True Random Number Generator Implemented in 130 nm CMOS Nanotechnology

Pedro Monteiro, Luís Oliveira
CTS-UNINOVA
FCT-NOVA
Monte da Caparica, Portugal
pag.monteiro@campus.fct.unl.pt

João Casaleiro
CTS-UNINOVA
DEETC, ISEL
Lisboa, Portugal
joao.casaleiro@isel.pt

2022 International Young Engineers Forum (YEF-ECE) | 978-1-6654-6731-5/22/$31.00 ©2022 IEEE | DOI: 10.1109/YEF-ECE55092.2022.9849994

***Abstract*—Random generators systems have the capacity to generate cryptographic keys which, when mixed with the information, hide it in an efficient and timely manner. There are two categories of RNG, being truly random (TRNG) or pseudorandom (PRNG). To study the entropy source based on the noise of an oscillator, and to achieve that, an RNG circuit was designed to have a low power consumption, a high randomness and a low cost and area usage. The chosen architecture for this paper is a hybrid RNG, which uses oscillators and a chaotic circuit to generate the random bits. With the simulation of the circuit, it was found to be at the objectives mark, having a low power consumption of 1.19 mW, a high throughput of 25 Mbit/s and an energy per bit of 47.6 pJ/bit. However, due to limitations with the simulation, it wasn't possible to run all the statistical tests, although all the ran tests were passed.**

***Index Terms*—PRNG, TRNG, oscillator, chaotic, random, cryptographic, information.**

I. Introduction

Since the beginning of civilization, ways of communicating between two or more interlocutors have been developed, which raised a necessity to package the information shared, so it does not get shared with more than the interested parties without compromising the contents of the information. That is what we call cryptography nowadays. With the development of the internet, society's dependency on technological systems grew gradually, currently being the IoT (Internet of Things) the most used systems. These systems made possible to generate, access, share and store data related to personal, business and enterprises in a fast and easy manner. This amount of information shared raises the need to develop new and robust cryptographic systems, to be able to secure the information on demand. This brings us to another problem: the more complex cryptographic systems get, the more energy it uses, and so, the smaller the devices the less power it can give, reducing the options to secure the information. All these concerns drive the objectives of the development of cryptographic systems, which are low area, low power consumption and high robustness. To understand how to develop cryptographic systems, we must understand what it is. We have the information which gets encrypted with a random key and transforms into the message and gets sent to the recipient, getting decrypted with the same key, enabling the second party to access the readable information. The key to encrypt and decrypt the information must be random, it does not matter how it is created, is it by software or hardware. The focus of this paper is the development of an entropy source to generate random keys.

II. RNGs and PUFs

One way to create random keys is by using a random number generator. These generators are based on algorithms and create a stream of bits which get processed, and a key is produced. There are two important tools to achieve cryptographic security: RNGs and PUFs.

A. Random Number Generators

There are two types of random number generators, ones that use the randomness of physical processes, true random number generators (TRNG), and ones that use complex mathematical algorithms, which are called pseudorandom generators (PRNG). The latter are called pseudorandom because if the mathematical system and the initial conditions are known, it is possible to calculate the output of the system. This does not happen on the TRNGs, which are truly random and impossible to replicate, even with the knowledge of the initial conditions. For instance, if we have a pseudorandom generator that works with prime numbers, if we know where it started and how many times it ran, with enough time, a computer can calculate it. On the other hand, if we have a TRNG based on the possibility of an earthquake, even if we ran it at the same place, it would never be the same. Usually, TRNG are hardware-based, needing to be well protected to physical attacks.

B. Noise

Randomness is the key to have a random output on an RNG and, to achieve it, we must tap into an entropy source. The noise of electrical circuits is, usually, a big drawback on the design of analog circuits, because it makes the circuit consume power that is used to spoil the signal, and so spoiling the information that the signal carries. In this work, the noise is a very important part, due to its randomness. There are several types of noise, such as shot, flicker and jitter. The noise types which we take advantage of in this paper are the thermal noise and the jitter. Thermal noise is very present on high frequency signals. This type is also known as $1/f^2$, because it is influenced by the square of the frequency. This type of

noise affects the amplitude of the signal, making it rise and fall in a small quantity randomly. Jitter, on the other hand, affects the frequency of the oscillator, which means in each cycle, the period can alter, affecting the signal. There is one more type of noise which is not random, although it seems random. This noise type is called chaotic, which is a deterministic type of noise, which means that given the time, it repeats with a frequency, therefore not random.

C. Oscillators

Oscillators are electronic circuits that receive a DC input and output a periodic signal (prof). There are several types of oscillators, but we will focus on ring oscillators. Ring oscillators function by delaying the input signal. This means that the oscillator needs some extra power at the beginning, so it can start the oscillation. A ring oscillator consists of delay cells (or inverters) connected in a ring configuration, which means the last delay cell is connected to the first one in a loop. The signal gets delayed each delay cell it passes, stopping at the opposing bit, which means if the signal input is 0, at the output, it needs to be 1 for the circuit to oscillate. We assure this by using an odd number of delay cells, starting with three. We can, however, use an even number of delay cells, if we use differential delay cells and cross the outputs of the second to last cell, this way we can be sure the output of the oscillator is the opposite of the input.

D. PUFs

PUFs (or physically unclonable functions) are circuits which save cryptographic keys which are unclonable. The difference between the TRNGs and PUFs is that PUFs are designed to have a set key size and repeats it, where TRNGs never repeats the sequence with a pattern. For instance, if we have a PUF and a TRNG that generate a 4 bit key and we run it for 4 sequences, the PUF would generate 4 equal sequences, i.e. 1011101110111011, and the TRNG would generate 4 different sequences, e.g. 1011010001111011, showing that the TRNG we would be able to get 4 keys from that sequence, unlike PUFs, where we get only one stable key. A very good use for PUFs is from authentic devices, because if the authentic device has an integrated PUF, it generates a unique key for that device, making it unique too.

III. State of the art

There are several types of RNGs, which we will present some recent works based on them, like free running oscillators, Johnson noise and a free running oscillators with chaotic noise. Free running oscillator based RNGs work by letting an oscillator run for a period of time, which is affected by the jitter of its frequency, making it possible to compare to its ideal frequency and registering the difference between the oscillator and the perfect clock, which will, in theory, be random. As developed in x, the free running oscillator is paired with a lower frequency oscillator to sample it and a D-flipflop to compare both frequencies and output the bits. Usually, this type of RNG are implemented on FPGAs or ASICs due to its modular nature, this means the free running oscillator module can be replicated any number of times (within the board resources) until the RNG behaves like a TRNG. This architecture has one big drawback, which is the power consumption. Although it has virtual infinite resources, if the power consumption is a constraint for some reason, for instance, to be installed on a battery power device, its infinite scalability stops being ideal. One example of a free running oscillator based RNG is proposed by Burak Acar and Salih Ergün [3], where they propose the use of SR latches 1 as a slow clock sampling a high frequency clock with a D-flipflop, all implemented on an FPGA. This work uses a free running oscillator approach, which uses the jitter of the oscillators to change its output.

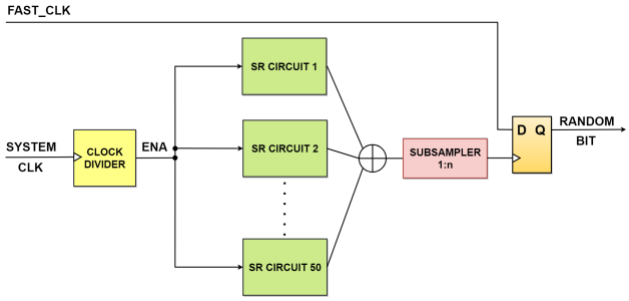

Fig. 1. Proposed RNG by Burak Acar and Salih Ergün in [3].

The authors affirm that their proposal uses a 1% of the FPGA logic resources, a throughput of 0.34 Mbits/s and has passed all the tests. This work is very compact and has a good throughput, however, it does not reference its power usage and it is implemented on an FPGA, making it hard to be used in a battery powered portable device.

Johnson's noise based RNG uses the thermal noise present on some components to generate entropy, making it possible to generate random numbers. For instance, a current is run through a resistor, which creates a voltage on its terminals, and the thermal noise will impact the voltage, making it shift up and down. Because the noise as a low magnitude compared to the voltage of the resistor, it needs to be amplified, so the resistor is then connected to an amplifier to make the noise impact greater. To finish, the amplified voltage is compared to the stable theoretical voltage at the terminals of the resistor, generating the bits related to the comparison. An example of this method is proposed by Laurenciu et al. in [11], proposing a circuit with two noise sources, an amplifier, a counter and a comparator, as seen in figure 2

According to the authors, this architecture has a 1GHz bit sampling, in CMOS 65nm and have passed all the NIST tests, showing no autocorrelation between different generated bit sequences. This work shows the power of $1/f^2$ noise, having a very good throughput and passing all the statistical tests. However, its architecture does not use oscillators, which did not align with our goals.

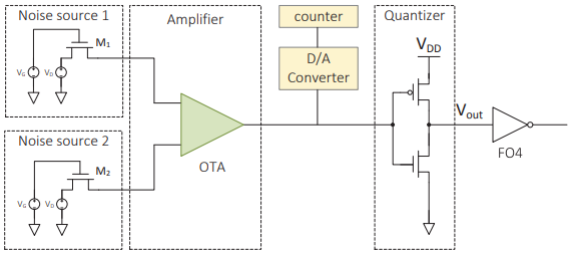

Fig. 2. Proposed RNG by Laurenciu et al. in [11].

IV. Circuit design

The goals of this work were to develop an entropy source using oscillators with CMOS $130nm$ technology to be integrated on a chip, having the best statistical results while having the lowest power consumption and area possible. We opted to use four pairs of free running oscillators connected to four chaotic cells, coupled by a XOR gate and sampled by a much slower oscillator.

Firstly, we began by choosing the delay cells for our clock oscillator. We chose a differential delay cell with an RC net to start the oscillation by generating a negative conductance, substituting the traditional latch with transistors. By doing this, we were trying to evade leaking currents through the latch transistors and so, drastically reduce the power dissipation. The delay cell is shown in figure 3.

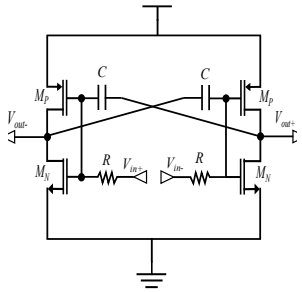

Fig. 3. Proposed delay cell for the clock.

We used 6 delay cells connected in a loop to form the clock oscillator and using the sizes of the transistors present on table I, we achieved a frequency of 26 MHz.

TABLE I
Clock components' sizing.

PMOS-L (μm	PMOS-W (μm)	NMOS-L (μm)	NMOS-W (μm)	R ($K\Omega$)	C pF
1.5	1	0.5	1	5	0.5

Secondly, we chose a delay cell for the fast oscillator. To be able to have the most noise coming from the supply source and not filter it through several transistors, we chose a simple CMOS inverter, shown in figure 4.

We designed it to have 7 delay cells connected in a loop, and achieved a frequency of 500 MHz with the component sizing in table II.

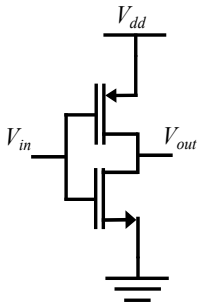

Fig. 4. Inverter delay cell used on the free running oscillator.

TABLE II
Inverter PMOS and NMOS sizing.

PMOS-L (μm	PMOS-W (μm)	NMOS-L (μm)	NMOS-W (μm)
0.5	3	0.5	1

To add more noise to the system, we decided to use a chaotic noise generator to pair with the fast oscillators. Although chaotic noise is deterministic, when added to stochastic noise, it amplifies it. We decided to pair 2 fast oscillators with the chaotic noise generator. The circuit of the chaotic noise generator is shown in figure 5 and the components size is shown in table III.

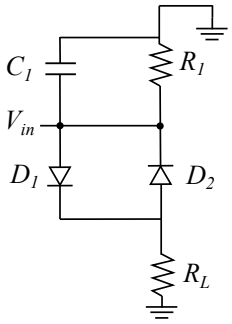

Fig. 5. Proposed chaos cell for the free running oscillator.

TABLE III
Chaotic cell components sizing.

R1 ($K\Omega$)	R2 ($K\Omega$)	C (pF)
100	10	1

We tried a couple of configurations of the circuit, firstly using only 2 pairs of fast oscillators (figure 6) and secondly using 4 pairs (figure 7). We found out that having more pairs helped the noise and did not affect considerably the power dissipation. We chose to alter the frequency between the pairs of oscillators but, to ease in fabrication, we decided to change the supply voltage of each pair instead of changing the number of delay cells. The fast oscillators are supplied with 0.8 V, 0.9 V, 1 V and 1.1 V, respectively, to obtain different oscillation frequencies.

TABLE IV
SIMULATED VALUES OF POWER, AREA, THROUGHPUT AND ENERGY PER BIT OF THE DESIGNED CIRCUIT

	Designed circuit
Power (mW)	1.19
Area (μm)	66
Throughput ($Mbits/s$)	25
Energy per bit (pJ/bit)	47.6

The final circuit is composed of 4 pairs of 7-stage fast oscillators connected to 4 chaotic noise generators to provide the raw data,3 XORs to combine the raw data, a 6 stage clock oscillator to sample and a D flip-flop to output the random data sequence, as shown in figure 7.

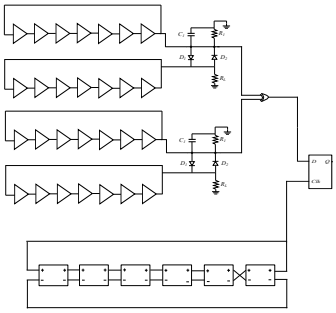

Fig. 6. Single stage circuit.

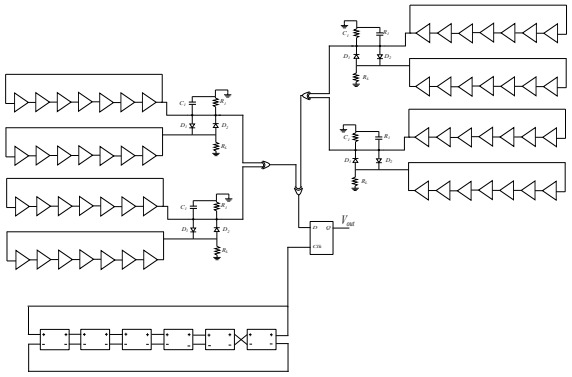

Fig. 7. Complete proposed circuit.

V. SIMULATION

We have simulated the proposed circuit with transient noise to be able to evaluate the behavior of our circuit, with the values of power, area of transistors, throughput and energy per bit on table V and the output signal on figure 8.

Evaluating table V we can say that we achieve our goals, having a low power design with a good throughput and a low energy per bit. We only show the area of the transistors because we did not proceed with the layout of the circuit, and did not estimate the full size of the circuit.

We ran a noise simulation of the fast oscillators to be able to evaluate how they improve by adding more pairs, and the jitter is shown in figures 9 and 10, respectively.

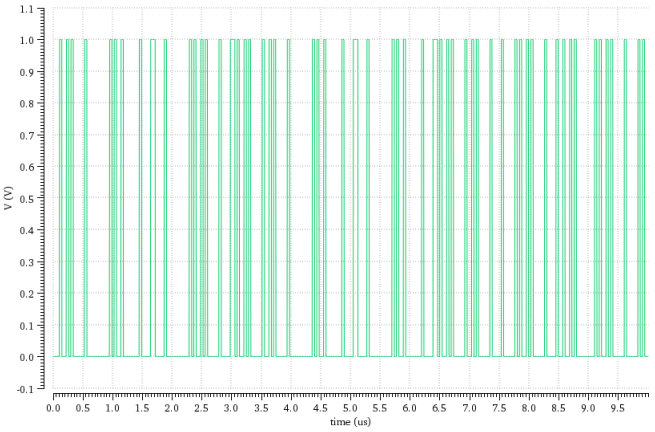

Fig. 8. Waveform of the complete circuit.

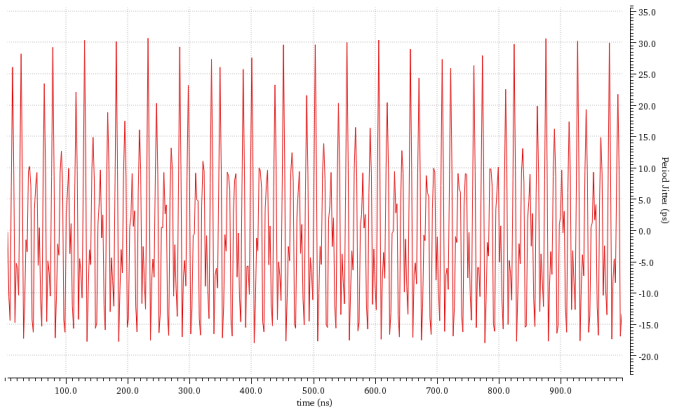

Fig. 9. Jitter from a single pair of fast oscillators.

We can see that there is a significant improvement by adding more oscillator pairs, by having significantly more jitter with 4 pairs instead of 1 pair.

To be able to run the NIST test suite, we withdraw the bits from the simulator to a text file, by processing the .csv file taken from the Cadence Virtuoso software (which had close to 20 million points) with a script on MATLAB that sampled the points with the ideal frequency of the clock, outputting a .txt file which we use to run the NIST SP 800-22 tests. Due to time and resources constraints, we were only able to take 3000 bits from the first and second versions of the circuit and 200 from the third, which made it impossible to run tests that needed a large number of bits. We were only able to run the Frequency test, the Block Frequency test, the Cumulative Sums test, the Longest Run test, the Runs test, the Non Overlapping test, the Serial test and the Approximate Entropy test. The next table shows the comparison between the versions of the circuit.

Lastly, we present a comparison between the work developed in [3], [4], [5] and the work developed in this work, with power consumption, area occupied and the result of the NIST SP 800-22 tests, present in table V.

TABLE V
COMPARISON OF STATISTICAL TESTS BETWEEN THE PROPOSED CIRCUIT AND THE STATE OF THE ART.

	Proposed		[3]		[4]		[5]	
	Pvalue	Result	Pvalue	Result	%Pass	Result	Pvalue	Result
Frequency	0.3317	Pass	0.1512	Pass	97/100	Pass	0.0533	Pass
Block Frequency	0.8114	Pass	0.1172	Pass	98/100	Pass	0.0030	Pass
Cumulative Sums	0.6562	Pass	0.6641	Pass	96/100	Pass	0.2958	Pass
Longest Run	0.1671	Pass	0.7099	Pass	98/100	Pass	0.8343	Pass
Runs	0.0606	Pass	0.0899	Pass	100/100	Pass	0.0444	Pass
NonOverlapping	1,0000	Pass	0.9341	Pass	98/100	Pass	0.6074	Pass
Serial	0.1561	Pass	0.7473	Pass	99/100	Pass	0.5486	Pass
Approx.Entropy	0.1167	Pass	0.0001	Pass	98/100	Pass	0.0069	Pass

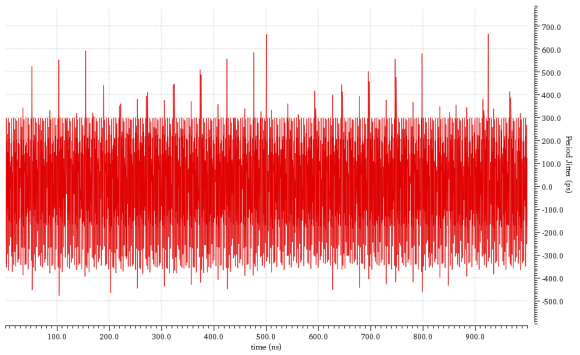

Fig. 10. Jitter from 4 pairs of fast oscillator.

VI. CONCLUSION

In this paper, we have designed and tested an entropy source based on oscillators which has potential to be a TRNG, having a low area usage and low power consumption. The circuit has the particularity of having an RC net ring oscillator used as a clock and a single-ended oscillator paired with a chaotic cell. The circuit has an area of 66 μm and a power consumption of 1.19 mW, with 26.28 MHz frequency of sampling. Although we were not able to run the whole NIST 800-22 test suite, the proposed work passed on all the tests we were able to run with the available data.

VII. ACKNOWLEDGEMENTS

This research work has been supported by FCT - Fundação para a Ciência e a Tecnologia, I.P., under the scope of the projects: CTS/UNINOVA (UIDB/00066/2020), foRESTER PCIF/SSI/0102/2017, and ROBUST EXPL/EEI-EEE/0776/2021.

REFERENCES

[1] J. F. Dooley. History of Cryptography and Cryptanalysis. Springer International Pu- blishing, 2018. isbn: 978-3-319-90442-9. d o i: 10.1007/978-3-319-90443-6 (ver p. 2).

[2] Y. Cao, L. Zhang, C. -H. Chang and S. Chen, "A Low-Power Hybrid RO PUF With Improved Thermal Stability for Lightweight Applications," in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, vol. 34, no. 7, pp. 1143-1147, July 2015, doi: 10.1109/TCAD.2015.2424955.

[3] B. Acar and S. Ergün, "A Random Number Generator Based on Irregular Sampling and Transient Effect Ring Oscillators," 2020 IEEE International Symposium on Circuits and Systems (ISCAS), 2020, pp. 1-5, doi: 10.1109/ISCAS45731.2020.9181193.

[4] V. K. Rai, S. Tripathy and J. Mathew, "TRGP: A Low-Cost Re-Configurable TRNG-PUF Architecture for IoT," 2021 22nd International Symposium on Quality Electronic Design (ISQED), 2021, pp. 420-425, doi: 10.1109/ISQED51717.2021.9424347.

[5] S. Robson, B. Leung and G. Gong, "Truly Random Number Generator Based on a Ring Oscillator Utilizing Last Passage Time," in IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 61, no. 12, pp. 937-941, Dec. 2014, doi: 10.1109/TCSII.2014.2362715.

[6] Stipčević, M., Koç, Ç.K. (2014). True Random Number Generators. In: Koç, Ç. (eds) Open Problems in Mathematics and Computational Science. Springer, Cham. https://doi.org/10.1007/978-3-319-10683-0_12

[7] Tuzlukov, V. (2002). Signal Processing Noise (1st ed.). CRC Press. https://doi.org/10.1201/9781315220147

[8] Simon Haykin. 2009. Communication Systems (5th. ed.). Wiley Publishing.

[9] L. B. Oliveira et al. Analysis and Design of Quadrature Oscillators. Dordrecht: Sprin- ger Netherlands, 2008. isbn: 978-1-4020-8515-4. doi: 10.1007/978-1-4020-8 516- 1. url: http://link.springer.com/10.1007/978-1- 4020- 8516- 1 (ver pp. 15, 16).

[10] U. L. Rohde. "Oscillator basics and low-noise techniques for microwave oscilla- tors and VCOs". Em: Proc. European GaAs and Other Semiconductors Application Symp.(now EuMIC). 2000 (ver pp. 14, 15).

[11] Laurenciu, Nicoleta Cucu and Cotofana, Sorin D., "Low cost and energy, thermal noise driven, probability modulated random number generator", 2015 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 2724-2727, 2015, doi:10.1109/ISCAS.2015.7169249