

A Fully-Synthesized TRNG with Lightweight Cellular-Automata Based Post-Processing Stage in 130nm CMOS

Juan Cartagena, Hector Gomez, Elkim Roa

Design Group of Integrated Systems CIDIC, Universidad Industrial de Santander, Bucaramanga, Colombia
{juan.cartagena,hector.gomez}@correo.uis.edu.co, efroa@uis.edu.co

Abstract—A fully-synthesized true-random number generator (TRNG) using cellular automata as post-processing stage is implemented in an FPGA and in a 130nm CMOS technology. A 3-edge ring oscillator provides the entropy source based on accumulative jitter. The post-processing stage uses a programmable array of cellular automatas and its performance is evaluated for all possible rules that can be constructed. 1.4Mbits are captured using the FPGA implementation and the randomness of data is tested applying NIST tests. One-dimensional cellular automata in a TRNG is reported reducing bias of output data. In addition, the fully-synthesized generator is completely verified for fabrication in 130nm CMOS technology and occupies a final area of 0.0098 mm² where the post-processing stage uses only 0.0023 mm².

Index Terms—TRNG, Digital design flow, Jitter, Random number generator.

I. INTRODUCTION

Random number generators are a fundamental part in cryptographic algorithms defining its proper operation. In recent years, crypto-systems for cloud computing and the internet of things have done increasing the interest in high quality random numbers. Furthermore, applications of randomness includes operations such as banking transactions, data encryption, logged on websites, secure shells and digital signatures.

Security in crypto-systems relies on the fact that the generated random numbers are practically unpredictable besides having uniformity and independence [1–3]. Pseudo random number generators (PRNGs) are a common solution to get the necessary random data, but these generators are based on deterministic algorithms which output data stream is repeated with a finite period. For this reason, true-random number generators (TRNGs) are used when there is a necessity of strong security. TRNGs make usage of physical randomness, as entropy source, which is the only way to produce truly non-deterministic data behavior.

A high quality entropy source, required to produce unpredictable numbers, needs to be robust. The robustness of physical sources requires a careful design that can be implemented in analog and digital domain. Based on analog domain such as a resistor-amplifier-ADC chain, TRNGs present some voltage bias and are very sensible to variations in process, voltage and temperature (PVT) [4]. The randomness also can be disrupted by device drift due to aging [5] and also, the analog ones have low scalability to another process technology.

Considering the disadvantages of the generators based on analog domain, digital and specially fully-synthesized based TRNGs are attractive alternatives to reduce implementation complexity besides overcoming effects of PVT.

Despite the fact that logic in digital TRNGs are less sensitive to PVT variations, the digital entropy sources can be affected by mismatch and another effects which may result in a tendency of the output data to certain values and low randomness. Therefore, digital generators, as the analog ones, may not get enough randomness to be used in cryptographic applications by themselves. For this reason, an additional post-processing step that ensures good random qualities, is usually added to the generator [3], [6].

In this work, we propose a fully-synthesized post-processing technique based on cellular automata (CA) that complements a TRNG based multimode ring oscillator [7] with four different length rings. The CA based post-processing stage provides a low-hardware overhead solution over other based on classical PRNGs. The post-processing stage can be programmed with different cellular automatas in order to identify which rules achieve better statistical properties. Verification of randomness is carried out through NIST tests where eight rules outperform the output of the TRNG. Implementation is performed in FPGA and fully synthesized in 130nm for fabrication.

II. THE TRUE RANDOM PROPOSED SCHEME

The proposed TRNG like most of the hardware-implemented generators are composed by an entropy source, an extraction stage and a post-processing stage. Fig. 1 shows the schematic of the proposed implementation indicating the four entropy sources, the capture stage and the cellular automata based post-processing stage called "Rule_X". The entropy source is based on the accumulative jitter of ring oscillators (RO) proposed by [7]. The capture stage operates as extractor of random numbers and the post-processing stage exploits the pseudo random behavior of different cellular automatas rules to improve statistical properties.

A. Entropy Source

The entropy source is based on jitter accumulation in multimode ring oscillators. The implemented entropy stage uses four sources that consist in different-length ROs getting

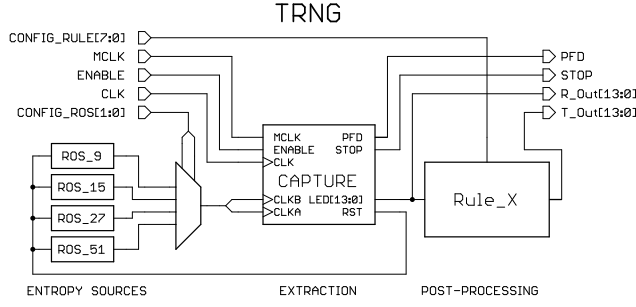


Fig. 1. TRNG schematic: the source of entropy (left), extraction (center), post-processing (right).

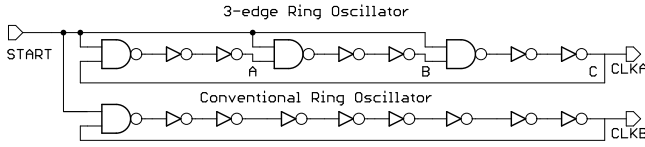


Fig. 2. Multimode RO (top), Conventional RO (bottom) [7]. Note the points A, B and C at the input in nand gates.

different oscillation frequencies and required cycles to collapse. Each source has a pair of multimode and normal RO with 9, 15, 27 and 51 stages. Fig. 2 shows the schematic of a 9 stages pair of ROs as example. ROs are multiplexed in order to probe the performance of the TRNG under different oscillations.

In a conventional RO, a NAND gate enables and disables the oscillation; when a logical one appears in the enable input *Start*, an edge begins to spread between the inverters until the last stage causing an oscillation with a nominal frequency. In a similar way, when a logical one appears in a multimode RO, injects 3 edges simultaneously in the NAND gates by the signal *Start* generating an oscillation at 3x nominal frequency spreading along the ring. As three edges accumulate jitter from thermal noise, eventually two neighboring edge collapse, forcing the RO to oscillate to the nominal 1x frequency mode.

Fig. 3 shows the time domain of signals in points A, B and C coming out of Fig. 2 to explain the concept. Signals at points A, B and C oscillate to a higher frequency than nominal for an unpredictable time (number of cycles) with a width pulse

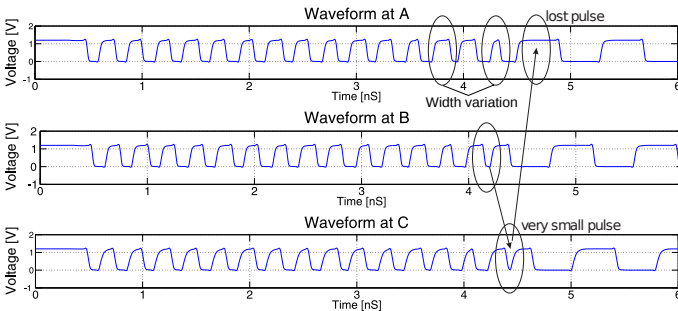


Fig. 3. Time graph of oscillations: Node A (a), Node B (b) and node C (c). from Fig. 2

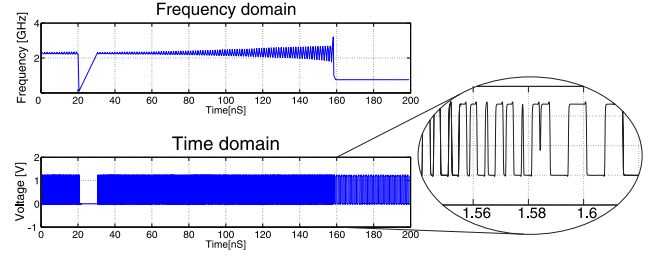


Fig. 4. Time and frequency graph of proposed 3-edge ring oscillator: frequency (top), time (bottom).

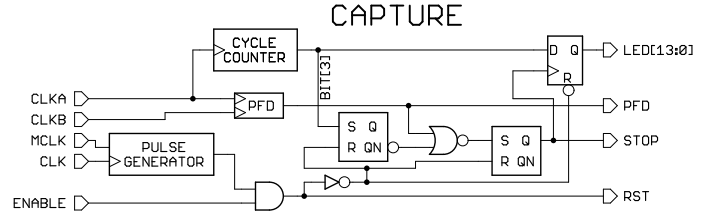


Fig. 5. Capture stage [7].

variation due to jitter. This variation causes two neighboring edges collapse, forcing the RO to oscillate with the 1x nominal frequency.

Other transient simulation is shown in Fig. 4 to analyze the frequency collapse, the upper figure shows the measurement of the frequency change of bottom figure. Simulations are performed with a greater number of inverter stages than in Fig. 3 to improve the collapse time. The upper figure shows how the frequency goes initially to zero due to inputs of the NANDs are put to zero by a short time and then goes up to 2.1 GHz approximately, value maintained until collapse occurs and frequency goes down to 700 MHz. Bottom figure illustrates the same behavior in time domain where the zoom allows us to see the collapse event in similar way than in Fig. 3.

The extraction of random bits is done through the CAPTURE stage as is shown in Fig. 5. The method to extract random bits consists in counting the cycles of the 3-edge RO until a collapse occurs with a 14-bit cycle counter. A digital frequency phase detector (PFD) identifies the collapse event of frequency and trigger the output register. There are two SC latches used to prevent false triggering and the Pulse Generator is an edge detector that delivers a constant width pulse to reset the latches and the output register.

B. Cellular Automates as Post-Processing

The cellular automata was proposed in 1982 by Stephen Wolfram and can be considered as a discrete differential equation that by iterations develops a series of binary numbers, where the coefficients of these differential equations determine the cellular automata behavior. A one-dimensional cellular automata has 3 coefficients that can be combined in 8 different forms [8]. Table I shows an example of how to calculate each possible combination with its corresponding output. There are

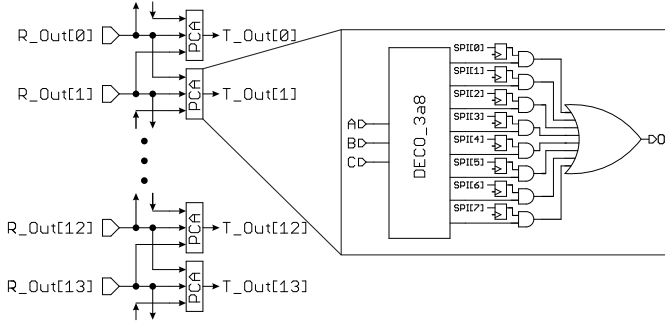


Fig. 6. Rule_X stage: Array of PCA used for post processing

8 different cellular automatats that can be configured in 256 different ways or **rules** [6].

The behavior of the 256 rules is different and each one has pseudo-random outputs, where some of them seem to have a more random output. This work proposes to study the behavior of all rules, one by one, in order to identify which ones accomplishes a better improvement in statistical properties of the TRNG output. This proposed stage is called Rule_X and consists in an array of 14 CAs that can be programmed with the 255 possible different rules. Each programmable CA (PCA) is composed by a 3 to 8 binary decoder, 8 registers to store the rule to be used and a logic to capture data as shown in Fig. 6, where input data come from the 14 bits counter of the TRNG. The boundary conditions are a closed circle as follows: the first automata is fed with the bits 1,0 and 13 and the last automata is fed with bits 0,12 and 13. The post processing stage allows to set a rule which is stored in the flip-flops and from there on the process is done through a combinational logic. This scheme enables the post-processing to be done in one clock cycle, in order that the dynamic power consumption occurs only when new data is generated. Furthermore, the maximum operation frequency varies according to the rule.

This stage was designed with the purpose of studying the performance of different rules. Moreover, by adding an external control (with a microcontroller for example) is possible to change the rule of post-processing during the TRNG operation. The stage uses in total 340 logic gates and a future implementation can be optimized as a function of some rules to reduce the total area. For example, a post-processing stage fixed with rule 150 uses only 112 logic gates. Finally, this stage also can be segmented to enhance the maximum frequency.

III. MEASUREMENTS RESULTS

The proposed TRNG is fully-synthesized in a FPGA and in a digital design flow with 130nm technology. For adequate performance, placement restrictions are added in both

TABLE I
DEFINITION OF A RULE. RULE 30.

State	111	110	101	100	011	010	001	000
Output	0	0	0	1	1	1	1	0
30 dec	128	64	32	16	8	4	2	1

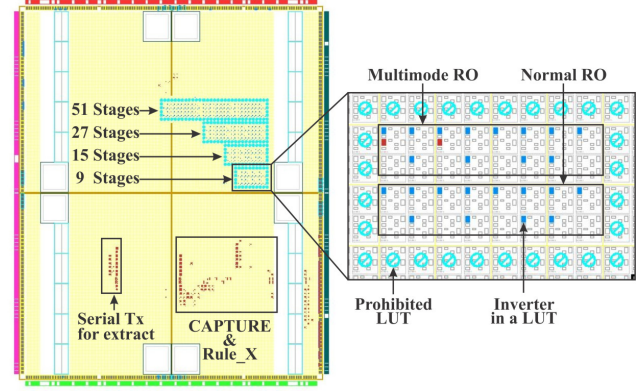


Fig. 7. Layout in FPGA: On the right top side the four ROs are shown, at right bottom side are the Capture and Rule_x modules and the Tx module use to extract the data is placed at bottom left side, at right side is a zoom of a RO layout. 233 Slices, 36 Flip Flops, 409 LUTs.

implementations to the ring oscillators avoiding any external interference that can be coupled with the rings. The FPGA implementation is made by using a Spartan 3AN and the ASIC Implementation was taped-out.

A. FPGA implementation

FPGA implementation helps to verify the performance of the proposed TRNG. By testing the concept of 3-edge ring oscillator, each inverter and nand gate are implemented by using Look Up Tables (LUTs) that allow x,y coordinate position using constraints in the synthesis environment. Fig. 7 shows the placement in FPGA with the restrictions for the four ring oscillators, considering the LUTs around rings are prohibited to avoid interference, and there are a zoom view of the smallest 9 stages RO.

For validation purpose, a total of 1.4Mbits were captured in order to apply the tests of National Institute of Standards and Technology (NIST) [9] to the output data. Outputs for the 256 rules were analyzed and only 30 showed some relevant result (nonzero numbers in any tests) with 6 of them outperforming the original data. Table II shows 14 results of the applied tests for the 6 outstanding rules. The first column shows the results without post-processing and the NonOverlappingTemplate row shows the sum of 148 different sub-tests. The final row presents the sum of all applied tests

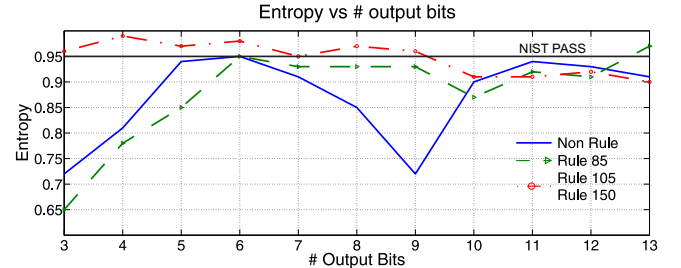


Fig. 8. Entropy vs Number of outputs bits: Approximated Entropy from NIST test results for Original data, rules 85,105 and 150 with different truncation.

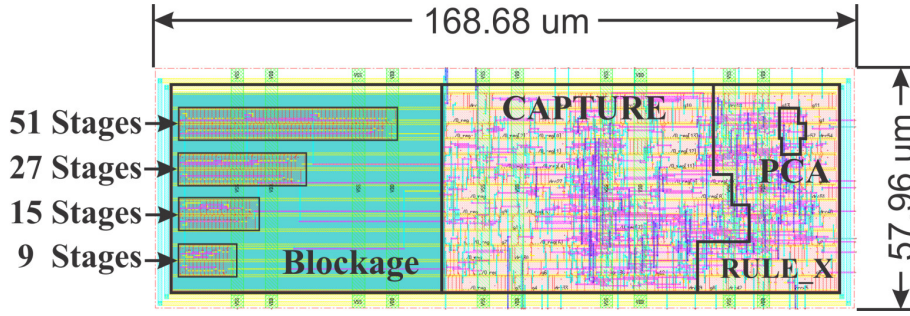


Fig. 9. Layout based standard cells in 130nm, total Area: $171\mu\text{m} \times 59\mu\text{m}$.

to identify the improvement over prior output and to select the rules with better performance. Table III shows the P-VALUES for Non-rule and rules 105 and 150.

It is possible to see that the rules 85, 105 and 150 obtained the best punctuation and the rules 75 and 180 were the best for the Approximate Entropy test. Rule 150 obtained perfect pass in six of the NIST tests. Rules 105 and 150 provided a full pass of the 148 sub-tests.

In spite of improving entropy approximation output test and that the rules 75, 85 and 180 passed the NIST criteria, none of them reach a value of 100% using non-truncated data. Fig. 8 shows the test result of Approximated Entropy (NIST test) according to the number of the output bits for data without post-processing and for rules 85, 105 and 150. These results were obtained by applying the NIST test to different truncation of the output bits. It should be clarified that the graph of entropy for rules 105 and 150 are exactly the same and therefore are super-placed on the red graphic. It is also possible to see that for a number of bits less than 9, these two rules get enough approximate entropy to pass the NIST test.

B. ASIC Implementation

The TRNG was fabricated in a 130nm CMOS standard technology with a final area of $170\mu\text{m} \times 58\mu\text{m}$ and the chip will be available for testing soon. The Power consumption of processing and capture stage obtained from synthesis to 1 GHz frequency is 2.66mW and uses 969 gates. Fig. 9 shows the

TABLE III
MEASURED NIST RANDOMNESS TEST RESULTS WITH P-VALUES FOR NON RULE AND RULES 105 AND 150.

Rule	Non Rule		105		150	
	P-VALUE	RATE	P-VALUE	RATE	P-VALUE	RATE
Frequency	0.000000	94	0.000818	100	0.000818	100
BlockFrequency	0.000001	95	0.983453	99	0.983453	99
CumulativeSums	0.000000	93	0.867692	100	0.867692	100
CumulativeSums	0.000000	93	0.350485	100	0.350485	100
Runs	0.202268	98	0.145326	99	0.145326	99
LongestRun	0.162606	99	0.798139	99	0.304126	100
Rank	0.145326	99	0.129620	97	0.005358	99
FFT	0.275709	99	0.275709	100	0.275709	100
NonOverlappingTemplate	0.514124	146	0.699313	148	0.779188	148
OverlappingTemplate	0.455937	100	0.137282	98	0.759756	99
Serial	0.000000	87	0.383827	98	0.383827	98
Serial	0.000145	98	0.595549	99	0.595549	99
LinearComplexity	0.213309	96	0.020548	97	0.366918	99
ApproximateEntropy	0.000000	91	0.000000	90	0.000000	90

layout where the four ROs are placed at the left side by forcing the position through synthesis constraints. Additional blockage is added to avoid synthesis process places other gates around ROs. These restrictions help to keep out external frequency noise sources.

Table IV shows the results in cells and area of synthesizing PCA array that uses 340 cells, whereas an individual PCA has 18 cells and the final area of the array is $2.319 \mu\text{m}^2$. This final area is low if it is compared with the area occupied by a fixed rule. For instance, the three most outstanding rules are fixed to make a comparative study with the final implementation and in the best case the rule 85 occupies more than 40% of final area. Therefore, the programmable array allows us to exploit better the area and give the possibility to study all the rules.

TABLE II
MEASURED NIST RANDOMNESS TEST RESULTS.

Rule	Non Rule	51	75	85	105	150	180
Frequency	94	94	98	97	100	100	98
BlockFrequency	95	95	100	100	99	99	100
CumulativeSums	93	93	98	97	100	100	98
CumulativeSums	93	93	99	97	100	100	99
Runs	98	98	90	96	99	99	90
LongestRun	99	98	100	98	99	100	100
Rank	99	99	99	97	97	99	98
FFT	99	99	100	99	100	100	100
NonOverlappingTemplate	146	146	142	146	148	148	142
OverlappingTemplate	100	99	98	100	98	99	98
Serial	87	87	95	96	98	98	95
Serial	98	98	97	97	99	99	97
LinearComplexity	96	99	100	98	97	99	99
ApproximateEntropy	91	91	98	97	90	90	98
Sum	1388	1389	1414	1415	1424	1430	1412

TABLE IV
AREA FOR PROGRAMMABLE AND FIXED RULE CA

Name	Cells	Area[μm^2]
Programmable Rule_X	340	2.319
Programmable CA	18	130
Fixed Rule_105	140	1.284
Fixed Rule_150	112	1.283
Fixed Rule_85	126	1.045
Fixed CA	7	90

Table V shows a comparison of this work and a PRNG implementation with post-processing using FPGA. The proposed TRNG is compact despite the programmable post-processing stage and offers the advantage of changing the pseudo-random

TABLE V
COMPARISON WITH ANOTHER FPGA IMPLEMENTATION.

Specifications	[3]	[10]	This work
LUT	308	272	409
FF	380	807	36
$f_{\text{real}}[MHz]$	275	-	50

processing during operation.

IV. CONCLUSIONS

This work offers a fully-synthesized TRNG which is implemented in FPGA and ASIC with 130nm technology. The proposed TRNG uses a post-processing stage based on cellular automatas, reducing the bias of the output data. Regarding the FPGA implementation, extracted data allow us to probe the improvement of using the post-processing stage for 8 specific rules. The improvement is studied basing on the results of applying NIST test to data for all possible rules that can be constructed. Furthermore, the enhancement in approximate entropy is enough to reach at least 90%, but only two rules pass the 148 sub-tests. It is necessary to review the post-processing stage and include combinations of the rules to seek better results. Finally, the fully-synthesized TRNG was fabricated to measure the performance of the proposed generator under ASIC implementation.

REFERENCES

- [1] T. Amaki and M. Hashimoto and T. Onoye, "A process and temperature tolerant oscillator-based true random number generator with dynamic 0/1 bias correction", Proceedings of the 2013 IEEE Asian Solid-State Circuits Conference, A-SSCC 2013, Singapore, 2013, pp.133-136.
- [2] Sunar, B.; Martin, W.J.; Stinson, D.R., "A Provably Secure True Random Number Generator with Built-In Tolerance to Active Attacks," in Computers, IEEE Transactions on , vol.56, no.1, pp.109-119, Jan. 2007.
- [3] Dabal, P.; Pelka, R., "An efficient post-processing method for pipelined pseudo-random number generator in SoC-FPGA," in Mixed Design of Integrated Circuits & Systems (MIXDES), 2015 22nd International Conference , vol., no., pp.607-611, 25-27 June 2015.
- [4] Srinivasan, S.; Mathew, S.; Ramanarayanan, R.; Sheikh, F.; Anders, M.; Kaul, H.; Erraguntla, V.; Krishnamurthy, R.; Taylor, G. , "2.4GHz 7mW all-digital PVT-variation tolerant True Random Number Generator in 45nm CMOS," in VLSI Circuits (VLSIC), 2010 IEEE Symposium on , vol., no., pp.203-204, 16-18 June 2010
- [5] S. K. Mathew et al., "2.4 Gbps, 7 mW All-Digital PVT-Variation Tolerant True Random Number Generator for 45 nm CMOS High-Performance Microprocessors," in IEEE Journal of Solid-State Circuits, vol. 47, no. 11, pp. 2807-2821, Nov. 2012.
- [6] Loza, S.; Matuszewski, L., "A true random number generator using ring oscillators and SHA-256 as post-processing," in Signals and Electronic Systems (ICSES), 2014 International Conference on , vol., no., pp.1-4, 11-13 Sept. 2014.
- [7] Kaiyuan Yang; Fick, D.; Henry, M.B.; Lee, Y.; Blaauw, D.; Sylvester, D., "16.3 A 23Mb/s 23pJ/b fully synthesized true-random-number generator in 28nm and 65nm CMOS," in Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014 IEEE International , vol., no., pp.280-281, 9-13 Feb. 2014.
- [8] S. Wolfram,"Cellular automata as models of complexity" Nature, , vol., no.55, pp.449, 1985.
- [9] NIST, "A Statistical Test Suite for Random and Pseudo- random Number Generators for Cryptographic Applications," Special Publication 800-22, Oct. 2000.
- [10] Mocanu D. et al., "Global Feedback Self-Programmable Cellular Automaton Random Number Generator" Rev. Tc. Ing. Univ. Zulia. Vol. 39, N 1, 1 - 9, 2016.