant noise perturbation, i.e. a delta-like noise density ρ_ν.

For the sufficiency, it is useful to introduce the shifted invariant density $\overline{\rho}_T^{\varepsilon} = \overline{\rho}(x-\varepsilon)$ so, from the linearity of PFO

$$\boldsymbol{P}_E \overline{\rho} = \boldsymbol{P}_E \overline{\rho}_T^{\varepsilon}, \; \forall \varepsilon \in N$$

i.e. $\boldsymbol{P}_E \overline{\rho}_T^{\varepsilon} = \overline{\rho} \, \forall \varepsilon$, and then (A.9) can be verified, namely

$$\begin{aligned} \boldsymbol{P}_E\left[\overline{\rho} * \rho_\nu\right](x) &= \boldsymbol{P}_E\left[\int_N \rho_\nu(\xi)\, \overline{\rho}(x-\xi)\, d\xi\right] \\ &= \int_N \rho_\nu(\xi)\, \boldsymbol{P}_E \overline{\rho}_T^{\xi}(x)\, d\xi = \int_N \rho_\nu(\xi)\, d\xi\, \overline{\rho}(x) = \overline{\rho}(x) \end{aligned}$$

□

Using the integral definition of $\boldsymbol{P}_E$ in (A.7) the condition from Lemma 1 can be rewritten as

$$\int_Y \Delta\overline{\rho}^{\varepsilon}(\xi)\, \delta\left(M_E(\xi) - x\right) d\xi = 0, \quad \forall x \in X, \; \forall \varepsilon \in N \tag{A.10}$$

Although of general applicability, condition (A.10) is again relatively difficult to use in practice, since it does not show an explicit dependence on the structure of the map. A simpler robustness condition can be obtained restricting the attention over maps with piecewise constant $\overline{\rho}$, thus including all PWAM maps.

THEOREM 3. *Let $M : X \to X$ be a chaotic map with a piecewise constant invariant density that can be expressed as*

$$\overline{\rho}(x) = \sum_{k=1}^{m} \beta_k \chi_{\left[X_k^-,\, X_k^+\right]}(x) \tag{A.11}$$

where, for $k = 1, \ldots, m$, $\left[X_k^-,\, X_k^+\right]$ is an arbitrary partition of X and β_k are suitable coefficients. The extended map M_E is noise robust if and only if

$$\sum_{k=1}^{m+1} (\beta_k - \beta_{k-1})\, \delta\left(M_E\left(X_k^- + \varepsilon\right) - x\right) = 0\,, \quad \forall x \in X,\ \forall \varepsilon \in N \tag{A.12}$$

where, for convenience, $\beta_0 = \beta_{m+1} = 0$ and $X_{m+1}^- = X_m^+$.

Proof. From (A.11)

$$\Delta\overline{\rho}^{\,\varepsilon}(x) = \sum_{k=1}^{m+1} (\beta_k - \beta_{k-1})\, \chi_{\left[X_k^-,\, X_k^- + \varepsilon\right]}(x)$$

where $\varepsilon > 0$ has been assumed (the case $\varepsilon < 0$ can be treated similarly). Hence, (A.10) can be rewritten as

$$\begin{aligned} &\int_Y \sum_{k=1}^{m+1} (\beta_k - \beta_{k-1})\, \chi_{\left[X_k^-,\, X_k^- + \varepsilon\right]}(\xi)\, \delta\left(M_E(\xi) - x\right)\, d\xi = \\ &\qquad \sum_{k=1}^{m+1} (\beta_k - \beta_{k-1}) \int_{\left[X_k^-,\, X_k^- + \varepsilon\right]} \delta\left(M_E(\xi) - x\right)\, d\xi = 0 \end{aligned} \tag{A.13}$$

Conditions (A.12) and (A.13) are equivalent.

If (A.12) is true, then (A.13) is certainly verified. Also, supposing (A.13) is verified and deriving both its terms

$$\begin{aligned} \frac{d}{d\varepsilon} \sum_{k=1}^{m+1} (\beta_k - \beta_{k-1}) &\int_{\left[X_k^-,\, X_k^- + \varepsilon\right]} \delta\left(M_E(\xi) - x\right)\, d\xi = \\ &\sum_{k=1}^{m+1} (\beta_k - \beta_{k-1})\, \delta\left(M_E\left(X_k^- + \varepsilon\right) - x\right) = 0 \end{aligned}$$

the condition (A.12) is verified.

□

Even if a discussion on the general case is possible (see [1] for it), it is interesting to consider the simplest possible case, i.e. a map with $\overline{\rho}$ uniform in X. In this case $\overline{\rho}(x) = \beta \chi_{[X^-,\, X^+]}(x)$ and, since $\beta \neq 0$, condition (A.12) can be recast as

$$\delta\left(M_E\left(X^- + \varepsilon\right) - x\right) - \delta\left(M_E\left(X^+ + \varepsilon\right) - x\right) = 0, \quad \forall\, x \in X,\ \forall \varepsilon \in N$$

and, due to Dirac generalized function properties can be easily verified that the condition is equivalent to say

$$M_E\left(X^- + \varepsilon\right) = M_E\left(X^+ + \varepsilon\right), \quad \forall \varepsilon \in N$$

This means that a map M_E is noise robust in N if and only if M_E is the restriction in Y of a periodic function, with period $\Pi = X^+ - X^- = \mu(X)$.

As an example, the only robust extension of the Bernoulli map (A.5) is

$$M(x) = \begin{cases} 2x + 3 & x < -1 \\ 2x + 1 & -1 \leq x < 0 \\ 2x - 1 & 0 \leq x < 1 \\ 2x - 3 & x \geq 1 \end{cases}$$

with $N = [-1, 1]$ and $Y = [-2, 2]$.

Note that the complexity of the extended map is increased, since now two additional branches are necessary.

Appendix B

Hardware and Algorithms used in this dissertation

IN THIS APPENDIX a brief overview of others physical based random number generators and pseudorandom generators used in this dissertation, as well as the hardware used for the data acquisition of the two implemented prototypes, is provided.

B.1 BBS Pseudorandom Generator

The Blum-Blum-Shub (BBS) is a pseudorandom number generator proposed in 1986 by Lenore Blum, Manuel Blum and Michael Shub [35].

It takes the form:

$$x_{n+1} = (x_n)^2 \bmod M$$

where $M = pq$ is the product of two large primes p and q. At each step of the algorithm, some output is derived from x_n; the output is commonly either the bit parity of x_n or one or more of the least significant bits of x_n.

The two primes, p and q, should both be congruent to 3 modulo 4 (remember that two integers a and b are called congruent modulo n if $a - b$ is divisible by n, or, equivalently, if they give the same remainder when divided by n); this guarantees that each quadratic residue has one square root which is also a quadratic residue. Also, the greatest common divisor between the totient function of $p - 1$ and $q - 1$ (the totient function of n is the number of positive integers less than n which are relatively prime to n) should be small; this makes the cycle period large.

The generator is appropriate for use in cryptography, because it is possible to prove its security. The security proof relates the quality of the generator to the computational difficulty of integer factorization. When the primes are chosen appropriately, and $O(\log_2 \log_2 M)$ lower-order bits of each x_n are considered as system output, then in the limit as M grows large, distinguishing the output bits from random will be at least as difficult as factoring M. Thus the primes p and q need to be large enough such that it computationally infeasible to factor N. As of today, this means that p and q should be at least 512 bits. This, however, makes the generator very slow, since every operation requires a high precision arithmetic module.

The code used is the following, which uses the GNU multiprecision library for achieving operations with an arbitrary precision.

```
/**************************************************************************/
/* Algoritmo de Blum-Blum-Shub para generación de números pseudoaleatorios. */
/*                                                                        */
/* Jaime Suarez <mcripto@bigfoot.com> 2003                                */
/* en http://www.matematicas.net                                          */
/**************************************************************************/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "gmp.h"

/* Este es el tamaño aproximado del módulo n */
#define BITS_MODULO 1024

/* Inicializa el generador de números aleatorios, s es una cadena con un
 * número en base 10 que sirve de semilla. Es importante que sea lo más
 * aleatorio posible y de un tamaño similar al de n */
void iniciarBBS(char *s);

/* Devuelve un bit aleatorio de un generador ya inicializado */
int bitBBS(void);

/* Devuelve un byte aleatorio de un generador BBS ya inicializado */
int byteBBS(void);


/*
 * Comienzo del programa
 */

int main(int argc, char *argv[])
{
        int i, nBytes;
        unsigned char s[256];
        FILE *fo;

        if (argc!=3) {
         printf("-------------------------------------------------------\n");
         printf(" Generador de números pseudoaleatorios Blum-Blum-Shub.\n\n");
         printf(" Uso: %s <n> <s>\n",argv[0]);
         printf(" n: numero de bytes deseados.\n");
         printf(" s: semilla, cadena de caracteres arbitrarios.\n");
```

```
         printf("-----------------------------------------------------\n");
         return 1;
        }

        /* Leer línea de comandos */
        nBytes=atoi(argv[1]);
        strncpy(s,argv[2],256);
        /* Convertir la cadena s en un entero */
        for (i=0; i<strlen(s); i++) s[i]=(s[i]%10)+'0';

        /* Inicializa generador y produce los bytes pedidos
         * guardándolos en bbs.out */
        iniciarBBS(s);
        fo = fopen("bbs.out", "wb");
        for (i=0; i<nBytes; i++) {
                fprintf(fo, "%c", byteBBS());
        }
        fclose(fo);
        puts("Resultados en el fichero: bbs.out");
        return 0;
}

mpz_t x;                        /* Último valor aleatorio */
mpz_t n;                        /* Módulo para BBS */

/*
 *  Inicia el generador de numeros aleatorios a partir de la cadena s
 *  que contiene un entero en base 10 que sirve como semilla.
 */
void iniciarBBS(char *s)
{
        mpz_t p, q, tmp;
        gmp_randstate_t estado;

        /* Inicializar rng de la librería gmp3 */
        gmp_randinit_default(estado);
        mpz_set_str(tmp, s, 10);
        gmp_randseed(estado, tmp);

        /* Inicializar enteros largos */
        mpz_init(x); mpz_init(n); mpz_init(p); mpz_init(q);

        /* Tenemos que generar n como producto
         * de dos grandes primos congruentes con 3 modulo 4 */
        do {
                mpz_urandomb(p, estado, BITS_MODULO/2);
                mpz_mul_ui(p,p,4);
                mpz_add_ui(p,p,3);

        } while (mpz_probab_prime_p(p,25)==0);

        do {
                mpz_urandomb(q, estado, BITS_MODULO/2);
                mpz_mul_ui(q,q,4);
                mpz_add_ui(q,q,3);

        } while (mpz_probab_prime_p(q,25)==0);
        mpz_mul(n,p,q);

        /* Ahora se produce la primera x = s^2 (mod n)  */
        mpz_set_str(x,s,10);
        mpz_mod(x,x,n);
        mpz_mul(x,x,x);
```

```
        mpz_mod(x,x,n);

        /* Limpiamos variables innecesarias en lo sucesivo */
        mpz_clear(p); mpz_clear(q);
        return;
}

/*                                                                    */
/*  Genera un bit pseudoaleatorio a partir de la variable global x    */
/*  previamente existente. Es necesario llamar a iniciarBBS antes de  */
/*  utilizarlo.                                                       */

int bitBBS(void)
{
        mpz_mul(x,x,x);
        mpz_mod(x,x,n);                 /* x = x^2 mod n                    */

        return mpz_tstbit(x, 0); /* devolver el bit menos significativo      */
}

/*
 * Devuelve un byte pseudoaleatorio. Debe llamarse una vez a iniciarBBS
 * antes de comenzar a pedir bytes.
 */
int byteBBS(void)
{
        int byte=0, i;

        for (i=0; i<8; i++)
                byte = byte*2 + bitBBS();

        return byte;
}
```

B.2 KISS Pseudorandom Generator

The KISS principle is a colloquial name for the empirical principle that simplicity is an essential asset and goal in any systems. It is popular in software and engineering in general. The term KISS is an acronym, corresponding in origin to the phrase “keep it simple, stupid”.

This generator was proposed by George Marsaglia as the combination of some easy pseudorandom generators, in order to get a much more complex behavior with respect to the original generators, while at the same time to maintain a very simple architecture. Many variants of this generator exist; the cose used here is the C/C++ code written by Marsaglia as a porting of the 16 bits Fortran code to 32 bits processors.

```
unsigned long KISS() {
static unsigned long x=123456789,
    y=362436, z=521288629, c=7654321;
unsigned long long t, a=698769069LL;
x=69069*x+12345;
y^=(y<<13);  y^=(y>>17);  y^=(y<<5);
t=a*z+c;  c=(t>>32);
return x+y+(z=t);  }
```

The generator is the combination of three simple generators:

- a congruential generator (represented by the variable `x` in the code)
- a 3-shift generator (the variable `y`)
- a multiply-with-carry generator the variable `z`

and the achieved period is about 2^{124} bits.

B.3 VIA PadLock Random Generator

All new VIA Technologies, Inc. processors (C3, C5P and the new C7) include VIA PadLock Security Engine, that is a block of hardware primitives designed to implement many security-related features. The VIA Padlock is composed by the VIA PadLock ACE (Advanced Cryptography Engine) and the VIA PadLock RNG [42]. The RNG is based on the jitter of two (the first one very fast, the second slow) free running oscillators; the slow oscillator is used to sample the fast one. Since the two oscillators cannot reach a synchronization, the sampled values depend on the *jitter* of the slow one, thus generating random bits. These random bits are post-processed with a von Neumann algorithm.

A short extract from VIA website is here reported.

> To address this need for good random numbers in security applications, VIA introduced the Nehemiah processor core in January 2003 that included the VIA Padlock RNG, integrating a high-performance hardware-based random number generator onto the processor die. The VIA PadLock RNG uses random electrical noise on the processor chip to generate highly random values at an extremely fast rate. It provides these numbers directly to security applications via a unique x86 instruction that has built-in multi-tasking support.
>
> Capable of creating random numbers at rates of between 800K to 1600K bits per second, the VIA PadLock RNG addresses the needs of security applications requiring high bit rates that algorithmically increases the quality (randomness) of the entropy produced, for example by applying hashing algorithms to the output.

> The VIA PadLock RNG uses a system of Asynchronous Multi-byte Generation, where the hardware generates random bits at its own pace. These accumulate into hardware buffers with no impact on program execution. Software may then read the accumulated bits at any time. This asynchronous approach allows the hardware to generate large amounts of random numbers completely overlapped with program execution. This is opposed to good software generators, which can be fast but consume a significant number of CPU cycles and have a negative affect on affecting overall system performance.

The processor used for testing was a VIA C3 1 GHz in an EPIA MX-II 10000 board. The generator speed was measured in about 1 Mbit/s.

B.4 Quantis Random Generator

The Quantis random generator by idQuantique SA, Geneva (CH), is a a random number generator based on the reflection of a single photon on a semi-transparent mirror [53]. Its characteristics can be found in the original idQuantique flyer.

> Quantis is a physical random number generator exploiting an elementary quantum optics process. Photons - light particles - are sent one by one onto a semi-transparent mirror and detected. The exclusive events (reflection - transmission) are associated to ”0” - ”1” bit values. The operation of Quantis is continuously monitored to ensure immediate detection of a failure and disabling of the random bit stream.
>
> Quantis is available as an OEM component for mounting on a printed circuit board, as a PCI card, and now also as a USB module. It comes with drivers for the main operating system platforms. Quantis is easily integrated in existing applications.
>
> Features
>
> - True quantum randomness (passes all randomness tests)
> - High bit rate of 4Mbits/sec (up to 16Mbits/sec for PCI card)
> - Low-cost device, compact and reliable
> - Continuous status check
> - PCI card comes with drivers for Windows (2000/XP), Linux (2.4, 2.6), FreeBSD (4, 5, 6) and Solaris (8, 9, 10 for SPARC, x86 and x64). A console application and a library for developpers are supplied. Moreover a Windows-based graphical application is supplied that acquires

random data in several formats and stores them in a file. A Labview VI is also available.

- USB module comes with drivers for: Windows(2000/XP) and Linux (2.4, 2.6). A console application and a library for developpers are supplied. Moreover a Windows-based graphical application is supplied that acquires random data in several formats and stores them in a file. A Labview VI is also available.

The version used is the PCI card, with a single module installed, capable of a speed of 4 Mbit/s.

B.5 Data Acquisition and Testing Procedure

To acquire data from the two designed prototypes, a specific hardware was necessary. This is mainly due to two reasons: *(a)* data come with a very high throughput, up to 100 Mbit/s for the 180 nm prototype; and *(b)* a very large amount of data need to be memorized, since many DieHard tests require about 80 Mbit (after the decimation introduced by the post-processing) to be performed.

These reasons pointed at the usage of a high-speed PCI acquisition card for PC. The card used is a PCI-1755 card from Advantech Co., Ltd. A brief description of its features ia available from Advantech website.

PCI-1755 Ultra-speed 32-ch Digital I/O Card Main Features:

- Bus-mastering DMA data transfer with scatter gather technology
- 32/16/8-bit Pattern I/O with start and stop trigger function, 2 modes Handshaking I/O Interrupt hand
- On-board active terminators for high speed and long distance transfer
- Pattern match and Change state detection interrupt function
- General-purpose 8-ch DI/O

The PCI-1755 supports PCI-bus mastering DMA for high-speed data transfer. By setting aside a block of memory in the PC, the PCI-1755 performs bus-mastering data transfers without CPU intervention, setting the CPU free to perform other more urgent tasks such as data analysis and graphic manipulation. The function allows users to run all I/O functions simultaneously at full speed without losing data.

This card should allow a maximum burst transfer rate of 30 Mword/s, with words up to 32 bits, and a continuous transfer speed of about 20 Mword/s.

Actually, a maximum continuous transfer speed of about 15 Mword/s was detected. Of course this speed, equivalent to a transfer rate of nearly half a Gbit/s, was sufficient for the purpose. However, due to the higher speed and the lower parallelism of the output data from the random generators, a high-speed serial to parallel converter was designed with standard high speed logic circuitry and used for data acquisition.

Data acquired from the prototypes have been memorized into 1.5 Gbits (i.e. 192 Mbytes) files, each file corresponding to a single, non-interrupted run of the generator. From each files, sequences of adequate length were extracted and post-processed, and then tested. The code used for the test was exactly line-by-line the code distributed by NIST (or by Marsaglia for the DieHard test suite); only some additional code has been written to interface via stdin/stdout the original NIST/Marsaglia code, thus providing a full automatization for the extract/post-processing/testing procedure. The tests were performed on a 64 CPUs cluster, featuring Intel Xeon 2400 CPUs.

Publications

Journal Publications

[1] FABIO PARESCHI, Riccardo Rovatti, and Gianluca Setti, "Periodicity as Condition to Noise Robustness for Chaotic Maps with Piecewise Constant Invariant Density", in *International Journal on Bifurcation and Chaos*, to appear in vol. 16, no. 11, November 2006.

Internation Conference Publications

[2] FABIO PARESCHI, Riccardo Rovatti, and Gianluca Setti "Second Level NIST Randomness Test for Improving Test Reliability", to appear in *Proceedings of 2007 IEEE International Symposium on Circuits and Systems* (ISCAS2007). New Orleans (USA), May 27–30, 2007.

[3] FABIO PARESCHI, Gianluca Setti, and Riccardo Rovatti "A Fast Chaos-based True Random Number Generator for Cryptographic Applications", in *Proceedings of 26th IEEE European Solid-State Circuit Conference* (ESSCIRC2006), pp. 130–133. Montreux (Switzerland) September 11–14, 2006.

[4] Luca Antonio De Michele, FABIO PARESCHI, Riccardo Rovatti, and Gianluca Setti "3 GHz Spread Spectrum Clock Generator for Serial ATA-II using Random Frequency Modulation", in *Proceedings of 2006 International Symposium on Nonlinear Theory and its Applications* (NOLTA2006), pp. 635–638. Bologna (Italy), September 11–14, 2006.

[5] FABIO PARESCHI, Riccardo Rovatti, and Gianluca Setti "Simple and Effective Post-Processing Stage for Random Stream Generated by a Chaos-Based RNG", in *Proceedings of 2006 International Symposium on Nonlinear Theory and its Applications* (NOLTA2006), pp. 383–386. Bologna (Italy), September 11–14, 2006.

[6] Michele Balestra, FABIO PARESCHI, Gianluca Setti, and Riccardo Rovatti "Design of a Low EMI Hysteretic Current-Controlled DC/DC Boost Converter Via Chaotic Perturbation", in *Proceedings of 2006 International Symposium on Nonlinear Theory and its Applications* (NOLTA2006), pp. 259–262. Bologna (Italy), September 11–14, 2006.

[7] Luca Antonio De Michele, FABIO PARESCHI, Riccardo Rovatti, and Gianluca Setti, "Chaos-based High-EMC Spread-Spectrum Clock Generator", in *Proceedings of 17th IEEE European Conference on Circuit Theory and Design* (ECCTD 2005), pp. 165–168. Cork (Ireland), August 29 – September 2, 2005. ***Winner of the best paper award.***

[8] FABIO PARESCHI, Gianluca Setti, and Riccardo Rovatti, "A macro-model for the efficient simulation of an ADC-based RNG", in *Proceedings of 2005 IEEE International Symposium on Circuits and Systems* (ISCAS2005), pp. 4349–4353. Kobe (Japan), May 23–26, 2005.

[9] FABIO PARESCHI, Gianluca Setti, and Riccardo Rovatti, "Noise Robustness condition for chaotic maps with Piecewise constant invariant density", in *Proceedings of 2004 IEEE International Symposium on Circuit and Systems* (ISCAS2004), vol. IV, pp. 681–684. Vancouver (Canada), May 23–26, 2004.

[10] FABIO PARESCHI, Luca Antonio De Michele, Riccardo Rovatti, and Gianluca Setti, "A PLL-based clock generator with improved EMC", in *Proceedings of 16th IEEE International Zurich Symposium on Electromagnetic Compatibility* (EMCZurich2005), pp. 367–372. Zurich (Swiss), February 13–18, 2005. ***Winner of the best student paper award.***

[11] Luca Antonio De Michele, FABIO PARESCHI, Riccardo Rovatti, and Gianluca Setti, "A chaos-driven PLL based spread spectrum clock generator", in *Proceedings of 2004 International Symposium on Nonlinear Theory and its Applications* (NOLTA2004), pp. 251–254. Fukuoka (Japan), November 29 – December 3, 2004.

Bibliography

References are organized in sections clustering them into homogeneous topics. Within each section, references are sorted chronologically (for the historical and general purpose references section) or alphabetically (for all other sections) according to authors' name.

Historical and General Purpose References

[12] R. Brown, "A brief account of microscopical observations made in the months of June, July, and August, 1827 on the particles contained in the pollen of plants; and on the general existence of active molecules in organic and inorganic bodies", in *Philosophical Magazine*, vol. 4, pp. 161–173, 1828.

[13] K. Pearson, "On the criterion that a given system of deviations from the probable in the case of a correlated system of variables is such that it can be reasonably supposed to hove arisen from random sampling", in *Philosophical Magazine*, no. 50, pp. 157–172, 1900.

[14] A. Einstein, "Über einen die erzeugung und verwandlung des lichtes betreffenden heuristischen gesichtspunkt" (On a heuristic point of view concerning the production and transformation of Light), in *Annalen der Physik*, vol. 17, pp. 132–148, 1905.

[15] A. Einstein, "Über die von der molekularkinetischen theorie der wärme geforderte bewegung von in ruhenden flüssigkeiten suspendierten teilchen" (On the movement of small particles suspended in a stationary liquid demanded by the molecular-kinetic theory of heat), in *Annalen der Physik*, vol. 17, pp. 549–560, 1905.

[16] A. Einstein, "Zur elektrodynamik bewegter körper" (On the electrodynamics of moving bodies), in *Annalen der Physik*, vol. 17, pp. 891–921, 1905.

[17] A. Einstein, "Ist die trägheit eines körpers von seinem energieinhalt abhängig?" (Does the inertia of a body depend upon its energy?), in *Annalen der Physik*, vol. 18, pp. 639–641, 1905.

[18] A. Einstein, "Zur theorie der Brownschen bewegung" (On the theory of Brownian motion), in *Annalen der Physik*, vol. 19, pp. 371–381, 1906.

[19] J. Perrin, "Les Atoms" (The Atoms), Librairie Felix Alcan, Paris, 1913.

[20] S. M. Ulam and J. von Neumann, "On combination of stochastic and deterministic process", in *Bulletin of American Mathematical Society*, no. 53, pp 1120–1132, 1947.

[21] C. E. Shannon, "A Mathematical Theory of Communication", in *The Bell system technical journal*, vol. 27, pp. 379–423, July 1948.

[22] F. J. Massey Jr., "The Kolmogorov-Smirnov test of goodness of fit", in *Journal of the American Statistical Association*, no. 46, pp. 68–78, 1951.

[23] J. von Neumann, "Various Technique Used in Connection with Random Digits", in *Applied Math Series*, notes by G. E. Forsythe, National Bureau of Standards, no. 12, pp. 36–38, 1951.

[24] RAND Corporation, "A Million Random Digits with 100,000 Normal Deviates", Free Press, New York, 1955.

[25] R. E. Kalman, "Nonlinear aspects of sampled-data control systems", in *Proceedings of Symposium of Nonlinear Circuit Analysis*, vol. VI, pp. 273–313. New York (USA), April 25–27, 1956.

[26] M. Klamkin, and D. Newman, "Extensions of the birthday surprise", in *Journal of Combinatorial Theory* no. 3, pp. 279–282, 1967.

[27] P. van Beeck, "An application of Fourier methods to the problem of sharpening the Berry-Esseén inequality", in *Probability Theory and Related Fields*, vol. 23, no. 3, pp. 187–196, September 1972.

[28] D. Bloom, "A birthday problem" in *American Mathematical Monthly* no. 80, pp. 1141–1142, 1973.

[29] I. S. Shiganov, "Refinement of the upper bound of the constant in the central limit theorem", in *Journal of Mathematical Sciences*, vol. 35, no. 3, pp. 2545–2551, November 1986.

[30] W. H. Press, B. P. Flannery, S. A. Teukolsky, and W. T. Vetterling, "Numerical Recipes in C: The Art of Scientific Computing", Cambridge University Press, 1992. Available at `http://www.nrbook.com/a/bookcpdf.php`

[31] A. N. Shiryaev, and R. P. Boas "Probability" (Graduate Texts in Mathematics), Springer-Verlag, 1995.

[32] D. E. Knuth, "The Art of Computer Programming", Volume 2: *Seminumerical Algorithms* (3rd edition), Addison-Wesley Professional, 1997.

Chaos and Random Number Related References

[33] A. M. Abo, P. R. Gray, "A 1.5-V, 10-bit, 14.3-MS/s CMOS Pipeline Analog-to-Digital Converter", in *IEEE Journal of Solid-State Circuits*, vol. 34, no. 5, pp. 599–606, May 1999.

[34] G. M. Bernstein, and M. A. Lieberman, "Secure Random Number Generation using Chaotic Circuit", in *IEEE Transaction on Circuit and Systems I: Fundamental Theory and Applications*, vol. 47, no. 9, pp. 1157–1164, September 2000.

[35] L. Blum, M. Blum, and M. Shub, "A Simple Unpredictable Pseudo-Random Number Generator", in *SIAM Journal on Computing*, vol. 15, pp. 364–383, May 1986.

[36] M. Blum, "Independent unbiased coin flips from a correlated biased source – A finite Markov chain", in *Combinatoria*, vol. 6, no.2, pp 97–108, 1986.

[37] S. Callegari, and R. Rovatti, "Analog chaotic maps with sample-and-hold errors", in *IEICE Transaction on Fundamentals of Electronics, Communications and Computer Sciences*, vol E82A, no. 9, pp. 1754-1761, September 1999.

[38] S. Callegari, R. Rovatti and G. Setti, "Efficient chaos-based secret key generation method for secure communications" in *Proceedings of 2002 International Symposium on Nonlinear Theory and its Applications* (NOLTA2002), Xi'an, China, October 7–11, 2002.

[39] S. Callegari, R. Rovatti, and G. Setti, "Embeddable ADC-Based True Random Number Generator for Cryptographic Applications Exploiting Nonlinear Signal Processing and Chaos", in *IEEE Transaction on Signal Processing*, vol. 53, no. 2, pp. 793–805, February 2005.

[40] S. Callegari, G. Setti, and R. Rovatti, *Robustness of Chaos in Analog Implementations*, Chapter 12 in M. P. Kennedy *et al.* "Chaotic Electronics in Telecommunications", pp. 397–442, CRC International, 2000.

[41] T. B. Cho, and P. R. Gray, "A 10 b, 20 Msample/s, 35mW Pipeline A/D Converter", in *IEEE Journal of Solid-State Circuits*, vol. 30, no. 3, pp. 166–172, March 1995.

[42] Cryptography Research, "Evaluation of VIA C3 Nehemiah Random Number Generator", white paper prepared by Cryptography Research, Inc., San Farncisco (USA). February 27, 2003. Available at `http://www.cryptography.com/resources/whitepapers/VIA_rng.pdf`

[43] J. Daemen, and V. Rijmen, *The Design of Rijndael: AES - The Advanced Encryption Standard* Springer-Verlag, 2002.

[44] M. Delgado-Restituto, F. Medeiro, and A. Rodríguez-Vázquez, "Nonlinear Switched-Current CMOS IC for Random Signal Generation", in *Electronics Letters*, vol. 29, pp. 2190–2191, December 1993.

[45] M. Delgado-Restituto, and A. Rodríguez-Vázquez, "Integrated chaos generator", in *Proceedings of the IEEE*, special issue on "Applications of Nonlinear Dynamics to Electronic and Information Engineering", vol. 90, no. 5, pp. 747-767, May 2002.

[46] R. Devaney, *An introduction to Chaotic Dynamical System*, Addison-Wesley, (Second Edition) 1989.

[47] D. E. Eastlake, S. D. Crocker, and J. I. Shiller, "RFC 1750: Randomness recommendation for security" in *Internet Society Request for Comments*, Internet Engineering Task Force, December 1994.

[48] D. E. Eastlake, and P. E. Jones, "RFC 3174: US Secure Hash Algorithm 1 (SHA1)" in *Internet Society Request for Comments*, Internet Engineering Task Force, September 2001.

[49] R. C. Fairfield, R. L. Mortenson, and K. B. Coulthard, "An LSI random number generator (RNG)", in *Advances in Cryptology - Proceedings of Crypto'84*, pp. 203-230, Springer-Verlag, 1984.

[50] K. Hamano, "The Distribution of the Spectrum for the Discrete Fourier Transform Test included in SP800-22", in *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, vol. E88, no. 1, pp. 67–73, January 2005.

[51] F. Hofbauer, G. Keller, *Ergodic Properties of Invariant Measures for Piecewise Monotonic Transformations*, Mathematische Zeitschrift, Springer-Verlag, vol. 180, no. 1, pp. 119–140, March 1982.

[52] W. T. Holman, J. A. Connelly, and A. B. Downlatadadi, "An Integrated Analog/Digital Random Noise Source", *IEEE Transaction on Circuit and Systems I: Fundamental Theory and Applications*, vol. 44, no. 6, pp. 521-528, June 1997.

[53] idQuantique, "Random Numbers Generation using Quantum Physics" white paper, 2004. Available at `http://www.idquantique.com/products/files/quantis-whitepaper.pdf`

[54] B. Jun and P. Kocher, "The Intel Random Number Generator", Crypt. Reasearch, Inc. white paper prepared by Cryptography Research, Inc. for Intel Corp., April 1999. Available at `http://www.cryptography.com/resources/whitepapers/IntelRNG.pdf`

[55] A. Johansson, and F. Heinrik, "Random number generation by chaotic double scroll oscillator on chip", in *Proceedings of 1999 IEEE International Symposium on Circuits and Systems* (ISCAS1999), vol. 5, pp. 407–409. Orlando (USA), May 30 – June 2, 1999.

[56] S. Kim, K. Umeno, A. Hasegawa, "On NIST Statistical Test Suite for Randomness", in *IEICE Technical Report*, Vol. 103, no. 449, pp. 21-27, 2003.

[57] B. P. Kitchens, *Symbolic Dynamics*, Springer-Verlag, 1998.

[58] T. Kohda, "Information Sources Using Chaotic Dynamics" in *Proceedings of the IEEE*, special issue on "Applications of Nonlinear Dynamics to Electronic and Information Engineering", vol 90, no. 5, pp. 641–66, May 2002.

[59] T. Kohda, and A. Tsundea, *Information Sources using Chaotic Dynamics*, Chapter 4 in M. P. Kennedy *et al.* "Chaotic Electronics in Telecommunications", CRC International, 2000.

[60] A. Lasota, and M. C. Mackey, *Chaos, Fractals, and Noise. Stochastic Aspects of Dynamics*, Springer-Verlag, 1994.

[61] A. Lasota, J. A. Yorke, "On the Existence of Invariant Measure for Piecewise Monotonic Transformations", in *Transactions of the American Mathematical Society*, vol. 186, pp 481-488, December 1973.

[62] G. Marsaglia, "The Marsaglia Random Number CD-ROM including the DieHard Battery of test of randomness". Available at `http://stat.fsu.edu/pub/diehard/`

[63] G. Marsaglia, "The diehard test suite", 2003. Available at `http://www.csis.hku.hk/~diehard/`

[64] G. Marsaglia, and A. Zaman, "The KISS generator", Technical Report, Department of Statistics, University of Florida, 1993.

[65] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996.

[66] Nevada Gaming Commission and State Gaming Control Board, "Gaming Statutes and Regulations". Available at `http://gaming.nv.gov/stats_regs.htm`

[67] National Institute of Standard and Technology, "Data Encription Standard", Federal Information Processing Standard (FIPS) publication 43-3, October 25, 1999. Available at `http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf`

[68] National Institute of Standard and Technology, "Security requirements for cryptographic modules," Federal Information Processing Standards 140-2, December 3, 2002. Available at `http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf`

[69] National Institute of Standard and Technology, "Advance Encription Standard", Federal Information Processing Standard (FIPS) publication 197, November 26, 2001. Available at `http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf`

[70] National Institute of Standard and Technology, "A statistical test suite for random and pseudorandom number generators for cryptographic applications", Special Publication 800-22, May 15, 2001. Available at `http://csrc.nist.gov/rng/SP800-22b.pdf`

[71] National Institute of Standard and Technology, "Random Number Generation and Testing". Available at `http://csrc.nist.gov/rng/`

[72] E. Ott, *Chaos in Dynamical Systems*, Cambridge University Press, 1993.

[73] S. Poli, S. Callegari, R. Rovatti, G. Setti, "Post-Processing of data generated by a chaotic pipelined ADC for the robust generation of perfectly random bitstreams", in *Proceedings of 2004 IEEE International Symposium of Circuit and Systems* (ISCAS2004), vol. IV, pp. 585-588. Vancouver (Canada), May 23–26, 2004.

[74] B. Razavi, *Principles of Data Conversion System Design*, Wiley-IEEE Press, November 1994.

[75] R. Rivest, "RFC 1321: The MD5 Message-Digest Algorithm" in *Internet Society Request for Comments*, Internet Engineering Task Force, April 1992.

[76] R. Rovatti, G. Setti, G. Mazzini, "Chaotic Complex Spreading Sequences for Asynchronous CDMA - Part II: Some Theoretical Performance Bounds", *IEEE Transaction on Circuit and Systems I: Fundamental Theory and Applications*, vol. 45, no. 4, pp. 496-505, April 1998.

[77] G. Setti, G. Mazzini, R. Rovatti, and S. Callegari, "Statistical modeling of discrete time chaotic processes: Basic finite dimensional tools and applications", in *Proceedings of the IEEE*, special issue on "Applications of Nonlinear Dynamics to Electronic and Information Engineering", vol. 90, no. 5, pp. 662-690, May 2002.

[78] T. Stojanovski, and L. Kocarev, "Chaos-Based Random Number Generators - Part I: Analysis", in *IEEE Transaction on Circuit and Systems I: Fundamental Theory and Applications*, vol. 38, no. 3, pp. 281-288, March 2001.

[79] T. Stojanovski, J. Pihl, and L. Kocarev, "Chaos-Based Random Number Generators - Part II: Practical Realization", in *IEEE Transaction on Circuit and Systems I: Fundamental Theory and Applications*, vol. 38, no. 3, pp. 382-385, March 2001.

[80] J. Schoukens, R. Pintelon, E. van der Ouderaa, J. Renneboog, "Survey of Excitation Signals for FFT Based Signal Analyzers", *IEEE Transactions on Instrumentation and Measurements*, vol. 37, no. 3, pp 342–352, September 1988.

[81] B. Sunar, W. J. Martin, and D. R. Stinson "A Provably Secure True Random Number Generator with Built-in Tolerance to Active Attacks", Technical Report CACR 2005-20, University of Waterloo (Canada), 2005

[82] L. Trevisan, and S. Vadhan, "Extracting randomness from samplable distributions", in *Proceedings of 41st IEEE Symposium on Foundations of Computer Science* (FOCS'00), pp. 32–42. Redondo Beach (USA), November 12-14, 2000.

EMI Reduction Related References

[83] M. Balestra, R. Rovatti, G. Setti, "Power Spectrum Density Tuning in Random and Chaos-based Timing Signal Modulation Techniques with Improved EMC" in *Proceedings of IEEE 11th Workshop on Nonlinear Dynamics of Electronic Systems* (NDES2003), pp 24–28. Scuol (Switzerland), May 18–21, 2003.

[84] R. E. Best, *Phase-locked Loops: Design, Simulation and Applications*, McGraw-Hill, 1999.

[85] S. Callegari, R. Rovatti, G. Setti, "Spectral Properties of Chaos-based FM Signals: Theory and Simulation results", *IEEE Transaction on Circuit and Systems I: Fundamental Theory and Applications*, vol. 50, no. 1, pp. 3–15, January 2003.

[86] Federal Communication Commission "FCC methods of measurement of radio noise emission from computing devices", *FCC/OST MP-4*, July 1987.

[87] K. B. Hardin, J. T. Fessler, D. R. Bush, "Spread spectrum clock generation for the reduction of radiated emission", *Proceedings of the IEEE International Symposium on Electromagnetic Compatibility* (EMC'94), pp. 227–231. Rome (Italy), September 13–16, 1994

[88] K. B. Hardin, J. H. Fessler, D. R. Bush, J. J. Booth, "Spread Spectrum Clock Generator and Associated Method," *U.S. Patent n. 5,488,627*, 1996.

[89] International Special Committee on Radio Interference (CISPR), Publication 16-1, 2002.

[90] F. Lin, D. Y. Chen, "Reduction of Power Supply EMI Emission by Switching Frequency Modulation", in *IEEE Transactions on Power Electronics*, vol. 9, no. 1, pp. 132–137, January 1994.

[91] R. Rovatti, G. Setti, S. Graffi, "Chaos based FM of clock signals for EMI reduction", in *Proceedings of 14th IEEE European Conference on Circuit Theory and Design* (ECCTD'99), vol 1, pp. 373-376. Stresa (Italy), August 29 – 2 September 2, 1999.

[92] S. Santi, R. Rovatti, G. Setti, "Advanced chaos based frequency modulations for clock signal tuning" in *Proceedings of 2003 IEEE International Symposium on Circuits and Systems* (ISCAS2003), vol 3, pp 116–119. Bangkok (Thailand), May 25–28, 2003.

[93] Serial ATA Workgroup, "Serial ATA II: Electrical Specificaion", Revision 1.0, May 2004.

FPGA and SCA Related References

[94] J. Coron, P. Kocher, and D. Naccache, "Statistics and Secret Leakage", in *Financial Cryptography* (FC2000), Lecture Notes in Computer Science, vol. 1962, pp. 157–173, February 2000.

[95] E. Hess, N. Janssen, B. Meyer, and T. Schuetze, "Information Leakage Attacks Against Smart Card Implementations of Cryptographic Algorithms and Countermeasures – a Survey", in *Proceedings of Eurosmart Security Conference*, pp. 55–64. Marseilles (France) June 13–15 June, 2000.

[96] P. Kocher, "Timing Attacks on Implementations of Diffie-Hellman, RSA, DSS, and Other Systems", in *Advances in Cryptology - Proceedings of Crypto'96*, Lecture Notes in Computer Science, vol. 1109, pp. 104–113, August 1996.

[97] P. Kocher, J. Jaffe, and B. Jun, "Differential Power Analysis", in *Advances in Cryptology - Proceedings of Crypto'99*, Lecture Notes in Computer Science, vol. 1666, pp. 388–397, August 1999.

[98] P. Kocher, R. Lee, G. McGraw, A. Raghunathan, and S. Ravi, "Security as a New Dimension in Embedded System Design", in *Proceedings of the 41st Design Automation Conference* (DAC 2004), pp. 753–760. San Diego (USA), June 7–11, 2004.

[99] J. Montanaro, R. Witek, K. Anne, A. Black, E. Cooper, D. Dobberpuhl, P. Donahue, J. Eno, W. Hoeppner, D. Kruckemyer, T. Lee, P. Lin, L. Madden, D. Murray, M. Pearce, S. Santhanam, K. Snyder, R. Stehpany, and S. Thierauf, "A 160-MHz, 32-b, 0.5-W CMOS RISC Microprocessor", in *IEEE Journal of Solid-State Circuits*, vol. 31, no. 11, pp. 1703–1712, November 1996.

[100] J. Quisquater, and D. Samyde, "ElectroMagnetic Analysis (EMA): Measures and Counter-measures for Smard Cards", in *Smart Card Programming and Security - Proceedings of Esmart 2001*, vol. 2140, pp. 200–210. Cannes (France), September 19–21, 2001.

[101] J. Rabaey, *Digital Integrated Circuits: A design perspective*, Prentice Hall, 1996.

[102] S. Ravi, A. Raghunathan, and S. Chakradhar, "Tamper Resistance Mechanisms for Secure, Embedded Systems", in *Proceedings of 17th International Conference on VLSI Design* (VLSID 2004), pp. 605–610. Mumbai (India), January 5–9, 2004.

[103] K. Tiri, M. Akmal, and I. Verbauwhede, "A Dynamic and Differential CMOS Logic with Signal Independent Power Consumption to Withstand Differential Power Analysis on Smart Cards", in *Proceedings of 28th European Solid-State Circuits Conference* (ESSCIRC 2002), pp. 403–406. Florence (Italy), September 24–26, 2002.

[104] Xilinx Inc., "Spartan and Spartan-XL Families Field Programmable Gate Arrays", available at `http://direct.xilinx.com/bvdocs/publications/ds060.pdf`, June 27, 2002.