Realization of True Random Number Generator Using Clock Divider on FPGA Platform

Kshanada Choudhary (Research Scholar), Dr. N.G. Narole (Guide)

Rajiv Gandhi College of Engineering & Research

ABSTRACT-Random Number Generators (RNGs) play an important role in cryptography. The security of crypto algorithms and protocols depend on the ability of RNGs to generate unpredictable secret keys and random numbers. The security of such systems highly relies on the quality of the random output produced by the generators.True Random Numbers are mostly used in generating non-reproducible and nondeterministic patterns used in different cryptographic protocols. They produce strong keys and challenges with high entropy and make procedures unpredictable. Conventional TRNGs often rely on the use of analog elements. It is our intention to avoid the use of analog elements (such as e.g. resistors or diodes) as noise source in order to streamline the adaption to arbitrary technologies and to allow the use of the standard digital design flow. However, digital circuits present comparatively limited number of sources of random noise, e.g., metastability of circuit elements, frequency of free-running oscillators, and jitters (random phase shifts) in clock signals. As would be evident, our proposed TRNG circuit utilizes the frequency difference of two free running ring oscillators, or in other words the beat frequency and oscillator jitter as sources of randomness. The random bit steam generated are undeterministic and unpredictable.

KEYWORDS: *True Random Number Generator (TRNG), Clock divider, field-programmable gate array (FPGA),beat frequency detection (BFD)*

1. INTRODUCTION:

Random Number Generators, an important security primitive in cryptography, are used to generate random numbers for different essential tasks like generation of secret keys, Initialization Vectors (IV), padding bits, seeds and nonce (numbers used once). An RNG uses a non-deterministic source (an entropy source), along with some processing function (an entropy distillation process) to produce randomness. As Random numbers are very crucial in cryptography and other areas, security of RNGs become a very important factor.

RNGs can be classified as Pseudo Random Number Generator (PRNG), True Random Number Generator (TRNG) and Hybrid Random Number Generator (HRNG). A PRNG requires an input called the seed. The randomness of the output bit streams of PRNG completely depends on randomness of the input seed. This means that for a given seed the output random bit stream of PRNG would always be same at any instance of time. PRNG based system uses system clock, mouse movement, network traffic etc. to generate a seed. Hence, this system is sensitive to security attacks [1]. In TRNG, no explicit input data such as seed is required since the entropy source is internal to the system itself. The internal entropy source is based on uncertain physical process. HRNG is a cascaded system with TRNG and PRNG, where the TRNG generates seed value for PRNG. HRNG would be useful when the rate of bit generation of TRNG is very less. In general, any TRNG system will have an entropy source for uncertainty and a sampling circuit which samples the random bit from entropy source without affecting the uncertainty. TRNGs are strongly recommended for applications that requires high level of security.

True Random Number Generators were and they still are a big and interesting area of research. They are used in different applications that rely on unpredictability from gaming, artificial intelligence, cryptography and so on. The main idea behind TRNGs is that no matter the circumstances that can be (re)created, the generator cannot generate the predicted output. When talking about modern True Random Number Generators we are referring to speed, stability and security. The most important idea behind this work is to provide a high speed TRNG, while using FPGA resources as possible but still maintaining the security and stability of it.

Various TRNG designs have been proposed and implemented on FPGAs. Each of these designs uses a different mechanism to extract randomness from an underlying physical phenomenon. Main source of randomness could be thermal or shot noise in the circuits, clock jitter, Metastability in circuits especially in flip-flops, Brownian motion, atmospheric noise and nuclear decay. Along with the noise source, a noise harvesting mechanism to derive the noise and a post processing stage to provide a uniform statistical distribution are other important components of the TRNG. Our focus is to design improved field-programmable gate array (FPGA) based TRNGs, using purely digital components. Using digital building blocks for TRNGs has the advantage that the designs are relatively simple and well suited to the FPGA design flow, as they can suitably leverage the CAD software tools available for FPGA design. FPGAs are popular platforms for implementing TRNGs because of their flexibility, small design cycle and re-configurability. However, digital circuits present comparatively limited number of sources of random noise, e.g., metastability of circuit elements, frequency of free-running oscillators, and jitters (random phase shifts) in clock signals. Our proposed TRNG circuit utilizes the frequency difference of two oscillators and oscillator jitter as sources of randomness.

This paper is structured as follows: the second chapter gives an overview on related work, the third chapter describes the concept of a TRNG, along with the steps needed to be taken into consideration, in order to create a generator, fourth chapter is structured to provide information on background and motivation of work and its shortcomings, proposed work has been introduced in fifth chapter followed with results and discussions on output.

2. RELATED WORK

Most of the TRNG implementations on FPGAs are based on free running oscillators. Oscillators provide a simple and effective method to build TRNGs because analog component like PLL is not required and it provides a great source of entropy which can be easily converted into digital form [5] [6]. Simple digital oscillator may be built by connecting an odd number of inverter gates in a ring configuration. Due to the feedback path, the output of any of the inverters will oscillate from a logic one to a logic zero and vice versa. Hence, a square wave signal is obtained by tapping the oscillator at any point in the ring. A theoretical basis of TRNGs based on sampling phase jitter in oscillator rings has been discussed by Sunar et.al. [5]. TRNG consisting of a number of 114 Free Running Ring Oscillators whose output were added modulo 2 by a free running block. They demonstrated that 114, as the number of ROs, was sufficient for that scheme and therefore for the stability and security of the generator[5].

Source of entropy is accumulated phase jitter on oscillations of RO and the entropy distillation process is XOR gate which receives input from multiple rings and a sampling Flip-flop. This design has been implemented with large number of oscillator rings of fixed length and a resilient function as post - processing unit in FPGA by Schellekenset.al. [7].The proposed and implemented design in [7] uses more FPGA resources as it needs many oscillator rings to obtain the required randomness in the generated bits.

Knut Wold [5] proposed a slightly modified design which introduces flip-flop pair at the end of each oscillator ring to remove metastability caused by sampling free running oscillator which has been illustrated in Fig. 1. This enhanced design eliminates the need for complicated post-processing required in Sunar's method detailed in [5].

Kohlbrenner and Kris presented in [8] a scheme that was based on Ring Oscillators implemented in FPGA. They obtained good statistical results and a decent output speed ratio. Another scheme, based on a slightly different approach was presented in [9], describing some good results as well.

In [10], there is presented the adaptation of Sunar's scheme for a Zybo Zynq™-7000 Development Board. The paper presented some conclusive results that strengthened and emphasized the principle described by Sunar.

Amaki described method using an oscillator that is sampled by a second one (So the jitter is deliberately enhanced in one of them [11][12]. It's possible to increase the entropy of the raw random stream generated by this circuit by means of a PRNG (or a Von Neumann extractor) [13]. In order to ensure a low power operation, the makeup between current and bandwidth must be take into account. Hence, the current budget initiates the limits of the "fast" and "slow" oscillators. In spite we can reuse external signals from the target system [14] to obtain the high speed oscillator, so the TRNG becomes more dependent of the specific target application.

3. TRUE RANDOM NUMBER GENERATORS

3.1. Preliminaries

The Generation of True Random Number implies the usage of three main components as follows:

- a reliable Noise Generator (that is based on a physical unpredictable phenomenon, like an oscillator's jitter or the unpredictability of a combinational function of a large number of ROs);
- a Randomness Extraction and testing (used for uniformly distribution of the output bits);
- an Entropy Extraction technique (techniques for measurement of jitter period)

All of these components are interconnected as the output of the first one is the input of the second one third one. The "approved" sequences (by the last component) represent the output of the whole generator.

3.2. Noise Generators

Noise Generators represent the pillar of the True Random Number Generator, being the component that actually provides the raw and random data, which has to be completely unpredictable. They are based on physical unpredictable and non-reproducible phenomenon, like cosmic radiations, thermal noise, the jitter of an oscillator, etc.

For this purpose, as a noise generator, the most common source is represented by the jitter found in the ring oscillators . Over the last years, the most discussed setup for a TRNG is presented by Sunar et al.in [5] (Fig. 2), where they are combining a large amount of ring oscillators, whose outputs are modulo 2 added, being sampled at a frequency given by another ring oscillator.

It is well known that each oscillator has a jitter. This jitter is influenced by different kind of factors, such as operating temperature, the stability of the power supply, fabrication imperfections for the components that make them to be slightly different, ambient thermal noise or vibration, etc.

The second most known oscillator for its jitter is the Ring Oscillator, consisting of an odd number of invertor gates that are connected in a ring structure (Fig. 4). This oscillator is probably the most used one because of its simplicity in FPGA implementation.

The difficulty of this approach is in finding a good scenario in which multiple oscillators that are oscillating at different frequencies can generate a non-deterministic output. For this, Sunar et al. proposed a solution based on a combination of a large amount of ring oscillators and demonstrated mathematically that his solution is stable and reliable.

Another good approach of an oscillator based TRNG is called TERO and has been described in [[15] and Fig 1. The advantage of this approaches is that they can be easily be implemented using a FPGA.

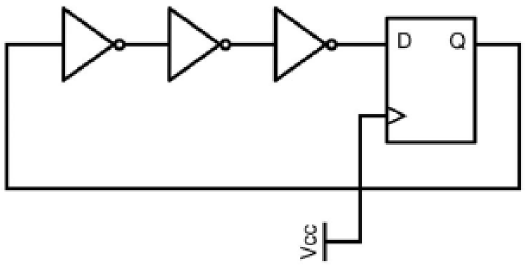

Fig 1. FPGA Schematic of a Ring Oscillator

The presence of the latch within the interconnected inverters is due to the compiler's optimizations which tends to reject such ring constructions. In this way, while using a latch whose clock is always 1, and who is logically considered as a wire, the ring oscillator can be thus implemented. Every Free Running Ring Oscillator (it doesn't require a clock signal as input), has its own running frequency which directly depends on different factors like the number of inverter gates, the physical distance between the components, the temperature of the components, etc. Due to this factors, if we instantiate two identical ROs, their frequency will slightly differ. Combining a large number of such different frequencies ROs in different setups, one can obtain a good TRNG. In [5], there are presented, in detail, two randomness extraction techniques that can be applied to acquire random data.

3.3. Randomness extraction and testing

Randomness extractors are used to reduce the generator's deviation from the ideal 50 percent

distribution of bits. They are commonly used among TRNGs because every Noise Generator's raw output tends to generate 0 or 1 with a higher probability. Each Noise generator comes with a bias, meaning that it generates one of the two states (0 and 1) with a higher probability. In sensible applications this bias has to be the ideal value of 0. For this to become true, the Randomness Extractor intervenes.

For the presented solutions, the Von Neumann randomness extractor was chosen, since it is the most common in the cryptographic field. This function works with pairs of two outputted bits, discarding every pair where the bits are the same (00 and 11) and outputting the first bit of the other pairs. This principle described by Von Neumann stated that if we play a heads or tails game with a biased coin, the probability of tossing first heads and second tails is the same as tossing first tails and second heads.

It is a good practice for testing each embedded device that generates Random Sequences. The role of the Randomness Testing is to determine if the sequence that is to be tested has some kind of mathematical pattern

3.4. Entropy Extraction technique

It is well known that random sources exist due to fabrication imperfections, operating temperature, the stability of the power supply, ambient thermal noise or vibration, etc. The hardest thing is extracting it[16]. Manipulating a phenomenon that lasts for 20 ns (as the jitter of an average ring oscillator)isn't that easy as it looks like.

Manipulating this king of phenomenon can imply different techniques for each source. For Ring oscillators we can use the following techniques:

- Counter Technique (Fig 2.) – is based on the measurement of the jitter's period. Usually the least significant bit of the result is used as the output of the generator.

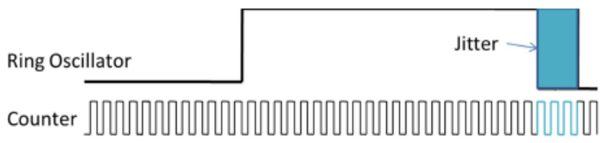

Fig 2. Counter Technique

A counter is used to count the number of temporary oscillations. The resulting bit of the generator is the least significant bit of the counter.

- Desynchronization Technique (Fig 3.) – is based on running a large number of ring oscillators that run on different frequencies. The output of the generator is the mod 2 sum of each ring oscillator output. This method implies a sample clock frequency that decides when to check the states of the oscillators.

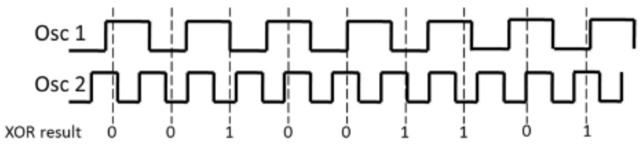

Fig 3. Desynchronization Technique

4. BACKGROUND AND MOTIVATION

This section briefly describes the basic BFD-TRNG model .The BFD-TRNG circuit [17] is a fully digital TRNG, which relies on jitter extraction by the BFD mechanism, originally implemented as a 65-nm CMOS ASIC. The structure and working of the (single phase) BFD-TRNG can be summarized as follows, in conjunction with Fig 4.

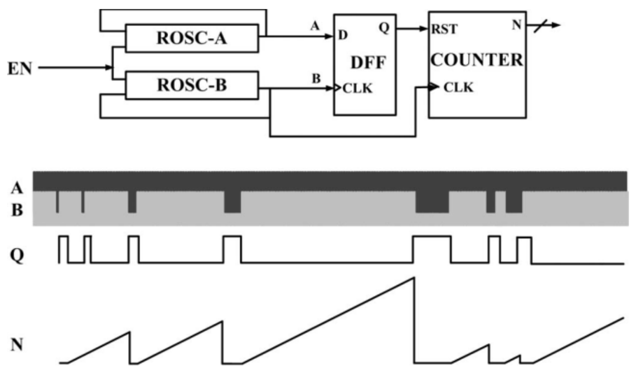

Fig 4. Architecture of the single-phase BFD-TRNG .

- The circuit consists of two quasi-identical ring oscillators (ROSC*A* and ROSC*B*), with similar design and placement. Due to built in physical randomness originating from *process variation* effects associated with deep sub micrometer CMOS manufacturing, one of the oscillators (e.g., ROSC*A*) oscillates slightly faster than the other oscillator (ROSC*B*).The authors [17] proposed to use trimming capacitors to further tune the oscillator output frequencies.
- The output of one of the ROs is used to sample the output of the other, using a D flip-flop (DFF). Output of ROSC*A* is fed to the *D*-input of the DFF, while the output of ROSC*B* is applied to the clock input of the DFF.
- At certain intervals (determined by the frequency difference of the two ROCs), the faster oscillator signal passes, catches up, and overtakes the slower signal in phase. Due to random jitter, these capturing events happen at random intervals, called "beat frequency intervals." As a result, the DFF outputs a logic-1 at different random instances.
- A counter controlled by the DFF increments during the beat frequency intervals and gets reset due to the logic-1output of the DFF. Due to the random jitter, the free running counter output inceases to different peak values in each of the count-up intervals before reset.
- The output of the counter is sampled by a sampling clock before it reaches its maximum value.
- The sampled response is then arrange to obtain the random bit stream.

One shortcoming of the previous BFD-TRNG circuit is that its statistical randomness depend on the design quality of the ring oscillators. Any design bias in the ring oscillators might seriously affect the statistical randomness of the bit stream generated. Designs with the same number of inverters but different placements resulted in varying counter maximas. Additionally, the same ring-oscillator-based BFD-TRNG implemented on different FPGAs of the same family shows different counter maxima. Unfortunately, since the ring oscillators are free-running, it is difficult to control them to eliminate any design bias. The problem is exacerbated in FPGAs, where it is often difficult to control design bias because of the lack of fine-grained designer control on routing in the FPGA design fabric.

5. PROPOSED WORK

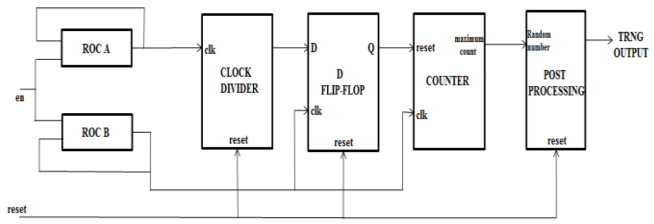

Fig 5. Overall architecture of the proposed TRNG

Fig 5 gives overall architecture of the proposed work. In order to eliminates the design bias generated by the ring oscillators clock divider has been introduced before the flip-flop. Clock divider is capable of fasten the oscillations of the one of the oscillator. Therefore the frequency of oscillations of ROC A has been made faster than ROC B. This capability provides the design good flexibility than the ring-oscillator-based BFD-TRNG. The difference in the frequencies of the two generated clock signals is captured using a D Flip-flop.

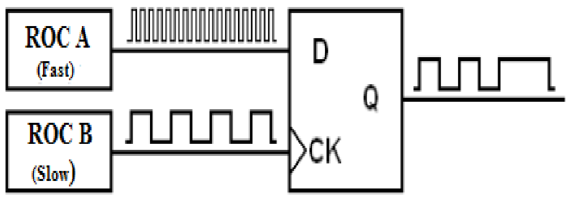

Fig 6. Behaviour of oscillations after clock divider

The D Flip-flop sets when the faster oscillator completes one cycle more than the slower one (at the beat frequency interval). A counter is driven by one of the generated clock signals and is reset when the D Flip-flop is set. Effectively, the counter increases the throughput of the generated random numbers. The last three LSBs of the maximum count values reached by the count were found to show good randomness properties.

Additionally, we have a simple post processing unit using a *Von Neumann corrector* (VNC) to eliminate any biasing in the generated random bits. VNC is a well-known low overhead scheme to eliminate bias from a random bit stream. Post-processing is needed to overcome any weakness in the entropy source that could result in the production of non-random bit streams. In this scheme, any input bit "00" or "11" pattern is eliminated; otherwise, if the input bit pattern is "01" or "10," only the first bit is retained. The last three LSBs of the generated random number are passed through the VNC. The VNC improves the statistical qualities at the cost of slight decrease in throughput.

6. RESULTS & DISCUSSION

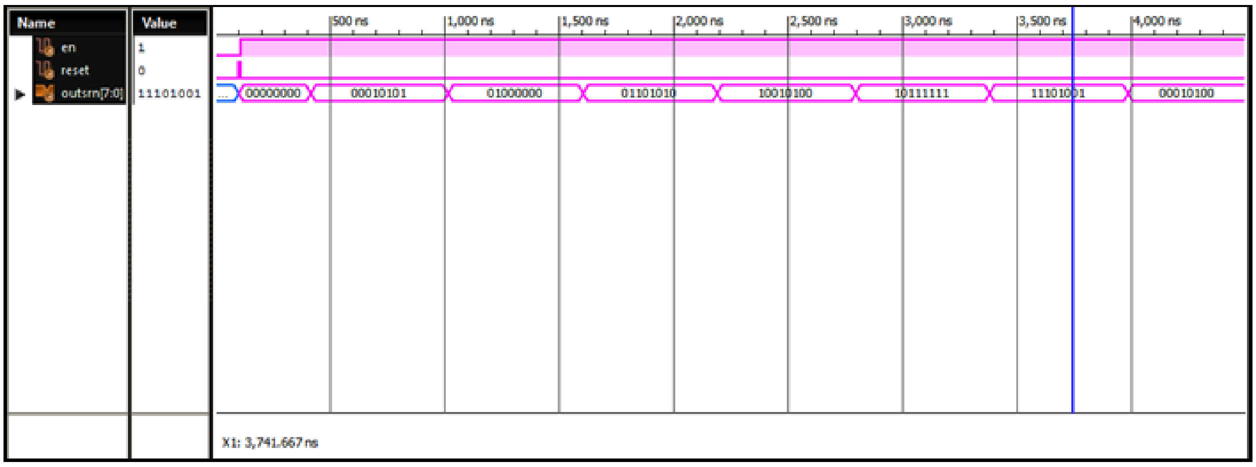

Fig 7. 8 bit TRNG

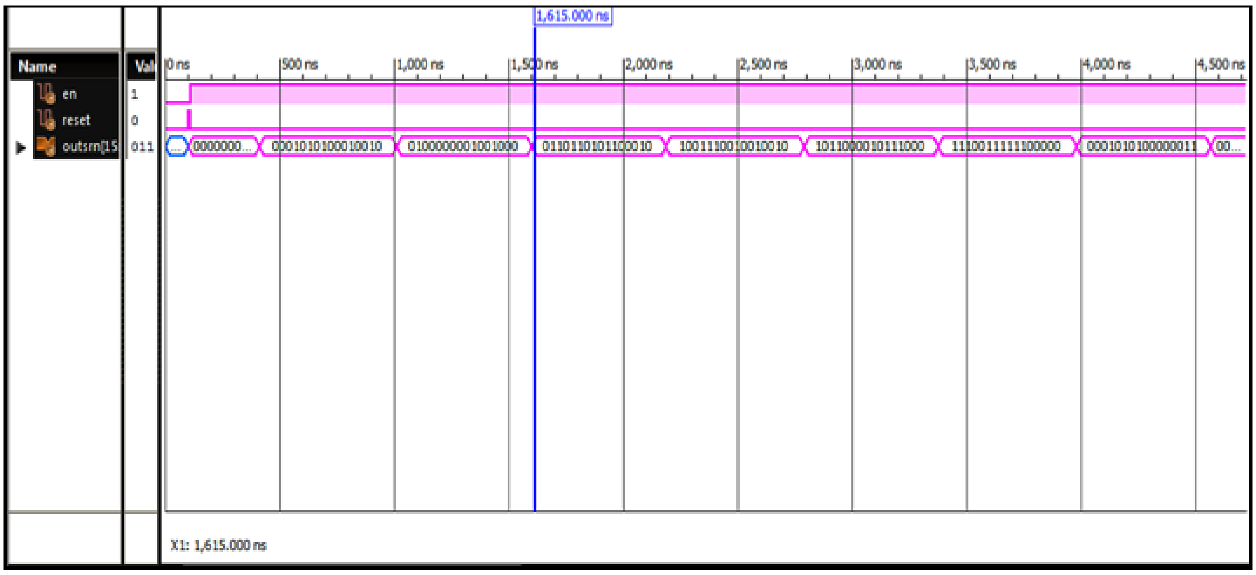

Fig 8. 16 bit TRNG

The proposed circuit is designed using VHLD (Very high speed integrated circuit Hardware Description Language) and implemented using Xilinx ISE (V 13.4) CA software application targeting Xilinx Spartan-6 FPGA platform.

Fig 7 shows 8 bit TRNG and fig 8 shows 16 bit TRNG generated. It can be seen from two generated TRNGs the relationship between the first random number with the next cannot be determined.

For Example:

Table 1 and table 2 shows that difference between the two numbers and the probability of 1' and 0's are unpredictable and cannot be determined.

Table 1. 8bit TRNG

Number	Difference	count 1's	count 0's
0	0	0	8
21	21	3	5
64	43	1	7
106	42	4	4
148	42	3	5
191	43	7	1
233	42	5	3
20	213	2	5

Table 2. 16 bit TRNG

Number	Difference	count 1's	count 0's
5394	0	5	11
16456	11062	3	13
28002	11546	8	8
40082	12080	7	9
45240	5158	7	9
5379	39861	5	11
14377	8998	6	10
59360	44983	9	7

Additionally the circuit is operating on a frequency of 495.93MHz with delay 2.016 ns.rom the above results , it is evident that the proposed TRNG shows the randomness properties. As the circuit is implemented on CAD software with fully digital circuits, it is having low hardware footprint.

CONCLUSION:

We have presented an improved fully digital TRNG on FPGA-platform application based on the principle of BFD-TRNG and clock jitter in addition to the clock divider to fasten the oscillations of ROC A as compared to ROC B. The jitter from two oscillators frequencies are extracted and random number generated are found out to be undeterministic and cannot be predicted.

REFERENCES:

[1] Vijay Bahadur, David Selvakumar, Vijendran, Sobha.P.M, “ Reconfigurable Side Channel attack resistant True Random Number Generator”, 2016 International Conference on VLSI Systems, Architectures, Technology and Applications (VLSI-SATA).

[2] John Kelsey, B. Schneier, David W Chris Hall, “Cryptanalytic Attacks on Pseudorandom number Generators”, Fifth Intl. Workshop Proc., Fast Software Encryption, Vol. 1372, LNCS, pp. 168-188, 1998.

[3] Markettos, Simon Moore,‘The frequency injection attack on ring oscillator-based true random number generators’. Proc. of 11th Int. Workshop on Cryptographic Hardware and Embedded Systems, pp. 317-331, September 2009.

[4] Bayon, Bossuet, Aubert, Fisher, “Electromagnetic Analysis on Ring Oscillator-Based True Random Generators”, IEEE Intl. Symp. On Circuits and Systems, pp. 19 54-1957, 2013.

[5] B.Sunar, W.J. Martin, Stinson D.R, “A Provably Secure True Random Number Generator with Built-in Tolerance to Active Attacks”, IEEE Trans. on Computers, pp. 109-119, 2007

[6] TarsaIonut Gabriel, Gigi-Daniel Budariu, ConstantinGrozea: “Study on a True Random Number Generator design for FPGA”, 8th Intl. Conf. on Communication, pp. 461-646, 2010

[7] Dries Schellekens, Bart Preneel, Verbauwhede, I, “FPGA Vendor Agnostic True Random Number Generator”, Intl. Conf. on Field Programmable Logic and Applicatons, pp. 1-6, 2006.

[8] Kohlbrenner, Paul, and Kris Gaj. "An embedded true random number generator for FPGAs." Proceedings of the 2004 ACM/SIGDA 12th international symposium on Field programmable gate arrays. ACM, 2004.

[9] Schellekens, Dries, Bart Preneel, and Ingrid Verbauwhede. "FPGA vendor agnostic true random number generator." Field Programmable Logic and Applications, 2006. FPL'06. International Conference on. IEEE, 2006.

[10] Marghescu, A.; Maimut, D.-S.; Teşeleanu, G.; Neacşa, T.; Svasta, P., "Adapting a ring oscillator-based true random number generator for Zynq system on chip embedded platform," in Design and Technology in Electronic Packaging (SIITME), 2014 IEEE 20th International Symposium for , vol., no., pp.197-202, 23-26 Oct. 2014;

[11] Amaki, Takehiko, Hashimoto, Masanori, Onoye, Takao, “An oscillator based true random number generator with jitter amplifier”, InternationalSymposium on Circuits and Systems (ISCAS), 2011.

[12] Craig S. Petrie and J. Alvin Connelly, “Modelling and simulation of oscillator based random number generators”, ISCAS ’96., IEEE International Symposium on Circuits and Systems, 1996.

[13] Wei Chen, Wenyi Che, Zhongyu Bi, Jing Wang, Na Yan, Xi Tan, Junyu Wang, Hao Min, Jie Tan, “A 1.04 _W Truly Random Number Generator for Gen2 RFID tag, Solid-State Circuits” Conference, 2009. A-SSCC 2009.

[14] Balachandran, G.K., Barnett, R.E., “A 440-nA True Random Number Generator for Passive RFID Tags”, IEEE Transactions on Circuits and Systems I: Regular Papers, 2008.

[15] Varchola, Michal, and Milos Drutarovsky. "New high entropy element for FPGA based true random number generators." *Cryptographic Hardware and Embedded Systems, CHES 2010*. Springer Berlin Heidelberg, 2010. 351-365;

[16] Andrei Marghescu, Paul Svasta, “Into Generating True Random Numbers - a Practical Approach using FPGA”, 2015 IEEE 21st International Symposium for Design and Technology in Electronic Packaging (SIITME)

[17] Q. Tang, B. Kim, Y. Lao, K. K. Parhi, and C. H. Kim, “True random number generator circuits based on single- and multi-phase beat frequency detection,” in Proc. IEEE Custom Integr. Circuits Conf., Sep. 2014, pp. 1–4.

[18] J. Von Neumann, “Various techniques used in connection with random digits,” Nat. Bureau Standards Appl. Math. Ser., vol. 12, pp. 36–38, 1951.

[19] Andrei Marghescu, Paul Svasta2, Emil Simion, “Optimising Ring Oscillator-based True Random Number GeneratorsConcept on FPGA”, IEEE, 2016