March - April 2020
ISSN: 0193-4120 Page No. 3996 - 4007

Efficient Implementation of Digital Standard Cells-Based True Random Number Generator for Securing FPGA Designs

Guard Kanda[1], Kwangki Ryoo[*2]

[1]Research Scholar, Department of Information and Communication Engineering, Hanbat National University, Daejeon, 34158, South Korea.

[*2]Professor, Department of Information and Communication Engineering, Hanbat National University, Daejeon, 34158, South Korea.

guardkanda@gmail.com[1], kkryoo@gmail.com[*2]

Article Info
Volume 83
Page Number: 3996 - 4007
Publication Issue:
March - April 2020

Article History
Article Received*: 24 July 2019*
Revised*: 12 September 2019*
Accepted*: 15 February 2020*
Publication*: 26 March 2020*

Abstract

Random numbers are fundamental resources in the field of computing and engineering. They have a wide scope of application including cryptography and simulation. True Random Number Generators (TRNGs) are considered to be the most secure based on the quality of its entropy sources. With the availability of several sources of entropy, ring oscillator architecture can easily be used as a quality source of entropy for a TRNG due to the inherent jitters and the simplicity of its design. Since there is still a possibility of a generator to generate random numbers that do not meet the required security metrics, hence, it is imperative for a TRNG to be able to quickly regenerate another set of random bit sequence. For these reasons, this research, proposes a high-speed array-sampling and a post-processing unit ring oscillator-based TRNG with improved statistical measures and throughput for securing FPGA devices. The core architecture consists of digital primitive cells – the ring oscillator, Q-Flip Flop, and the CubeHash algorithm. These are used as the building blocks for constructing the proposed TRNG architecture. This proposed hardware architecture reveals an improvement in throughput and the statistical measure of the quality of generated bits. The architecture was modeled and simulated using Verilog HDL, Modelsim SE, and Xilinx's ISim simulation tools. This architecture was designed using both the Xilinx ISE and Vivado tools. The proposed design was implemented on the Spartan 6 and Cyclone IV FPGA devices and occupies an area of 3287 LUTs and 1714 Slice registers and had a maximum throughput of 1422 Mbps. Sampled bitstreams' statistical accuracies were ascertained using NIST's statistical Test package program.

Keywords: *TRNG, Ring Oscillator, Cryptography, FPGA, CubeHash, Hardware Security.*

1. Introduction

Internet-of-things (IoT), has gradually developed into what is arguably the largest technological platform having huge potential benefits. It has woven itself into the very fiber of our everyday life. This is the case owed to the varied computational capabilities and sizes they exhibit. They span a range of passively powered wearable health-care monitoring devices to powerful edge devices or nodes. These devices or sensors can be located in homes, cars, farms, factories, laboratories, and hospitals to increase productivity and results. These potential benefits they offer have shown that an estimated 50 billion connected devices are expected by the year 2020 [1] These devices such as the ones used in the hospital to monitor a patient's vital organs, usually possess and process a large amount of data that is highly sensitive and confidential. These make the IoT platforms a

March - April 2020
ISSN: 0193-4120 Page No. 3996 - 4007

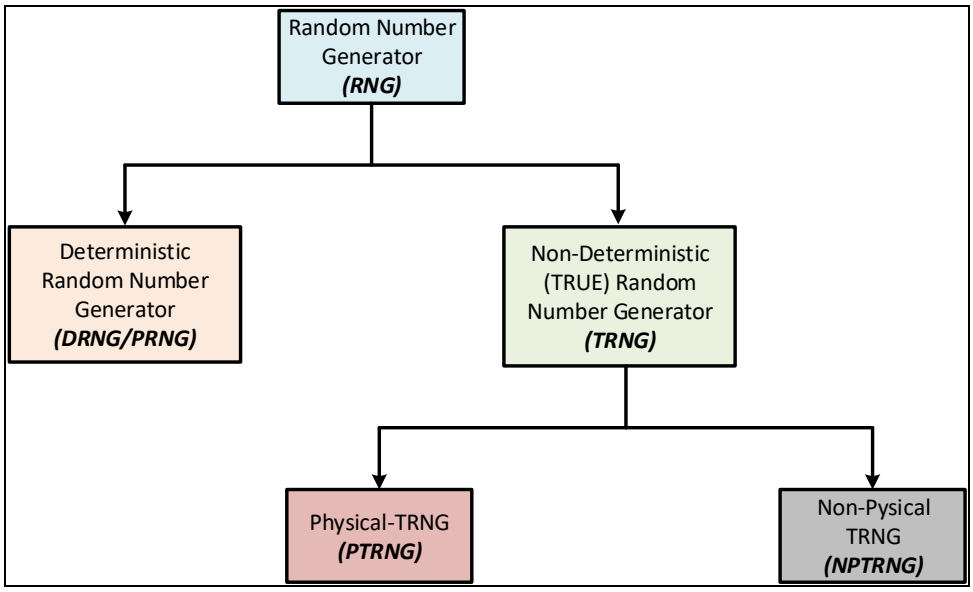

Figure 1. Taxonomy of Random Number Generator

breeding ground for adversaries and attackers to ply their trade. The information-rich data they possessed by these IoT nodes and sensor have necessitated that the security and privacy of these platforms must be treated with a great deal of attention to mitigate majority, if not all, of the current and emerging security attacks such as the IoT based distributed denial-of-service (DDoS) attacks [2].

A means by which the security and privacy of the IoT platform can be preserved is through cryptography. where messages or information is encoded (locked) with a key (ephemeral keys, session keys, signatures) and is only decoded (unlocked) by the intended recipient who has access to this key. These keys can be generated by running a Random Number Generator (RNG). From Fig. 1, there are at least two main types of the RNG: Pseudo-Random Number Generator (PRNG) and the True Random Number Generator (TRNG). PRNG, also known as Deterministic Random Number Generator (DRNG) rely on complex algorithms or mathematical procedures alongside a seed (an initial value) to generate random bit sequences. These bit sequences of the PRNG are completely deterministic and hence when one uses a "weak" seed for generation, then the amount of time taken for a bit sequence to be regenerated is shorter and undesirable. On the other hand, TRNG uses purely random and non-deterministic electronic effects as the source of its randomness as compared to the algorithmic-based PRNG. The sources of randomness (entropy) for the Physical-TRNG (PTRNG) include thermal noise from semiconductors, metastability (Quantum mechanics), Timing Jitters, and Chaos circuits whereas that for the Non-Physical-TRNG (NPTRNG) can include system time, RAM contents, keyboard loggers, etc. This implies that the NPTRNGs are software-based whereas the PTRNGs are hardware-based.

A ring oscillator (RO) is a chain connection of an odd number of inverter gates in series and having their final output fed back into the input of the first inverter gate [3]. This setup causes the output of the inverter ring to oscillate between the two voltage levels of high and low [4]. These ring oscillators are extensively used in hardware and electronics design because of the simplicity of their structure, ease of design and the low cost of implementation associated with it. They are desired because they present a simpler and

effective method of building TRNGs [5-8]. The entropy of any state or message as represented in Equation (1), measures the average amount of information needed to represent an even taken from a probability distribution for a random variable.[9]. From Equation (1), p_r represents the rth state or message out of a total of t possible states or messages. The optional value K can be evaluated to the value $^1/_{\log(2)}$. A random number generator that generates K-bits of binary sequences has the probability that, an output will equal r to be p_r, As required by every RNG, the source of entropy in the case of a ring oscillator is the presence of jitters [10]. According to [11], Due to the many storage elements arising from the multiple stages of inverters in a ring oscillator coupled with the delay component of each stage, which depends on the capacitance (stray and junction) and carrier transit times, absolute frequency stability is therefore not guaranteed.

$$H = -K\sum_{r=i}^{t} p_r log p_r \quad (1)$$

The applicability of random numbers ranges from the field of arts to cryptography and these random numbers generators (RNGs) are critical components of a cryptographic ecosystem. Great cryptography requires quality random numbers. The random number generator for any cryptographic application should seem to adversaries as close to perfect RNG. It is therefore crucial for a cryptographic application to generate PRBS which cannot be predicted even by the toughest adversaries. Generally, such quality true random number generators have a generic architecture as shown in Fig. 2. The component blocks of the generic architecture include the entropy source, the harvesting technique, the post-processing block, the total failure test, and the online tests. The source of entropy is the critical component of the architecture because this is where the "the randomness" is generated. As mentioned earlier, several sources of entropy exist which determines the class of random number being generated. The time-continuous signals obtained from the entropy sources are harvested (digitized) for form what is termed digitized analog signals. Since some of these entropy sources have some form of bias, the post-processing stage is implemented to reduce or eventually eliminate some of these weaknesses that may be present.

This paper, therefore, presents an efficient TRNG architecture based on the generic TRNG architecture, by using a vector array-sampling approach to accumulate and harvest the jitters that are inherent in the ring oscillator and then finally using the CubeHash hashing algorithm for postprocessing. This approach promises results that pass the statistical test as well as having great throughput. The remaining portions of this article are organized as follows: From Section 2, a brief review of some related works. The TRNG hardware architecture is discussed in the 3rd Section. In Section 4, the proposed architecture's statistical quality matrices, as well as result analysis, are covered. Recommendations regarding the trade-off between hardware area and throughput are discussed briefly in Section 5. Finally, conclusion and future work are discussed in Section 6 of the paper.

2. Related Work

Numerous random number generators have been designed and proposed based on either pure digital electronics or a mixture of digital and analog electronics. A typical example is that proposed by [12]. This design relied on a blend of both the analog and digital electronics components to amplify and sample white noise. The main drawback of this design was related to the stage of amplification. This stage consumed more power to be able to raise the level of noise a few orders of

March - April 2020
ISSN: 0193-4120 Page No. 3996 - 4007

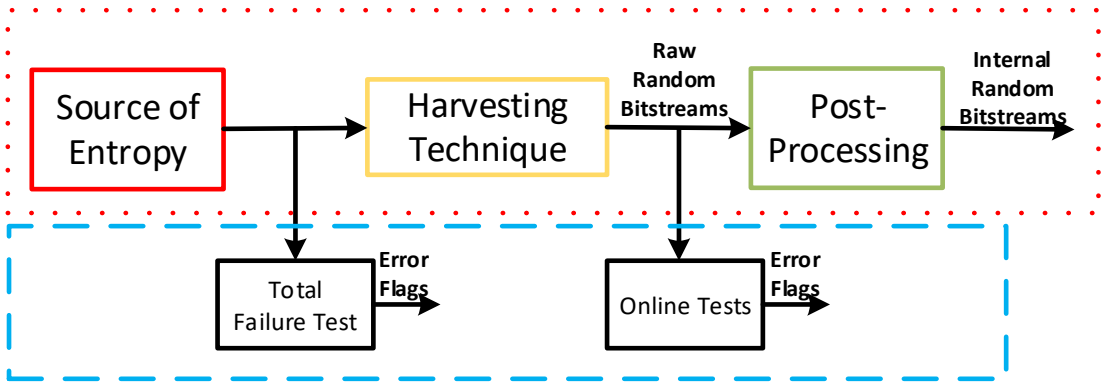

Figure 2. TRNG Generic Architecture

magnitude in order to allow digitization. The Intel random number generator [13] shown in Fig. 3 implemented a similar concept whereby the Johnson noise (thermal noise) from resistors is amplified in order to drive a voltage-controlled oscillator that is in turn sampled by a high-speed oscillator and then post-processed digitally with the von Neumann algorithm. A purely digital electronics-based architecture was proposed by [14]. In this architecture, the outputs of Linear feedback shift registers (LFSR) and cellular automata are randomly sampled to obtain and measure the randomness that is associated with the jitters in the ring oscillators. For this architecture, due to the complexity of the harvesting scheme implemented, the harvested samples were difficult to verify although this proposed architecture had no amplification stage or step and the source of entropy was obviously limited to the two ring oscillators implemented. Not all, a simple architecture was proposed by [15] which was based on metastability of circuits but had the disadvantage of combining a large number of such circuit in order to pass statistical tests. Also, two novel hardware random number generator architectures were presented by [16] that also relied on the metastability of latches. The first of these was the RNG with the capability of nullifying direct current (DC) for the operations that require extremely low power consumption. In addition to the DC-nulling RNG, a finite impulse response (FIR)-based RNG that implements the predictive whitening filter to be able to separate non-random components from the generated bits sequences was proposed. The RO-based TRNG was proposed by [6]. This architecture relied on a design that was similar in several ways to the Linear Feedback Shift Generator (LFSR) in which the registers are replaced with inverters. The positions of the feedbacks are labeled with switch values making them open if 0 or low and closed if 1 or high. The authors also proposed the use of self-controlled LFSR as the post-processing unit.

3. Proposed Architecture

As depicted in Fig. 2, the proposed TRNG comprise of the three key components that are required for a TRNG. Because this is a purely digital TRNG, we do not use amplifiers to enable the harvesting or sampling of the randomness. The various components are presented in detail in the sub-sections that follow. Fig. 4 shows the block diagram of the proposed TRNG. The block diagram consists of 4 multi-mode ring oscillator architectures as the main source of entropy. These individual ring oscillators are XORed to form a single source of entropy signal which is sampled by an array of sampling units. The resulting bitstreams from the multisampling are then passed through a cryptographic hash algorithm for post-processing to increase the rate as well as the quality of bits generated.

March - April 2020
ISSN: 0193-4120 Page No. 3996 - 4007

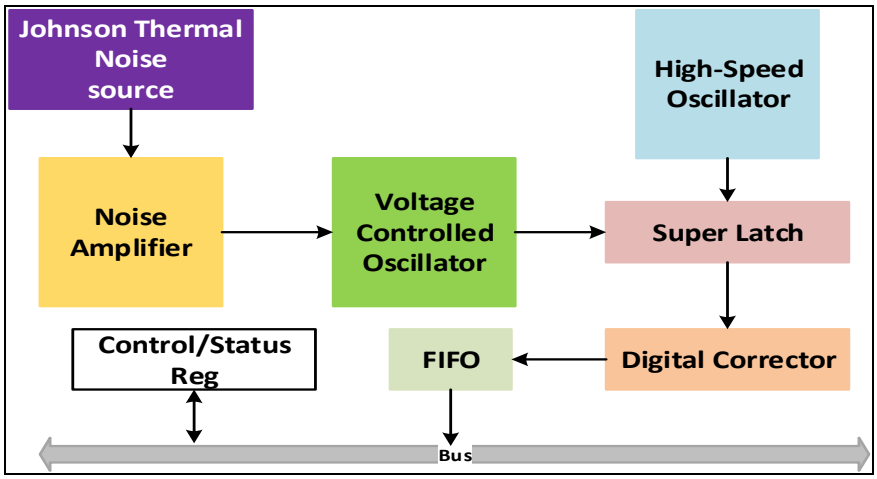

Figure 3. The Architecture of Intel's Random Number Generator

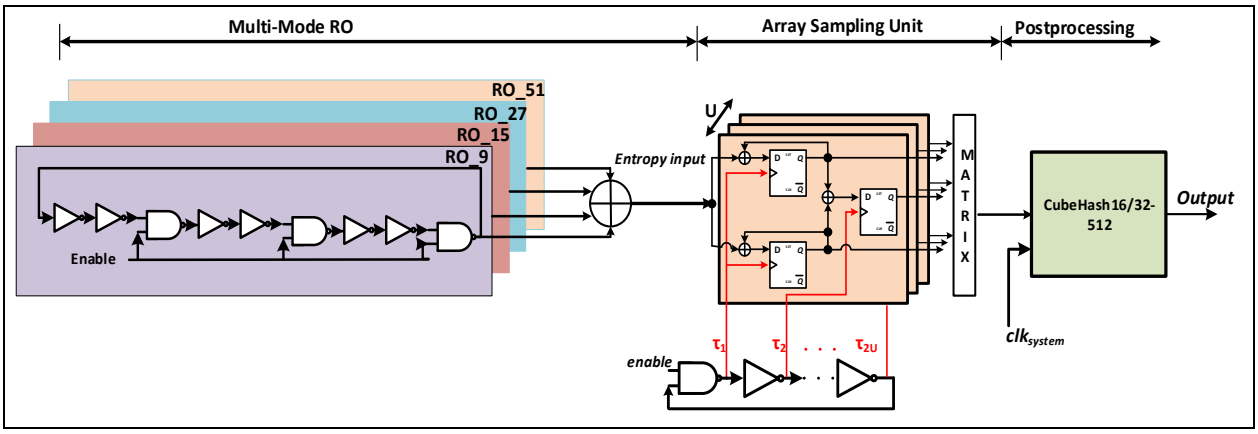

Figure 4. Proposed TRNG Architecture

3.1. Multi-Edge Ring Oscillator

The source of entropy for the proposed architecture is the Ring Oscillator (RO). The ring oscillator is employed due to the inherent jitters that are present and accumulate as a result of their operation. Conventional ring oscillators have a single NAND gate and an even number of inverter gates that pulses a single edge signal to propagate through the ring oscillator. However, the architecture proposed by [17] showed a 3-input node ring oscillator that pulses three different edges for a single ring oscillator simultaneously with each edge propagating through the chain as in a conventional ring oscillator architecture. The three edges are 120 degrees phase-shifted and boost the resulting frequency by a factor of three (3). The accumulated jitters cause the pulse width between two neighboring edges to increase in variation with each completed cycle. With time the two neighboring edges will merge into a single edge, taking the multi-edge ring oscillator into a single edge ring oscillator. The time taken to collapse is the time taken to accumulate the jitters that are sampled to generate random numbers. The entropy source implemented includes 4 sets of ring oscillator chains in the multimode and having different stages for each ring oscillator chain. The NAND gate replaces an inverter to allow for easy control of the ring oscillator. When a value high is present at the enable input, 3 pulse edges are

propagated simultaneously through the ring oscillator chain. Visibly, the multi-mode ring oscillator can be regarded as shorter individual ring oscillators brought together to form a longer ring oscillator. For the ring oscillator shown in Fig. 5 (b), the 9-stage ring oscillator is 3 separate ring oscillators of length 3-each- a NAND gate and 2 inverter gates. The 4 sets of ring oscillators designed for this research begins with one with 9 stages. The subsequent ring oscillators are 2 times the number of stages of the preceding ring oscillator less three stages. Therefore, the next ring oscillator is of length (2x9) – 3 = 15. The second ring oscillator is then broken into 3 stages consisting of 4-inverters and a NAND gate to generate one of the three edges. The remaining two ring oscillators are of length 27 and 51 based on the same computation. Because the architecture in Fig. 5. is not easily simulated, Fig. 6 shows the operation of the architecture on a Xilinx's Spartan 6 FPGA board using the ChipScope internal logic analyzer tool to show the output after the sampler array unit.

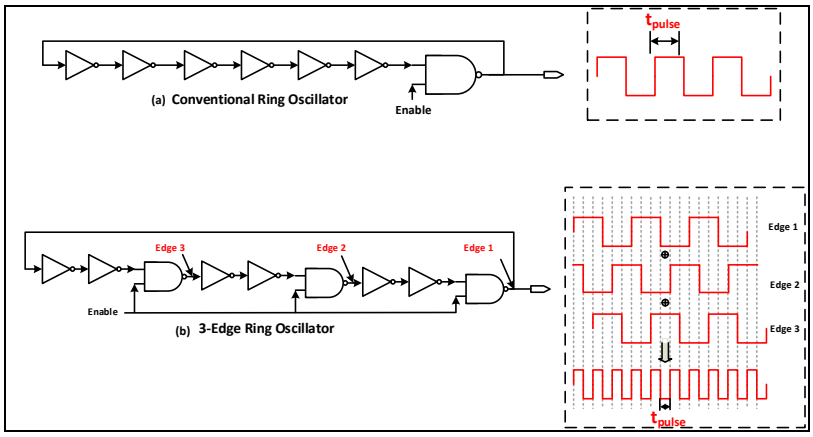

Figure 5. The Architecture of Intel's Random Number Generator

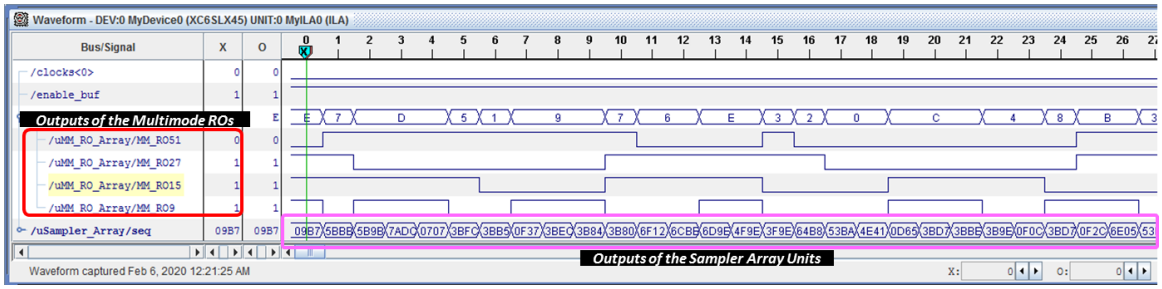

Figure 6. Proposed TRNG Architecture

3.2. Array-Sampling Unit

To increase the rate at which random bits are generated in the proposed architecture, a simple array-sampling unit is proposed. The array-sampler, shown in Fig. 7, is a simple architecture consisting of three registers (Flip-Flops) and then a simple clock generator. Each sampling unit is controlled by three clocks from the K phase clock signals within the array-sampling unit. A total of *3U* registers (Flip-Flops) - where U is the number of sampling-unit in the array - sample the input data from the XOR concatenated signal of the

March - April 2020
ISSN: 0193-4120 Page No. 3996 - 4007

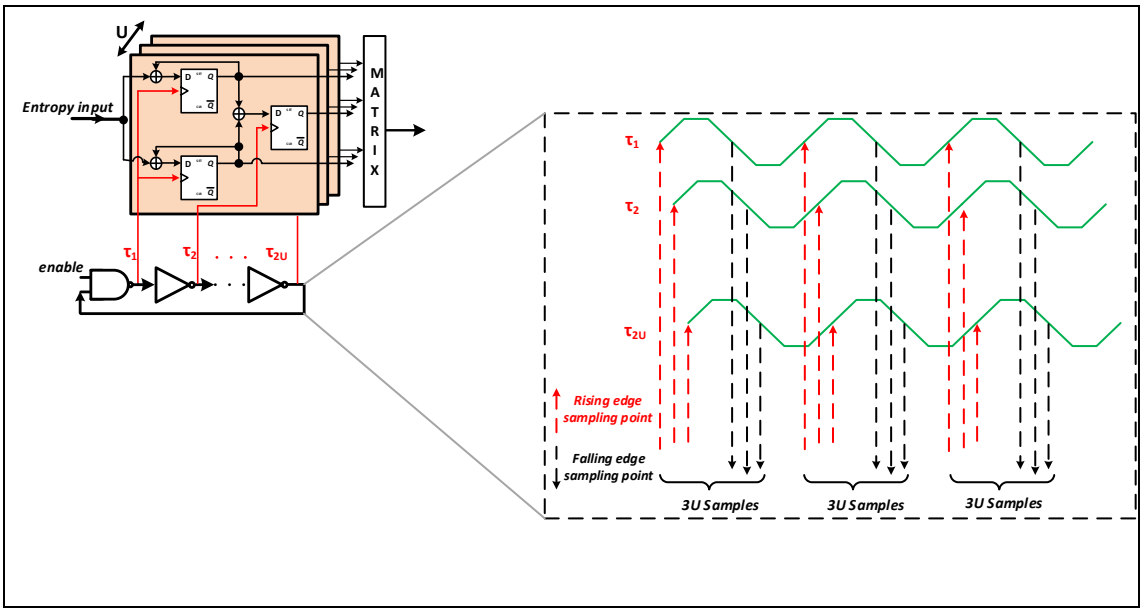

Figure 7. Array Sampling Architecture and Timing Diagram

multi-mode ring oscillator at both edges of the K-phase clock generator. Due to the very short period between the clocks of the clock generator – τ_1, τ_2, τ_3 – the probability of the data signal being sampled at the threshold voltage increases and this introduces a meta-stable state which in turn increases the entropy of the TRNG. The presence of this meta-stable condition implies that the number of ring oscillators used in the entropy source stage of the TRNG can be reduced [18].

3.3. Post-Processing Unit

The postprocessing unit implemented for this TRNG architecture is the CubeHash function. To ensure that the bitstreams from the multi-mode ring oscillator have good statistical randomness and a high bit rate, the sampled outputs of the multi-mode ring oscillator can be fed to a cryptographically secured hash function. The CubeHash is a collection of hash functions proposed and designed by Daniel J. Bernstein [19]. This set of hash functions was one of NIST's SHA-3 competition candidates eliminated in the second round but is yet to be broken [20]. A key advantage of this algorithm is its simplicity. This hash algorithm uses a uniform structure for processing message digests of lengths of up to 512 bits, using tweakable number of rounds and message block sizes. Six parameters namely parameters *i, f h, r, b,* and *m* specify the exact tweak or setup of the CubeHash algorithm. The *i*-parameter specifies the number of rounds of the compression function to be executed to obtain the initialization vector. This parameter spans the range of 1 up to ∞ but it is typically 16 whereas the parameter f denotes the number of rounds to be computed for the final block of message to be processed. This value is typically 32 but ranges from 1 to ∞. The h determines the width of the message digest in bits and ranges from 8-bits to 512-bits in multiples of 8-bits and is typically 512-bits. The r determines the number of rounds compressions to be performed per message block. The r ranges from 1 to ∞. Not all, the parameter b determines the number of bytes per block message. Finally, m is the parameter that denotes the length of the message that can be processed and it is a string of bits between 0 to $(2^{128} - 1)$ bits. Generally, the CubeHash notation is written as CubeHash*i+r/b+f-h(m)* to describe a specific variant of the algorithm.

March - April 2020
ISSN: 0193-4120 Page No. 3996 - 4007

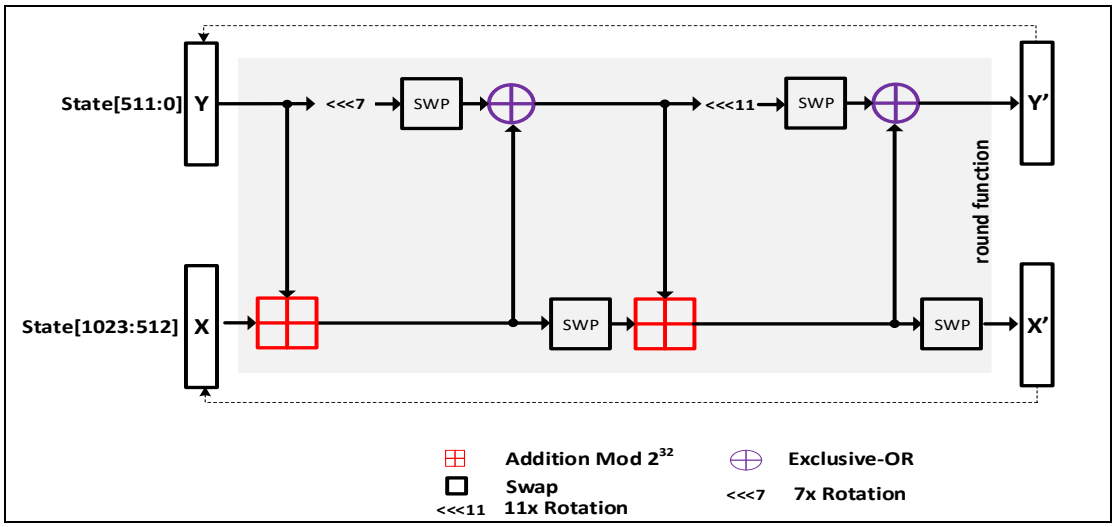

Figure 8. CubeHash Compression Function

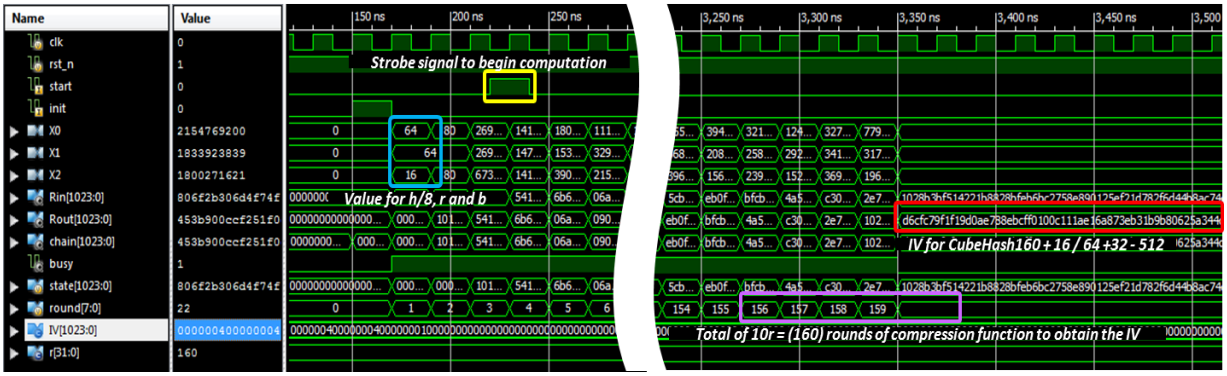

Figure 9. Simulation of the Computation of the Initialization Vector

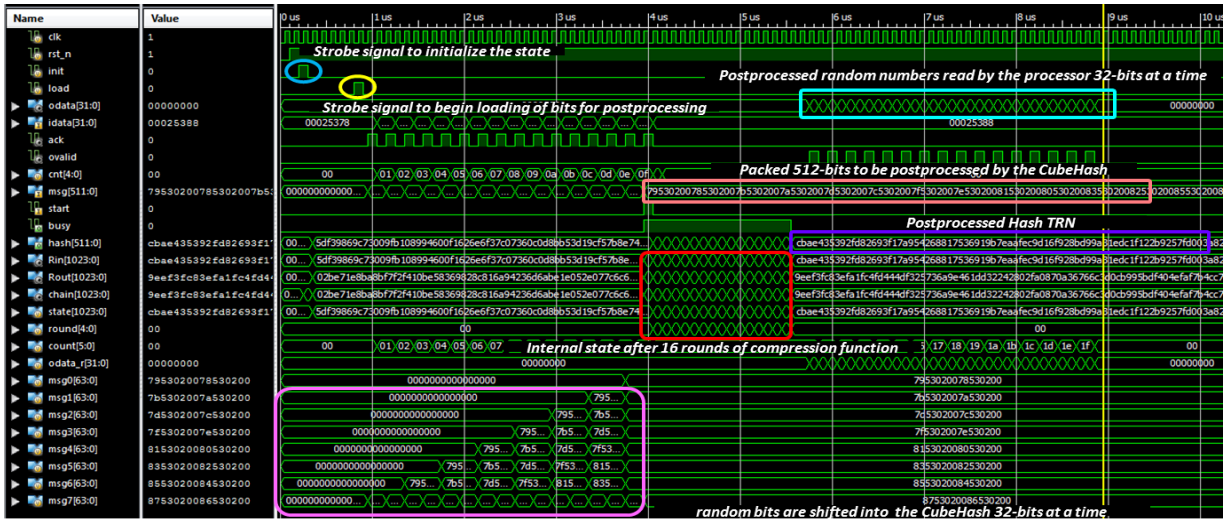

Figure 10. Simulation of the Computation of a Hash of a Message

The variant of CubeHash implemented in this research is the *CubeHash160+16/64+32-512*.To initialize the internal state of the CubeHash, 10r rounds are performed. Afterward, the first block of data to be hashed, b-byte in size, is XORed into the first b-bytes of the internal state. In the end, r-round(s) of the compression function is performed to obtain the message hash. The width of the internal state is 1024-bits and this forms the core of the algorithm's implementation. The round compression function, shown in Fig. 8, operates on the 1024-bit internal state organized as 32 long words, each being 32-bits wide. The State is divided into two halves, each of size 512 bits and labeled as X and Y. This division is performed because the compression function only performs 10 simple operations on half of the internal state which is (512-bits) during each of the 10 compression rounds. At the end of each compression round the outputs *X'* and *Y'* are obtained from their respective *X* and *Y* halves. The *X'* and *Y'* outputs are fed back to *X* and *Y* if multiple rounds of the compression are required. The compression function consists of 2 addition modulo 2^{32} , 2 XOR operations, 2 rotation operations and 4 swapping operations as shown in Fig. 8. The precomputed initialization vector (IV) is *0xd6cfc79f1f19d0ae788ebcff0100c111ae16a873eb31b9b80625a344d07f2fe269f7245d7aaa6f126fd54233a7447386c59e65d1c33c6ecb09b82faea211166f8fc1addc5343afe5ee724b803565179e24f7ff604687a9b653e0c307b06405f8623e77acf75b428f14c22fe6290a39c63e581e2dfd52b75937eb14d8522588b3*. Fig. 9 is the simulation for the initialization vector and shows the setting or tweaking values used for the first three register x0, x1, x2. For test purposes, a randomly generated message of length 512-bit long; *0x79530200785302007b5302007a5302007d5302007c5302007f5302007e5302008153020080535302008253020085530200845302008753020086530200* is passed through the CubeHash, using the initialization vector as the one shown in Fig. 10. The resulting hash that is generated for this message equals to *0x6982fd925343aecb5317882654a9173f169decafeab719691fdc1ea399bd28f9823a00fd57922b126f3cf8fa40fa58f54126955750322deee9fa2443336b31a0*. Fig. 11, on the other hand, shows the internal logic capture of the postprocessing using in operation on the Spartan 6 test board. The logic analyzer shows the postprocessing of the first 512-bit generated by the proposed entropy and sampler array units. The first 512-bits generated is shown in the red rectangle in Fig. 11 in little-endian format.

4. Discussion of Results

The proposed architecture was implemented using Verilog HDL with Xilinx's ISE and Vivado Tools. The Modelsim SE 10.6d and Xilinx's ISIM were also used for the functional and timing simulations of the proposed architecture. The design was again tested on Altera's DE2-115 Cyclone IV board. A total of 1GiB of True random number samples were generated continuously at 50MHz and 25 MHz clock frequencies. A simple architecture comprising of a MicroBlaze microcontroller system fitted with a UART module is used to write the generated samples to the PC for analysis and examination of their statistical properties using the statistical test suite by NIST- NIST's SP 800-22 [21]. Results from this test are tabulated in Table. 1 and shows the p-value alongside the success rate of the generated samples. The results prove the right operation of the proposed architectures and its suitability for use in other systems that require the use of true random numbers. The hardware overhead for the proposed architecture's implementation on Xilinx's Spartan 6 and Altera's Cyclone IV FPGA devices are recorded in Table 2 and Table 3 respectively. From Equation (2), the maximum throughput of the design is determined to be 1422 Mbps considering that the bits are sampled at 512 bits into the CubeHash core using a total of 18 clock cycles and a minimum of 553 Mbps if the bits are sampled into the CubeHash core 32-bits at a time, which requires a total of 48 clock cycles.

March - April 2020
ISSN: 0193-4120 Page No. 3996 - 4007

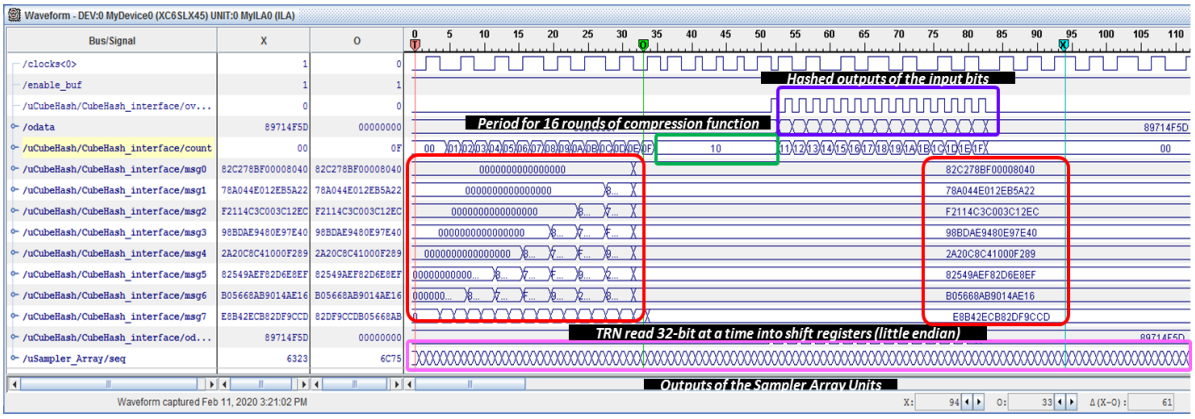

Figure 11. Simulation of the Computation of a Hash of a Message

Table 1: Results Comparison Between Designs

NIST Test Package	CubeHash @ 50 MHz	
	P-Value	Success Rate
Frequency	0.9005	98/100
Block Frequency	0.6961	99/100
Cumulative Sums (Forward)	0.1216	99/100
Cumulative Sums (Reverse)	0.2083	98/100
Runs	0.0688	98/100
Longest Run	0.1507	98/100
Rank	0.6937	98/100
FFT	0.2070	98/100
Overlapping	0.6038	99/100
Universal	0.0248	99/100
Approximate Entropy	0.9992	99/100
Serial (m = 16, n= 1024)	0.7599	99/100
Linear Complexity	0.9022	98/100

Table 2: Hardware Results from FPGA Implementation (Spartan 6)

Architecture	Slice Registers		Slice LUTs		Fully utilized LUT-FF pairs		Max Operating Freq (MHz).
	# Used	% Utilization	#Used	% Utilization	#Used	%Utilization	
Multi-Mode RO	107	0	233	0	0	0	-
Multi Array Sampler	10	0	15	0	10	66	812
CubeHash	1035	1	2325	10	1050	33	155
TRNG	***1714***	***3***	***3287***	***17***	***1138***	***29***	***120***

Table 3: Hardware Results from FPGA Implementation (Cyclone IV)

Architecture	Total Logic Elements	Combinational Functions	Dedicated Logic Registers
Multi-Mode RO	6	6	0
Multi-Array Sampler	15	15	15
CubeHash	3960	3444	1591
TRNG	***3029***	***2994***	***1156***

$$\textbf{Throughput} = (Operating\ frequency.x\ Number\ of\ Bits) \,/\, Number\ of\ Clock\ Cycle \quad (2)$$

5. Recommendations

Compared to other light-weight TRNG, the proposed TRNG has a high throughput. This is as a result of the postprocessing unit; CubeHash. The disadvantage of the CubeHash is the area of hardware it occupies. This is, therefore, a tradeoff between throughput and size (hardware area). Several options for the post-processing unit are available. Some of the recommended options for use in place of the CubeHash to reduce the area overhead while decreasing the throughput are the shrink-generator or the linear feedback shift register (LFSR) which are both cryptographic post-processing methods and the Parity filter or the debiasing Von Neumann algorithm [22] which are typical algorithmic post-processing methods

6. Conclusion and Future Work

In the paper, digital standard cells -based TRNG architecture that is able to accumulate jitters to use as a source of randomness has been proposed. This proposed architecture is completely digital as it employs the ring oscillator to exploit the embedded entropy to generate random numbers. The design uses a simple yet effective digital sampler to sample the bits that are generated from the ring oscillator arrays that were implemented. The final sample bits are then fed to the post-processing unit of choice. For this architecture, the CubeHash cryptographic hash function was used to improve upon the raw sampled bits generated for the statistical tests. The design was implemented on the Spartan-6 FPGA device by Xilinx and also on the Cyclone IV DE2-115 Altera board with a maximum throughput of 1422 Mbps. The statistical test suit from NIST shows that the design architecture passes all the tests and of high quality. With the growing adoption of IoT systems in this era, it has become imperative that the security-related aspects of these systems be met by designers. There is an inherent challenge of obtaining a suitable solution to integrate these TRNG into low-end area constrained devices such as edge and sensor nodes. For future work, we will seek to explore various post-processing algorithms or architectures that will drastically reduce the hardware overhead while keeping the required throughput. Not all, the proposed architecture will be included in the design of an ECIES architecture being developed [23] to generate True Random Numbers (TRNG) for the generation of shared keys.

References

[1] Majzoobi M, Kharaya A, Koushanfar F, and Devadas S. Automated design, implementation, and evaluation of arbiter-based PUF on FPGA using programmable delay lines. In IACR Cryptology ePrint Archive. 2014: 639, [cited 2020 Feb 17] Available from: https://eprint.iacr.org/2014/639.pdf

[2] Rührmair U, Sölter J, Sehnke F, Xu X, Mahmoud A, Stoyanova V, Dror G, Schmidhuber J, Burleson W, and Devadas S. PUF modeling attacks on simulated and silicon data, IEEE Transaction on Information. Forensics Security. 2013 Nov;8(11):1876–1891. DOI: 10.1109/TIFS.2013.2279798

[3] CMOS Inverter Ring Oscillators [Internet] [place unknown; publisher Analog Devices]; [updated 2019 Sept 25; cited 2020 Feb 17]. Available from: https://wiki.analog.com/university/courses/alm1k/alm-lab-ring-osc

[4] Ring Oscillators [Internet] [place unknown; publisher T.U. Wien]; [cited 2020 Feb 17]. Available from: http://www.iue.tuwien.ac.at/phd/entner/node35.html

[5] Schulz RA, Random Number Generator Circuit Feb. 1990. [Internet], [place unknown; publisher Google Patent]; [cited 2020 Feb 17]. Available from: https://patents.google.com/patent/US4905176A/en

[6] Golić JDJ, New methods for digital generation and postprocessing of random data. In, IEEE Transactions on Computers, 2006 Oct; 55(10):1217- 1229. DOI: 10.1109/TC.2006.164

[7] Wu J, O'Neill M, Ultra-lightweight true random number generators. In Electronics Letters. 2010

March - April 2020
ISSN: 0193-4120 Page No. 3996 - 4007

Jul 12;46(14):988-990. DOI: 10.1049/el.2010.0893

[8] Güler Ü, Ergün S, Dündar G, A digital IC random number generator with logic gates only. In IEEE International Conference on Electronics, Circuits, and Systems, ICECS. 2010 Dec 12-15; 239- 242. DOI: 10.1109/ICECS.2010.5724498

[9] Shannon, CE, A Mathematical Theory of Communication. The Bell System Technical Journal. 1948 Jul; 27(3):379-423. DOI: 10.1002/j.1538-7305.1948.tb01338.x

[10] Abidi AA, Phase Noise and Jitter in CMOS Ring Oscillators. In IEEE Journal of Solid-State Circuits. 2006 Jul. 24;41(8): 1803-1816. DOI: 10.1109/JSSC.2006.876206

[11] McNeill JA, Jitter in ring oscillators. In IEEE Journal of Solid-State Circuits. 1997 Jun.; 32(6): 870-879. DOI: 10.1109/4.585289

[12] Bagini V, Bucci M, A Design of Reliable True Random Number Generator for Cryptographic Applications. In: Koç ÇK, Paar C. (eds) Cryptographic Hardware and Embedded Systems. CHES 1999. Lecture Notes in Computer Science.,2002 Feb. 08;1717: 204-218. DOI: 10.1007/3-540-48059-5_18

[13] Jun B, Kocher P, The Intel Random Number Generator. Intel Corporation, 1999 Apr.22 1999. [cited 2020 Feb 17] Available from: https://www.rambus.com/wp-content/uploads/2015/08/IntelRNG.pdf

[14] Tkacik TE, A Hardware Random Number Generator. In: Kaliski B.S., Koç.K., Paar C. (eds) Cryptographic Hardware and Embedded Systems - CHES 2002. CHES 2002. Lecture Notes in Computer Science. 2003 Feb. 17;2523: 450-453. DOI: 10.1007/3-540-36400-5_32

[15] Epstein M, Hars L, Krasinski R, Rosner M, Zheng H, Design and Implementation of a True Random Number Generator Based on Digital Circuit Artifacts In: Walter C.D., Koç Ç.K., Paar C. (eds) Cryptographic Hardware and Embedded Systems - CHES 2003. CHES 2003. Lecture Notes in Computer Science. 2003; 2779:152-165. DOI: 10.1007/978-3-540-45238-6_13

[16] Holleman J, Otis B, Bridges S, Mitros A, Diorio C, A 2.92μW Hardware Random Number Generator. In Proceedings of the 32nd European Solid-State Circuits Conference, Montreux, 2006; 134-137. DOI: 10.1109/ESSCIR.2006.307549

[17] Yang K, Fick D, Henry MB, 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 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC). 2014 Mar. 06; 280-281. DOI: 10.1109/ISSCC.2014.6757434

[18] Güler Ü, Ergün S, Dündar G, A digital IC random number generator with logic gates only. In IEEE International Conference on Electronics, Circuits, and Systems, ICECS. 2011 Mar. 07; 239- 242. DOI: 10.1109/ICECS.2010.5724498

[19] Bernstein DJ, CubeHash specification (2.B.1). [cited 2020 Feb 17] Available from http://cubehash.cr.yp.to/submission/spec.pdf

[20] National Institute of Standards and Technology, Announcing request for candidate algorithm nominations for a new cryptographic hash algorithm (SHA-3) family. Federal Register. 2007 Feb. 11; 72 (212): 66212-66220

[21] Bassham LE, Rukhin AL, Soto J, Nechvatal JR, Smid ME, Leigh SD, Levenson M, Vangel M, Heckert NA, Banks DL, A statistical test suite for random and pseudorandom number generators for cryptographic applications. National Institute of Standards and Technology (NIST). 2010 Sept.16.

[22] VON Neumann, J. Various techniques used in connection with random digits. In Monte Carlo Method (Washington, D.C.: U.S. Government Printing Office, 1951), A. Householder, G. Forsythe, and H. Germond, Eds., National Bureau of Standards Applied Mathematics Series, 12, pp. 36–38

[23] Kanda G, Ryoo K. Securing Ubiquitous Hardware Devices with Elliptic Curve Integrated Encryption Scheme. Journal of Advanced Research in Dynamical and Control Systems. 2018 Dec;10(14): 314-324.