FPGA-Based True Random Number Generator Architecture Using 15-Bit LFSR and ADPLL

Huirem Bharat Meitei and Manoj Kumar

1 Introduction

Random numbers are required in virtually all cryptographic operations. Initialization vectors, block padding, nonces, and keys are all encrypted structures that require a randomly generated sequence of numbers. Since the majority of the data created by the random number generator (RNG) are communicated in the open domains, a passive hacker has enough opportunity to evaluate the RNG's findings and exploit any flaws discovered. Consequently, RNG employed in cryptographic operations must constantly be regarded as a crucial element of the encrypted algorithm. A defect or malfunction in the RNG could cause the entire system to fail [1]. The infamous Netscape V2.0 website compromise is a prime example of an effective assault on a vulnerable RNG architecture [2]. Despite an increasing reliance on data collected from different applications, digital phones, and devices, communication infrastructure security becomes crucial. Consumers' privacy must be appropriately secured by implementing an RNG that is both secure and reliable, such as the TRNG. RNGs are essential parts of every cryptographic algorithm, as used in block ciphers, digital signatures [3] as well as in one-time padding. The field of RNG also addressed the deterministic random bit generators (DRBGs) built on a hash algorithm and the SHA-256 encryption technique [4]. Our proposal has two significant advantages over previous concepts: It is vendor-independent and eliminates the need for human installation and channeling throughout the production process, making the generator more compact. A TRNG's development shouldn't be dependent on any specific technology. Recently, Cherkaoui et al. [5], inspired by Sunar et al., devised a novel design in which ROs were replaced with a Self-Timed Ring (STR). Cret et al. expand on the core concept of utilizing only 2 ROs [6]. In this method, the writer uses

H. B. Meitei (✉) · M. Kumar
Department of ECE, NIT Manipur, Imphal 795004, India
e-mail: thinktank453@gmail.com

B. P. Swain and U. S. Dixit (eds.), *Recent Advances in Electrical and Electronic Engineering*, Lecture Notes in Electrical Engineering 1071,
https://doi.org/10.1007/978-981-99-4713-3_27

a multiplexer to split the sampling pulse. All digital phase lock loops (ADPLL)-based TRNGs have a number of advantages over PLL-based TRNGs, including low-power consumption, reduced area requirements, simple synthesizability, as well as the capability to be quickly modified. In contrast, LFSR has the most widespread application in communications and cybersecurity.

LFSRs are typically composed of a D flip-flop and two input XOR gates. It could be done in two steps: using either the Fibonacci algorithm or the D flip-flop algorithm. The LFSR uses a feedback loop to shift the bits of binary data rely on the current state of the register and a predetermined set of feedback rules. This allows for the generation of a sequence of random numbers that are difficult to predict or reproduce Murali Krishna et al. [7] Outcome assessment suggests that envisaged LFSR with and without seed value provides superior results, reduced power intake, as well as increased unpredictability in runtime with Partial Reconfiguration (PR).TRNG based on an LFSR and an ADPLL is a hardware-based random number generator that uses the principles of both LFSR and an ADPLL to generate random numbers. It offers a higher degree of randomness and unpredictability than an LFSR alone, making it useful for applications where true randomness is important, such as cryptography. Therefore, TRNGs employing FPGA digital logic design have additional freedom, performance, and convenience than TRNGs employing analog circuitry [8]. Because of the different sources of unpredictability utilized in the production of stochastic randomized sequence, such as acquired Jitter [9], metastability [10], and transitional impacts [11] from different resources like PLL, ROs, and FFs which all greatly influence the TRNG's speed. The strong level of security provided by the cryptographic system is based on generations of random and unique digital key sequences [12]. This work aims to establish and implement a 15-bit LFSR with ADPLL-based TRNG (15-LAT) architecture on the FPGA platform, as explained in the sections below: Sect. 1: Introduction, Sect. 2: Summary of an ADPLL used in proposed TRNG (15-LAT) Architecture, Sect. 3: Proposed design for implementing TRNG based on ADPLL with LFSR, Sect. 4: FPGA realization of TRNG design centered on ADPLL with 15-bit LFSR, Sect. 5: Experiment results, Sect. 6: Comparison among different TRNGs, Sect. 7: Conclusion.

1.1 Random Number Generators Type

RNGs are typically divided into 2 different categories.

Pseudo-Random Number Generators (PRNGs)
A PRNG is a probabilistic algorithm that generates an unpredictable bitstream of data from a non-repetitive sequence of random bits. It is generated via specific software instructions and is activated by a function that generates a predictable key stream [13]. As a consequence, a continuous, cyclical, and repeated sequence of a random sequence is generated that approximates the properties of earlier generated arbitrary sequences. PRNGs employ a preset technique to generate a series of outcomes based

on the previous entropy seed [14]. It takes an initial number (the "seed"), performs a series of calculations on it, and then outputs a sequence of numbers that seem to be random. The sequence of numbers is dependent on the seed, so if the seed thus generated is known, then the generated numbers can be reproduced.

True Random Number Generators (TRNGs)
Whereas TRNG is a method of producing unexpected randomized numbers that rely on both physical processes and a non-deterministic source. These physical processes range from measuring thermal noise in resistors to observing radioactive decay. Additionally, the absence of similarity between the recently generated bitstream as well as the most recently created data sequence [10] provides TRNG with high robustness over PRNG. This architecture receives its output entirely from an asynchronous physical process occurring under the surface. Because no intermediate data is preserved in the generator, the result is entirely dependent on the physical operation and never upon any predefined information. TRNGs provide a guarantee of unpredictability that PRNGs cannot provide.

2 A Summary of an ADPLL Used in the Proposed TRNG (15-LAT) Architecture

We combine our LFSR using an ADPLL layout comprising ring oscillators and flip-flops to provide the requisite total entropy seed for the stochastic sequence generation of the proposed TRNG. ADPLL is an electronic circuitry technique that permits the functional reproduction of the basic digital block on the Field programming Gate Array board. ADPLL implements phase-locked loops (PLLs) entirely digitally [15]. It is consist of 3 basics element, i.e., Phase detectors (PDs), Loop filters (LFs) as well as Digital Control Oscillators (DCOs). XOR-Gate act as the phase detector (PD) [16]. ADPLL makes use of PD to reduce the amount of difference that exists between the 2 streams.

To remove undesired frequency components, K counter is utilized as a loop filter [18]. ID counters operate in the same way to DCOs such that they modify the frequencies based on the LF input result. Figure 1 depicts the overall circuit diagram of an ADPLL. Mfo, the clock frequency driving the K counter, is the same as the K clock. The ID counter's clock signal is 2Nfo, while M and N represent the K counter and ID counter's modulus controls, correspondingly. M is often set to 8, 16, 32, …, with $M = 2N$ used to fix the N values. The XOR gate's XOR out signal is fed into the K counter, which generates a carry signal (ca) [15].

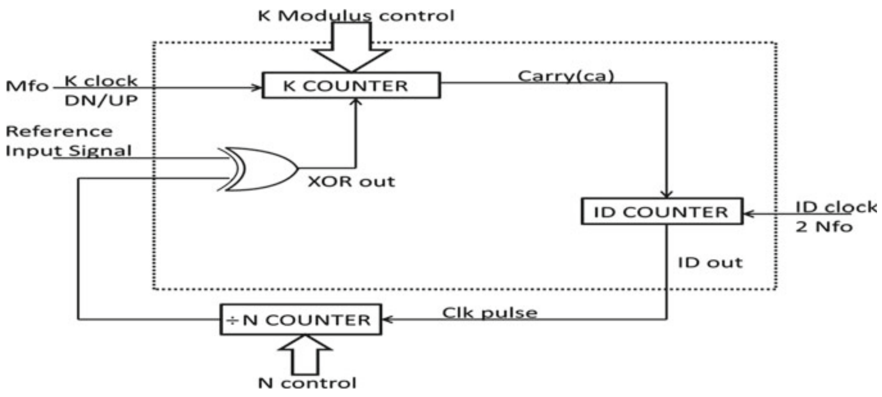


Fig. 1 Circuit diagram of an ADPLL used in ACT architecture [17]

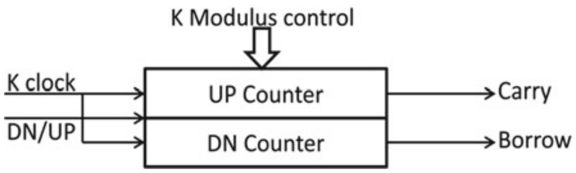


Fig. 2 Block diagrammatic form of *K* counter used as loop filters in our proposed (15-LAT) architecture [16]

2.1 K Counter as Loop Filter (LF) Used for ADPLL

A *K* counter, as shown in Fig. 2, is a type of LF or integrator that operates in tandem with an EXOR or JK phase detector. It is made up of 2 distinct counters, an up counter as well as a down counter, that both count upward. The *K*-modulus counter has a k range of 0 to $k - 1$, and its value equals M times the center frequency. For the down counter to be enabled, the DN/UP condition must always be high, while for the up counter to be enabled, the DN/UP condition must always be low [19].

2.2 Digital Control Oscillator (DCO) Used in ADPLL

DCOs are a form of adaptive oscillator that use the output of the loop filter to alter the input signal's frequency [19]. Figure 3 depicts the DCO network configuration of the ADPLL. The final output of DCO is known as id out [15].

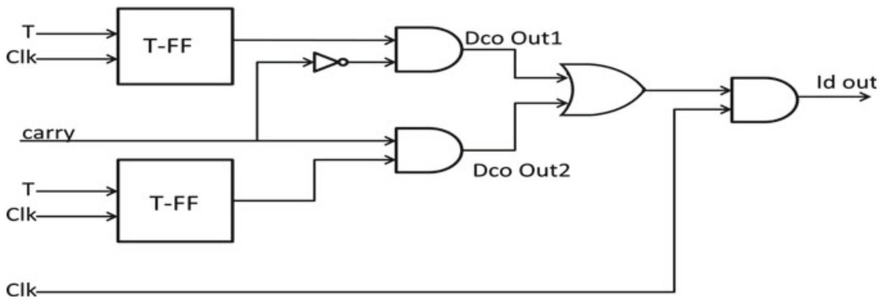
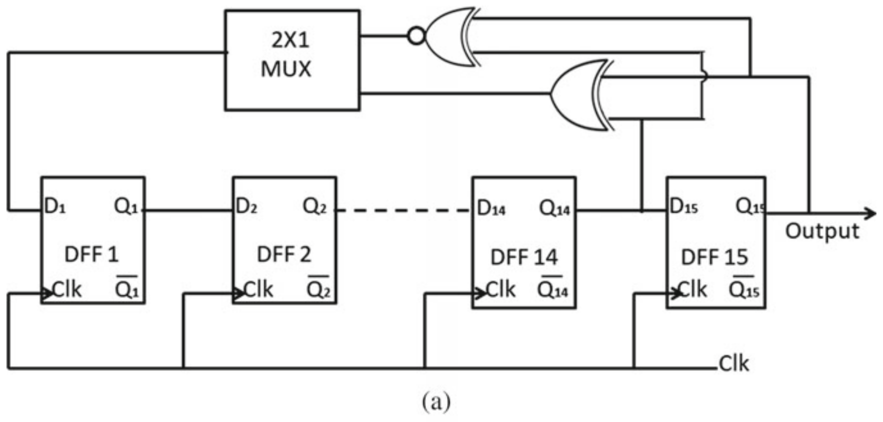


Fig. 3 DCO circuit diagram [17]

3 Proposed Design for Implementing TRNG Architecture Based on ADPLL with 15-LFSR

A LFSR is a form of a shift register that generates a sequence of binary numbers via feedback. The sequence is decided by the feedback function, which is a Boolean function that accepts the current state of the shift register as an input and returns a new bit that is shifted into the register. It is possible to develop an LFSR capable of generating a stream of random numbers by carefully configuring the feedback function. Theoretically, an LFSR is a series of connected FFs, with each flip-input flop's being the output of the previous one [20]. It is produced by combining XOR gates within the feedback of a series of flip-flops. The initial number of the LFSR, also known as the seed number, consists of both 1s and 0s. Even if the seed number influences following random variables, it is essential to choose a number with minimal energy usage. Figures 4a and b exhibit the architectures of LFSR1 and LFSR2, respectively. Exclusive-OR (XOR) with a single bit is used as a linear function. LFSR is composed of XOR gates with DFF. Typical LFSR polynomials are characterized by XOR positions. $P(x) = x5+x4+x+1$. An LFSR with a properly selected feedback mechanism can generate a seemingly unpredictable bitstream. The starting value of the LFSR is referred to as the seed; the sequence of integers generated by the shift register depends on its prior or present configuration. Due to the register's limited number of stages, it subsequently reaches a cycle. The maximum size of an LFSR series is $2n$ minus one, producing a randomized periodic pattern [21]. The feedback function of an LFSR is often represented by a simple polynomial. The maximum feasible LFSR generates the greatest number of PRPG configurations with a design count of $2n - 1$, where n is the number of register components in the LFSR.

Figure 4 illustrates the fundamental structural design of the 15-bit-LFSR utilized in our suggested architecture. Additionally, TRNG shouldn't be overly dependent on technology. Nevertheless, because the PLL is not available across all FPGA types, it is challenging for engineers. In addition, PLL-based TRNG needs additional energy and occupies greater storage [21] over ADPLL-based TRNG. Due to its ADPLL-based digitized structure, our Noval 15-LAT design may be created as well as improved in a short time.



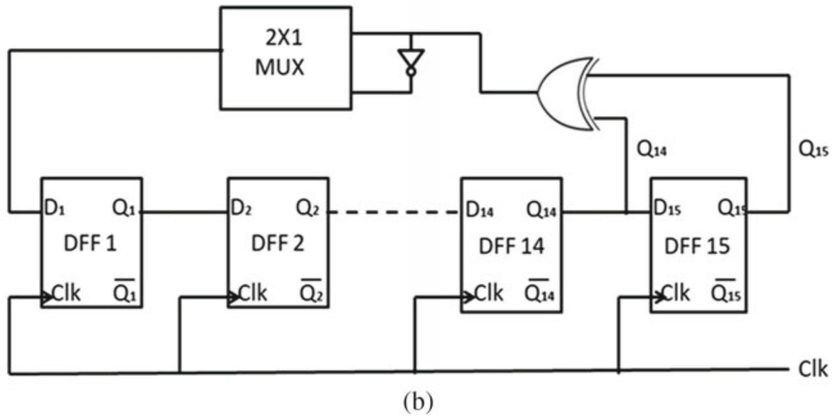


Fig. 4 **a** 15-bit linear feedback sift register 1 (15-Bit LFSR1) [7]. **b** 15-bit linear feedback sift register 2 (15-Bit LFSR2) [7]

ADPLLs configured separately using dual 15-bit LFSRs connected in series using the ring oscillator comprise our design proposal. In our RO architecture, we implemented a pulse generator consisting of 51 inverters in order to generate the clock signal for the suggested 15-LAT structure. Using VHDL, 15-bit LFSR and ADPLL-based TRNG implementations are constructed. In additional to the flip-flop and metastability criterion, explore all important types of entropy, such as disruption by ADPLLs [22] and proposed rings oscillators [17]. Figure 5 depicts the suggested TRNG design-based ADPLL with 15-bit LFSR. The system clock is set to 100 MHz and is supplied to the divide by two counters on the pulse generator circuit.

Here, 100 MHz is used as the operating system CLK and is sent to the pulse generator circuit's divide by two counter. Now, a 50 MHz pulse is received by a circuit in which a pulsing signal oscillates amongst 2 voltage states signifying true and false. XORing the jitter output produced by two cascading ring-oscillators and a 15-bit LFSR using the 400 MHz ID output pulse (DCO output) of the ADPLL as

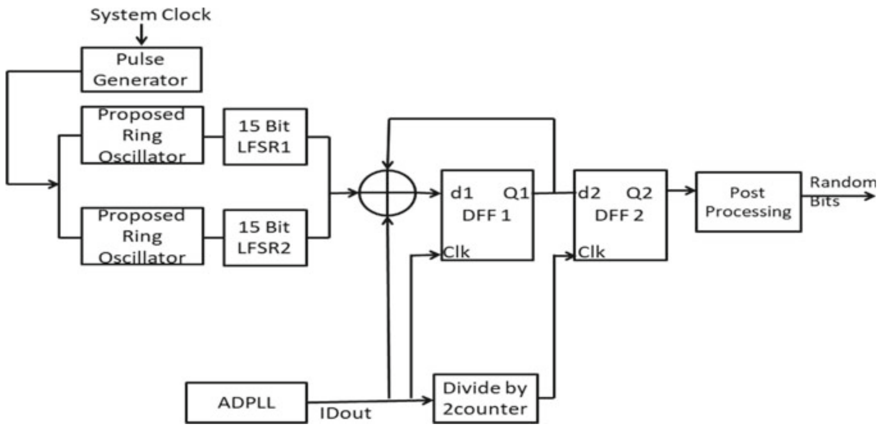


Fig. 5 Proposed TRNG architecture with 15-bit LFSR based on ADPLL (15-LAT)

well as the Q1 of DFF1 feedback loop. The outcome of Q1 of DFF1 is therefore supplied to d2 of DFF2 together with the counter's produced CLK pulse. By using an ADPLL in conjunction with an LFSR, it is possible to create a TRNG that has a higher degree of randomness and unpredictability than an LFSR alone. Q2 of DFF2 yields a random bitstream that is transmitted for post-processing to verify that the sampler's outputs random numbers are unbiased.

4 FPGA Realization of TRNG Design Based on ADPLL with 15-Bit LFSR

The inquiry utilizes an Artrix-7 FPGA system (XC7A35T CPG236-1) and an oscilloscope (DSO-X3012A) to record the pattern. Table 1 lists the FPGA pinouts for the TRNG solution based on ADPLL with CS phenomenon. Figure 6 shows the experimental setup for an ACT-TRNG. The total system clock is generated utilizing the W5 input mode as well as the V17 T-FF inputs. The outcomes are connected to JB1:A14, which serves as the DSO's live probe, and JB5:GND, which serves as the ground probe.

Table 1 Pins information for the integrated proposed TRNG architecture centered on ADPLL with 15-bit LFSR

Symbol	Details	Mode	Pinned mode
CLK	Operating-clock	Input	W5
t	T-FF	Input	V17
rst	Reset	Input	V16
q3	Resulted random data	Output	A14
Vcc	Power supply		USB port

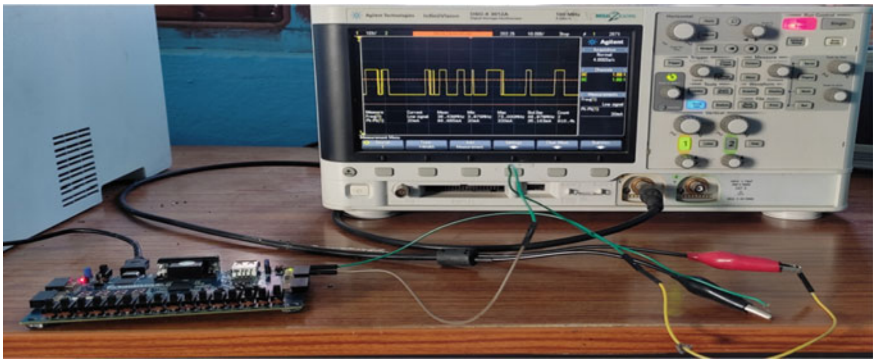


Fig. 6 Experimental configuration of the FPGA-DSO interface for the 15-LAT architecture

5 Experiment Results

The suggested TRNG centered on ADPLL with 15-bit LFSR is schematically depicted in Fig. 7. All schematic designs were generated in Vivado v.2015.2, and simulations were run on an Artrix-7 FPGA board with the xc7a35tcpg236-1 module. Figure 8 and Fig. 9 illustrates the TRNG output waveforms along with the FFT waveform respectively collected by DSO of the proposed 15-LAT architecture (Table 2).

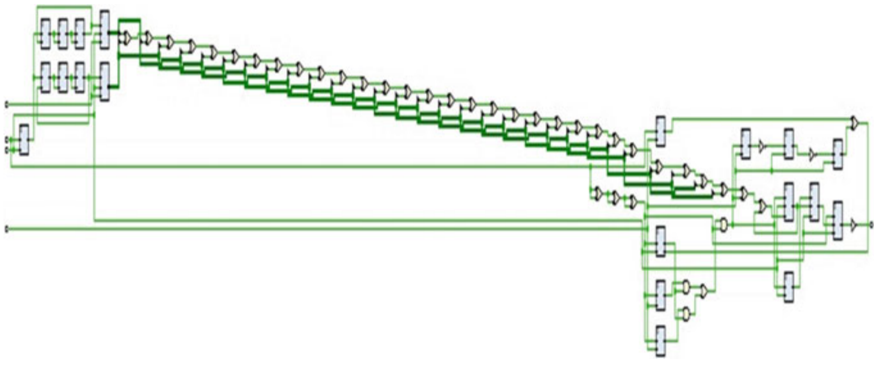
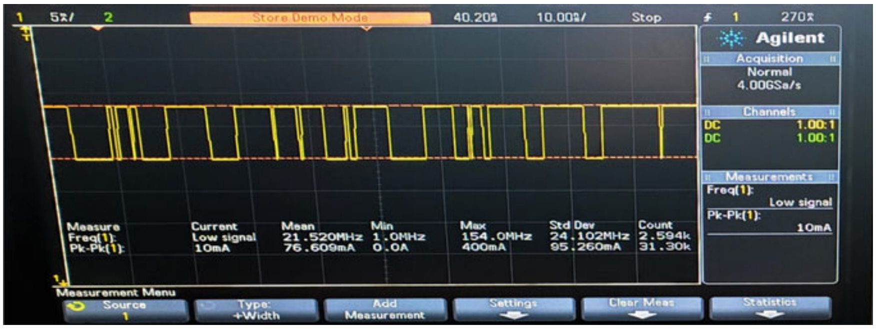


Fig. 7 Schematic diagram for the proposed 15-LAT architecture



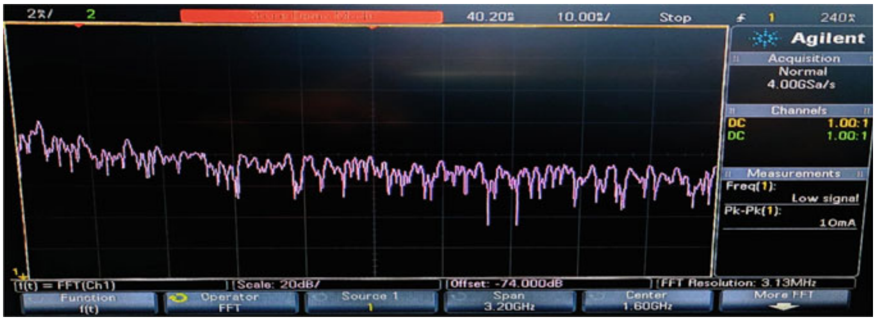
Acquisition
Normal
4.00GSa/s

Channels
DC 1.00:1
DC 1.00:1

Measurements
Freq(1): Low signal
Pk-Pk(1): 10mA

Measure	Current	Mean	Min	Max	Std Dev	Count
Freq(1):	Low signal	21.520MHz	1.0MHz	154.0MHz	24.102MHz	2.594k
Pk-Pk(1):	10mA	76.609mA	0.0A	400mA	95.260mA	31.30k

Fig. 8 The envisioned TRNG's output waveform is constructed upon ADPLL and LFSR (15-LFSR)



Acquisition
Normal
4.00GSa/s

Channels
DC 1.00:1
DC 1.00:1

Measurements
Freq(1): Low signal
Pk-Pk(1): 10mA

f(t) = FFT(Ch1) | Scale: 20dB/ | Offset: -74.000dB | FFT Resolution: 3.13MHz
Span 3.20GHz
Center 1.60GHz

Fig. 9 FFT output waveform of TRNG build on ADPLL along with LFSR (15-LFSR) architecture

Table 2 NIST (SP 800-22) test result

NIST-test	*p*-value	Result
Frequencies	0.9990	Passed
Block-frequencies	0.9999	Passed
Run	0.9990	Passed
Rank	0.0000*	Failed
DFT	0.o218	Passed
Serial test	0.9936	Passed
Linear-complexities test	0.0000*	Failed
Longest-run test	0.0500	Passed
Approximate-entropy test	0.0000*	Failed
Cumulative-sum test	0.9990	Passed

* NIST failed whenever the *p* value is 0.000

6 Comparison Between Existing TRNG with Proposed TRNG Based on ADPLL with 15-Bit LFSR

Tables 3 and 4 compares the new 15-LAT architecture to the previous TRNGs architectures. The performance of numerous TRNGs is compared in the Table 3. Our presented TRNG architecture-based ADPLL with 15-bit LFSR made better use of existing hardware capacity even though utilizing minimum power.

7 Conclusion

Our method makes a substantial contribution toward the growing utilization of FPGAs in encryption methods. The system's overall security is strengthened by having the capacity to completely enclose a TRNG architecture with the FPGA. In this paper, we introduced a unique entropy source for the 15-LAT architecture that significantly reduces engineering complexity. We empirically confirmed the architecture's viability and showed that achieving the stochastic model's entropy needs is always possible, especially whenever placing constraints are removed or the architecture is ported to a different generation of FPGAs. This characteristic provides the TRNG a good candidate for inclusion into wider crypto methodologies. Randomized bits are generated and are statistically valid using NIST 800-22 test. By using 15-bit LFSR along with ADPLL in the TRNG design, we can reduce hardware resources while increasing the efficacy of the FPGA chips, as shown in Table 3. With this study, the potential of cybersecurity via ADPLL-based TRNG with 15-LFSR appears to be optimistic, giving it a more dependable and secure solution for a variety of applications.

Table 3 A comparative study of different TRNGs architectural efficiency

Paper	Entropy	Device	Hardware-resource	Post-processing
15-LAT	Jitter and metastability	Artrix-7	0 LUT, 4 FFs	Yes
[23]	Latches-oscillatory and metastability	Artrix-7	40 LUTs	No
[24]	Self-time ring-oscillator	Artrix-6	56 LUTs	No
[25]	Jitters and metastability	Artrix-7	50 LUTs	No
[10]	Jitters or metastability	XCKU040	1 PLL, 5 Primitives, 5 Slices	Yes
[14]	Jitters or metastability	XC7A35T CPG236-1	1 LUT	Yes
[26]	Chaotic ring-oscillator	XC6SLX16	44 LUTs	Yes
[27]	Metastability	Virtex5	0 LUT	No
[28]	Ring-oscillator	Spartan-6	10 LUTs, 5 FFs	No
[17]	Jitter or metastability	XC7A35T	2 LUTs, 1 Slice	No

Table 4 Synthesis results in comparison among different TRNG's

Parameter	Paper [17]	Paper [29]	Paper [30]	Paper [31]	15-LAT
Platform	FPGA	FPGA	FPGA	0.35 μm CMOS	FPGA
Power consumption (W)	0.076	0.125	0.095	0.026	0.072
Post processing	No	No	No	Yes	Yes
Testing	NIST-SP800-22	NIST-SP800-22	NIST-SP800-22	NIST-SP800-22	NIST-SP800-22

Acknowledgements The author would like to extend his heartfelt gratitude to Dr. Manoj Kumar for all of his support and assistance throughout the process of producing this research article.

Availability of Data There is no data available.

References

1. Eastlake D, Crocker S, Schiller J (1994) Randomness recommendations for security—RFC 1750. Available at http://www.faqs.org
2. Goldberg I, Wagner D (1996) Randomness and the Netscape Browser. Dr. Dobb's Journal, January 1996
3. Buchmann J, Dahmen E, Szydlo M (2009) Hash-based digital signature schemes. Post-Quantum Cryptogr, 35–93. https://doi.org/10.1007/978-3-540-88702-7_3
4. Loza S, Matuszewski L (2014) A true random number generator using ring oscillators and SHA-256 as post-processing. In: 2014 international conference on signals and electronic systems (ICSES). https://doi.org/10.1109/icses.2014.6948739
5. Cherkaoui A, Fischer V, Fesquet L, Aubert A (2013) A very high speed true random number generator with entropy assessment. In: Proceedings of international workshop on cryptographic hardware and embedded systems (CHES'13). Lecture notes in computer science, vol 8086. Springer, pp 179–196
6. Cret O, Suciu A, Gyrfi T (2008) Practical issues in implementing TRNGs in FPGAs based on the ring oscillator sampling method. In: 10th international symposium on symbolic and numeric algorithms for scientific computing (SYNASC'08). IEEE Computer Society, pp 433–438
7. Murali Krishna B, Madhumati GL, Khan H (2019) FPGA based pseudo random sequence generator using XOR/XNOR for Communication cryptography and VLSI testing applications. Int J Innov Technol Exploring Eng 8(4):485–494
8. Gupta R, Pandey A, Baghel RK (2018). Efficient design of chaos based 4 bit true random number generator on FPGA. Int J Eng Technol 7(3):1783. https://doi.org/10.14419/ijet.v7i3.16586
9. FIPS 140-1 (n.d.) Security requirements for cryptographic modules. Csrc.nist.rip. Retrieved 13 April 2021, from https://csrc.nist.rip/publications/fips/fips140-1/fips1401.htm
10. Stanchieri G, De Marcellis A, Palange E, Faccio M (2019) A true random number generator architecture based on a reduced number of FPGA primitives. AEU Int J Electron Commun 105. https://doi.org/10.1016/j.aeue.2019.03.006
11. Kohlbrenner P, Gaj K (2004) An embedded true random number generator for FPGAs. In: Proceeding of the 2004 ACM/SIGDA 12th international symposium on field programmable gate arrays—FPGA'04. https://doi.org/10.1145/968280.968292
12. Vasyltsov I, Hambardzumyan E, Kim Y-S, Karpinskyy B (n.d.) Fast digital TRNG based on metastable ring oscillator. In: Cryptographic hardware and embedded systems—CHES 2008, pp 164–180. https://doi.org/10.1007/978-3-540-85053-3_11
13. Internet resource: AMD random number generator library. https://developer.amd.com/wordpress/media/2013/12/AMD-Number-Generator-User-Guide.pdf. Accessed Sept 2018
14. Meitei H, Kumar M (2022) FPGA implantations of TRNG architecture using ADPLL based on FIR filter as a loop filter. SN Appl Sci 4:96. https://doi.org/10.1007/s42452-022-04981-6
15. Chaudhary AK, Kumar M (2017) Design and implementation of ADPLL for digital communication applications. IEEE Xplore. https://doi.org/10.1109/I2CT.2017.8226159
16. Kumar M, Lata K (2012) All digital phase locked loop (ADPLL): a survey. In: Proceeding the 4th IEEE international conference on electronics computer technology (ICECT 2012), 6–8 April 2012, Kanyakumari, India

17. Meitei HB, Kumar M (2021) FPGA implementation of true random number generator architecture using all digital phaselocked loop. IETE J Res. https://doi.org/10.1080/03772063.2021.1963333
18. Lata K, Kumar M (2013) ADPLL design and implementation on FPGA. IEEE Xplore. https://doi.org/10.1109/ISSP.2013.6526917
19. Lata K, Kumar M (2013) All digital phase-locked loop (ADPLL): a Survey. Int J Future Comput Commun, 551–554. https://doi.org/10.7763/ijfcc.2013.v2.225
20. Hollmann H (1990) Design of test sequences for VLSI self-testing using LFSR. IEEE Trans Inf Theory 36(2):386–392
21. Arnault F, Berger T, Minier M, Pousse B (2011) Revisiting LFSRs for cryptographic applications. IEEE Trans Inf Theory 57(12):8095–8113
22. Radhapuram S, Yoshihara T, Matsuoka T (2019) Design and emulation of all-digital phase-locked loop on FPGA. Electronics (Basel) 8:1307
23. Fujieda N (2020) On the feasibility of TERO-based true random number generator on Xilinx FPGAs. In: Proceedings of the 2020 30th international conference on field-programmable logic and applications (FPL), Gothenburg, Sweden, 31 August–4 September 2020, pp 103–108
24. Wang X, Liang H, Wang Y, Yao L, Guo Y, Yi M, Huang Z, Qi H, Lu Y (2021) High-throughput portable true random number generator based on jitter-latch structure. IEEE Trans Circ Syst I Regul, Pap. 2021 68:741–750
25. Lin J, Wang Y, Zhao Z, Hui C, Song Z (2020) A new method of true random number generation based on Galois ring oscillator with event sampling architecture in FPGA. In: Proceedings of the 2020 IEEE international instrumentation and measurement technology conference (I2MTC), Dubrovnik, Croatia, 25–29 May 2020, pp 1–6
26. Yang Y et al (2017) A reliable true random number generator based on novel chaotic ring oscillator. In: 2017 IEEE international symposium on circuits and systems (ISCAS). www.semanticscholar.org/paper/A-reliable-true-random-number-generator based-on-Yang-Jia/c3e65e27fd09968934977d250f0ead2c13e60b35. https://doi.org/10.1109/ISCAS.2017.8050843
27. Ben-Romdhane M, Graba T, Danger J-L (2013) Stochastic model of a metastability-based true random number generator. Trust Trust Comput. https://doi.org/10.1007/978-3-642-38908-5_7
28. Yang B, Rožic V, Grujic M, Mentens N, Verbauwhede I (2018) ESTRNG: a high-throughput, low-area true random number generator based on edge sampling
29. Cicek I, Pusane AE, Dundar G (2014) A new dual entropy core true random number generator. Analog Integr Circ Sig Process 81(1):61–70
30. Petrie CS, Connelly JA (2000) A noise-based IC random number generator for applications in cryptography. IEEE Trans Circ Syst I: Fund Theory Appl 47(5):615–621. https://doi.org/10.1109/81.847868
31. Ergun S, Ozoguz S (2010) Truly random number generators based on non-autonomous continuous-time chaos. Int J Circuit Theory Appl 38(1):1–24