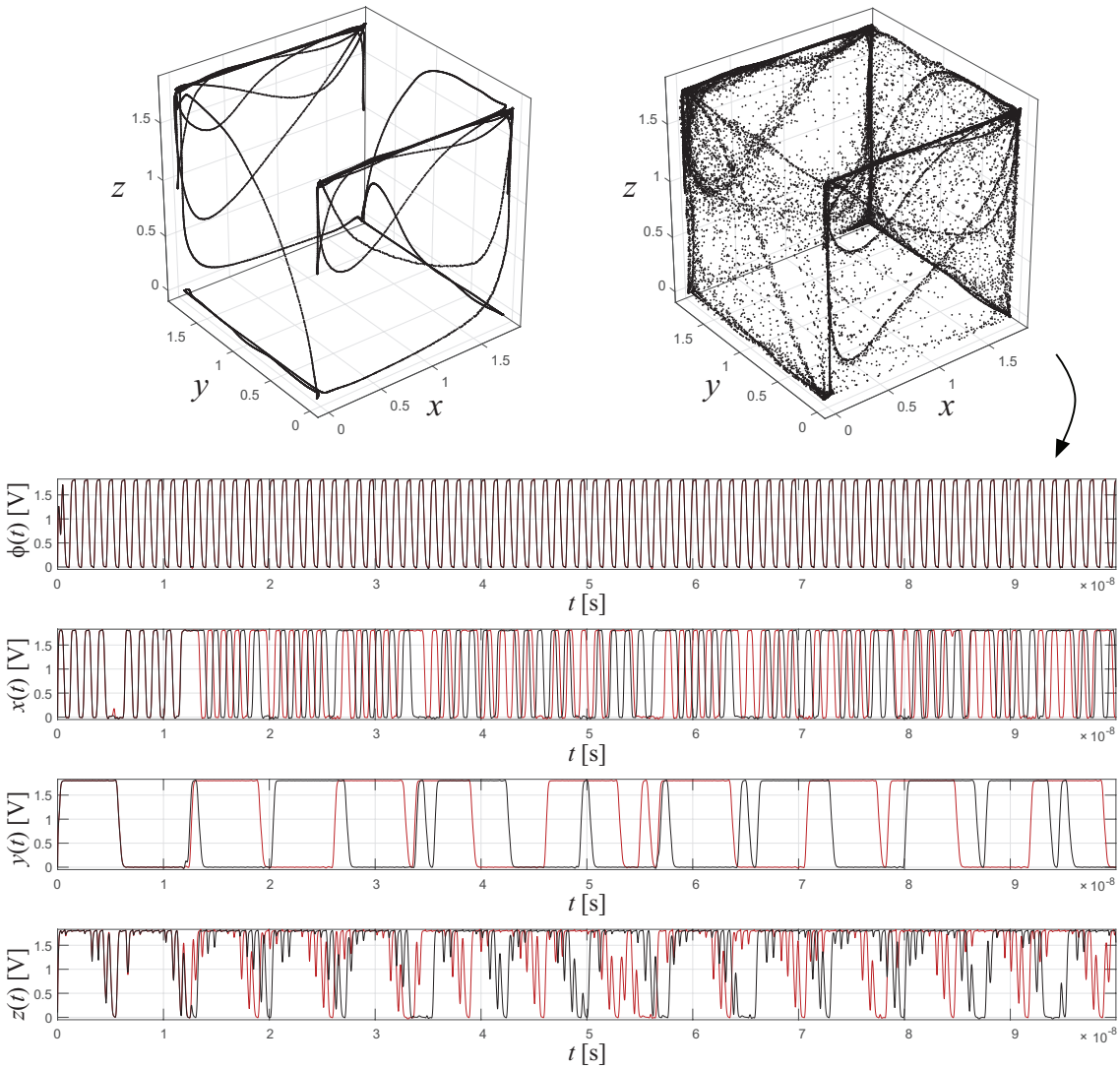

Figure 4.15: In the upper plots, the 3D-projections of the voltages x, y, z in Fig. 4.14 under different realizations (mismatches and process variability). Upper-left: complex periodic dynamics. Upper-right: chaotic dynamics exhibiting sensitivity to initial conditions, highlighted in the trajectories below.

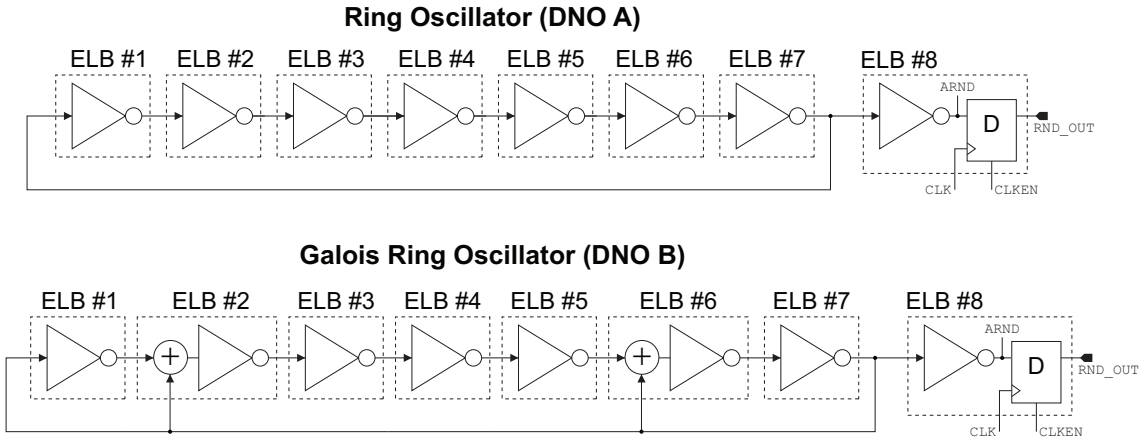

Figure 4.16: The reference DNO architectures considered for the analysis of the proposed topology. DNO A is a 7-nodes Ring Oscillator, DNO B is a 7-nodes Galois Ring Oscillator.

designed, which were located in different areas of the chip; in this way it is possible to evaluate the effect of intra-device variability on the performance of the circuits. Furthermore, to also evaluate the effect of inter-device variability, the design of the 16 instances was implemented in six different FPGAs, always using the same positions for the circuits. Following this procedure, each DNO was evaluated through 96 different implementations. Finally, taking control of the place and route policies using the special directives shown in Section 3.5, we forced a compact layout for the three oscillators, selecting LUTs belonging to couples of slices of single CLBs, such to minimize the propagation times associated to signal routing. Fig. 4.17 shows the resulting layout for DNO C. DNOs A and B have a similar layout, as the used LUTs are the same; the difference between DNOs is in the routing used for the connections between the LUTs.

For each oscillator, we performed acquisitions of sequences of 1 million bits at different frequencies, defined on a range between 100 kHz and 100 MHz. The acquisition was carried out through an architecture designed on the FPGA, which collects the acquired bits in a RAM and subsequently transmits them via RS232 serial interface to a PC, on which a virtual instrument was implemented with National Instruments LabVIEW. The virtual instrument stores the data in binary files, which are then processed with MathWorks MATLAB.

4.5.1 Performance Comparison

The acquired sequences were used to calculate the Average Shannon Entropy (ASE) and the Decorrelation Time τ, already introduced in Section 3.2. Figs. 4.18 and 4.19 show the results obtained in terms of ASE, calculated on 10 bits long symbols, and in terms of Decorrelation Time, calculated for an energy ratio equal to 99.9% of the total energy evaluated over a 10 μs window. Table 4.1 instead summarizes the statistics related to the figures of merit, comparing the DNOs in

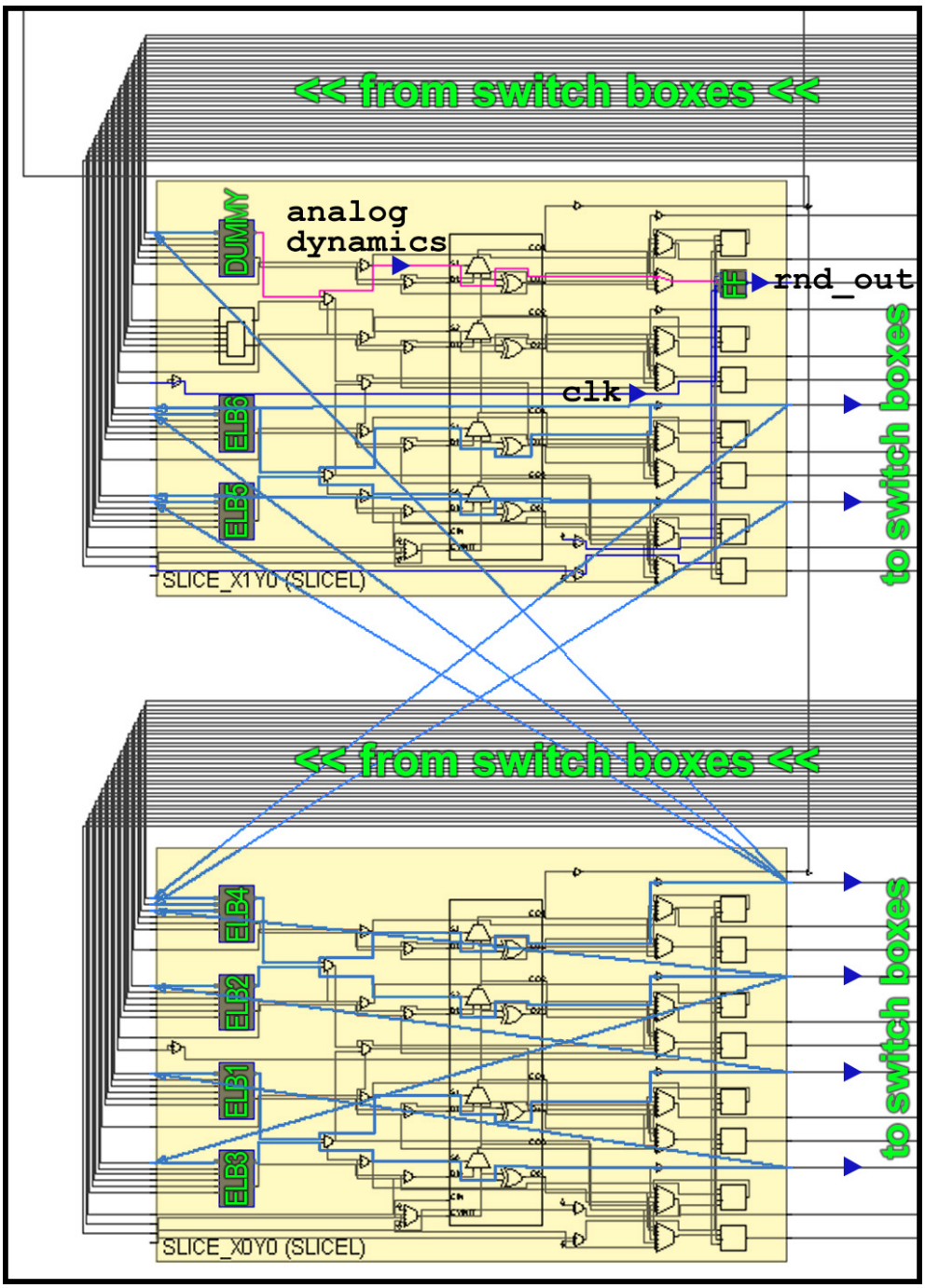

Figure 4.17: The condensed layout implementing the DNO if Fig.4.14 using two slices, including the Synchronization Interface realized with a single D flip-flop and a transparent (dummy) LUT.

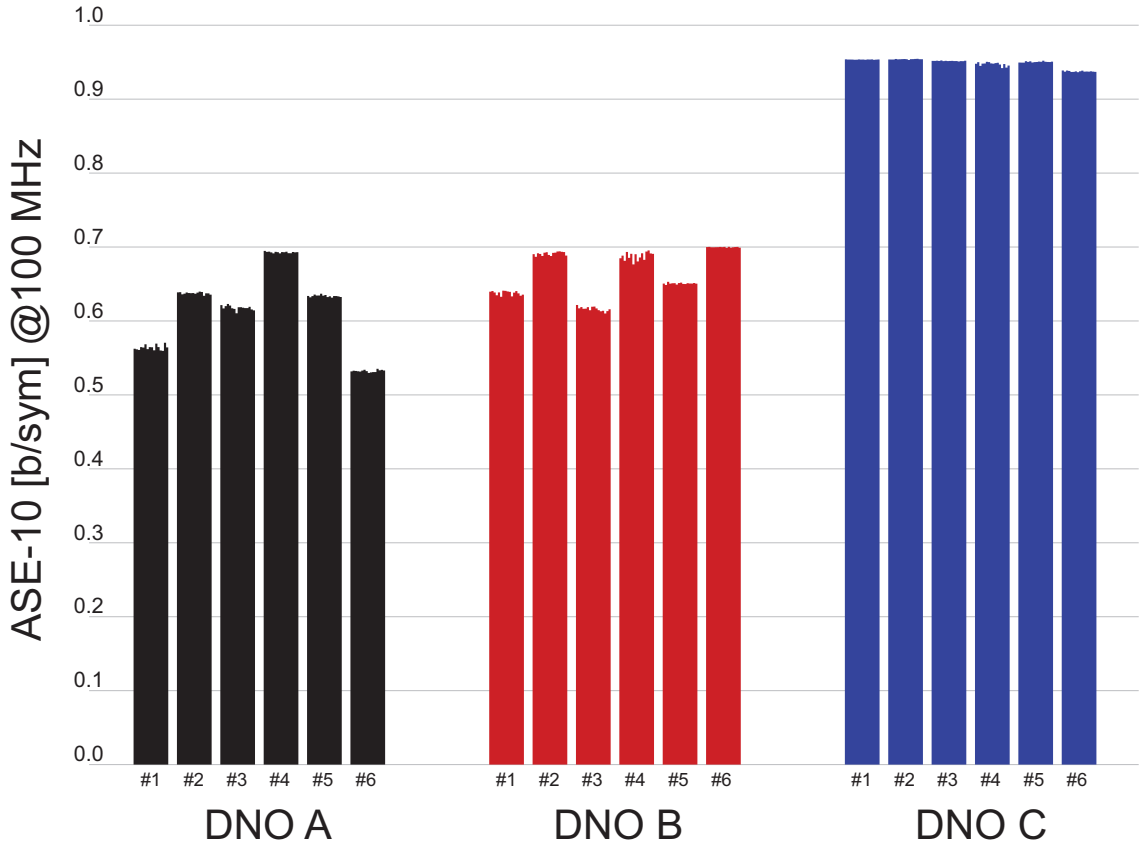

Figure 4.18: Average Shannon Entropy of the three compared DNOs, estimated on the basis of binary words of 10 bits (ASE-10) acquired from 96 instances for each topology, implemented on 6 Xilinx Artix 7 xc7a35 FPGAs. For each DNO, the ASEs of all the instances are reported, organized according to the chip on which they are implemented.

terms of achieved performance and consumed FPGA hardware resources.

The results shown in the figures and in the table show that the DNO C achieves higher performance than the two reference DNOs, both in terms of ASE and Decorrelation Time. In all cases, the bits collected at a sampling rate of 100 MHz from the DNO C were found to have negligible or undetectable correlation, as also shown in Fig. 4.20. Consequently, the DNO C does not reach an entropy exactly equal to 1 only due to a residual offset of the sequences. This aspect is related to the average value of the analog output signal of the DNO, which depends on the shape of the trajectory, and on the level of quantization thresholds of the flip-flops D used for sampling. This problem is more evident in DNO B and less important in the Ring Oscillator (DNO A), as it generates a square wave with a duty cycle of approximately 50%: although DNO B has a Decorrelation Time on average lower than that of DNO A, the bias it is affected by limits its ASE. In DNO C the bias is maintained on adequate values thanks to the symmetrical structure of the topology and the mutual interaction through the XOR3 gate of the two feedback loops. Since

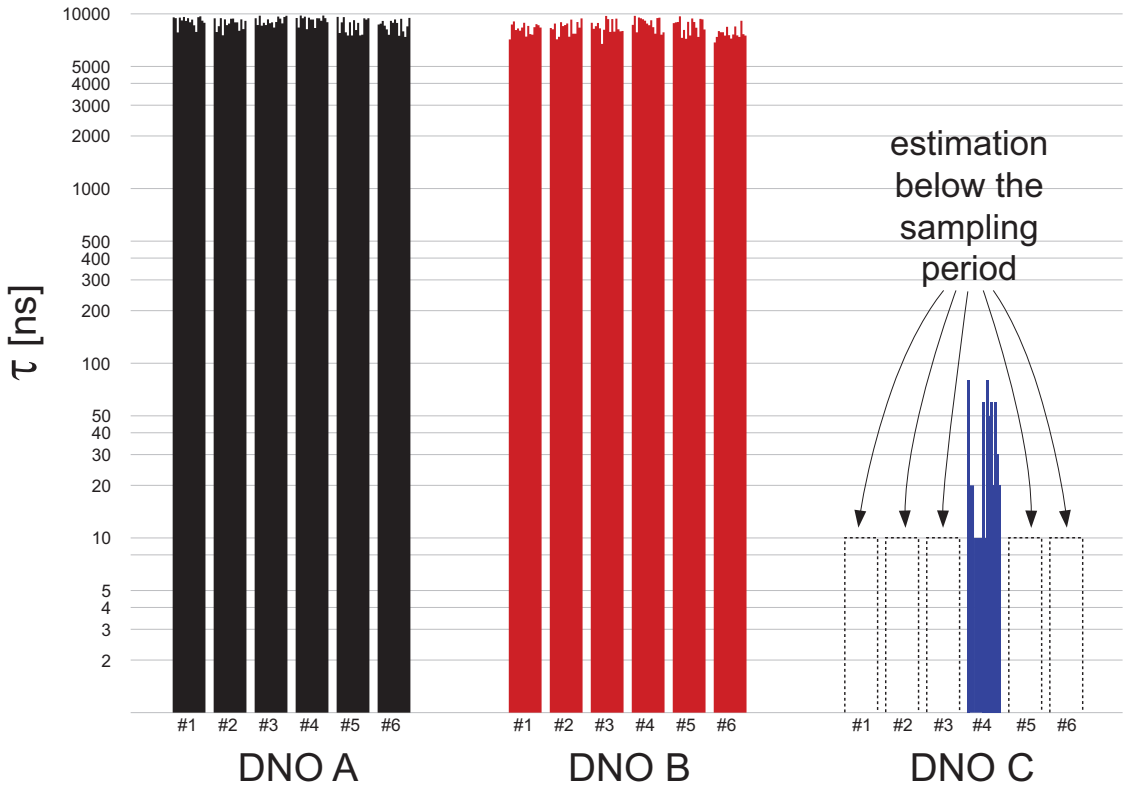

Figure 4.19: Decorrelation Time τ of the three DNOs, estimated as the time at which the autocorrelation function of the binary source expresses the 99.9% of its variation-energy, referring to an observation time-window of 10 μs. For each DNO, the Decorrelation Times of 96 instances implemented on 6 Xilinx Artix 7 xc7a35 FPGAs are reported, organized according to the chip on which they are implemented.

	Ring Oscillator (DNO A)	Galois Ring Oscillator (DNO B)	Proposed Circuit (DNO C)
CLBs	1	1	1
Slices	2	2	2
ELBs (LUTs)	7 (+1)	7 (+1)	6 (+1)
ASE-$10_{\max}$ [bit/sym]	0.695	0.700	0.955
ASE-10_{mean} [bit/sym]	0.613	0.664	0.949
ASE-$10_{\min}$ [bit/sym]	0.530	0.610	0.937
$\tau_{\min}$ [ns]	7380	6730	<10
τ_{mean} [ns]	8893	8350	53
$\tau_{\max}$ [ns]	9780	9740	80

Table 4.1: Comparison of the device utilization and the measurements results for the three DNOs. The device utilization is described in terms of required LUTs taking into account a compact layout, the measurements are compared in terms of maximum, minimum and average ASEs and Decorrelation Times.

DNO C achieves such results, we can assume that all implementations worked in structurally stable chaotic regions.

To better appreciate the comparison between the three topologies, Fig. 4.21 shows the byte patterns for the three DNOs evaluated for the 10-bits maximum ASE and, for DNO C, the 10-bits minimum ASE implementations, respectively with 100 kHz and 100 MHz sampling.

4.5.2 Performance Dependency on the Implementation Layout

Figs. 4.18 and 4.19 demonstrate that the DNO C is able to achieve much higher performance than the reference oscillators. However, a variability of the performance of the circuit implementations is observed both between different positions and chips.

This variability depends on the differences in the hardware resources related to the chip manufacturing process, but there is also an influence given by routing and, consequently, by the layout.

To evaluate the impact of the selected layout on the variability of the circuit performance, we decided to repeat the measurement campaign for DNO C, evaluating two different designs, one adopting a condensed layout (i.e. the one already used in the analysis reported in Subsection 4.5.1) and one adopting a scattered layout. The CLBs utilization map in the two analyzed cases is shown in Fig. 4.22.

The two layouts were evaluated by acquiring 1 million bits long sequences from 96 implementations of the DNO, obtained using six Xilinx Artix 7 xc7a35 FPGAs and implementing 16 oscillators on each of them. The sequences were acquired at a sampling rate of 400 MHz, achieved by implementing a PLL in the FPGA architecture. We increased the sampling frequency with respect to the previous analyzes to obtain more correlated sequences of bits, thus highlighting the variability associated with the two layouts. The acquired sequences were used to calculate the Average Shannon Redundancy (ASR) of 10-bits symbols. In Information Theory, the ASR is a figure of merit that is complementary to the ASE, since it is defined as:

$$ASR(n) = 1 - ASE(n), \tag{4.10}$$

where the ASE(n) is given in (3.10). In an ideal binary random source the ASR(n) is equal to 0 bit/sym. for any $n > 0$. In practical cases, the lower is the ASR, the higher is its entropy. For the analysis of high-entropy sources it is often convenient to report results referring to redundancy, instead of entropy, for a clearer presentation. Indeed, when the values of ASE are close to 1 bit/sym, a logarithmic representation of the ASR allows for a better comparison of different solutions.

Fig. 4.23 shows the experimental results obtained with the condensed layout, evaluating its intra-device and inter-device variabilities. The percentile levels L_x, expressed in bit/sym for $x = 10, 50, 80, 90, 95$, were estimated for the whole set of 96 instances, while the red squares highlight the ASR obtained for the DNO in position 1 (plot A) and in chip 1 (plot B). From the figure it can be appreciated that 90% of the DNOs are capable of providing impressive levels of ASR lower than $L_{90} = 0.077$

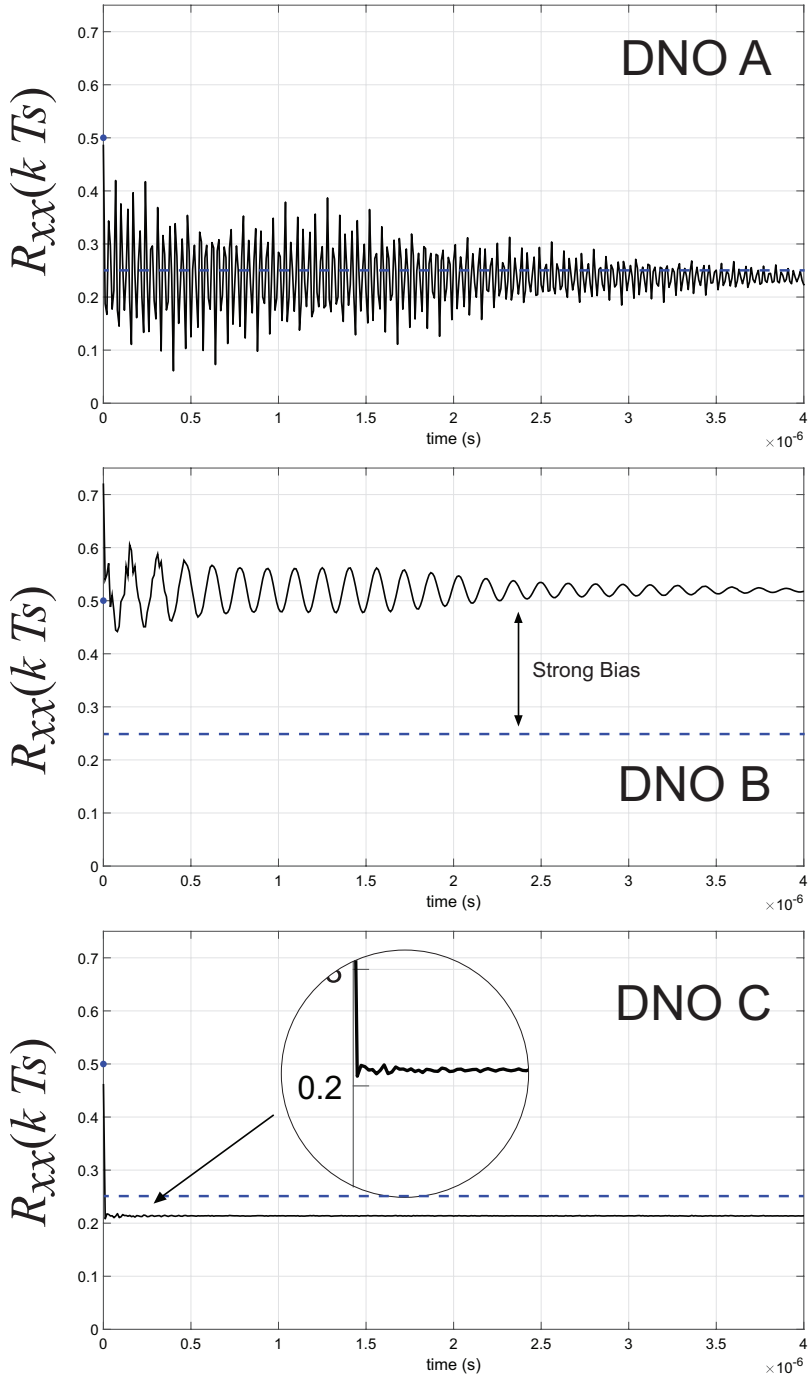

Figure 4.20: Autocorrelation functions for the three DNOs (typical results for a sampling frequency of $1/T_s = 100$ MHz). For an ideal random binary source the autocorrelation is $R_{xx}(kT_s) = 0.5$ if $k = 0$, 0.25 otherwise (blue/dashed levels) [37].

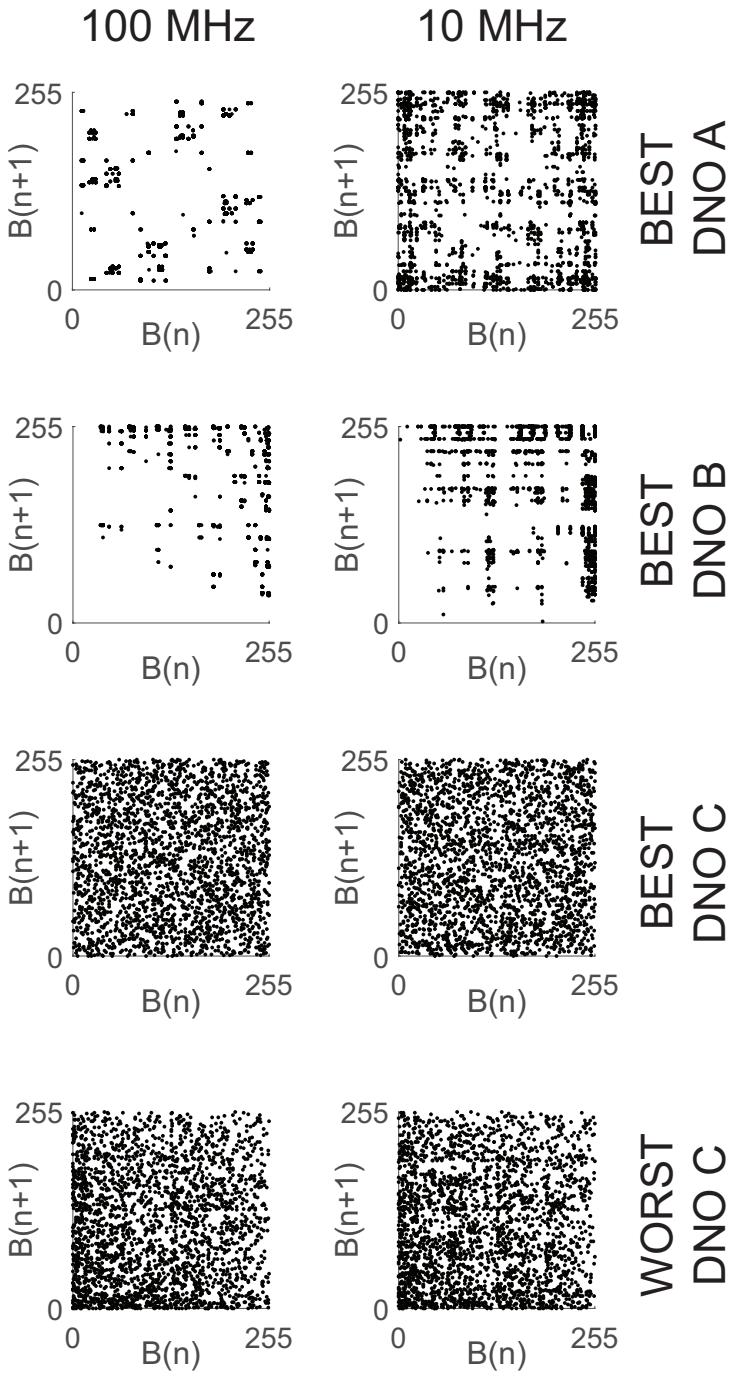

Figure 4.21: Comparison between the byte-pattern generations, reporting the result for the DNOs with the highest ASE-10 entropy, including the worst DNO with the lowest ASE-10 entropy for the DNO C type (proposal), for 100 MHz and 10 MHz sampling rates.

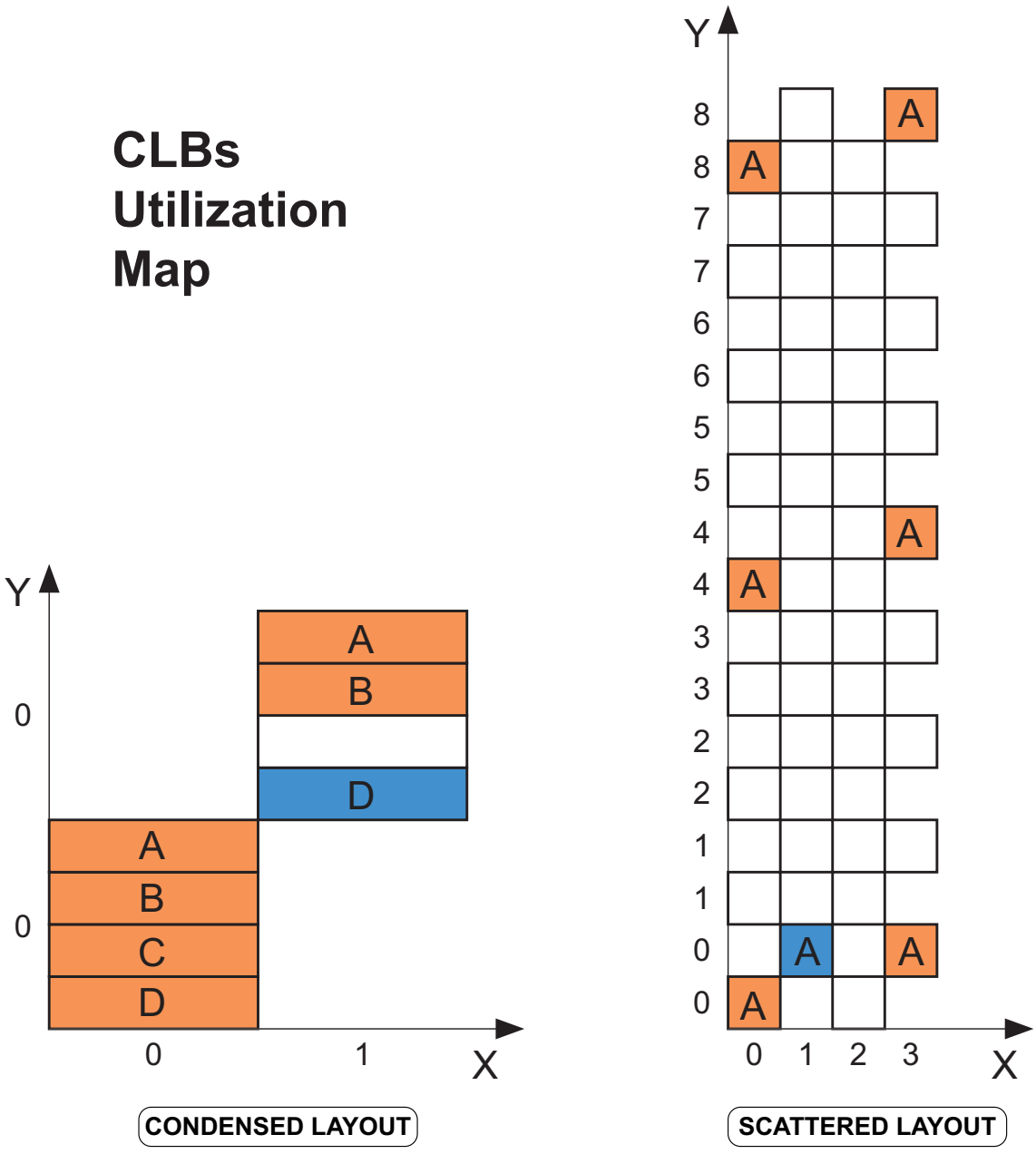

Figure 4.22: Comparison of the CLBs resource mapping for the evaluated condensed and scattered layouts (one DNO instance). An orange color was used to denote those resources used for the ELBs (6 LUTs); a blue color for the Synchronization Interface (1 transparent LUT and 1 flip-flop D).

bit/sym with a sampling frequency of 400 MHz, which corresponds, in principle, to 369.2 Mbit/s of true random information, generated with the minimum use of two slices of the FPGA. By reducing the set, in half of the cases (ASR lower than $L_{50} = 0.0067$ bit/sym) the throughput rises to 397.3 Mbit/s.

Fig. 4.24 reports the same results as Fig. 4.23 for the scattered layout. In this case, a slight deterioration in performance is observed, as 90% of the cases provide an ASR lower than $L_{90} = 0.17$ bit/sym, corresponding to a throughput of 332.0 Mbit/s of true random information, and in half cases the ASR is less than $L_{50} = 0.03$ bit/sym, with throughput equal to 388.0 Mbit/s. However, these are still exceptional result, given the small number of used hardware resources.

The equivalent throughputs reported for the analyzed data are values that can be reached by applying post-processing techniques compressing the input data to remove information redundancy. Unfortunately, the use of these techniques increases the complexity of the whole structure. In particular, to reach the considered throughputs, lossless algorithms must be used; this algorithms require large quantities of hardware resources and power. As an alternative, lossy algorithms can be used, which are less demanding from the hardware and power consumption point of view, but reduce the resulting throughput.

What is more, considering the variability between implementations, a correct design of TRNGs requires to ensure adequate performance against the source implementation worst case. Referring to the results achieved in the condensed case, this would require to adapt the overall project to the maximum found ASR, equal to 0.4 bit/sym, corresponding to a throughput of 160 Mbit/s, which is much less than the throughput achieved by the majority of the analyzed cases.

A possible solution to this problem derives from the simplicity of the considered circuit: since the DNO occupies only one CLB of the FPGA, it is much more convenient, compared to the implementation of a post-processing architecture, to duplicate the structure of the DNO (i.e. to implement two DNOs) and to XOR the random output bits by applying a 2:1 lossy compression. The result, shown in Fig. 4.25, drastically reduces the variability, allowing to reach much higher performance.

4.5.3 Performance Dependency on Temperature Variations

In the previous subsections, we evaluated the performance variability of the topology under study with respect to the hardware used for its implementation and to the layout design. These two sources of variability can be considered static: their influence on a DNO performance is defined by the hardware resources and the layout chosen for its implementation, and does not change over time.

However, there is a third source of variability that we still must consider, whose effect on a DNO performance changes over time, that is temperature.

Transistors are characterized by different temperature-dependent parameters, such as mobility, threshold voltage, saturation velocity, parasitic drain/source resistances [51–54]. Taking into account digital devices, this dependency is directly transferred to the gates high-low and low-high propagation time, affecting therefore

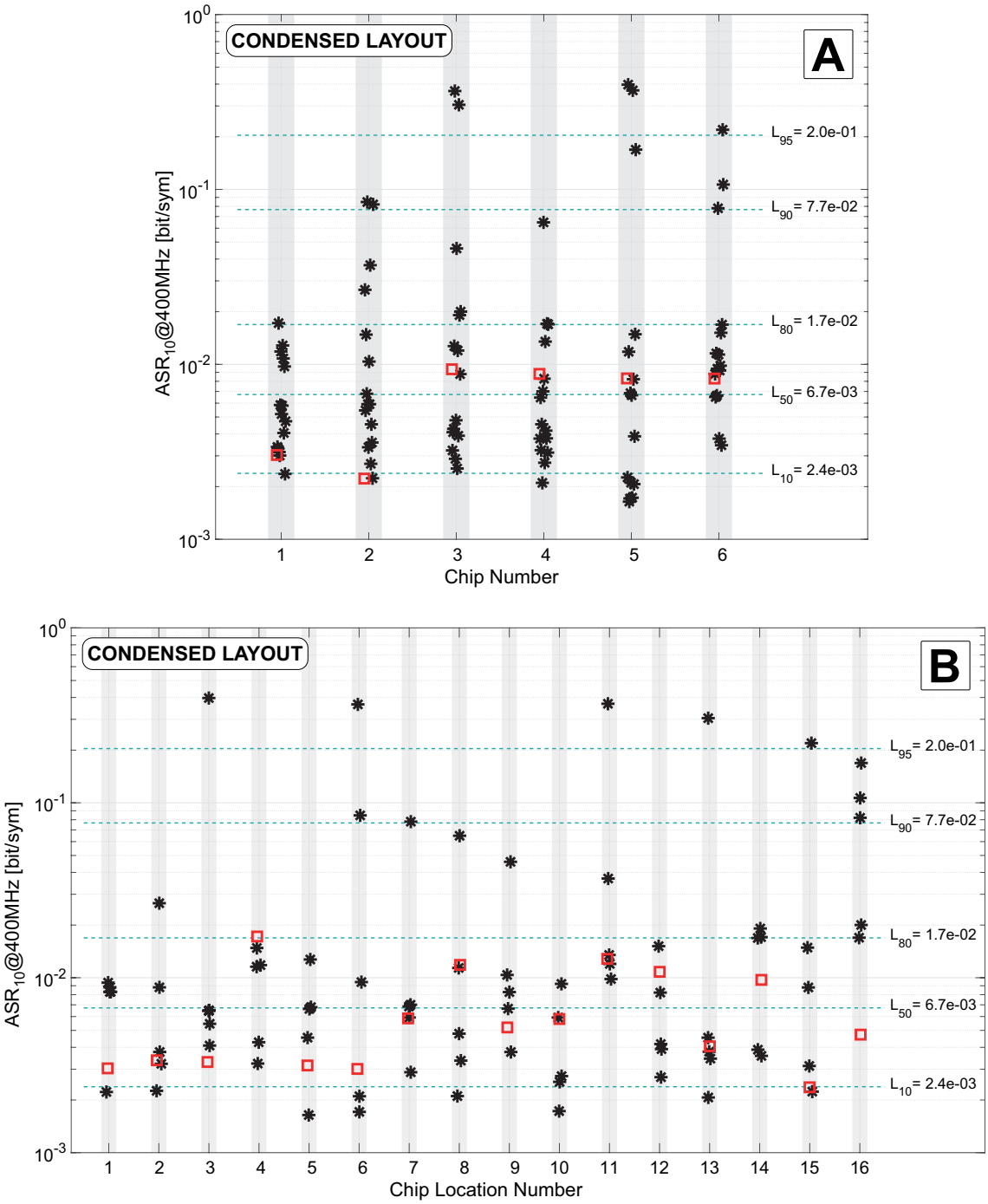

Figure 4.23: Experimental results highlighting the effects on the ASR of both the chip-to-chip variability and the intra-device variability, for a condensed layout. The percentile levels L_x, expressed in bit/sym, were estimated on the base of the entire data set (96 DNO instances). Red square symbols were used to highlight the chip location 1 (upper plot A) or the chip number 1 (lower plot B).

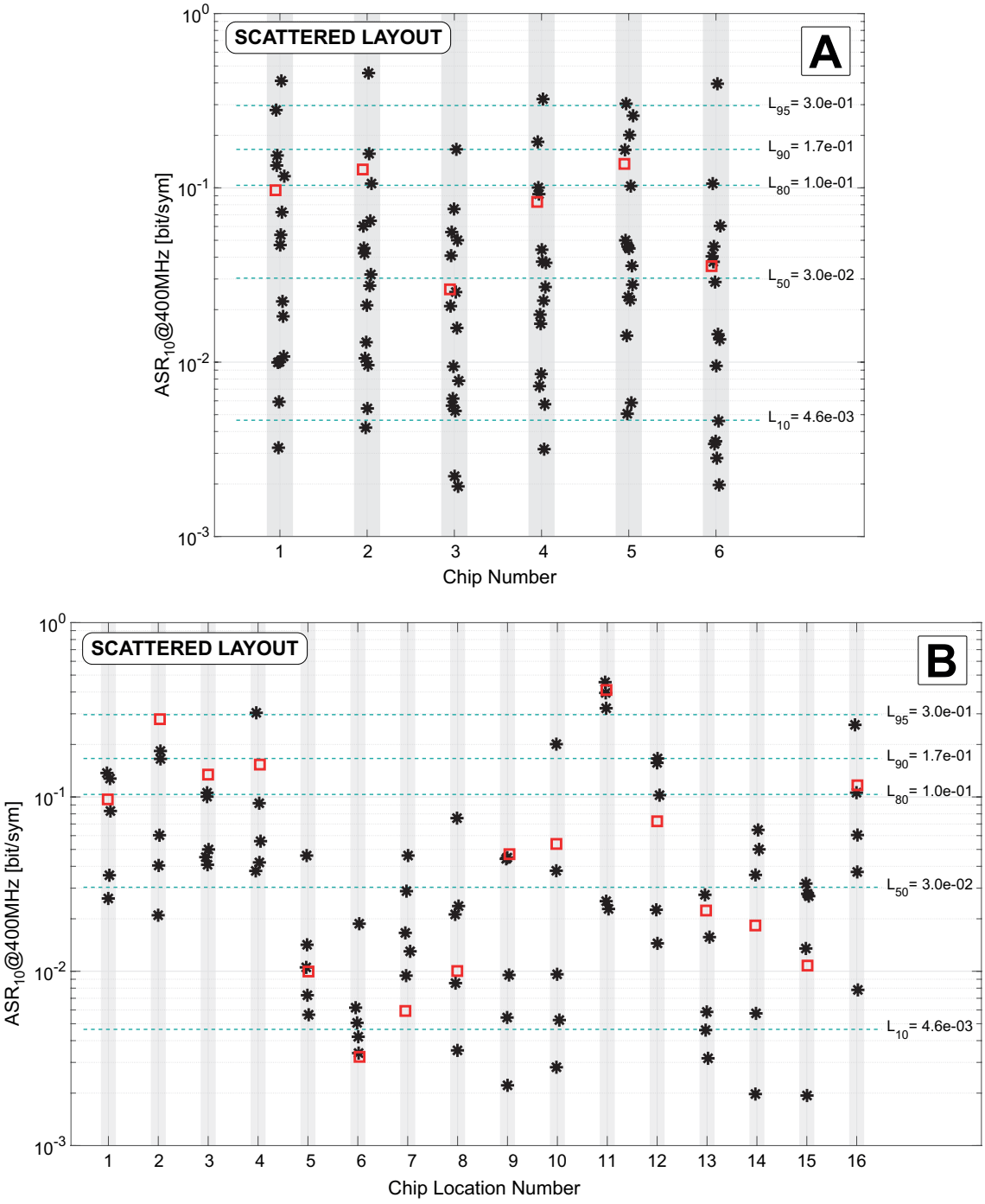

Figure 4.24: Experimental results highlighting the effects on the ASR of both the chip-to-chip variability and the intra-device variability, for a scattered layout. The percentile levels L_x, expressed in bit/sym, were estimated on base of the entire data set (96 DNO instances). Red square symbols were used to highlight the chip location 1 (upper plot A) or the chip number 1 (lower plot B).

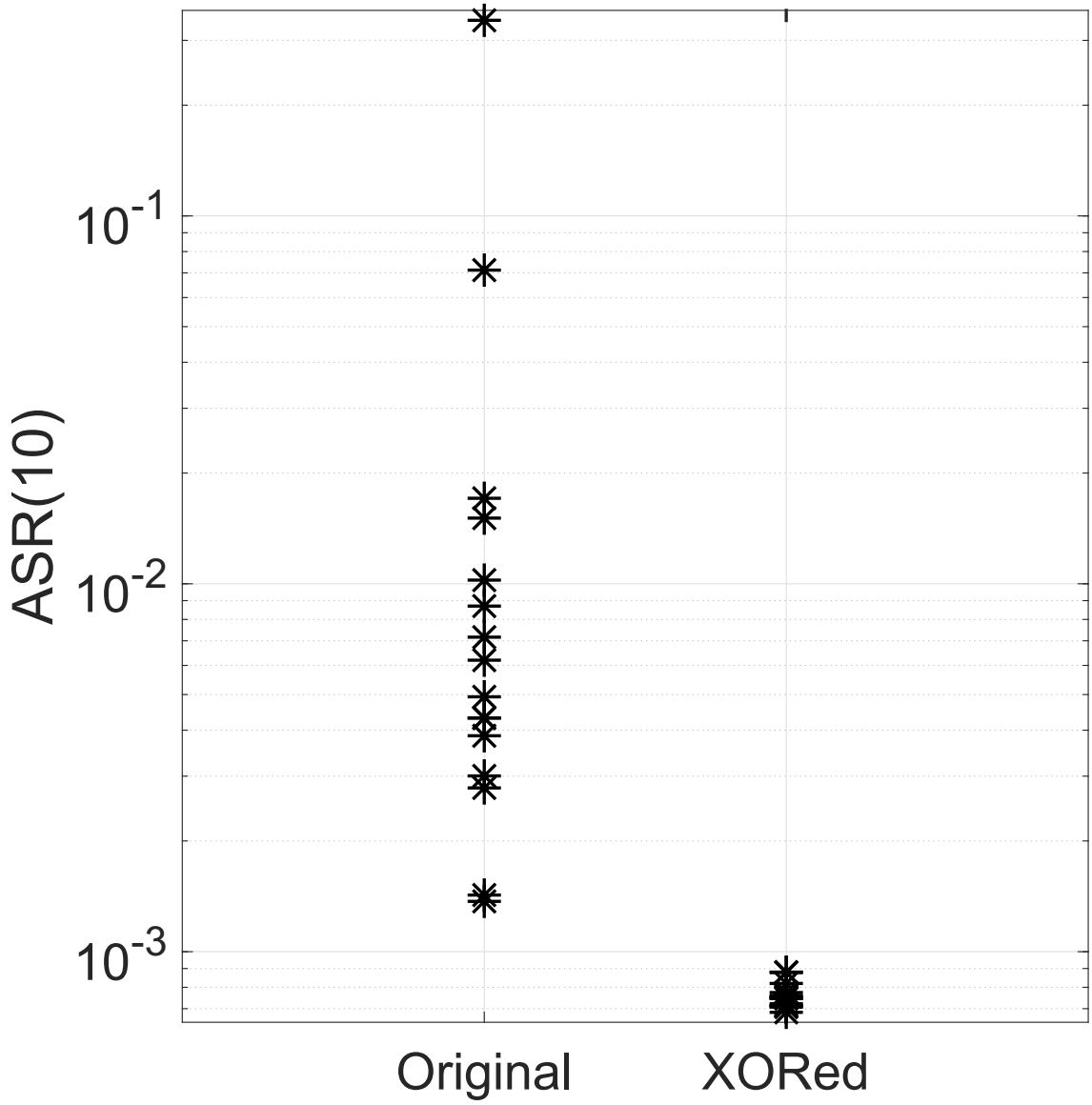

Figure 4.25: Rejection of intra-device variability by XORing the binary stream generated exploiting two instances of the discussed DNO (16 tested locations, four FPGA slices for each obtained generator).

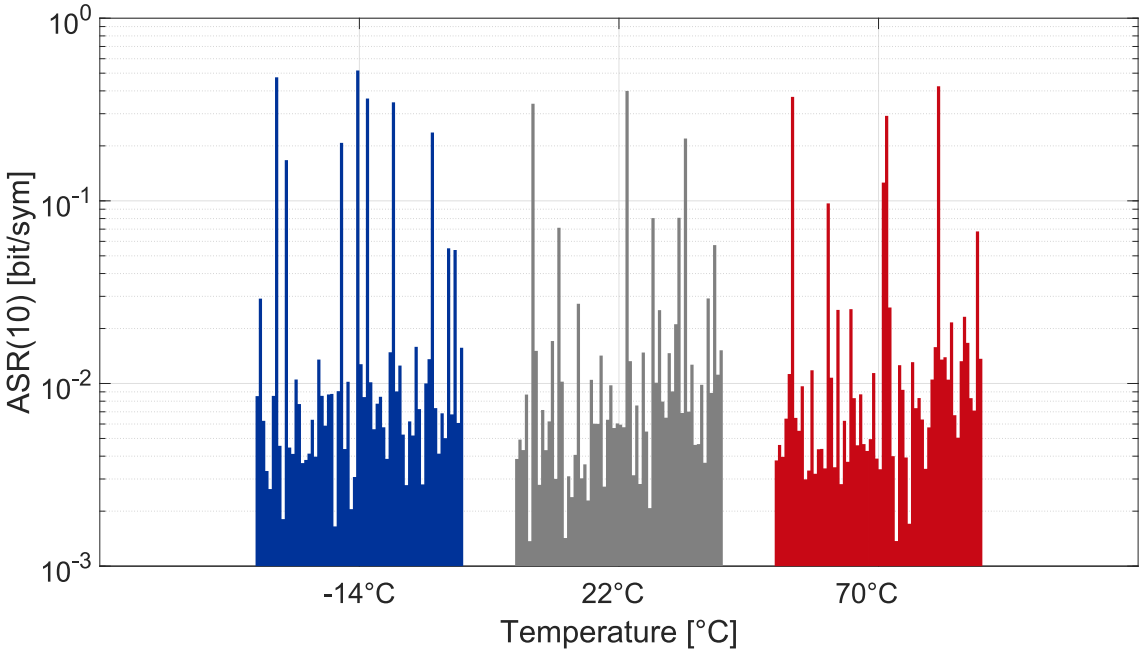

Figure 4.26: 10-bits Average Shannon Redundancy of 64 implementations of the DNO shown in Fig. 4.14 at three different ambient temperatures, i.e. -14 °C, 22 °C and 70 °C.

	-14 °C	22 °C	70 °C
ASR-10_{min} [bit/sym]	0.0017	0.0014	0.0014
ASR-10_{mean} [bit/sym]	0.0442	0.0269	0.0293
ASR-10_{max} [bit/sym]	0.5137	0.3998	0.4240

Table 4.2: Measurements results for the DNO shown in Fig. 4.14 at three different ambient temperatures, i.e. -14 °C, 22 °C and 70 °C. The measurements are compared in terms of maximum, minimum and average ASRs.

the performance of dynamic systems based on digital hardware, such as DNOs.

To evaluate the effect of temperature fluctuation on the performance of the DNO under investigation, we designed 64 implementations of the DNO on four Xilinx Artix 7 xc7a35 FPGAs (16 oscillators per FPGA), and we acquired 1 million bits long sequences at a sampling rate of 400 MHz from each implementation at three different ambient temperatures, i.e. -14 °C, 22 °C and 70 °C. The acquired sequences were used to calculate the Average Shannon Redundancy (4.10) of 10-bits symbols.

Fig. 4.26 shows the obtained ASR values and Table 4.2 reports the performance variability with respect to temperature in terms of maximum, minimum and average ASRs. From the measurements we can observe that, in general, the DNO is more affected by lower temperatures with respect to higher ones, as we have slightly worse ASRs at -14°C. However, looking at the ASR mean values in the table, we note also that the temperature-dependent loss in perfomance is almost neglectable, resulting lower than 0.001 bit/sym.

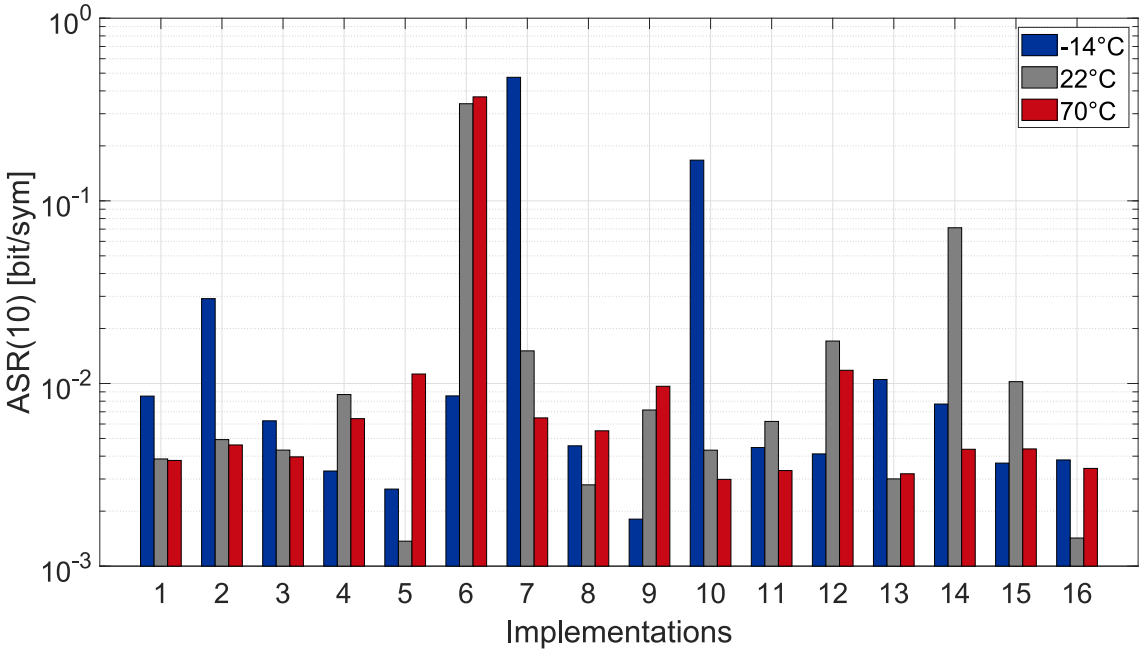

Figure 4.27: Comparison of the 10-bits Average Shannon Redundancy of 16 implementations of the DNO shown in Fig. 4.14 on a Xilinx Artix 7 xc7a35 FPGA at three different ambient temperatures, i.e. -14 °C, 22 °C and 70 °C.

What is more, from Fig. 4.27, in which the comparison of the ASRs of a single FPGA implementations at the three different temperatures is reported, we can observe that the ASR does not change monotonically with the temperature.

4.5.4 Statistical Testing

Besides comparing the performance of DNO C with two reference topologies using the figures of merit, we tested it statistically with the standard NIST 800.22 tests.

In general, a TRNG consists of an entropy source and a post-processing block with the purpose of reducing the information redundancy, e.g. through a compression operation, and to mask residual statistical defects, e.g. by means of stream cyphers [4, 55, 56]. The minimum post-processing necessary to pass all the NIST 800.22 tests required by the DNO C in each of its implementation was a bit-by-bit XORing of the collected bits with an 8-bits PRNG (Fibonacci LFSR based on the primitive polynomial $x^8 + x^6 + x^5 + x^4 + 1$). In 90% of the implementation a 4-bits XORing [57] is sufficient and in general any post-processing of greater complexity proposed in literature allows the DNO C to pass the tests (obviously at the cost of a greater consumption of hardware resources). Table 4.3 shows typical results obtained with respect to NIST tests, evaluated on the basis of 100 binary sequences of 1 million bits collected for each run.

Test Name	p-Value	Proportion	Result
Frequency	0.474986	1.00	pass
Block Frequency	0.911413	0.97	pass
Cumulative Sums[a]	0.191687	0.99	pass
Runs	0.935716	1.00	pass
Longest Run	0.015598	1.00	pass
Rank	0.474986	0.99	pass
FFT	0.534146	0.98	pass
Non Overlapping Template[a]	0.955835	0.96	pass
Overlapping Template	0.350485	0.99	pass
Universal	0.699313	0.99	pass
Approximate Entropy	0.798139	0.98	pass
Random Excursions[a]	0.888137	0.96	pass
Random Excursions Variant[a]	0.324180	0.96	pass
Serial[a]	0.145326	0.97	pass
Linear Complexity	0.289667	0.99	pass

[a] Worst case reported for tests with multiple outcomes.

Table 4.3: NIST 800.22 Rev.1a statistical tests results for the DNO under test, evaluated on the basis of 100 binary sequences of 1 million bits collected for each run.

4.5.5 Inspection of Physical Signals

Since the dynamical speed of the implemented DNO is too high to be able to acquire the signals directly, to evaluate its performance we must resort to the figures of merit applied on binary sequences acquired by sampling its output signal internally to the FPGA.

However, by varying the number of delay elements in each loop, it is possible to artificially slow down the dynamics to a few MHz of frequency and propagate the z signal directly to the FPGA I/O pins. This operation avoids the distortion effect caused by the parasitic capacities offered by the I/O pins.

Using this expedient, we carried out two study campaigns.

We initially implemented the circuit in Fig. 4.28 in a Xilinx Artix 7 xc7a35 FPGA, placing cascades of logically transparent LUTs in the loops. The external excitation ϕ was generated by a Rigol Waveform Generator controlled by a LabVIEW virtual instrument, with the purpose of configuring the oscillation frequency and automatically characterizing the dynamics of the resulting device. The output z signal was acquired via a Textronix MSO64 Oscilloscope at a sampling rate of 1.25 GS/s. Observation of the acquired signal confirmed that the excitation-free DNO ($\phi = 0$ V) exhibits complex oscillations, while it is possible to observe different types of dynamics when the excitation is on, as a function of the frequency of ϕ.

For example, in Fig. 4.29 two typical cases are reported, obtained for excitation frequencies equal to 1.208 MHz (A) and 1.160 MHz (b) by adding a cascade of 1200 delay elements in cascade to the ELBs#(2,3) of Fig. 4.28. Depending on the

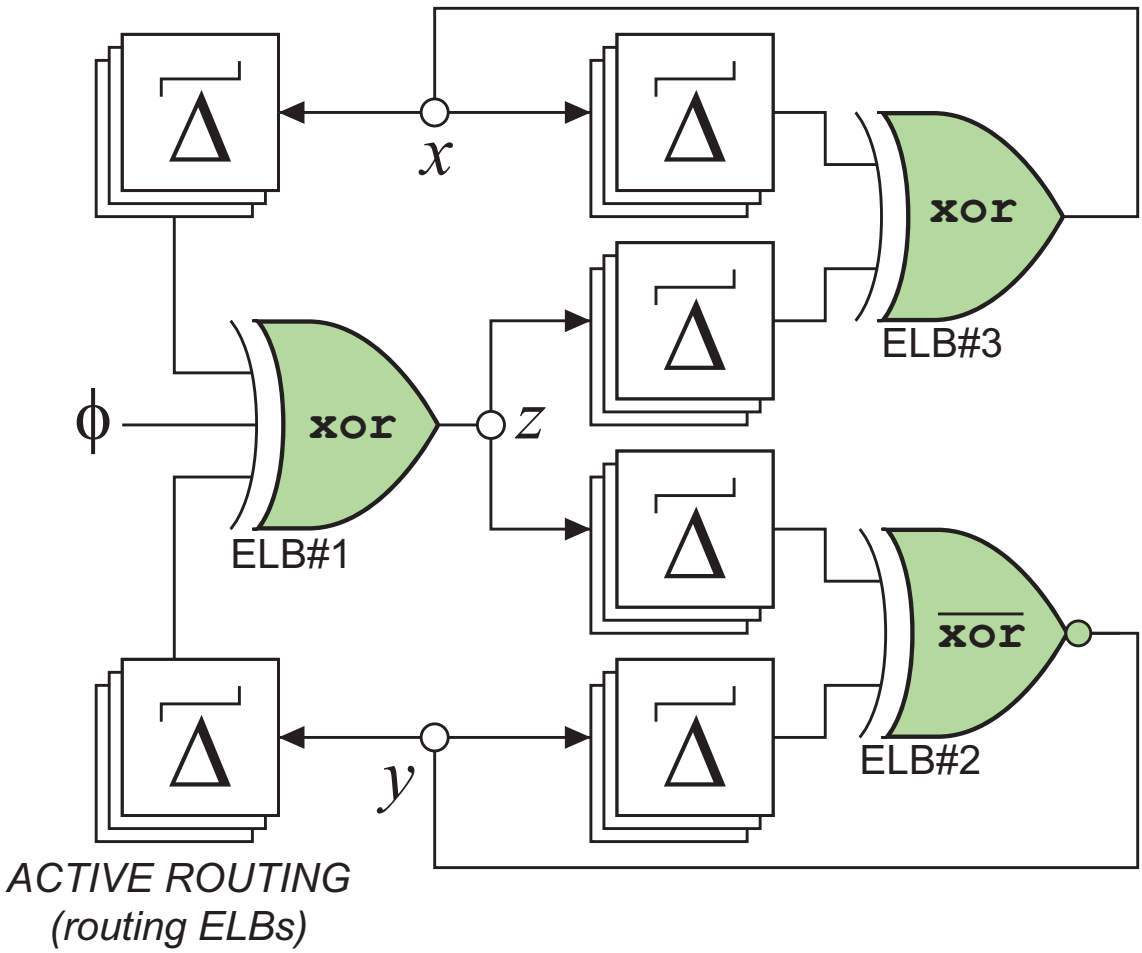

Figure 4.28: DNO topology implemented on a Xilinx Artix 7 xc7a35 FPGA, slowed down adding transparent delay LUTs in the loops, for the evaluation of the output signal dynamics in function of the oscillation frequency of an externally generated excitation signal.

oscillation frequency, the nonlinear oscillator can be forced to work in periodic or chaotic bifurcation windows. The observation of the autocorrelation function for the two analyzed cases, shown in Fig. 4.30, reveals that in the periodic case (A) the fundamental period of the signal z is about 1 kHz, far below the frequency of excitation. On the other hand, in the case of the chaotic signal (B) it is not possible to detect relevant periodic components.

By applying an increment step of 10 Hz, we experimentally evaluated the information entropy of the forced nonlinear oscillator over an oscillation frequency range between 1.15 MHz and 1.25 MHz. The acquired signal was sampled at 1 bit by applying a threshold quantization of 1.65 V and for each tested frequency we estimated the 8-bit ASE starting from sequences of 1 million acquired bits, obtaining the result shown in Fig. 4.31. We indicated in the figure the cases (A) and (B) of Figs. 4.29 and 4.30. The experiments show that the circuit generates information by exploiting two possible sources of entropy, namely jitter and chaos, as a function of the excitation frequency. While the electronic noise generating jitter is always present, chaos is activated only under certain conditions, which in the tested circuit appear to be structurally robust. This result is unexpected, as usually in the presence of complex dynamics small parametric perturbations cause significant changes in the dynamical behavior [58].

The second study campaign focused on the analysis of the circuit in Fig. 4.14, implemented on a Xilinx Artix 7 xc7a35 FPGA. We added transparent delay LUTs to the loops and to the Ring Oscillator providing the excitation signal. We acquired the z signal on 20 ms windows by adopting a sampling rate equal to 3.125 GS/s. The sampling frequency was selected considering the band of the acquired signal, in order to avoid aliasing.

By varying the number of LUTs in the three loops, we observed periodic and chaotic dynamical behaviors, as shown in Fig. 4.32. The autocorrelation function related to the chaotic case (b) has some residual periodicities. These harmonics are due to the Ring Oscillator, which influences the output signal as it supplies the excitation to the system, and by some active components mounted on the board used for interfacing with the FPGA (Digilent Arty), such as the switched power supplies. These periodicity elements can be removed by reducing the number of transparent LUTs in the Ring Oscillator (by increasing its oscillation frequency and pushing its fundamental harmonic towards higher bands) and by appropriately selecting the electronic components used to power the FPGA.

4.5.6 Comparison with the State of the Art

To conclude the analysis relating to the proposed DNO, in Table 4.4 we report a comparison of the analyzed solution with the most relevant recent works published in the literature. The TRNG resulting from DNO C and the minimum post-processing required to pass the NIST tests is made up of 15 LUTs and provides a throughput of 6.66 Mbit/s per LUT. A result of this type far exceeds the performance of any other proposal. This fact is justified by the simplicity of the topology, that increases

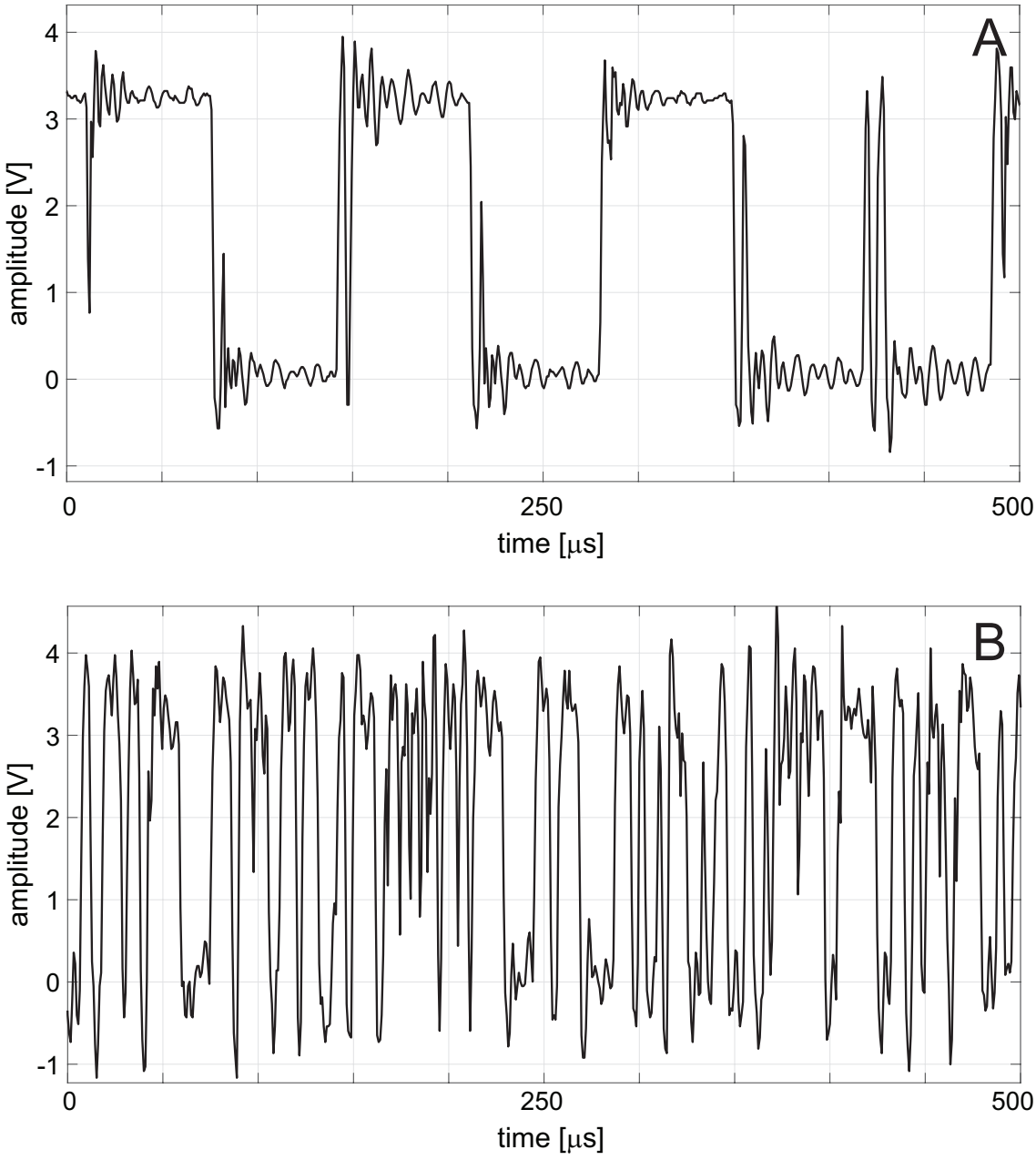

Figure 4.29: The forced-oscillator signal z extracted from an I/O pin of the FPGA, for excitation frequencies of 1.208MHz (A) and 1.160MHz (B), using 1200 delay elements for each loop in Fig. 4.28 to slow down the dynamics speed.

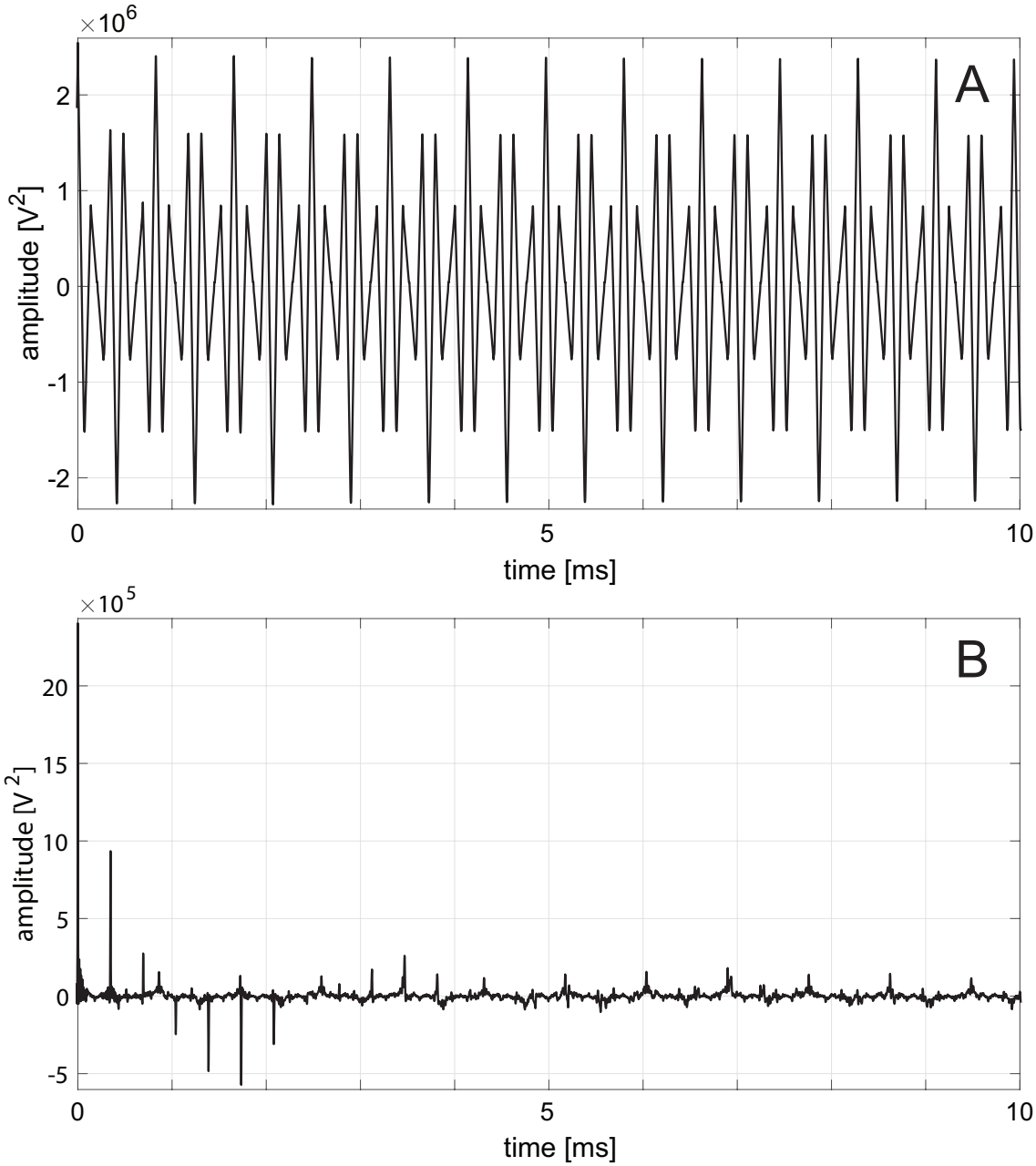

Figure 4.30: The unscaled autocorrelation function estimated for the two signals (A) and (B) reported in Fig. 4.29.

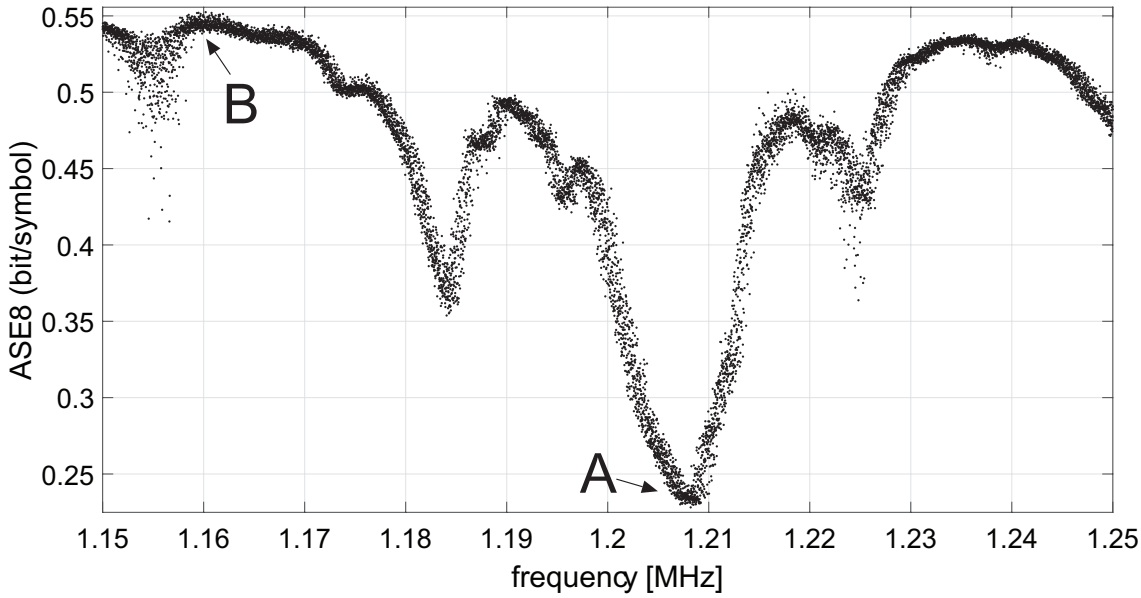

Figure 4.31: The 8-bit Average Shannon Entropy estimated on the basis of 1 million bits, obtained 1-bit quantizing the signal z acquired at a 1.25GHz sampling frequency.

the dynamics speed of the resulting nonlinear dynamical system, which operates, according to the design, in structurally stable chaotic regions.

4.6 Conclusion

We presented the complete workflow followed for the design of a DNO characterized by chaotic dynamical behaviors; the DNO achieves high performance in terms of generated entropy, downstream of a reduced hardware complexity and high sampling rates.

The topology was formalized on the basis of the theoretical evaluations carried out through the simplified dynamical model on the primitive subcircuit presented in Subsection 3.3.1 and of the theory related to forced oscillators.

We analyzed the topology using the simplified dynamical model, focusing mainly on the forced part of the DNO. This analysis highlighted how the dynamical system resulting from the model constitutes a stable oscillator in the absence of the excitation signal, while in the presence of periodic excitation it is possible to observe transitions from periodic to chaotic behaviors as a function of the ratio between the oscillation frequency of the excitation signal and the natural oscillation frequency of the forced part of the DNO.

We built the topology in Cadence Virtuoso using the basic component library defined starting from the UMC 180 nm technology, and we repeated the dynamics analysis of the resulting circuit in the absence and in the presence of an excitation signal. The obtained results appeared to be consistent with the simplified dynamical

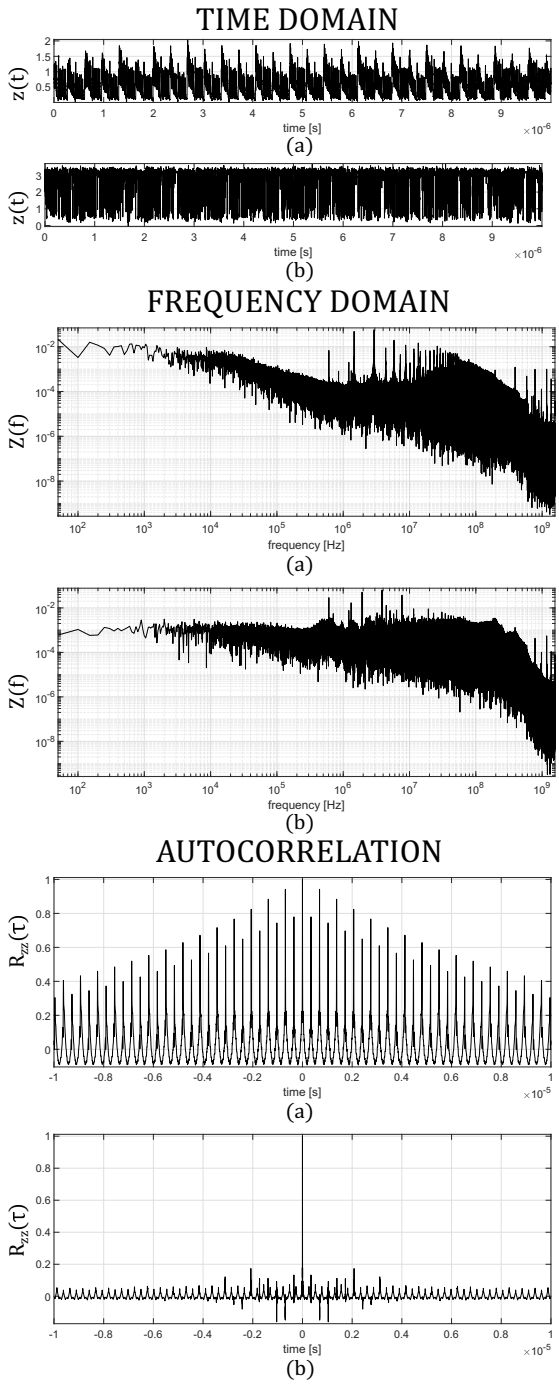

Figure 4.32: DNO output signal acquisitions from the digital I/O pins of a Xilinx Artix 7 FPGA mounted on a Arty board. The upper plots show the time behavior of configurations with strong periodicities (a) and with chaotic behavior (b). The middle plots show the spectrum of the above signals, from which we can observe the presence of harmonics in case (a) and the relatively flat spectrum in case (b). The lower plots show the autocorrelation functions of the two signals, confirming the periodic behavior in case (a), while in case (b) the autocorrelation function is almost a delta.

model, as we obtained a stable periodic oscillator in the absence of excitation and a complex dynamical system capable of passing from periodic to chaotic behaviors as a function of the ratio between the oscillation frequencies when the excitation signal is turned on.

The topology was finally implemented on FPGA. This implementation was analyzed from different points of view.

We compared the DNO under study with a Ring Oscillator and a Galois Ring Oscillator of equivalent hardware complexity, implementing for each of them 96 instances using 6 Xilinx Artix 7 FPGAs. The output signals of this implementations were sampled at 100 MHz to acquire sequences of bits, which were then used to investigate the Decorrelation Time and the Average Shannon Entropy. The comparison according these figures of merit proved that the proposed circuit is capable to reach outstanding values of entropy with negligible correlation between samples, suggesting that in every implementation the circuits were working in chaotic conditions.

We investigated the effect of the DNO circuit layout in the FPGA on the variability of the ASE between different implementations. We designed two different LUTs layouts (one condensed and one scattered), and for each of them we implemented 96 instances using 6 Xilinx Artix 7 FPGAs. We sampled the output signals of these implementations at 400 MHz to enhance the correlation between samples with respect to the previous analysis, we computed the Average Shannon Entropy on the acquired sequences and we evaluated the variability between the entropies. We observed that the condensed layout provides slightly lower variability with respect to the scattered one, reaching in any case high values of throughput of information. To minimize this variability, we proposed a 2:1 lossy compression of the generated information, by XORing the output bits of two DNO implementations working in parallel.

We evaluated the effect of temperature fluctuation on the performance of the DNO under investigation. We designed 64 implementations of the DNO on four Xilinx Artix 7 xc7a35 FPGAs (16 oscillators per FPGA), and we acquired 1 million bits long sequences at a sampling rate of 400 MHz from each implementation at three different ambient temperatures, i.e. -14 °C, 22 °C and 70 °C. The acquired sequences were used to calculate the Average Shannon Redundancy (4.10) of 10-bits symbols. We can observe that the DNO is more affected by colder temperatures with respect to hotter ones, even if the temperature-dependent loss in perfomance is almost neglectable.

We subjected the DNO to the NIST 800.22 standard statistical tests, observing that a bit-by-bit XORing of the collected bits with an 8-bits PRNG is sufficient to pass the tests with each implementation of the circuit.

By applying a reduction in the dynamical speed of the circuit adding transparent delay LUTs to the circuit loops, we observed directly the output signals of the DNO, both by providing external excitation and by internal excitation. The signal observation confirmed that, according to the ratio between the natural frequency of the oscillator and the excitation signal frequency, the DNO can pass from periodic

to chaotic dynamics.

Finally, we compared the proposed DNO with the state of the art, observing that a circuit of this type outperforms the most relevant works recently published in literature.

Reference	Chief Entropy Source	FPGA Device	Hardware Resources[a]	Throughput [Mb/s]	Post-Processing
[59]	Jitter	Xilinx Spartan-3A	528 LUTs	6	Von Neumann
[60]	Jitter and Metastability	Xilinx UltraScale	1PLL + 5 primitives + 17 LUTs	100	XOR Compression
[61]	Jitter	Xilinx Virtex-6	131202 LUTs	167.4	Stream Ciphering
[62]	Metastability	Altera Cyclone IV	298 LUTs	150	Hashing
[63]	Metastability	Xilinx Spartan-6	1 Dig. Clock Manager + 36 LUTs	12.6	Custom
[57]	Timing Skew	Xilinx Virtex-6	224 Slices	50	XOR Mixing
[64]	DNO (Undetermined Complex Dynamics)	Altera Cyclone IV	≈120 LUTs	200	Stream Ciphering
This Work	DNO (Chaos Evidence)	Xilinx Artix-7	15 LUTs	100	Stream Ciphering

Overall hardware resources necessary to design the TRNG subsystem.

Table 4.4: Comparison of the proposed solution with similar recently proposed TRNGs (NIST-tests passing)

Chapter 5

A Maximum Worst-Case Entropy Selection Algorithm and its Hardware Implementation

In this chapter we present an algorithm, called Maximum Worst-Case Entropy Selector (MWCES), that aims to identify, within a set of entropy sources, which offers the best performance in terms of worst-case entropy, also known in literature as "min-entropy". This algorithm is designed to be implemented in low-complexity digital architectures, suitable for lightweight cryptographic applications, thus allowing online maximization of the performance of a random number generation system based on Digital Nonlinear Oscillators. This chapter presents the theoretical premises underlying the algorithm formulation, some notable examples of its generic application and, finally, considerations related to its hardware implementation in FPGA.

5.1 Sub-Optimal Entropy Estimation

The analysis of the DNO presented in Chapter 4 demonstrates how an in-depth study of this class of circuits requires the use of complex and varied tools.

Different techniques to assess the statistical characteristics of True Random Number Generators were proposed in literature, as also discussed in Chapter 3 [4, 5, 56]. Most of these solutions investigate the statistical properties of the random source collecting and inspecting long binary sequences, and are not suitable for being implemented in lightweight digital hardware systems. In this chapter we present a low-complexity algorithm addressing the online evaluation of different random sources that may be available in a same chip, aiming to select the one with the highest entropy. As it is made clearer in the following, to reduce algorithmic/hardware complexity, it is necessary to resort to sub-optimal methodologies aimed at providing a rough estimate of the statistical characteristics of a circuit under examination.

The primary objective of the class of circuits analyzed in this thesis is the generation of entropy. According to Shannon [65], given an ergodic source X of symbols taking values from the set $A = \{x_1, x_2, \ldots, x_k\}$ with probability $P(X = x_i) = p_i$, $i = 1, 2, \ldots, k$, its entropy is defined as:

$$\mathcal{E}(X) = -\sum_{i=1}^{k} p_i \log_2 p_i. \tag{5.1}$$

An estimate of this expression appears particularly complex to achieve, as it requires defining a method capable of extracting information relating to the probability distribution of the analyzed source.

According to the recommendations for entropy source provided by NIST [56], the essential figure of merit to establish how unpredictable is a source is its min-entropy, defined as:

$$I(X) = \min_{1 \le i \le k}(-\log_2 p_i) = -\log_2(\max_{1 \le i \le k} p_i). \tag{5.2}$$

If X has min-entropy I, the probability for X to generate any of its symbols cannot be greater than 2^{-I}. It is observed that, unlike entropy, min-entropy depends solely on the probability of the most probable symbol of the distribution; this fact suggests that the estimate of this figure of merit can be made in a much less complex way than the estimate of the Shannon entropy of the source.

In this chapter, we present an algorithm for the assessment of the Shannon entropy, based on the estimation of the probability of the most probable symbol of the distribution associated with the evaluated source. This algorithm can be implemented on PLDs for the online evaluation of the performance of a DNO or, alternatively, for the automatic identification, within a set of sources, of the implementation with maximum entropy.

At the time of writing this thesis, the results presented in this chapter are under development for forthcoming submission to peer-reviewed journals for possible publication.

5.2 Estimation-Based Entropy Bounds

The Shannon entropy (5.1) of a random source is dependent on the probability distribution associated to the symbol generation. Attempting to assess the Shannon entropy starting from the estimation of the generation probability of the most probable symbol introduces uncertainty, since different distributions can be related to different Shannon entropy levels, even if their most probable symbols have the same generation probability. However, as shown in the following, starting from the knowledge of this probability, we can identify lower and upper bounds of the Shannon entropy.

5.2.1 Theoretical Framework

Let us consider an ergodic stochastic source S of independent and identically distributed (i.i.d.) symbols belonging to the alphabet $\mathcal{A} = \{s_1, s_2, \ldots, s_N\}$. We denote with $\mathbb{M}(N) \subset [0,1]^N \subset \mathbb{R}^N$ the set of probability mass functions expressing different generation probabilities for the symbols in $\mathcal{A}$. Accordingly, if $\boldsymbol{P} = (p_1, p_2, \ldots, p_N) \in \mathbb{M}(N)$, we have $\sum_{i=1}^{N} p_i = 1$ and $0 \leq p_i \leq 1$. Furthermore, the following properties hold.

Proposition 5.1. *Let $\boldsymbol{P} = (p_1, p_2, \ldots, p_N) \in \mathbb{M}(N)$. By denoting $p_L = \min_i p_i$ and $p_H = \max_i p_i$, it results:*

$$p_L \leq \frac{1}{N} \leq p_H. \tag{5.3}$$

Proof. Since $p_L \leq p_i \leq p_H$ for all $1 \leq i \leq N$, we have:

$$1 = \sum_{i=1}^{N} p_i \geq \sum_{i=1}^{N} p_L = N p_L, \tag{5.4}$$

or $p_L \leq 1/N$. Analogously, we have:

$$1 = \sum_{i=1}^{N} p_i \leq \sum_{i=1}^{N} p_H = N p_H, \tag{5.5}$$

or $p_H \geq 1/N$. □

We define the Shannon entropy of the source S as:

$$\mathcal{E}_S(\boldsymbol{P}) = \sum_{i=1}^{N} h(p_i) \quad [\text{bit/sym}], \tag{5.6}$$

where if $h(x) = -x \log_2 x, x \in [0,1]$. In the specific case in which the symbols are evenly distributed, i.e. $\boldsymbol{U} = (1/N, 1/N, \ldots, 1/N) \in \mathbb{M}(N)$, then $\mathcal{E}(\boldsymbol{U}) = \log_2 N = \mathcal{E}_U$.

As shown in Fig. 5.1, function h is infinitely differentiable and strictly concave for $x \in (0,1)$, having maximum value:

$$h_{\max} = h(x_{\max}) = \frac{\log_2 e}{e} \approx 0.53, \tag{5.7}$$

where $x_{\max} = 1/e \approx 0.37$. Consequently, h is strictly monotonic increasing for $x \in [0, x_{\max}]$.

Lemma 5.1. *Given two arbitrary probability mass functions $\boldsymbol{P} = (p_1, p_2, \ldots, p_N) \in \mathbb{M}(N)$ and $\boldsymbol{Q} = (q_1, q_2, \ldots, q_N) \in \mathbb{M}(N)$, it results:*

$$\mathcal{E}_S(\boldsymbol{P}) \leq -\sum_{i=1}^{N} p_i \log_2 q_i, \tag{5.8}$$

with the equality holding if and only if $\boldsymbol{P} = \boldsymbol{Q}$.

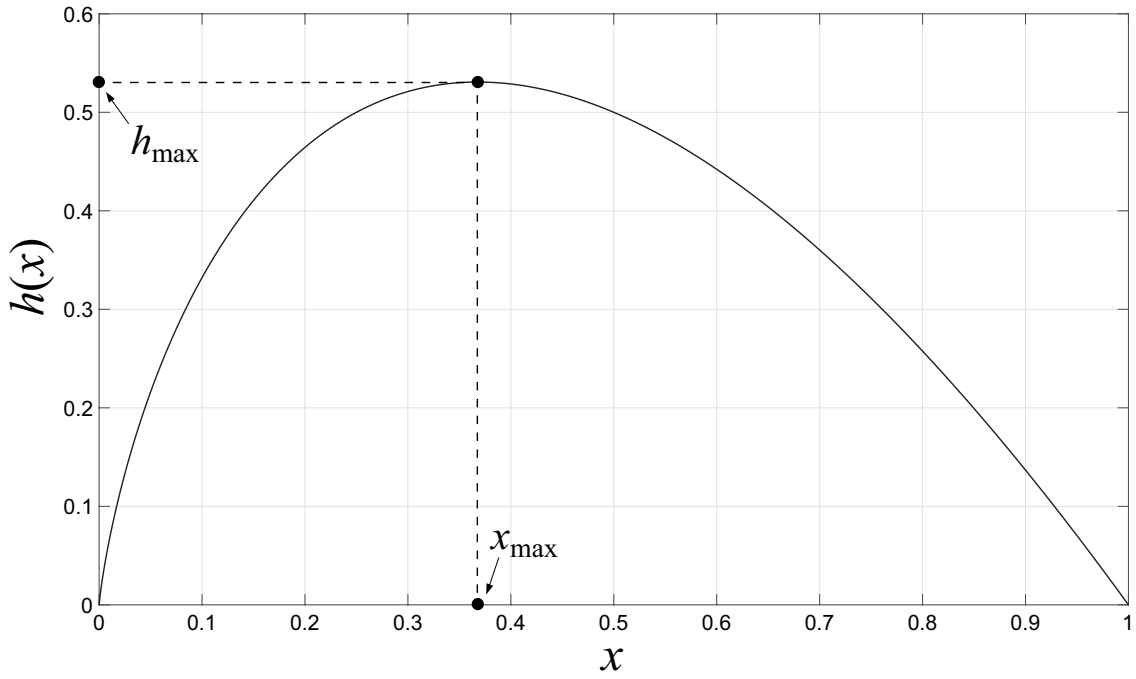

Figure 5.1: The function $h(x) = -x \log_2 x$ is monotonically increasing in the interval $[0, 1/e] = [1, x_{\max}]$.

Proof. It can be easily proved that:

$$\log_2 t \leq t - 1 \quad \forall t \in \mathbb{R}^+. \tag{5.9}$$

Indeed, $f(t) = \log_2 t$ and $g(t) = t - 1$ are infinitely differentiable functions with $f'(t) = 1/t$ and $g'(t) = 1$. It results $g(t) - f(t) = g(1) - f(1) + \int_1^t (1 - \frac{1}{\theta}) d\theta = \int_1^t h(\theta) d\theta$. We observe that:

- if $t \geq 1$, $h(t) \geq 0$ and $g(t) - f(t) = \int_1^t h(\theta) d\theta \geq 0$;
- if $0 < t < 1$, $h(t) < 0$ and $g(t) - f(t) = -\int_t^1 h(\theta) d\theta > 0$;

Accordingly to (5.9), given two real numbers $p, q \in (0, 1)$ we have $\log_2(\frac{q}{p}) \leq \frac{q}{p} - 1$ or, equivalently:

$$p - p \log_2 p \leq q - p \log_2 q. \tag{5.10}$$

Since (5.10) holds for all p_i, q_i with $i = 1, 2, \ldots, N$, we have:

$$\sum_{i=1}^{N} p_i - \sum_{i=1}^{N} p_i \log_2 p_i \leq \sum_{i=1}^{N} q_i - \sum_{i=1}^{N} p_i \log_2 q_i. \tag{5.11}$$

Recalling that $\sum_{i=1}^{N} p_i = \sum_{i=1}^{N} q_i = 1$, the inequality (5.8) is proven.

If $\boldsymbol{P} = \boldsymbol{Q}$, (5.8) reduces to an identity.

On the other hand, if (5.8) is an equality, we have to prove that $\boldsymbol{P} = \boldsymbol{Q}$. Indeed from (5.10) we note that:

$$q - p \log_2 q - p + p \log_2 p \geq 0. \tag{5.12}$$

If (5.8) is an equality, we have:

$$-\sum_{i=1}^{N} p_i \log_2 p_i = -\sum_{i=1}^{N} p_i \log_2 q_i, \tag{5.13}$$

or:

$$1 - \sum_{i=1}^{N} p_i \log_2 p_i = 1 - \sum_{i=1}^{N} p_i \log_2 q_i. \tag{5.14}$$

Recalling that $\sum_{i=1}^{N} p_i = \sum_{i=1}^{N} q_i = 1$, (5.14) can be rewritten as:

$$\sum_{i=1}^{N} (q_i - p_i \log_2 q_i - p_i + p_i \log_2 p_i) = 0. \tag{5.15}$$

As noticed in (5.12), since the summation involves non-negative terms, (5.15) implies that, for $i = 1, 2, \ldots, N$, $(q_i - p_i \log_2 q_i - p_i + p_i \log_2 p_i) = 0$, that can be rearranged as:

$$\log_2 \frac{q_i}{p_i} = \frac{q_i}{p_i} - 1 \quad i = 1, 2, \ldots, N, \tag{5.16}$$

that is true only if $p_i = q_i$. □

From Lemma 5.1 we have the following theorem.

Theorem 5.1. *For any probability mass function $\boldsymbol{P} \in \mathbb{M}(N)$ we have:*

$$\mathcal{E}_S(\boldsymbol{P}) \leq \mathcal{E}_U, \tag{5.17}$$

with the equality holding if and only if if $\boldsymbol{P} = \boldsymbol{U}$.

Proof. The proof derives directly from Lemma 5.1, noting that if in (5.8) $\boldsymbol{P} = \boldsymbol{U}$, we have $q_i = 1/N, i = 1, 2, \ldots, N$ and:

$$\mathcal{E}_S(\boldsymbol{P}) \leq -\sum_{i=1}^{N} p_i \log_2 \frac{1}{N} = \log_2 N \sum_{i=1}^{N} p_i = \mathcal{E}_U. \tag{5.18}$$

□

Theorem 5.1 states that a random source of evenly distributed independent symbols has maximal entropy equal to $\mathcal{E}_U = \log_2 N$ bits/sym.

5.2.2 Worst Case Entropy

Given a probability mass function, we can reduce the complexity related to the accurate computation of (5.6) focusing on the concept of worst case entropy. We start first to prove the following lemma.

Lemma 5.2. *Consider K real numbers $x_1, x_2, \ldots, x_K$ such that $1 > x_1 \geq x_2 \geq \cdots \geq x_K > 0$. For any K-tuple of nonnegative real numbers $\{\delta_1, \delta_2, \ldots, \delta_K\}$ such that $\delta_1 = \sum_{i=2}^{K} \delta_i$, with $0 \leq \delta_1 \leq 1 - x_1$ and $0 \leq \delta_i \leq x_i, i = 2, 3, \ldots, K$, it results:*

$$\sum_{i=1}^{K} h(x_i) \geq h(x_1 + \delta_1) + \sum_{i=2}^{K} h(x_i - \delta_i). \tag{5.19}$$

Proof. We can rewrite (5.19) as:

$$h(x_1 + \delta_1) - h(x_1) \leq \sum_{i=2}^{K} [h(x_i) - h(x_i - \delta_i)]. \tag{5.20}$$

Since the prime derivative of h:

$$h'(x) = -\log_2 x - \frac{1}{\ln 2} \tag{5.21}$$

is a strictly decreasing function, we have $h(x_1+\delta_1)-h(x_1) \leq \delta_1 h'(x_1)$. On the other hand, $h(x_i) - h(x_i - \delta_i) \geq \delta_i h'(x_i)$ and:

$$\sum_{i=2}^{K} [h(x_i) - h(x_i - \delta_i)] \geq \sum_{i=2}^{K} \delta_i h'(x_i) \geq h'(x_2) \sum_{i=2}^{K} \delta_i = \delta_1 h'(x_2). \tag{5.22}$$

Since $x_1 \geq x_2 \Rightarrow h'(x_2) \geq h'(x_1)$, we have:

$$h(x_1 + \delta_1) - h(x_1) \leq \delta_1 h'(x_1) \leq \delta_1 h'(x_2) \leq \sum_{i=2}^{K} [h(x_i) - h(x_i - \delta_i)]. \tag{5.23}$$

□

The following theorem provides the theoretical worst case entropy for a stochastic source S in which the most probable symbol has generation probability p_H.

Theorem 5.2. *Let $\boldsymbol{P} = (p_1, p_2, \ldots, p_N) \in \mathbb{M}(N), N > 2$. It results:*

$$\mathcal{E}_S(\boldsymbol{P}) \geq \mathcal{E}_{WC} = F h(p_H) + h(1 - F p_H), \tag{5.24}$$

where $p_H = \max_i p_i, i = 1, 2, \ldots, N$ and $F = \lfloor 1/p_H \rfloor$.

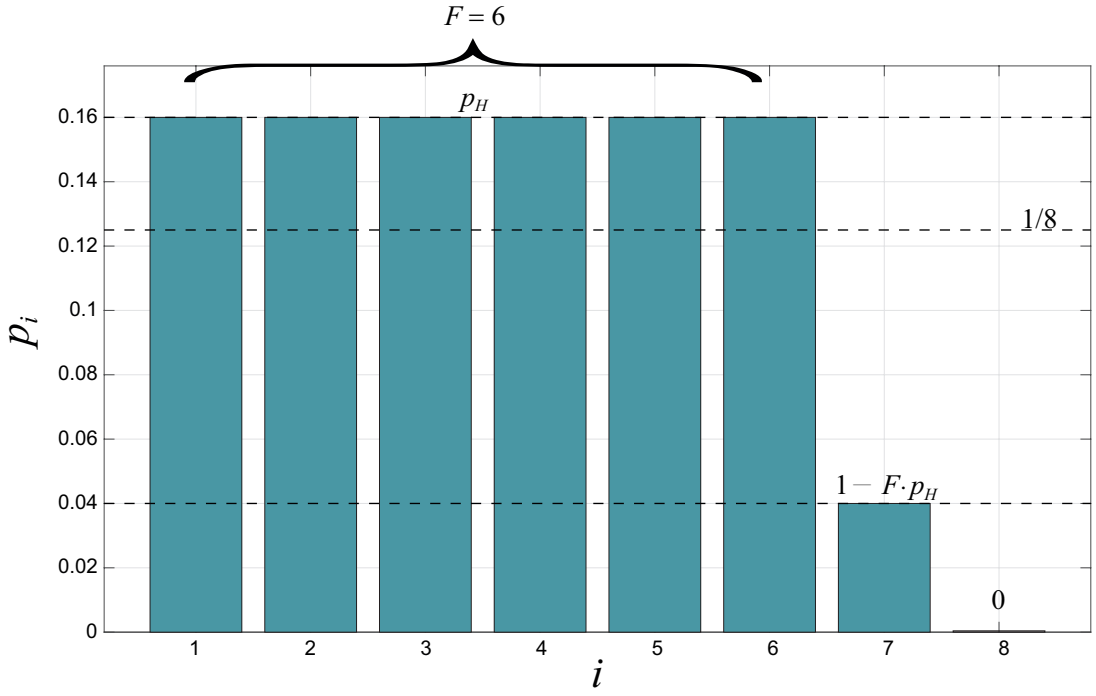

Figure 5.2: The probability mass function $\mathbf{Q}_{p_H} \in \mathcal{O}_{p_H}$ for $p_H = 0.16, N = 8$. According to the Theorem 5.2 this probability mass function provides minimum entropy in $\mathcal{O}_{p_H}$.

Proof. Entropy (5.6) is a summation that does not depend on the probability distribution ordering. Accordingly, $\mathbb{M}(N)$ can be partitioned in equivalence classes, each one containing permutations of probability mass functions, and we can focus on the subset $\mathcal{O}_{p_H} \subset \mathbb{M}(N)$ of the class representatives with weakly decreasing probabilities, i,e. satisfying the condition:

$$\boldsymbol{T} = (t_1, t_2, \ldots, t_N) \in \mathcal{O}_{p_H} \Leftrightarrow p_H = t_1 \geq t_2 \geq t_N. \tag{5.25}$$

Every probability mass function $\boldsymbol{P} \in \mathbb{M}(N)$ has a unique class representative in $(O)_{p_H}$ such that $\mathcal{E}_S(\boldsymbol{P}) = \mathcal{E}_S(\boldsymbol{T})$.

Among the probability mass functions in $\mathcal{O}_{p_H}$ we choose the element $\boldsymbol{Q}_{p_H} = (q_1, q_2, \ldots, q_N)$ for which:

$$q_i = \begin{cases} p_H, & \text{if } 1 \leq i \leq F \\ 1 - Fp_H, & \text{if } i = F+1 \\ 0, & \text{otherwise} \end{cases}. \tag{5.26}$$

An example of $\boldsymbol{Q}_{p_H}$ is shown in Fig. 5.2.

Applying (5.6), we observe that $\mathcal{E}_S(\boldsymbol{Q}_{p_H}) = Fh(p_H) + h(1 - Fp_H) = \mathcal{E}_{\text{WC}}$.

To conclude the proof, we have to show that:

$$\mathcal{E}_S(\boldsymbol{Q}_{p_H}) = \min_{\boldsymbol{T} \in \mathcal{O}_{p_H}} \mathcal{E}_S(\boldsymbol{T}). \tag{5.27}$$

To this aim, for any $\boldsymbol{P} = (p_1, p_2, \ldots, p_N) \in \mathbb{M}(N)$ we build a sequence of $F+1$ probability mass functions $\{\boldsymbol{T}^1, \boldsymbol{T}^2, \ldots, \boldsymbol{T}^{F+1}\}$, in which $\boldsymbol{T}^{F+1} = \boldsymbol{Q}_{p_H}$ and $\boldsymbol{T}^1$ is the class representative of $\boldsymbol{P}$ in $\mathcal{O}_{p_H}$, such that:

$$\mathcal{E}_S(\boldsymbol{P}) = \mathcal{E}_S(\boldsymbol{T}^1) \geq \mathcal{E}_S(\boldsymbol{T}^2) \geq \cdots \geq \mathcal{E}_S(\boldsymbol{T}^{F+1}) = \mathcal{E}_{\mathrm{WC}}. \tag{5.28}$$

Proceeding by induction, assume that for $1 \leq k \leq F$ the probability mass density $\boldsymbol{T}^k = (t_1^k, t_2^k, \ldots, t_N^k) \in \mathcal{O}_{p_H}$ has the first k probability levels equal to p_H, namely $t_1^k = t_2^k = \cdots = t_k^k = p_H$. Such element $\boldsymbol{T}^k$ exists in $\mathcal{O}_{p_H}$ since $kp_H \leq 1$ and the weakly decreasing nonnegative levels t_i^k for $k+1 \leq i \leq N$ can be arbitrarily chosen to satisfy the normalization property $\sum_{i=1}^N t_i^k = 1$.

Comparing $\boldsymbol{T}^k$ with $\boldsymbol{Q}_{p_H}$, we observe that $q_{k+1} \geq t_{k+1}$. Indeed, if $q_{k+1} < p_H$, we recall that, according to (5.26), $q_i = p_H$ for $i = 1, 2, \ldots, F$, $q_i = 0$ for $i = F+2, F+3, \ldots, N$ and q_{k+1} is the unique value lower than p_H that satisfies the normalization property for $\boldsymbol{Q}_{p_H}$. Thus, the normalization property for $\boldsymbol{T}^k$ can be satisfied only if $q_{k+1} \geq t_{k+1}$. If $q_{k+1} = p_H$ we have nothing to prove.

Starting from $\boldsymbol{T}^k$, we now build a new mass probability function $\boldsymbol{W}^{k+1} = (w_1^{k+1}, w_2^{k+1}, \ldots, w_N^{k+1} \in \mathbb{M}(N)$ such to have:

$$w_i^{k+1} = \begin{cases} q_i, & \text{if } 1 \leq i \leq k+1 \\ t_i^k - \delta_i, & \text{otherwise} \end{cases}, \tag{5.29}$$

where $0 \leq \delta_i \leq t_i^k, i = 1, 2, \ldots, N$ are arbitrary values such to satisfy $\sum_{i=k+2}^N \delta_i = q_{k+1} - t_{k+1}^k$.

The construction of $\boldsymbol{W}^{k+1}$ is well defined since $w_i^{k+1} \geq 0$ and $1 = \sum_{i=1}^N t_i^k = kp_H + \sum_{i=k+1}^N t_i^k = kp_H + \sum_{i=k+1}^N t_i^k + q_{k+1} - t_{k+1}^k - \sum_{i=k+2}^N \delta_i = kp_H + q_{k+1} + \sum_{i=k+2}^N (t_i^k - \delta_i) = \sum_{i=1}^N w_i^{k+1}$.

As a result, since by construction the first k probabilities of $\boldsymbol{Q}_{p_H}$, $\boldsymbol{W}^{k+1}$ and $\boldsymbol{T}^k$ are equal, we have:

$$\begin{aligned} \mathcal{E}_S(\boldsymbol{W}^{k+1}) - \mathcal{E}_S(\boldsymbol{T}^k) &= \sum_{i=k+1}^N [h(w_i^{k+1}) - h)t_i^k)] = \\ &= h(q_{k+1}) - h(t_{k+1}^k) + \sum_{i=k+2}^N [h(t_i^k - \delta_i) - h(t_i^k)] = \\ &= h(t_{k+1}^k + \delta_{k+1}^k) - h(t_{k+1}^k) + \sum_{i=k+2}^N [h(t_i^k - \delta_i) - h(t_i^k)]. \end{aligned} \tag{5.30}$$

According to Lemma 5.2, we have $\mathcal{E}_S(\boldsymbol{W}^{k+1}) \leq \mathcal{E}_S(\boldsymbol{T}^k)$.

Finally, we define $\boldsymbol{T}^{k+1}$ as the class representative of $\boldsymbol{W}^{k+1}$ in $\mathcal{O}_{p_H}$. Since $\mathcal{E}_S(\boldsymbol{T}^{k+1}) = \mathcal{E}_S(\boldsymbol{W}^{k+1}), k = 1, 2, \ldots, F$, proceeding iteratively we proved (5.28), noting that $\boldsymbol{T}^{F+1} = \boldsymbol{Q}_{p_H}$. Being the initial probability mass function $\boldsymbol{P}$ arbitrary in $\mathbb{M}(N)$, also (5.27) is true. □

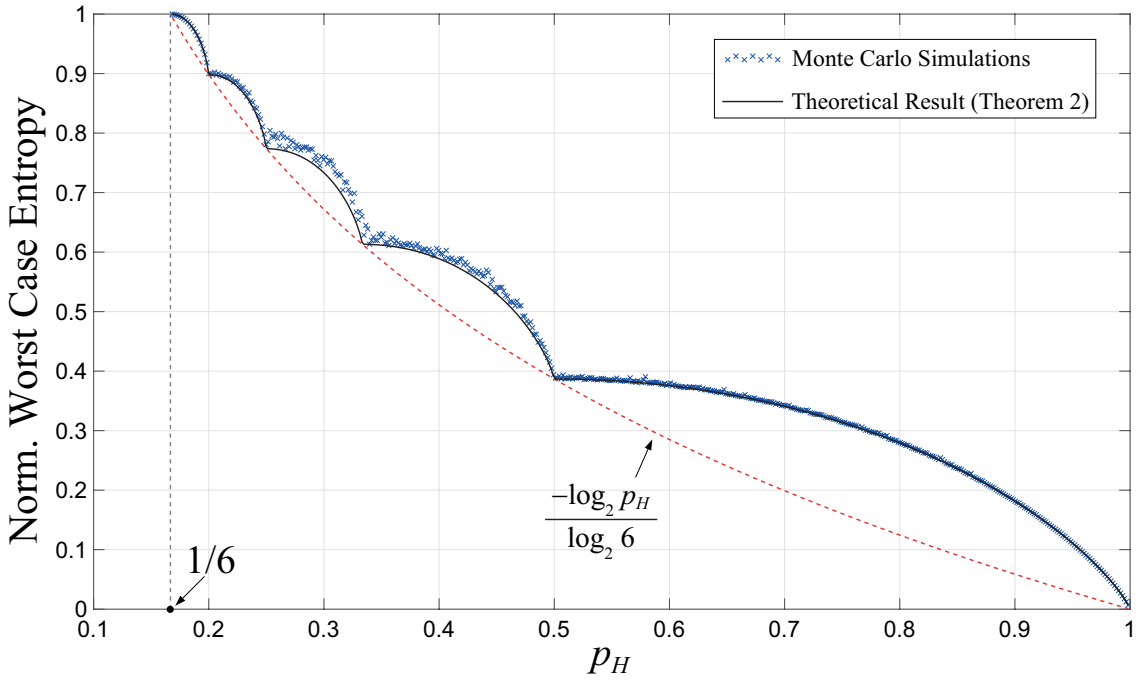

Figure 5.3: Normalized worst case entropy $\mathcal{E}_{WC}/\mathcal{E}_U$ for $N = 6$ and p_H ranging in the interval $(\frac{1}{N}, 1-\frac{1}{N})$. Theoretical bounds are compared with Monte Carlo simulations (minimum entropy among 1000 probability mass functions randomly chosen, for each p_H, in $\mathcal{O}_{p_H}$).

Fig. 5.3 reports as an example the normalized worst case entropy for $N = 6$ and p_H ranging in the interval $(\frac{1}{N}, 1 - \frac{1}{N})$.

A simpler expression providing a lower bound for the worst case entropy is presented in the following corollary.

Corollary 5.1. *Let* $\boldsymbol{P} = (p_1, p_2, \ldots, p_N) \in \mathbb{M}(N), N > 2$. *It results:*

$$\mathcal{E}_S(\boldsymbol{P}) \geq \mathcal{E}_{WC} \geq -\log_2 p_H, \tag{5.31}$$

where $p_H = \max_i p_i, i = 1, 2, \ldots, N$.

Proof. The min-entropy can be rewritten as follows:

$$\begin{aligned} -\log_2 p_H &= -\log_2 p_H \left(1 - \left\lfloor \frac{1}{p_H} \right\rfloor p_H + \left\lfloor \frac{1}{p_H} \right\rfloor p_H\right) = \\ &= -\left\lfloor \frac{1}{p_H} \right\rfloor p_H \log_2 p_H - \left(1 - \left\lfloor \frac{1}{p_H} \right\rfloor p_H\right) \log_2 p_H = \\ &= \left\lfloor \frac{1}{p_H} \right\rfloor h(p_H) + h\left(1 - \left\lfloor \frac{1}{p_H} \right\rfloor p_H\right) \frac{\log_2 p_H}{\log_2 \left(1 - \left\lfloor \frac{1}{p_H} \right\rfloor p_H\right)}. \end{aligned} \tag{5.32}$$

The quantity $1 - \left\lfloor \frac{1}{p_H} \right\rfloor p_H$ is the reminder of the division $1/p_H$ and is lower than $p_H < 1$. As a result, we have $\frac{\log_2 p_H}{\log_2 \left(1 - \left\lfloor \frac{1}{p_H} \right\rfloor p_H\right)} \leq 1$ and $-\log_2 p_H \leq \mathcal{E}_{\mathrm{WC}}$. □

5.2.3 Best Case Entropy

Following a similar reasoning, it is possible to calculate the theoretical maximum entropy for the probability mass functions in $\mathcal{O}_{p_H}$, proving the following theorem.

Theorem 5.3. *Let* $\boldsymbol{P} = (p_1, p_2, \ldots, p_N) \in \mathbb{M}(N), N > 2$. *It results:*

$$\mathcal{E}_S(\boldsymbol{P}) \leq \mathcal{E}_{BC} = (N-1)h\left(\frac{1 - p_H}{N-1}\right) + h(p_H), \tag{5.33}$$

where $p_H = \max_i p_i, i = 1, 2, \ldots, N$.

Proof. Consider a generic probability mass function $\boldsymbol{P} = (p_1, p_2, \ldots, p_N) \in \mathbb{M}(N)$, $N > 2$ having maximum symbol probability p_H. Since entropy is a summation that does not depend on the probability distribution ordering, we can substitute $\boldsymbol{P}$ with its unique class representative $\boldsymbol{T} \in \mathcal{O}_{p_H}$.

In this way we know that the first element t_1 of $\boldsymbol{T}$ is equal to p_H and that $\sum_{i=2}^{N} t_i = 1 - p_H$.

Consider now the probability mass function $\boldsymbol{R}_{p_H} = \{r_1, r_2, \ldots, r_N\}$ for which:

$$r_i = \begin{cases} p_H, & \text{if } i = 1 \\ \frac{1-p_H}{N-1}, & \text{otherwise} \end{cases}, \tag{5.34}$$

whose entropy is equal to:

$$\mathcal{E}_S(\boldsymbol{R}_{p_H}) = h(p_H) + (N-1)h\left(\frac{1-p_H}{N-1}\right). \tag{5.35}$$

The theorem is proved simply applying Lemma 5.1 on $\boldsymbol{T}$ and $\boldsymbol{R}_{p_H}$:

$$\begin{aligned}
\mathcal{E}_S(\boldsymbol{P}) = \mathcal{E}_S(\boldsymbol{T}) &\leq -\sum_{i=1}^{N} p_i \log_2 r_i = \\
&= -p_H \log_2 p_H - \sum_{i=2}^{N} p_i \log_2 \left(\frac{1-p_H}{N-1}\right) = \\
&= -p_H \log_2 p_H - \log_2 \left(\frac{1-p_H}{N-1}\right) \sum_{i=2}^{N} p_i = \\
&= -p_H \log_2 p_H - (1-p_H) \log_2 \left(\frac{1-p_H}{N-1}\right) = \\
&= -p_H \log_2 p_H - (N-1)\frac{1-p_H}{N-1} \log_2 \left(\frac{1-p_H}{N-1}\right) = \\
&= h(p_H) + (N-1)h\left(\frac{1-p_H}{N-1}\right) = \mathcal{E}_S(\boldsymbol{R}_{p_H}) = \mathcal{E}_{\mathrm{BC}}.
\end{aligned} \tag{5.36}$$

□

Fig. 5.4 reports as an example the normalized best and worst case entropies for $N = 6$ and p_H ranging in the interval $(\frac{1}{N}, 1 - \frac{1}{N})$.

5.3 Sequence Length and Symbols Generation Probabilities

In Section 5.2 we addressed the uncertainty given by the attempt to estimate the entropy of a source based solely on the knowledge of the generation probability of its most likely symbol.

Targeting the design of an estimator for this probability, it is also necessary to take into account another factor that can affect the quality of the estimate: typically an estimator bases its functioning on the observation of events, each of which provides a certain amount of information necessary to refine the accuracy of the estimate with respect to the parameter of interest. In our case, this involves the need to observe sequences of symbols generated by the analyzed source whose length is long enough to allow for an adequate estimation of the probability of the most likely symbol evaluating its frequency in the sequence under observation.

From a theoretical point of view, it is possible to establish what the minimum length of the sequences must be to obtain an adequate estimate, by evaluating the

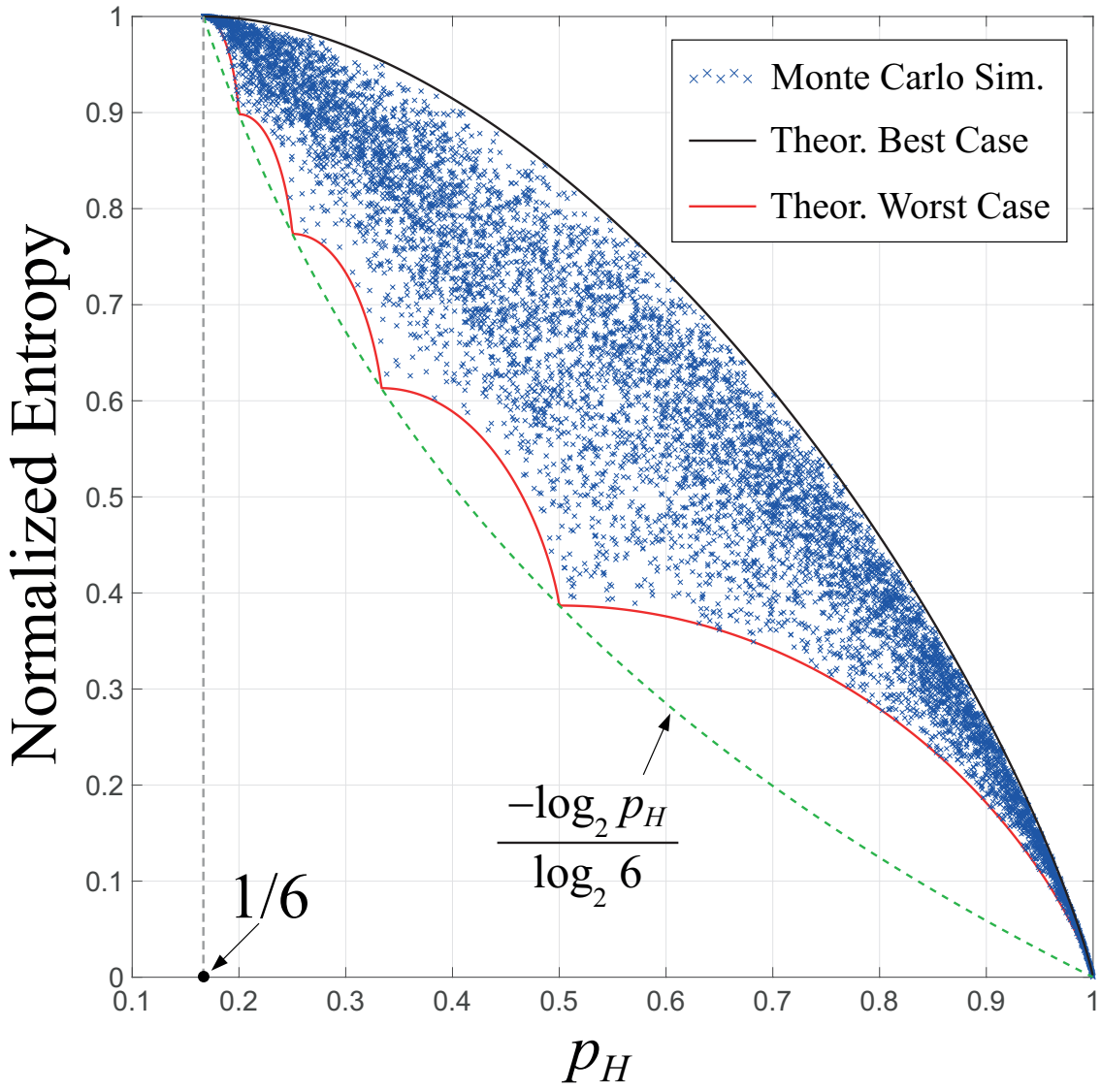

Figure 5.4: Normalized entropy $\mathcal{E}_S/\mathcal{E}_U$ for $N = 6$ and p_H ranging in the interval $(\frac{1}{N}, 1 - \frac{1}{N})$. Theoretical bounds are compared with Monte Carlo simulations (5000 random values for p_H, uniformly distributed, one random probability mass function in $\mathcal{O}_{p_H}$ for each p_H).

minimum number of elements that a source must generate to ensure that a symbol with a certain probability is generated a defined number of times.

Consider again an ergodic stochastic source S of i.i.d. symbols belonging to the alphabet $\mathcal{A} = \{s_1, s_2, \ldots, s_N\}$. Since the alphabet $\mathcal{A}$ is indexed, there exists a look-up table:

$$\phi : \mathcal{A} \to \{1, 2, \ldots, N\} \subset \mathbb{N} \tag{5.37}$$

that associates any symbol $s \in \mathcal{A}$ to its index in $\mathbb{N}$. For example, if $\mathcal{A} = \{A, B, C\}$, the index of the symbol B is $\phi(B) = 2$.

For any $L \in \mathbb{N}$, we define the set:

$$\Omega(N, L) = \left\{ \boldsymbol{o} \in \{0, 1, \ldots, L\}^N : \sum_{i=1}^{N} o_i = L \right\} \tag{5.38}$$

as the set of all the possible integer vectors counting the occurrences for the N symbols in $\mathcal{A}$ when observing sequences of L elements. Hereafter, vectors $\boldsymbol{o} \in \Omega(N, L)$ are named occurrences vectors.

For example, if $\mathcal{A} = \{A, B, C\}$ and $L = 4$, the set $\Omega(3, 4)$ contains integer vectors in $\{0, 1, \ldots, 4\}^3$ enumerating the symbols occurrences when observing all the possible sequences of $L = 4$ elements, as shown in Table 5.1.

Any sequence of L elements $\sigma = \{s_1, s_2, \ldots, s_L\}$, being $s_i \in \mathcal{A}$, is described by the sequence of alphabet indexes identifying the symbols in $\mathcal{A}$ that are generated according to the sequence order, i.e.:

$$\sigma = \{s_1, s_2, \ldots, s_L\} \Leftrightarrow \{i_1 = \phi(s_1), i_2 = \phi(s_2), \ldots, i_L = \phi(s_L)\}. \tag{5.39}$$

Since the elements in the sequence are i.i.d., the sequence σ has generation probability equal to:

$$P(\sigma) = \prod_{j=1}^{L} p_{\phi(s_j)} = \prod_{j=1}^{L} p_{i_j}. \tag{5.40}$$

Consider now the occurrences vector $\boldsymbol{o} \in \Omega(N, L)$. We define the function $H : \Omega(N, L) \to \mathcal{A}^L$ providing the set of sequences of L elements counting the same symbol occurrences given by $\boldsymbol{o}$. For example, referring to Table 5.1:

$$H(\boldsymbol{o}_2) = \{AAAB, AABA, ABAA, BAAA\}. \tag{5.41}$$

Accordingly, the generation probability for any occurrences vector $\boldsymbol{o} \in \Omega(N, L) = \{o_1, o_2, \ldots, o_N\}$ is equal to the generation probability of any sequence counting the same symbol occurrences given by $\boldsymbol{o}$, i.e.:

$$P(\boldsymbol{o}) = \sum_{\sigma \in H(\mathbf{0})} P(\sigma). \tag{5.42}$$

Vectors	$\boldsymbol{o}_i \in \Omega(3,4)$	Sequences σ (Realizations)
$\boldsymbol{o}_1 =$	$\{4,0,0\} \rightarrow$	$AAAA$
$\boldsymbol{o}_2 =$	$\{3,1,0\} \rightarrow$	$AAAB, AABA, ABAA, BAAA$
$\boldsymbol{o}_3 =$	$\{3,0,1\} \rightarrow$	$AAAC, AACA, ACAA, CAAA$
$\boldsymbol{o}_4 =$	$\{2,2,0\} \rightarrow$	$AABB, ABAB, ABBA, BAAB, BABA, BBAA$
$\boldsymbol{o}_5 =$	$\{2,1,1\} \rightarrow$	$AABC, AACB, ABAC, ABCA, ACAB, ACBA$ $BAAC, BACA, BCAA, CAAB, CABA, CBAA$
$\boldsymbol{o}_6 =$	$\{2,0,2\} \rightarrow$	$AACC, ACAC, ACCA, CAAC, CACA, CCAA$
$\boldsymbol{o}_7 =$	$\{1,3,0\} \rightarrow$	$ABBB, BABB, BBAB, BBBA$
$\boldsymbol{o}_8 =$	$\{1,2,1\} \rightarrow$	$ABBC, ABCB, ACBB, BABC, BACB, BBAC$ $BBCA, BCAB, BCBA, CABB, CBAB, CBBA$
$\boldsymbol{o}_9 =$	$\{1,1,2\} \rightarrow$	$ABCC, ACBC, ACCB, BACC, BCAC, BCCA$ $CABC, CACB, CBAC, CBCA, CCAB, CCBA$
$\boldsymbol{o}_{10} =$	$\{1,0,3\} \rightarrow$	$ACCC, CACC, CCAC, CCCA$
$\boldsymbol{o}_{11} =$	$\{0,4,0\} \rightarrow$	$BBBB$
$\boldsymbol{o}_{12} =$	$\{0,3,1\} \rightarrow$	$BBBC, BBCB, BCBB, CBBB$
$\boldsymbol{o}_{13} =$	$\{0,2,2\} \rightarrow$	$BBCC, BCBC, BCCB, CBBC, CBCB, CCBB$
$\boldsymbol{o}_{14} =$	$\{0,1,3\} \rightarrow$	$BCCC, CBCC, CCBC, CCCB$
$\boldsymbol{o}_{15} =$	$\{0,0,4\} \rightarrow$	$CCCC$

Table 5.1: The set $\Omega(3,4)$ containing the integer vectors in $\{0,1,\ldots,4\}^3$ associated to different symbol occurrences, depending on the different possible realizations for sequences of $L = 4$ elements generated by a source of i.i.d. symbols taken from the alphabet $\mathcal{A} = \{A, B, C\}$. The number of different sequences is $N^L = 3^4 = 81$, in the right column.

Since the sequences in $H(\boldsymbol{o})$ share the same generation probability, we have:

$$\begin{aligned} P(\boldsymbol{o}) &= \sum_{\sigma \in H(\mathbf{0})} P(\sigma) = \#H(\boldsymbol{o}) \prod_{j=1}^{L} p_{i_j} = \\ &= \#H(\boldsymbol{o}) \prod_{j=1}^{N} p_j^{o_j}, \end{aligned} \tag{5.43}$$

being $\#H(\boldsymbol{o})$ the cardinality of the set $H(\boldsymbol{o})$ and p_j the probability of the j-th symbol in alphabet $\mathcal{A}$.

As a result, to calculate the generation probability $P(\boldsymbol{o})$ we need to express the cardinality $\#H(\boldsymbol{o})$, that is equal to:

$$\#H(\boldsymbol{o}) = \frac{L!}{\prod_{j=1}^{N} o_j!}. \tag{5.44}$$

For example, referring to Table 5.1, $\#H(\boldsymbol{o}_2) = \frac{4!}{3!1!0!} = 4$.

Summarizing, the generation probability for any occurrences vector $\boldsymbol{o} \in \Omega(N, L)$ $= \{o_1, o_2, \ldots, o_N\}$ is equal to:

$$P(\boldsymbol{o}) = \frac{L!}{\prod_{j=1}^{N} o_j!} \prod_{j=1}^{N} p_j^{o_j}. \tag{5.45}$$

Fig. 5.5 reports a comparison between the calculations of (5.45) for the set $\Omega(3, 4)$ shown in Table 5.1 and Monte Carlo simulations.

5.3.1 Event Space Size

The set $\Omega(N, L)$ satisfies the following properties.

The number of different sequences composed by L elements taken from an alphabet of N symbols is given by:

$$\sum_{\boldsymbol{o} \in \Omega(N,L)} \#H(\boldsymbol{o}) = N^L. \tag{5.46}$$

For example, the different sequences in Table 5.1 are $N^L = 3^4 = 81$.

On the other hand, the cardinality of $\Omega(N, L)$ can be calculated noting that, since $\Omega(N, L)$ is defined as follows:

$$\Omega(N, L) = \left\{ \boldsymbol{o} = \{o_1, o_2, \ldots, o_N\} : o_i \in \{0, 1, \ldots, L\}, \sum_{i=1}^{N} o_i = L \right\}, \tag{5.47}$$

for $N = 1$ the set $\Omega(1, L)$ is always composed by a single element:

$$\boldsymbol{o} = \{L\}, \tag{5.48}$$

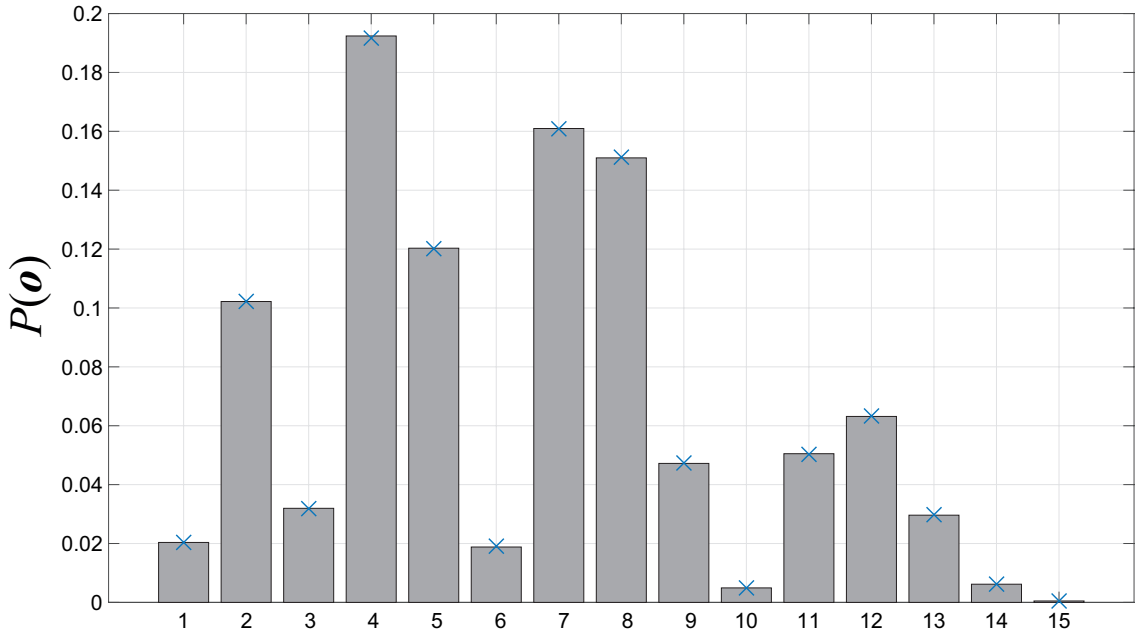

Figure 5.5: Numerical calculation of (5.45) and comparison with Monte Carlo simulation results, in blue crosses (one million sequences, $N = 3, L = 4$ for a random probability mass function $\mathbf{P} \in \mathbb{M}(3)$).

while for $N \geq 2$ each element of $\Omega(N, L)$ can be expressed in the following way:

$$\begin{cases} \boldsymbol{o} = \left\{ o_1, o_2, \ldots, o_{N-1}, L - \sum_{i=1}^{N-1} o_i \right\} \\ o_1 \in \{0, 1, \ldots, L\} \\ o_2 \in \{0, 1, \ldots, L - o_1\} \\ \vdots \\ o_{N-1} \in \left\{ 0, 1, \ldots, L - \sum_{i=1}^{N-2} o_i \right\} \end{cases} . \tag{5.49}$$

As a consequence, the cardinality of the set $\Omega(N, L)$ is given by the following expression:

$$\#\Omega(N, L) = \begin{cases} 1, & \text{if } N = 1 \\ L + 1, & \text{if } N = 2 \\ \sum_{o_1=0}^{L} \sum_{o_2}^{L-o_1} \cdots \sum_{o_{N-2}=0}^{L-\sum_{i=1}^{N-3} o_i} (o_{N-2} + 1), & \text{if } N > 2 \end{cases} . \tag{5.50}$$

5.3.2 Events Counting Statistics

Now that we defined the set $\Omega(N, L)$ of vectors that count the occurrences of the symbols generated within a sequence, we can tackle the problem that we initially set ourselves, which is to establish how many elements a source must generate in order to generate a certain symbol $s_j \in \mathcal{A}$ M times.

If p_j is the generation probability for the symbol s_j, we expect to have in T trials (observations), in average, Tp_j occurrences of that symbol. Therefore, the mean number of observations we have to perform such to have M occurrences for the j-th symbol must satisfy the relation $E[T_j]p_j = M$, that is:

$$E[T_j] = \overline{T}_j = \frac{M}{p_j}. \tag{5.51}$$

This result is limited to average values, for specific symbols.

In the following we refer to a more precise statement of our problem, considering the statistical event $\Psi_{N,L}^M$, corresponding to a source with alphabet $\mathcal{A}$ of size N that has to generate L elements to have at least a symbol in $\mathcal{A}$ generated M times.

Regarding the probability of the event $\Psi_{N,L}^M$, we note the following properties:

- if $L < M$, the probability $P(\Psi_{N,L}^M) = 0$ since the number of generation steps is smaller than M and no symbol can be generated more than $L < M$ times;
- if $L > N(M-1)+1$, the probability $P(\Psi_{N,L}^M) = 0$ since at least one symbol was generated M times in previous generation steps; indeed, in the worst case, after $N(M-1)+1$ steps we have all the N symbols generated $M-1$ times, plus one occurrence of one symbol that brings its count to M.

These considerations imply that:

$$L \notin \{M \leq q \leq N(M-1)+1, q \in \mathbb{N}\} \Rightarrow P(\Psi_{N,L}^M) = 0. \tag{5.52}$$

Assuming to focus on the interval $M \leq L \leq N(M-1)+1$, we can calculate the probability of $\Psi_{N,L}^M$ partitioning the event space according to the last generated symbol s_L in the sequence, i.e.:

$$\begin{aligned} P(\Psi_{N,L}^M) &= \sum_{i=1}^{N} P(\Psi_{N,L}^M | x_L = s_i) P(x_L = s_i) = \\ &= \sum_{i=1}^{N} P(\Psi_{N,L}^M | x_L = s_i) p_i. \end{aligned} \tag{5.53}$$

The term $P(\Psi_{N,L}^M | x_L = s_i)$ is the probability that at the L-th step of the generation process the event $\Psi_{N,L}^M$ occurs thanks to the last symbol s_i being generated for its M-th time.

The conditioned event $\Psi_{N,L}^M | (x_L = s_j)$ implies that:

- in the first $L-1$ steps of the generation process the symbol s_j appears $M-1$ times;
- in the first $L-1$ steps of the generation process any other symbol $s_i \neq s_j$ was generated no more than $M-1$ times;

- the occurrences of the symbols in the first $L-1$ elements of the sequence, described by the occurrences vector $\boldsymbol{o} \in \Omega(N, L-1)$, are such that the following constraints are satisfied:

$$\begin{cases} \sum_{i=1}^{N} o_i = L-1 \\ o_j = M-1 \\ 0 \leq o_i \leq M-1 \text{ for } i \neq j \end{cases} . \tag{5.54}$$

  We denote with $\Omega_j \subset \Omega(N, L-1)$ the set of all occurrences vectors $\boldsymbol{o} \in \Omega(N, L-1) \subset \mathbb{N}^N$ satisfying the contraints (5.54).

Recalling that the occurrences vectors $\boldsymbol{o} \in \Omega_j$ represent disjoint events and recalling (5.45), the probability of the conditioned event $\Psi_{N,L}^M | x_L = s_j$ is:

$$P(\Psi_{N,L}^M | x_L = s_j) = \sum_{\boldsymbol{o} \in \Omega_j} \frac{(L-1)!}{\prod_{i=1}^{N} o_i!} \prod_{i=1}^{N} p_i^{o_i}. \tag{5.55}$$

Combining (5.55) with (5.53) we have, if $M \leq L \leq N(M-1)+1$:

$$P(\Psi_{N,L}^M) = \sum_{j=1}^{N} p_j \sum_{\boldsymbol{o} \in \Omega_j} \frac{(L-1)!}{\prod_{i=1}^{N} o_i!} \prod_{i=1}^{N} p_i^{o_i}, \tag{5.56}$$

that can be rewritten as:

$$P(\Psi_{N,L}^M) = \frac{(L-1)!}{(M-1)!} \sum_{j=1}^{N} \sum_{\boldsymbol{o} \in \Omega_j} p_j^{M-1} \prod_{i=1, i \neq j}^{N} \frac{p_i^{o_i}}{o_i!}. \tag{5.57}$$

This expression provides the complete statistical characterization of the event $\Psi_{N,L}^M$ for $M \leq L \leq N(M-1)+1$, recalling that out of this bound $P(\Psi_{N,L}^M) = 0$. The necessary conditon to have $P(\Psi_{N,L}^M) > 0$ can be also rewritten as:

$$M_{\min} = \left\lceil \frac{L-1}{N} \right\rceil < M < L+1 = M_{\max}. \tag{5.58}$$

Fig. 5.6 shows the comparison between the numerical calculation of (5.57) and Monte Carlo simulation results for $N = 5, L = 20$ and the uniform probability mass function $\mathbf{P} \in \mathbb{M}(5)$.

5.4 Maximum Worst-Case Entropy Selection Algorithm

In Sections 5.2 and 5.3 we evaluated the range of uncertainty associated with the attempt to assess the Shannon entropy of an information source based on the estimation of the probability p_H of the most probable symbol of the source, and we defined what should be the length L of the sequence of generated symbols that the estimator must analyze to correctly recognize what the value of this probability is.

On the basis of the obtained results, we can establish that:

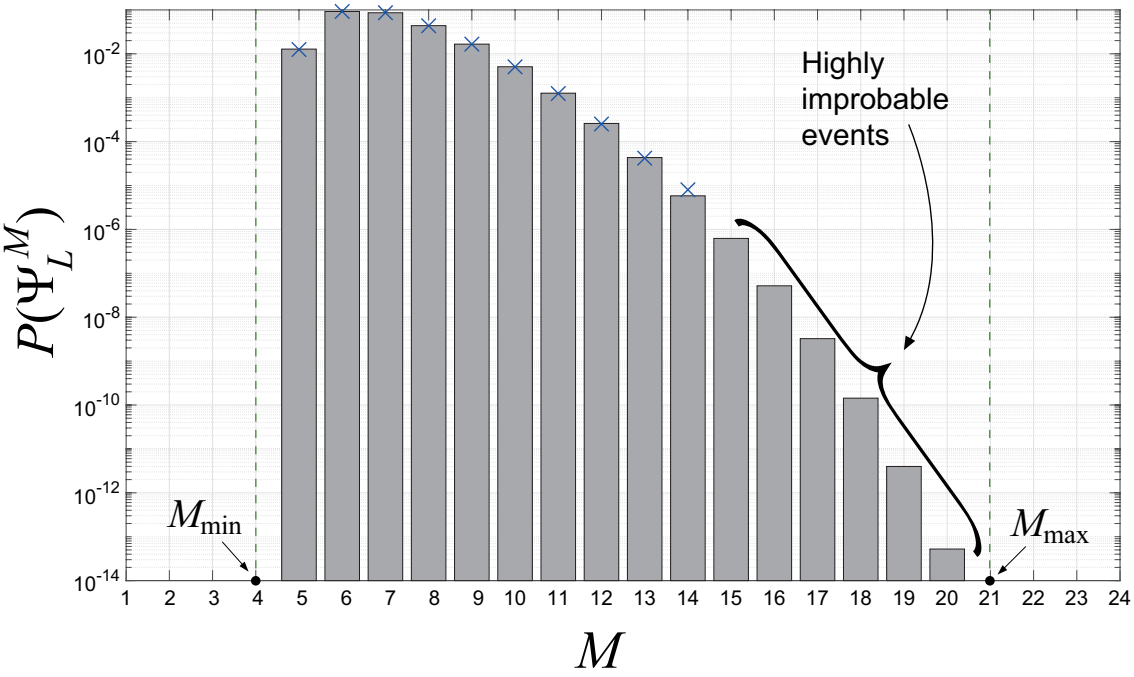

Figure 5.6: Numerical calculation of (5.57) and comparison with Monte Carlo simulation results (blue crosses), for $N = 5, L = 20$ and the uniform probability mass function $\mathbf{P} \in \mathbb{M}(5)$ (one million sequences). Highly improbable events require orders of magnitude longer Monte Carlo simulations for being detected.

- the uncertainty given by an entropy estimator based on the evaluation of the probability p_H of the most probable symbol is not constant, as it depends on the value of p_H;
- according to the values of M and L, the calculation of the probability $P(\Psi_{N,L}^{M})$ through the application of (5.57) can become unfeasible because of the magnitude of the involved parameters.

However, the principle underlying an estimator of this type can be used for comparative purposes applied to sets of sources, rather than for the assessment of the entropy of a single source.

Consider a set of sources with different maximum symbol generation probabilities.

Since both the best-case and the worst-case entropies are monotonically decreasing functions of p_H, the source in the set with the lowest p_H will be more likely to have higher entropy with respect to the other sources.

Furthermore, if we limit our goal to the identification of the source with the lowest p_H in the set, we can find heuristically a minimum value of L for which the source with the lowest p_H in the set is identified.

These considerations are the starting point of a procedure for identifying the information source with the maximum worst-case entropy within a set, defined as the Maximum Worst-Case Entropy Selector (MWCES).

5.4.1 Source Selection Procedure

The source selection procedure is described by the algorithm shown in Fig. 5.7.

Consider a set of N entropy sources, each one generating K-bits long symbols.

To evaluate the sources entropy, we define 2^K counters (`symbol_count`) with overflow value equal to 2^L (one counter per symbol in the dictionary) and another counter (`total_count`) able to count up to 2^{K+L}.

The algorithm scans sequentially all the sources to understand which one has the lowest p_H and, therefore, the maximum entropy.

The procedure requires to sample the selected entropy source output and to increase by one the values of the counter corresponding to the generated symbol and of `total_count`. These operations are repeated until one of the symbol counters reaches the 2^L overflow value.

The p_H estimation is given by the value stored in `total_count` when the overflow event happens, as it corresponds to the time needed to reach the overflow.

The source with the highest `total_count` value is, according to the selector, the one with the lowest p_H and, consequently, the one with the highest entropy.

5.4.2 Algorithm Optimization Strategies

The algorithm complexity depends on three parameters, i.e. the number of sources N, the symbols bit length K and the overflow value bit length L. The

```
ALGORITHM A1:
 1. best_source = 0;
 2. best_count = 0;
 3. for(source = 0; source < N; source++)
 4. {
 5.   overflow = 0;
 6.   total_count = 0;
 7.   for(symbol = 0; symbol < 2^K; symbol++)
 8.   {
 9.     symbol_count[symbol] = 0;
10.   }
11.   while(overflow == 0)
12.   {
13.     symbol = get_sample(source);
14.     symbol_count[symbol]++;
15.     total_count++;
16.     if(symbol_count[symbol] == 2^L)
17.     {
18.       overflow = 1;
19.     }
20.   }
21.   if(total_count > best_count)
22.   {
23.     best_count = total_count;
24.     best_source = source;
25.   }
26. }
```

Figure 5.7: Maximum Worst-Case Entropy Selector algorithm.

maximum possible execution time is reached when, for every source, at the iteration preceding the overflow all the `symbol_count` array elements are equal to $2^L - 1$. This means that, referring to the worst case, the proposed algorithm has a time complexity order $O(N2^{K+L})$.

The time complexity grows with K and L as a power of 2, so these two parameters should be kept as small as possible without affecting the selector performance. It is not possible to understand a priori which are the minimum values of K and L that match this condition, because they depend on the statistical properties of the sources on which the MWCES is employed. At the same time, an a posteriori configuration based on the knowledge of the sources statistical properties would make the use of the MWCES useless, since in that condition we would already know which is the source with the maximum ASE.

To overcome these problems, we defined a procedure for configuring K and L which takes into consideration the specific sources to be analyzed without having to directly evaluate their statistical properties.

The configuration procedure is described by the algorithm shown in Fig. 5.8.

Suppose that we want to apply the MWCES to a group of N sources with the purpose of generating K_{MAX}-bits long symbols. To configure K and L we start implementing M groups of these N sources. In each group, the sources are characterized by slightly different statistical properties with respect to their equivalents in the other groups; such differences are an effect of their implementation process.

For each source, a sequence of $2^{K_{MAX}+L_{MAX}}$ K_{MAX}-bits long symbols is generated. L_{MAX} is the exponent of the maximum overflow threshold, defined as a power of two, we can accept for the MWCES that we are implementing.

According to [55], a good rule of thumb for 2^L is to set it greater than 10. The minimum power of two greater than this value is 16, i.e. $L = 4$. For this reason the algorithm takes 4 as a lower bound for L.

The generated sequences are used to compute the K_{MAX}-bits ASEs. According to this computation, we identify the maximum ASE of every group and we compute the mean value of these maxima. The obtained parameter (`ase_m`) is taken as a reference of the best performance the MWCES can reach on the considered sources.

At this point, we employ the MWCES on each group of sources, testing all the possible combinations of K and L in the sets $\{1, 2, \ldots, K_{MAX}\}$ and $\{4, 5, \ldots, L_{MAX}\}$ respectively. For each combination, we take the sources chosen by the MWCES in every group and we compute the mean value `ase_s` of their K_{MAX}-bits ASEs, calculated in the previous phase. `ase_s` is compared with `ase_m` through the calculation of the relative error between them. If the error is below the error tolerance ERR_{MAX}, the actual (K, L) combination is considered valid.

All the valid (K, L) combinations must be compared to understand which is the one for which the MWCES would have the minimum complexity. To do so, we defined a cost function:

$$C(K, L) = 2^K \cdot L + 2 \cdot (K + L). \tag{5.59}$$

(5.59) refers to a raw implementation of the MWCES based on counting registers: in this case the selector would employ 2^K L-bits long registers (`symbol_count`) and

```
ALGORITHM A2:
 1. for(grp = 0; grp < M; grp++)
 2. {
 3.   max_ase[grp] = 0;
 4.   for(src = 0; src < N; src++)
 5.   {
 6.     for(samp = 0; samp < 2^(K_MAX+L_MAX); samp++)
 7.     {
 8.       seq[grp][src][samp] = get_symbol(grp,src,K_MAX);
 9.     }
10.     ase[grp][src] = compute_ase(grp,src,K_MAX);
11.     if(ase[grp][src] > max_ase[grp])
12.     {
13.       max_ase[grp] = ase[grp][src];
14.     }
15.   }
16. }
17. ase_m = mean(max_ase);
18. cost = 2^K_MAX*L_MAX+2*(K_MAX+L_MAX);
19. K = K_MAX;
20. L = L_MAX;
21. for(k = 1; k <= K_MAX; k++)
22. {
23.   for(l = 4; l <= L_MAX; l++)
24.   {
25.     for(grp = 0; grp < M; grp++)
26.     {
27.       src = execute_mes(k,l,grp);
28.       sel_ase[grp] = ase[grp][src];
29.     }
30.     ase_s = mean(sel_ase);
31.     mes_err = (ase_m-ase_s)/ase_m;
32.     if(mes_err < ERR_MAX)
33.     {
34.       new_cost = 2^k*l+2*(k+l);
35.       if(new_cost < cost)
36.       {
37.         cost = new_cost;
38.         K = k;
39.         L = l;
40.       }
41.       else if(new_cost == cost && k < K)
42.       {
43.         K = k;
44.         L = l;
45.       }
46.     }
47.   }
48. }
```

Figure 5.8: Proposed procedure to configure the parameters used in MWCES algorithm (Fig. 5.7).

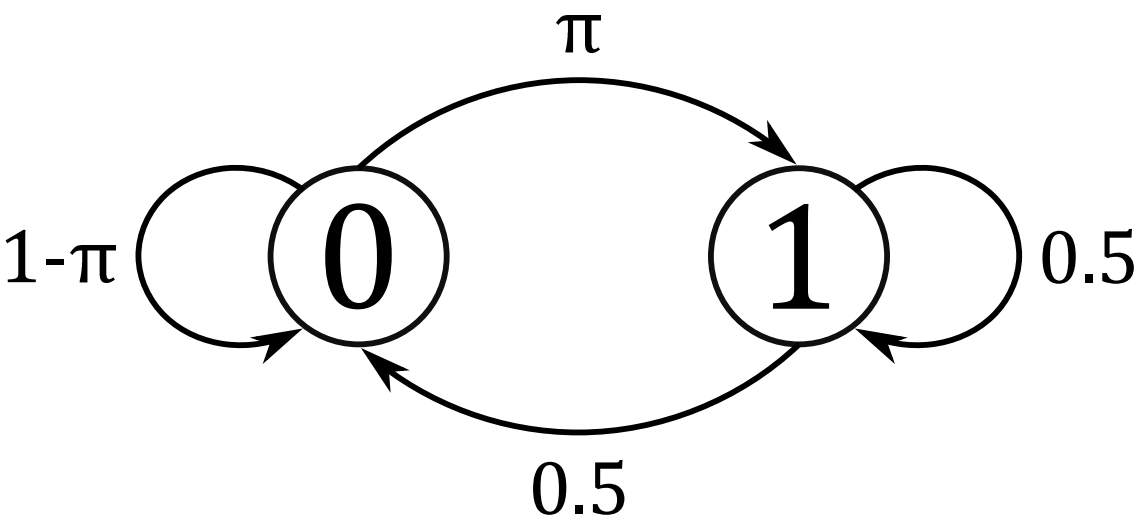

Figure 5.9: Two-states Markov chain for the generation of random bit sequences.

two $(K+L)$-bits long registers (`total_count` and `best_count`). The cost function corresponds to the total number of bits required by these registers, i.e. a first evaluation of the hardware resources required by the MWCES.

The valid combination (K, L) with the lowest cost is the one chosen for the considered application. If two combinations have the same cost, we choose the one with the lowest K, because a lower K implies a lower number of registers, which simplifies the overall MWCES architecture.

To better understand the presented procedure, an example based on Markov chains is given in the following section.

5.5 Example Case: Markov Chain-Based MWCES Test

To provide a demonstration of the procedure to be followed to configure and apply the Maximum Worst-Case Entropy Selector (MWCES) algorithm, in this section we present an example analysis performed on entropy sources based on Markov chains. We chose this kind of sources because they allow, through a proper definition of their transition matrix, to control the resulting ASE, as shown in the following.

Let's consider the two-states Markov chain in Fig. 5.9.

The chain transition matrix Π is parametrized dependently on the probability π to pass from state 0 to state 1:

$$\Pi = \begin{bmatrix} p_{00} & p_{10} \\ p_{01} & p_{11} \end{bmatrix} = \begin{bmatrix} 1-\pi & 0.5 \\ \pi & 0.5 \end{bmatrix}. \tag{5.60}$$

The steady-state probabilities to get a 0 or a 1 for a Markov chain of this kind can

be computed solving the following system:

$$\begin{bmatrix} 1-\pi & 0.5 \\ \pi & 0.5 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \end{bmatrix} = \begin{bmatrix} P_0 \\ P_1 \end{bmatrix}, \tag{5.61}$$

where P_i is the steady-state probability to get the symbol i.

Remembering that $P_0 + P_1 = 1$, we obtain that:

$$\begin{cases} P_0 = \frac{0.5}{\pi+0.5} \\ P_1 = \frac{\pi}{\pi+0.5} \end{cases}. \tag{5.62}$$

π is a probability, therefore it is a real number limited between 0 and 1. The obtained steady-state probabilities are continuous on this domain and according to the value of π it is possible to get both a deterministic source:

$$\pi = 0 \Rightarrow \begin{cases} P_0 = 1 \\ P_1 = 0 \end{cases}, \tag{5.63}$$

and a uniformly distributed source:

$$\pi = 0.5 \Rightarrow \begin{cases} P_0 = 0.5 \\ P_1 = 0.5 \end{cases}. \tag{5.64}$$

The Average Shannon Entropy is a combination of continuous functions in π, so it is again a continuous function:

$$ASE = -P_0 \log_2 P_0 - P_1 \log_2 P_1, \tag{5.65}$$

and since π can be tuned to get both a deterministic (ASE = 0) and a uniformly distributed (ASE = 1) sources, the ASE image is continuously defined between 0 and 1.

This result can be extended to K-bits long symbols. In this case, the probability to get a K-bits long symbol starting from an initial state $P(b_0)$, $b_0 \in \{0, 1\}$ is:

$$\begin{aligned} P(b_0, b_1, \ldots, b_{K-2}, b_{K-1}) &= P(b_0) p_{b_0 b_1} p_{b_1 b_2} \ldots p_{b_{K-2} b_{K-1}} = \\ &= P(b_0) p_{00}^{\#00} p_{01}^{\#01} p_{10}^{\#10} p_{11}^{\#11}, \end{aligned} \tag{5.66}$$

where $p_{b_{i-1} b_i}$ is the probability to get a bit b_i after a bit b_{i-1}, and $\#ij$ is the number of i to j bit transitions in the considered symbol.

Taking into account the proposed Markov chain, it can be observed that the obtained probabilities are again continuous in π and at the steady-state they are equal to:

$$P(b_0, b_1, \ldots, b_{K-2}, b_{K-1}) = P_{b_0} (1-\pi)^{\#00} \pi^{\#01} 0.5^{\#10+\#11}, \tag{5.67}$$

where P_{b_0} denotes the steady-state probabilities obtained in (5.62). Setting π equal to 0 and to 0.5 we obtain again a deterministic and a uniformly distributed sources respectively:

$$\pi = 0 \Rightarrow \begin{cases} P_{s_0} = 1 \\ P_{s_i} = 0, \quad i = 1, \ldots, 2^K - 1 \end{cases}, \tag{5.68}$$

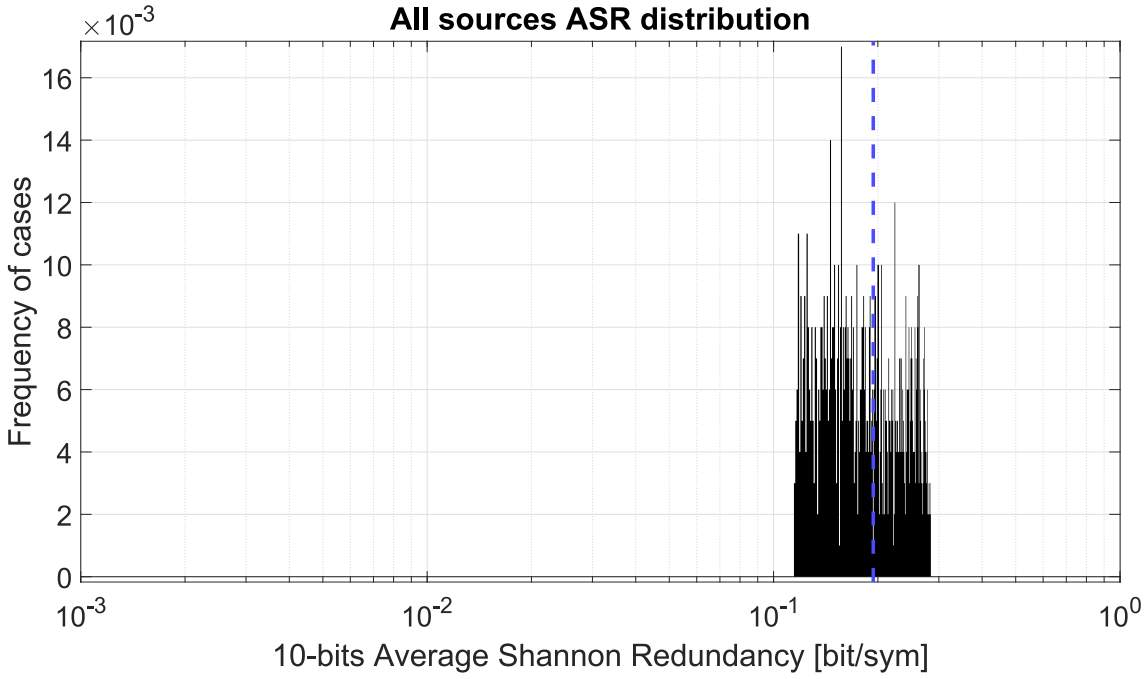

Figure 5.10: 10-bits ASR distribution of the whole set of Markov chain-based entropy sources. The blue dashed line indicates the mean value of the distribution.

$$\pi = 0.5 \Rightarrow P_{s_i} = \frac{1}{2^K}, \quad i = 0, \ldots, 2^K - 1, \tag{5.69}$$

therefore the ASE image is continuously defined between 0 and 1 also when we evaluate it for K-bits long symbols. The probabilities P_{s_i}, $i = 0, \ldots, 2^K - 1$ are the steady-state probabilities of the K-bits long symbols.

5.5.1 Experiments and Results

We simulated 1000 sources based on Markov chains, organized in $M = 100$ groups of $N = 10$ sources. The designed Markov chains are of the type shown in Fig. 5.9, with π randomly chosen on the interval $0.3 \pm 20\%$.

We fixed the maximum symbol bit length at $K_{MAX} = 10$ and the maximum threshold bit length at $L_{MAX} = 10$.

The resulting 10-bits Average Shannon Redundancies (ASRs), as shown in Fig. 5.10, are distributed on the interval $[0.115, 0.284]$ bit/sym, with a mean value equal to 0.194 bit/sym and a standard deviation equal to 0.048 bit/sym.

Looking only at the 10-bits ASE of the best sources for each group, shown in Fig. 5.11, we observe that an optimal selector would increase the performance narrowing the distribution on the interval $[0.115, 0.175]$ bit/sym, with a mean value equal to 0.128 bit/sym and a standard deviation equal to 0.012 bit/sym.

The error tolerance was set at $ERR_{MAX} = 1\%$. The (K, L) combination matching the error threshold condition that minimizes the cost function (5.59) is $(K, L) =$

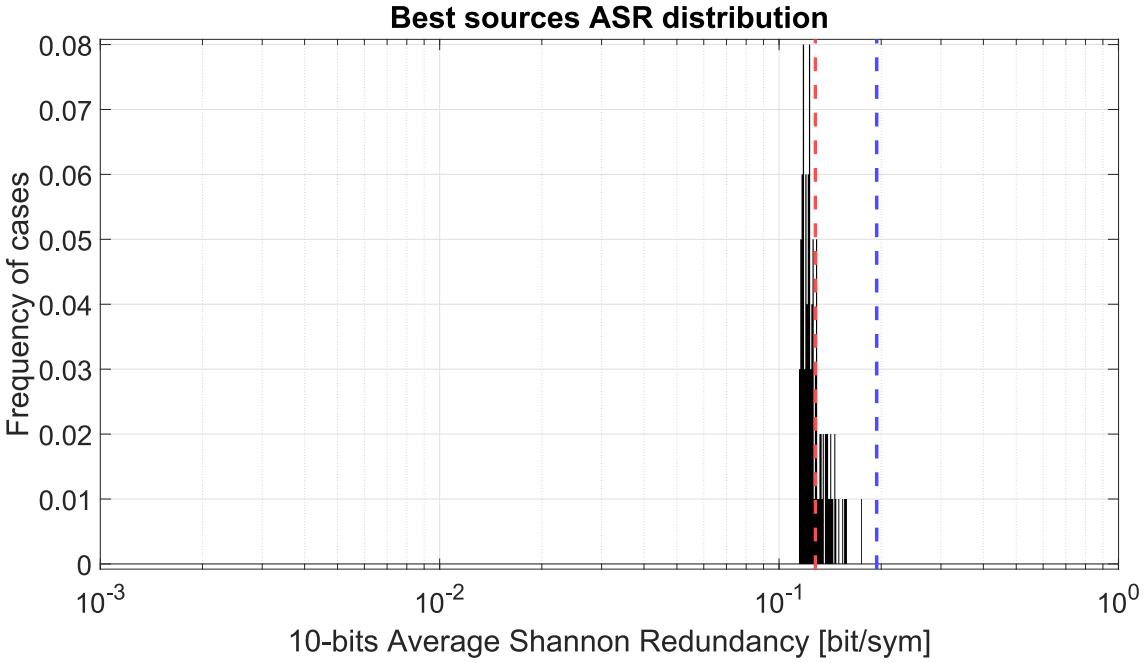

Figure 5.11: 10-bits ASR distribution of the best sources for each group of Markov chain-based entropy sources. The blue dashed line indicates the mean value of the whole set of sources, while the red dashed line corresponds to the mean value of the best sources for each group.

$(2, 9)$. The 10-bits ASR distribution of the sources selected by the MWCES configured with these parameters is shown in Fig. 5.12.

The ASE of the sources selected by the MWCES is distributed on the interval $[0.116, 0.208]$ bit/sym, with a mean value equal to 0.134 bit/sym and a standard deviation equal to 0.018 bit/sym. With respect to the optimal solution, we observe that the MWCES introduces an error on the mean value of the 0.7% with an increase of the standard deviation of 0.006 bit/sym.

5.6 Tunable Digital Nonlinear Oscillator

The test case of the MWCES based on Markov chains presented in Section 5.5 highlighted the possibility of using this estimation methodology to ensure a maximization, net of a controlled error tolerance, of the entropy extractable from a set of entropy sources, based solely on an approximate a priori knowledge of the statistical characteristics of the involved sources. In this section we show how this principle can be applied to Digital Nonlinear Oscillators, with the aim of mitigating the performance variability introduced by their physical implementation.

Let us consider again the DNO topology analyzed in Chapter 4, whose structure is shown again in Fig. 5.13.

In Section 4.5 we highlighted that, although it is capable of achieving particularly high performance, the circuit is affected by a variability in terms of generated

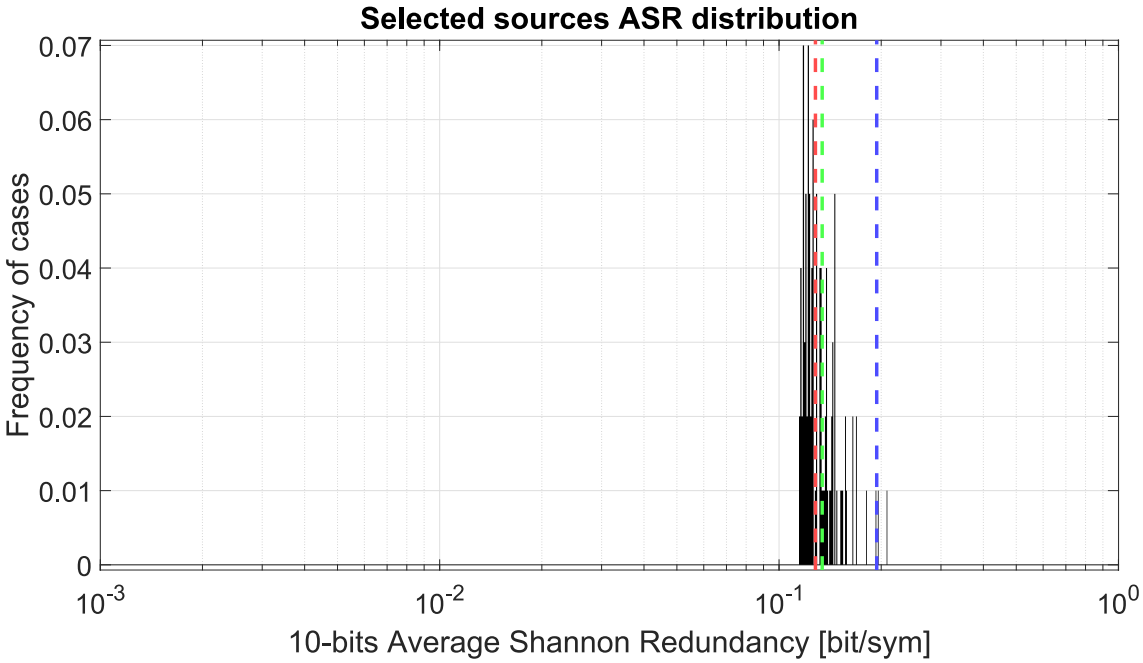

Figure 5.12: 10-bits ASR distribution of the sources selected by a MWCES configured with $K = 2$ and $L = 9$ for each group of Markov chain-based entropy sources. The blue dashed line indicates the mean value of the whole set of sources, the red dashed line corresponds to the mean value of the best sources for each group, the green dashed line is the mean value of the sources selected by the MWCES.

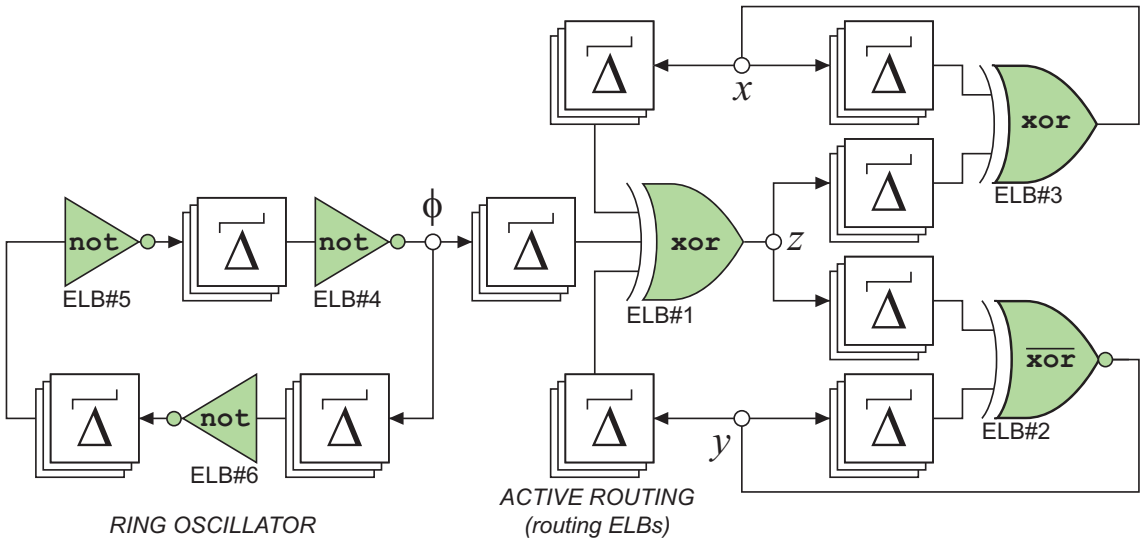

Figure 5.13: The complete topology studied in Chapter 4, in which a nonlinear oscillating structure (the nonlinear oscillator in Fig.4.8), is excited by a ring oscillator to produce complex dynamics.

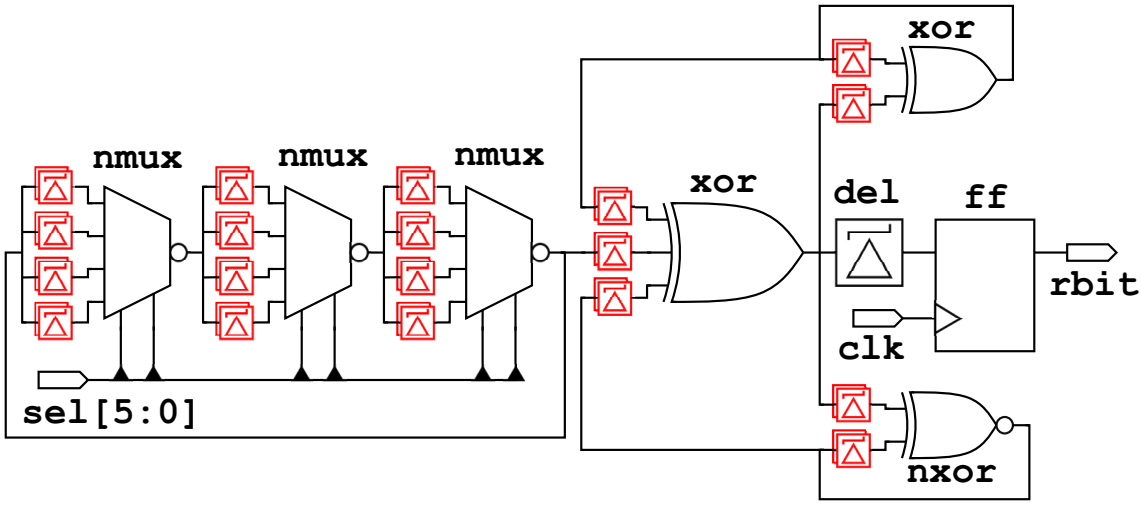

Figure 5.14: DNO topology inspired by the topology shown in Fig. 5.13, modified to be tunable through the MWCES. The red elements represent the active routing delay introduced by its hardware implementation. The `del` and `ff` blocks represent respectively a rectifying gate and a flip-flop, forming a 1-bit A/D converter.

entropy, caused by the differences in the hardware used to implement the resources that compose it and by the routing between the pins of the LUTs.

Although in the specific analyzed case an optimized solution was presented that reduces this variability, that is a 2:1 lossy compression based on the XORing of the output bits of two implementations of the DNO operating in parallel, this problem distinguishes any DNO.

In this context, the MWCES can be used as a generic solution, applicable to any topology simply by applying small changes to the configuration of some used LUTs, such as to make the routing of the circuit tunable, as described below for the circuit of Fig. 5.13.

Consider the DNO topology shown in Fig. 5.14.

The represented circuit is a modified version of the topology presented in Chapter 4. In fact, we can immediately recognize the three XOR and NXOR gates connected in loops. The main difference is the presence of three negated multiplexers (NMUXes) connected to form an independent loop.

From a logic point of view, this loop is equivalent to the three-stages Ring Oscillator, because the four inputs of each multiplexer are short circuited, forming a single logic path. From an analog point of view, however, each line is characterized by a different propagation time (represented by the red delay blocks). For this reason, the NMUXes loop can be seen as a frequency-programmable excitation signal generator, whose frequency can be chosen among a set of 64 different values, obtained changing the value of the multiplexers control signal `sel`.

According to our analyzes, we expect the DNO dynamics to change in dependence of the selected frequency. In this sense, the presented DNO topology describes a set of 64 different complex dynamical systems, each one employable as an entropy

source, which can be selected simply fixing the value of `sel`.

A structure of this kind constitutes a clear application for the MWCES: it is not possible to know a priori which is the best multiplexers configuration from an entropy generation point of view, since the propagation times associated to the DNO lines are different for each implementation; using the MWCES we expect to be able to increase on average the circuit entropy without having to worry about the specific characteristics of the implementation under consideration.

5.6.1 Experiments and Comments

To evaluate the performance of the MWCES in tuning the DNO proposed in the previous section, we designed the circuit to be implemented on Xilinx Artix 7 FPGAs.

As shown in Fig. 5.14, the proposed circuit is composed by seven logic gates and a flip-flop. Exploiting the attributes to be included in the RTL design already described in Section 3.5, we took precise control on the resources placement to force the FPGA synthesizer to implement the designed DNO in the most compact way, programming only LUTs belonging to a single CLB. In this way, we tried to maximize the repeatability between different implementations of the same circuit in different locations of the CLB matrix. Unfortunately, we already know that this operation does not allow to have control on the routing between the gates, since the ELBs are interconnected through the switch matrices, whose internal connections are not programmable. This lack of control, together with the parasitic effects introduced by the hardware, make it impossible to determine a priori the performance of the specific implementation of the circuit.

For this reason, we had to configure the MWCES in function of the Xilinx Artix 7 FPGA implementation of the DNO, following the procedure shown in Fig. 5.8.

We designed the circuit in five Xilinx Artix 7 xc7a35 FPGAs. In each device we implemented sixteen different instances of the DNO, selecting their locations to cover different areas of the matrix. All the implementations were sampled at a frequency of 400 MHz. Referring to the MWCES procedure and remembering the DNO topology is controlled by a 6-bits selection line (sel), with this configuration we can analyze $M = 80$ groups of $N = 2^6 = 64$ sources.

We fixed the maximum symbol bit length at $K_{MAX} = 10$ and the maximum threshold bit length at $L_{MAX} = 9$.

The resulting 10-bits ASRs, as shown in Fig. 5.15, are distributed on the interval $[0.020, 0.893]$ bit/sym, with a mean value equal to 0.083 bit/sym and a standard deviation equal to 0.122 bit/sym.

The figure clearly shows that, even if on average the DNO implementation is able to reach very high entropy values, a wrong configuration of the multiplexers can lead to a marked deterioration in the performance. In fact, if we consider only the best multiplexers setup for each implementation, the 10-bits ASE distribution shrinks significantly, as shown in Fig. 5.16.

An optimal configuration of the multiplexers provides a 10-bits ASE distribution

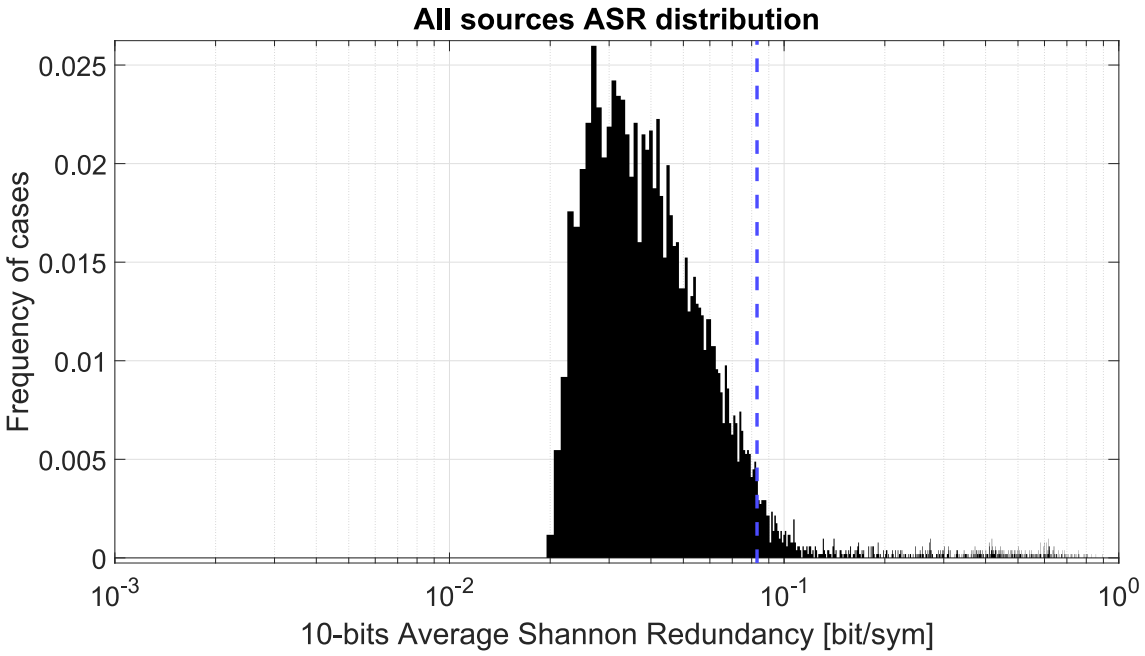

Figure 5.15: 10-bits ASR distribution of the whole set of DNOs, taking into account any possible configuration of the multiplexers. The blue dashed line indicates the mean value of the distribution.

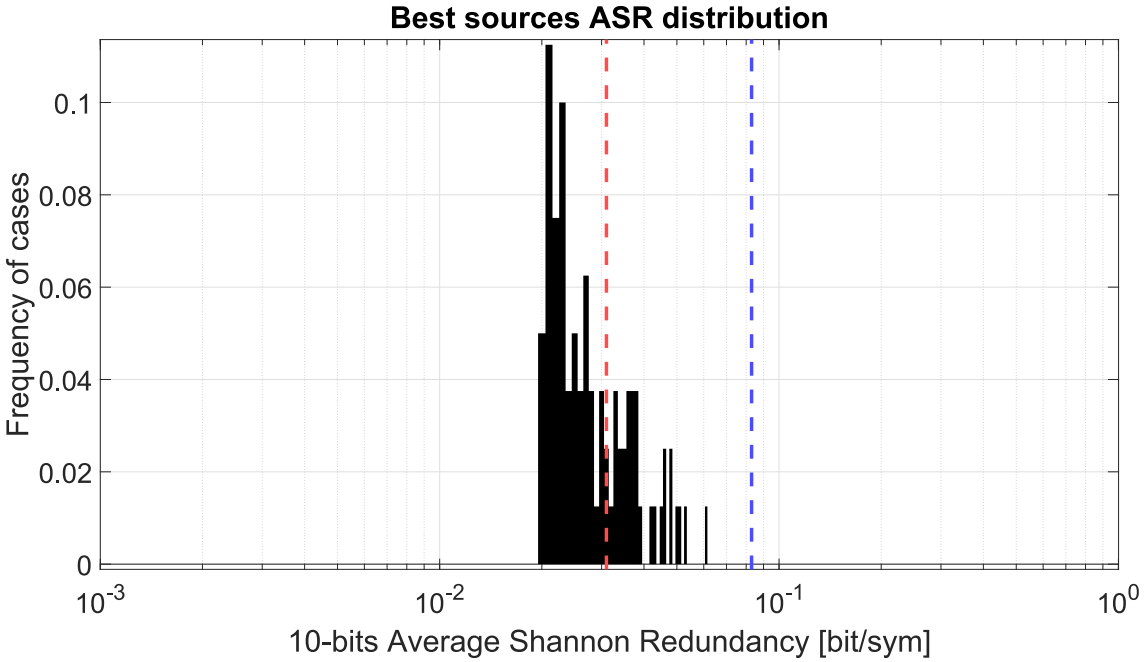

Figure 5.16: 10-bits ASR distribution of the DNOs, taking into account the best configurations of the multiplexers. The blue dashed line indicates the mean value of the whole set of sources, while the red dashed line corresponds to the mean value of the best configuration for each implementation.

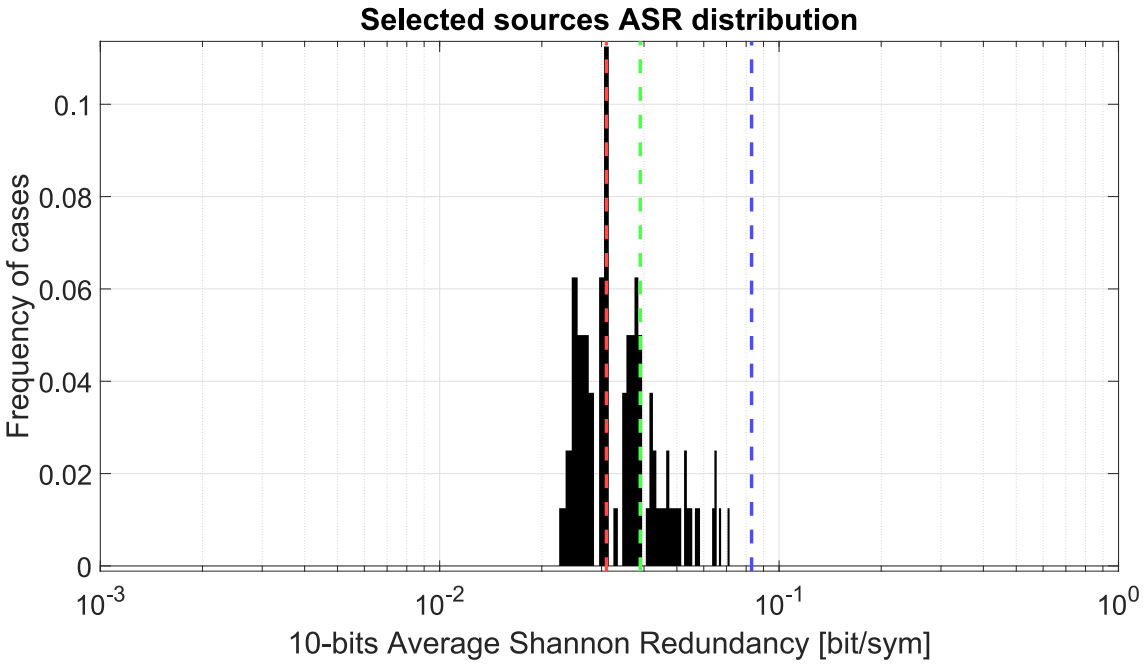

Figure 5.17: 10-bits ASR distribution of the sources selected by a MWCES configured with $K = 3$ and $L = 8$ for each DNO implementation. The blue dashed line indicates the mean value of the whole set of sources, the red dashed line corresponds to the mean value of the best multiplexer configuration for each instance, the green dashed line is the mean value of the multiplexer configurations selected by the MWCES.

defined on the interval $[0.020, 0.061]$ bit/sym, with a mean value equal to 0.031 bit/sym and a standard deviation equal to 0.009 bit/sym. From this result we can expect therefore that, with a proper selection of the multiplexers inputs, even a suboptimal setup could keep the DNO performance high.

To configure the MWCES, we set an error tolerance $ERR_{MAX} = 1\%$. The (K, L) combination matching the error threshold condition that minimizes the cost function (5.59) is $(K, L) = (3, 8)$. The 10-bits ASE distribution of the sources selected by the MWCES configured with these parameters is shown in Fig. 5.17.

The ASE of the sources selected by the MWCES is distributed on the interval $[0.024, 0.072]$ bit/sym, with a mean value equal to 0.038 bit/sym and a standard deviation equal to 0.012 bit/sym. With respect to the optimal solution, we observe that the MWCES introduces an error on the mean value of the 0.8% with an increase of the standard deviation of 0.003 bit/sym.

This result represents a suboptimal solution, since the ASE that we can reach with a configuration of this kind is slightly worse with respect to the maximum one. However, if we compare it with the ASE distribution of the whole set of DNOs, taking into account any possible configuration of the multiplexers, the performance gain we get is evident.

5.7 MWCES Hardware Implementation

To conclude the analyzes related to the MWCES algorithm, we decided to evaluate the hardware resources consumption associated with an implementation on FPGA.

Here we report the VHDL description of the MWCES, assuming to use it to compare 16 different sources.

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.NUMERIC_STD.ALL;

entity MaximumEntropySelector is
  generic (
    K : natural;
    L : natural );
  port (
    clk : in std_logic;
    en : in std_logic;
    sources : in std_logic_vector(15 downto 0);
    bestsource : out std_logic_vector(3 downto 0);
    done : out std_logic );
end MaximumEntropySelector;

architecture Behavioral of MaximumEntropySelector is
  type t_symcnt is array(integer range 2**K-1 downto 0) of natural range 0 to 2**L-1;
  signal symbol : std_logic_vector(K-1 downto 0);
  signal totcnt : natural range 0 to 2**(K+L)-1;
  signal symcnt : t_symcnt;
  signal bestcnt : natural range 0 to 2**(K+L)-1;
  signal bestpartialsrc : natural range 0 to 15;
begin
  MWCES: process(clk)
    type t_state is (BUILDSYM,COUNTSYM,CHECKTIME,UPDATESRC,STOP);
    variable state : t_state;
    variable srcsel : natural range 0 to 15;
    variable bitcnt : natural range 0 to K-1;
    variable symindex : natural range 0 to 2**K-1;
  begin
    if rising_edge(clk) then
      if en = '0' then
        state := BUILDSYM;
        srcsel := 0;
        bitcnt := 0;
        symcnt <= (others => 0);
        totcnt <= 0;
        bestcnt <= 0;
        bestsource <= (others => '0');
        done <= '0';
      else
        case state is
          when BUILDSYM =>
            symbol(K-1 downto 1) <= symbol(K-2 downto 0);
            symbol(0) <= sources(srcsel);
            if bitcnt = K-1 then
              bitcnt := 0;
              state := COUNTSYM;
            else
              bitcnt := bitcnt+1;
            end if;
          when COUNTSYM =>
            symindex := to_integer(unsigned(symbol));
            totcnt <= totcnt+1;
            if symcnt(symindex) = 2**L-1 then
              symcnt <= (others => 0);
```

```
            state := CHECKTIME;
          else
            symcnt(symindex) <= symcnt(symindex)+1;
            state := BUILDSYM;
          end if;
        when CHECKTIME =>
          if totcnt >= bestcnt then
            bestcnt <= totcnt;
            bestpartialsrc <= srcsel;
          end if;
          totcnt <= 0;
          state := UPDATESRC;
        when UPDATESRC =>
          if srcsel = 15 then
            srcsel := 0;
            bestsource <= std_logic_vector(to_unsigned(bestpartialsrc,4));
            state := STOP;
          else
            srcsel := srcsel+1;
            state := BUILDSYM;
            end if;
        when STOP =>
          done <= '1';
      end case;
    end if;
  end if;
 end process MWCES;
end Behavioral;
```

We can consider this specific case without losing generality, as the number of compared sources only affects the size of the ports `sources` and `bestsource`, indicating the sources output signals and the selected source respectively, of the signal `bestpartialsource`, that stores the best source number along the algorithm computations, and of the variable `srcsel`, that selects the source to be analyzed. A different number of analyzed sources would therefore determine a small variation in the LUTs and flip-flops required for the implementation with respect to the hardware resources that are dependent on the choice of K and L.

We used the above VHDL description of the MWCES to implement it on a Xilinx Artix 7 xc7a65 FPGA using the Xilinx Vivado Design Suite. We synthesized the VHDL design to evaluate the hardware resources consumption required by the circuit for different values of K and L. Tables 5.2 and 5.3 report the number of LUTs and flip-flops required by the the circuit for $K = 2, 3, \ldots, 8$ and for $L = 4, 5, \ldots, 10$.

If we compare the flip-flop numbers indicated in Table 5.3 with the minimum theoretical flip-flop consumption defined by the cost function (5.59), reported in Table 5.4, we observe that the FPGA implementation is coherent with the theoretical expectations, since the flip-flops number grows linearly in function of L for fixed K and exponentially in function of K for fixed L.

The flip-flops number required according to the Vivado design is higher with respect to what indicated in Table 5.4 because the cost function takes into account only the register required to store the symbols occurrences and to count the estimation time, while in the implementation we have some additional logic.

Furthermore, the increment of flip-flops for increasing values of K and L is slightly irregular, especially for high values of K and L. This is because of the synthesizer strategies, that was left to freely operate according to a balanced area/time

LUT

		L						
		4	5	6	7	8	9	10
	2	35	38	42	50	47	50	53
	3	49	55	59	64	70	69	71
	4	70	74	81	97	102	107	105
K	5	118	130	143	145	180	177	190
	6	231	216	231	267	307	330	312
	7	338	370	408	448	482	519	572
	8	630	704	775	856	923	1000	1206

Table 5.2: Number of LUTs required to synthesize the MWCES algorithm in Xilinx Vivado for $K = 2, 3, \ldots, 8$ and for $L = 4, 5, \ldots, 10$

.

FF

		L						
		4	5	6	7	8	9	10
	2	49	55	61	67	73	79	85
	3	69	79	89	99	109	119	129
	4	104	122	140	158	176	194	212
K	5	172	206	240	274	308	342	376
	6	303	369	435	501	569	635	699
	7	562	694	824	956	1087	1217	1347
	8	1084	1344	1602	1861	2124	2379	2641

Table 5.3: Number of flip-flops required to synthesize the MWCES algorithm in Xilinx Vivado for $K = 2, 3, \ldots, 8$ and for $L = 4, 5, \ldots, 10$.

.

$\boldsymbol{C(K, L)}$

		L						
		4	5	6	7	8	9	10
	2	28	34	40	46	52	58	64
	3	46	56	66	76	86	96	106
	4	80	98	116	134	152	170	188
K	5	146	180	214	248	282	316	350
	6	276	342	408	474	540	606	672
	7	534	664	794	924	1054	1184	1314
	8	1048	1306	1564	1822	2080	2338	2596

Table 5.4: Theoretical minimum number of flip-flops required to implement the MWCES algorithm according to the cost function (5.59) for $K = 2, 3, \ldots, 8$ and for $L = 4, 5, \ldots, 10$.

.

design optimization strategy. In any case, the overall resource consumption is in agreement with the expected theoretical one.

5.8 Conclusion

We presented an algorithm, called Maximum Worst-Case Entropy Selector (MWCES), that aims to identify, within a set of entropy sources, which offers the best performance in terms of worst-case entropy, also known in literature as "min-entropy". The algorithm aims to assess the minimum entropy of a source, estimating what is the probability of the most likely symbol that can be generated by the source.

We rigorously investigated what are the levels of uncertainty associated with an assessment of the entropy of an information source on the basis of an estimate of this type. Taking into account the properties of the Shannon entropy, we studied which are the worst and best entropies of a source with a maximum symbol generation probability p_H. We also studied, from a statistical point of view, which is the minimum number of observations required by an estimator to properly estimate this probability.

On the basis of these studies, we established that an estimator of the maximum symbol generation probability of a sources has two main limitations:

- the range between the worst and best entropies given by p_H is a not constant function of p_H;
- the optimal number of observations required by the estimator cannot be computed because of the complexity of the required calculation.

For this reason, we established a better use of the estimator for the assessment of the source with the maximum entropy among a set, observing that:

- the worst and best entropies are monotonically decreasing function of p_H, therefore a source with lower p_H with respect to another is more likely to have higher entropy:
- it is possible to find heuristically a number of observations such to identify which is the source in a set with the lowest p_H, regardless of its value.

Therefore, we defined a procedure for selecting the maximum worst-case entropy source starting from a set of sources. This procedure counts the occurrences of the symbols generated by a source and takes note of the time required by them to reach an overflow value. The slowest source is the one with the lowest p_H.

We proposed also a procedure for the correct choice of the selector parameter. This probabilistic procedure adopts a statistical approach to establish which is the minimum symbols bit length and the minimum overflow level for which the selection error on the output Shannon entropy is lower than a certain threshold.

The selection algorithm was then applied in two notable examples, one based on a set of sources defined starting from Markov chains, and the other based on a modified version of the DNO presented in Chapter 4.

In the examples, we applied the selector configuration procedure to propose an implementation of the selector with minimum hardware resources consumption and with a selection relative error on the output Shannon entropy lower than 1%.

Both cases showed that, downstream of a correct configuration, the selector is able to maximize the output Shannon entropy of the set, net of a predefined relative error threshold.

Finally, we proposed a design of the selector for its implementation in FPGAs.

Chapter 6

Conclusion

In this thesis we introduced a new class of circuits that can be used as entropy sources for True Random Number Generators, called Digital Nonlinear Oscillators (DNOs), which constitute dynamical systems capable of supporting complex dynamics (periodic or chaotic) in the analog time-continuous domain, although they are made of digital circuits.

The objective of this study was to demonstrate that circuits of this type can define high-performance entropy sources suitable for the design of True Random Number Generators on purely digital devices with limited resources, suitable for lightweight cryptographic applications.

For this purpose, we initially showed through notable examples how different circuits belonging to the DNO class can be characterized by different performance.

Subsequently, we introduced a set of tools that allow to analyze and design DNOs. More in detail:

- two figures of merit, namely the Decorrelation Time and the Average Shannon Entropy, for the comparative evaluation of the statistical characteristics of DNOs were defined. This figures of merit provide additional information with respectc to standard statistical tests, which are limited to providing saturated binary outcome;

- a simplified theoretical investigation approach based on low-complexity dynamical models, whose purpose is to investigate which conditions favor compatibility with complex dynamics for a DNO starting from the analysis of the stability of its fixed points, was proposed;

- an advanced numerical simulation setup in Cadence Virtuoso based on UMC 180 nm technology that deepens the results given by the study of the simplified dynamical models was designed;

- design methodology for the implementation of a DNO on FPGAs, aimed at taking control of the synthesizer place and route policies for the implementation of precise layouts were discussed.

The introduced tools were used to investigate a novel DNO topology inspired by the theory of forced nonlinear oscillators, characterized by chaotic dynamics. We showed, through theoretical studies, numerical simulations and experiments, that this topology achieves high performance, outperforming the most relevant recent results proposed in literature both in terms of generated entropy and hardware resources consumption.

We studied this topology extensively, using simplified dynamical models and in-depth numerical simulations to establish under which conditions the circuit exhibits chaotic behaviors, and testing experimentally its dynamical behavior through FPGA implementations.

Specifically, the FPGA implementations allowed to evaluate the performance in terms of generated entropy, the influence of routing on the considered performance, the temperature sensitivity, to inspect physical output signals for a comparison with the results obtained in simulation.

The proposed DNO was also able to pass the standard NIST 800.22 statistical tests, requiring only a minimum post-processing, such that the complete system (DNO and post-processing) presents a negligible complexity compared to the solutions currently proposed in literature.

Finally, we presented an algorithm capable of identifying within a set of entropy sources which one offers the maximum entropy, called Maximum Entropy Selector. The algorithm aims to assess the worst-case entropy of a source, estimating what is the probability of the most likely symbol that can be generated by the source.

We rigorously investigated what are the levels of uncertainty associated with an assessment of the entropy of an information source on the basis of an estimate of this type. We also studied, from a statistical point of view, which is the minimum number of observations required by an estimator to properly estimate this probability.

On the basis of these investigations, we defined a procedure for selecting the maximum entropy source starting from a set of sources. We provided also a procedure for the correct choice of the selector parameter.

The selection algorithm was then applied in two notable examples, one based on a set of sources defined starting from Markov chains, and the other based on a modified version of the high-performance DNO.

Finally, we proposed a design of the selector for its implementation in FPGAs.

On the basis of the presented results, it is possible to conclude that DNOs represent a class of circuits that can be used for the design of high-performance True Random Number Generators based on purely digital hardware, opening new perspectives in the field of lightweight cryptography regarding the integration of TRNGs compliant to cryptographic security standards even in devices with limited resources.

Bibliography

[1] S. Vaudenay, *A classical introduction to cryptography: Applications for communications security*. Springer Science & Business Media, 2006.

[2] J. Katz and Y. Lindell, *Introduction to modern cryptography*. CRC press, 2014.

[3] D. R. Stinson and M. Paterson, *Cryptography: theory and practice*. CRC press, 2018.

[4] L. E. Bassham III, A. L. Rukhin, J. Soto, J. R. Nechvatal, M. E. Smid, E. B. Barker, S. D. Leigh, M. Levenson, M. Vangel, D. L. Banks *et al.*, *Sp 800-22 rev. 1a. a statistical test suite for random and pseudorandom number generators for cryptographic applications*. National Institute of Standards & Technology, 2010.

[5] A. J. Acosta, T. Addabbo, and E. Tena-Sánchez, "Embedded electronic circuits for cryptography, hardware security and true random number generation: an overview," *International Journal of Circuit Theory and Applications*, vol. 45, no. 2, pp. 145–169, 2017.

[6] M. Delgado Restituto and Á. B. Rodríguez Vázquez, "Integrated chaos generators," in *Proceedings of the IEEE*. Institute of Electrical and Electronics Engineers, 2002.

[7] M. P. Li, *Jitter, noise, and signal integrity at high-speed*. Pearson Education, 2007.

[8] T. Yamazaki and A. Uchida, "Performance of random number generators using noise-based superluminescent diode and chaos-based semiconductor lasers," *IEEE Journal of Selected Topics in Quantum Electronics*, vol. 19, no. 4, pp. 0 600 309–0 600 309, 2013.

[9] T.-K. Kuan, Y.-H. Chiang, and S.-I. Liu, "A 0.43 pj/bit true random number generator," in *2014 IEEE Asian Solid-State Circuits Conference (A-SSCC)*. IEEE, 2014, pp. 33–36.

[10] S. Yasuda, H. Satake, T. Tanamoto, R. Ohba, K. Uchida, and S. Fujita, "Physical random number generator based on mos structure after soft breakdown," *IEEE journal of solid-state circuits*, vol. 39, no. 8, pp. 1375–1377, 2004.

[11] J. Holleman, S. Bridges, B. P. Otis, and C. Diorio, "A 3 μw cmos true random number generator with adaptive floating-gate offset cancellation," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 5, pp. 1324–1336, 2008.

[12] M. Perić, P. Milićević, Z. Banjac, V. Orlić, and S. Milićević, "High speed random number generator for section key generation in encryption devices," in *2013 21st Telecommunications Forum Telfor (TELFOR)*. IEEE, 2013, pp. 117–120.

[13] C. De Roover and M. Steyaert, "A 500 mv 650 pw random number generator in 130 nm cmos for a uwb localization system," in *2010 Proceedings of ESSCIRC*. IEEE, 2010, pp. 278–281.

[14] H. Zhun and C. Hongyi, "A truly random number generator based on thermal noise," in *ASICON 2001. 2001 4th International Conference on ASIC Proceedings (Cat. No. 01TH8549)*. IEEE, 2001, pp. 862–864.

[15] A. Khanmohammadi, R. Enne, M. Hofbauer, and H. Zimmermanna, "A monolithic silicon quantum random number generator based on measurement of photon detection time," *IEEE Photonics Journal*, vol. 7, no. 5, pp. 1–13, 2015.

[16] H. Hata and S. Ichikawa, "Fpga implementation of metastability-based true random number generator," *IEICE TRANSACTIONS on Information and Systems*, vol. 95, no. 2, pp. 426–436, 2012.

[17] J.-L. Danger, S. Guilley, and P. Hoogvorst, "High speed true random number generator based on open loop structures in fpgas," *Microelectronics journal*, vol. 40, no. 11, pp. 1650–1656, 2009.

[18] B. Sunar, W. J. Martin, and D. R. Stinson, "A provably secure true random number generator with built-in tolerance to active attacks," *IEEE Transactions on computers*, vol. 56, no. 1, pp. 109–119, 2006.

[19] M. Raitza, M. Vogt, C. Hochberger, and T. Pionteck, "Raw 2014: Random number generators on fpgas," *ACM Transactions on Reconfigurable Technology and Systems (TRETS)*, vol. 9, no. 2, pp. 1–21, 2015.

[20] K. H. Tsoi, K. Leung, and P. H. W. Leong, "Compact fpga-based true and pseudo random number generators," in *11th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, 2003. FCCM 2003.* IEEE, 2003, pp. 51–61.

[21] K. Wold and C. H. Tan, "Analysis and enhancement of random number generator in fpga based on oscillator rings," in *2008 International Conference on Reconfigurable Computing and FPGAs*. IEEE, 2008, pp. 385–390.

[22] Ü. Güler, S. Ergün, and G. Dündar, "A digital ic random number generator with logic gates only," in *2010 17th IEEE International Conference on Electronics, Circuits and Systems*. IEEE, 2010, pp. 239–242.

[23] T. Addabbo, M. Alioto, A. Fort, S. Rocchi, and V. Vignoli, "The digital tent map: Performance analysis and optimized design as a low-complexity source of pseudorandom bits," *IEEE Transactions on Instrumentation and Measurement*, vol. 55, no. 5, pp. 1451–1458, 2006.

[24] M. Dichtl and J. D. Golić, "High-speed true random number generation with logic gates only," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2007, pp. 45–62.

[25] H. Martin, T. Korak, E. San Millán, and M. Hutter, "Fault attacks on strngs: Impact of glitches, temperature, and underpowering on randomness," *IEEE transactions on information forensics and security*, vol. 10, no. 2, pp. 266–277, 2014.

[26] M. Baudet, D. Lubicz, J. Micolod, and A. Tassiaux, "On the security of oscillator-based random number generators," *Journal of cryptology*, vol. 24, no. 2, pp. 398–425, 2011.

[27] T. Amaki, M. Hashimoto, Y. Mitsuyama, and T. Onoye, "A worst-case-aware design methodology for noise-tolerant oscillator-based true random number generator with stochastic behavior modeling," *IEEE transactions on information forensics and security*, vol. 8, no. 8, pp. 1331–1342, 2013.

[28] J. Yang, Y. Ma, T. Chen, J. Lin, and J. Jing, "Extracting more entropy for trngs based on coherent sampling," in *International Conference on Security and Privacy in Communication Systems*. Springer, 2016, pp. 694–709.

[29] T. Addabbo, A. Fort, M. Mugnaini, V. Vignoli, and M. Garcia-Bosque, "Digital nonlinear oscillators in plds: Pitfalls and open perspectives for a novel class of true random number generators," in *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2018, pp. 1–5.

[30] Xilinx, *7 Series FPGAs CLB User Guide - UG474 (v1.8)*, Sep. 2016.

[31] I. Vasyltsov, E. Hambardzumyan, Y.-S. Kim, and B. Karpinskyy, "Fast digital trng based on metastable ring oscillator," in *International Workshop on Cryptographic Hardware and Embedded Systems*. Springer, 2008, pp. 164–180.

[32] V. Fischer, F. Bernard, N. Bochard, and M. Varchola, "Enhancing security of ring oscillator-based trng implemented in fpga," in *2008 International Conference on Field Programmable Logic and Applications*. IEEE, 2008, pp. 245–250.

[33] R. Sivaraman, S. Rajagopalan, J. B. B. Rayappan, and R. Amirtharajan, "Ring oscillator as confusion–diffusion agent: a complete trng drove image security," *IET Image Processing*, vol. 14, no. 13, pp. 2987–2997, 2020.

[34] M. A. Prada-Delgado, C. Martínez-Gómez, and I. Baturone, "Auto-calibrated ring oscillator trng based on jitter accumulation," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2020, pp. 1–4.

[35] N. Bochard, F. Bernard, V. Fischer, and B. Valtchanov, "True-randomness and pseudo-randomness in ring oscillator-based true random number generators," *International Journal of Reconfigurable Computing*, vol. 2010, 2010.

[36] J. D. Golic, "New methods for digital generation and postprocessing of random data," *IEEE transactions on computers*, vol. 55, no. 10, pp. 1217–1229, 2006.

[37] T. Addabbo, A. Fort, R. Moretti, M. Mugnaini, V. Vignoli, and M. G. Bosque, "Lightweight true random bit generators in plds: Figures of merit and performance comparison," in *2019 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2019, pp. 1–5.

[38] T. Addabbo, A. Fort, R. Moretti, M. Mugnaini, and V. Vignoli, "Analysis of a circuit primitive for the reliable design of digital nonlinear oscillators," in *2019 15th Conference on Ph. D Research in Microelectronics and Electronics (PRIME)*. IEEE, 2019, pp. 189–192.

[39] F. Pareschi, R. Rovatti, and G. Setti, "On statistical tests for randomness included in the nist sp800-22 test suite and based on the binomial distribution," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 2, pp. 491–505, 2012.

[40] Xilinx, *Vivado Design Suite Properties Reference Guide - UG912 (v2020.1)*, Jul. 2020.

[41] T. Addabbo, A. Fort, R. Moretti, M. Mugnaini, H. Takaloo, and V. Vignoli, "Chaos in fully digital circuits: A novel approach to the design of entropy sources," in *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2020, pp. 1–5.

[42] ——, "A new class of chaotic sources in programmable logic devices," in *2020 IEEE International Workshop on Metrology for Industry 4.0 & IoT*. IEEE, 2020, pp. 6–10.

[43] ——, "A new class of digital circuits for the design of entropy sources in programmable logic," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 7, pp. 2419–2430, 2020.

[44] P. Bak, T. Bohr, and M. H. Jensen, "Mode-locking and the transition to chaos in dissipative systems," *Physica Scripta*, vol. 1985, no. T9, p. 50, 1985.

[45] L. Cveticanin, "Forced pure nonlinear symmetrical oscillators," *Mathematical and Computer Modelling*, vol. 55, no. 3-4, pp. 1580–1593, 2012.

[46] C. Miwadinou, A. Monwanou, J. Yovogan, L. Hinvi, P. N. Tuwa, and J. C. Orou, "Modeling nonlinear dissipative chemical dynamics by a forced modified van der pol-duffing oscillator with asymmetric potential: chaotic behaviors predictions," *Chinese journal of physics*, vol. 56, no. 3, pp. 1089–1104, 2018.

[47] C. Ainamon, C. Miwadinou, A. Monwanou, and J. C. Orou, "Analysis of multiresonance and chaotic behavior of the polarization in materials modeled by a duffing equation with multifrequency excitations," *Applied Physics Research*, vol. 6, no. 6, p. 74, 2014.

[48] I. Bashkirtseva and L. Ryashko, "Sensitivity and chaos control for the forced nonlinear oscillations," *Chaos, Solitons & Fractals*, vol. 26, no. 5, pp. 1437–1451, 2005.

[49] E. Ott, *Chaos in dynamical systems*. Cambridge university press, 2002.

[50] J. K. Hale and H. Koçak, *Dynamics and bifurcations*. Springer Science & Business Media, 2012, vol. 3.

[51] N. Paydavosi, T. H. Morshed, D. D. Lu, W. M. Yang, M. V. Dunga, X. J. Xi, J. He, W. Liu, M. C. Kanyu, X. Jin *et al.*, "Bsim4v4. 8.0 mosfet model," *University of California, Berkeley (CA)*, 2013.

[52] E. A. Gutiérrez-D, *Nano-scaled semiconductor devices: physics, modelling, characterisation, and societal impact*. The Institution of Engineering and Technology, 2016.

[53] M. Grundmann, *Physics of semiconductors*. Springer, 2010, vol. 11.

[54] S. M. Sze, *Semiconductor devices: physics and technology*. John wiley & sons, 2008.

[55] T. Addabbo, M. Alioto, A. Fort, S. Rocchi, and V. Vignoli, "A variability-tolerant feedback technique for throughput maximization of trbgs with predefined entropy," *Journal of Circuits, Systems, and Computers*, vol. 19, no. 04, pp. 879–895, 2010.

[56] M. S. Turan, E. Barker, J. Kelsey, K. A. McKay, M. L. Baish, and M. Boyle, "Recommendation for the entropy sources used for random bit generation," *NIST Special Publication*, vol. 800, no. 90B, 2018.

[57] X. Yang and R. C. Cheung, "A complementary architecture for high-speed true random number generator," in *2014 International Conference on Field-Programmable Technology (FPT)*. IEEE, 2014, pp. 248–251.

[58] T. Addabbo, A. Fort, D. Papini, S. Rocchi, and V. Vignoli, "Invariant measures of tunable chaotic sources: Robustness analysis and efficient estimation," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 56, no. 4, pp. 806–819, 2008.

[59] N. N. Anandakumar, S. K. Sanadhya, and M. S. Hashmi, "Fpga-based true random number generation using programmable delays in oscillator-rings," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 67, no. 3, pp. 570–574, 2019.

[60] G. D. P. Stanchieri, A. De Marcellis, E. Palange, and M. Faccio, "A true random number generator architecture based on a reduced number of fpga primitives," *AEU-International Journal of Electronics and Communications*, vol. 105, pp. 15–23, 2019.

[61] M. Tuna, A. Karthikeyan, K. Rajagopal, M. Alcin, and İ. Koyuncu, "Hyperjerk multiscroll oscillators with megastability: analysis, fpga implementation and a novel ann-ring-based true random number generator," *AEU-International Journal of Electronics and Communications*, vol. 112, p. 152941, 2019.

[62] X. Wu and S. Li, "A new digital true random number generator based on delay chain feedback loop," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 2017, pp. 1–4.

[63] P. Z. Wieczorek, "An fpga implementation of the resolve time-based true random number generator with quality control," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 61, no. 12, pp. 3450–3459, 2014.

[64] S. Tao, Y. Yu, and E. Dubrova, "Fpga based true random number generators using non-linear feedback ring oscillators," in *2018 16th IEEE International New Circuits and Systems Conference (NEWCAS)*. IEEE, 2018, pp. 213–216.

[65] C. E. Shannon, "A mathematical theory of communication," *The Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.