FPGA implantations of TRNG architecture using ADPLL based on FIR filter as a loop filter

Huitem Bharat Meitei¹ · Manoj Kumar¹

Received: 14 September 2021 / Accepted: 3 February 2022

Published online: 07 March 2022

© The Author(s) 2022

Abstract

This article describes about the design, implementation, and analysis of a true random number generator (TRNG) employing an all-digital phase-locked loop (ADPLL) based on a finite impulse response (FIR) filter as the digital loop filter and implemented on the Artix 7(XC7A35T-CPG236-1) field programmable gate array (FPGA) board using the Xilinx Vivado v.2015.2 design suite. The coefficients of a 3rd-order broadcast low-pass digital FIR filter are computed using the Keiser window method. The MATLAB-FDA tool is used to calculate the filter coefficient. After reducing the bias in the sequence using the XOR-corrector post-processing approach, the suggested ADPLL-based TRNG designs created an unbiased stochastic random number with an overall throughput of 200 Mbps for both designs. The first proposed FIR-ADPLL-based TRNG design (FAT-1) consumes 0.072 W of power, whereas the second proposed FIR-ADPLL-based TRNG design (FAT-2) consumes 0.074 W. The resulting bitstream is verified for randomness using national institute of standards and technology (NIST) test following post-processing. Digital storage oscilloscope (DSO) is interfaced to the Artix-7 FPGA board in order to capture the TRNG output waveforms. Both the proposed FIR-based ADPLL-TRNG designs passed the NIST SP 800-22 test, indicating that they are well-suited for a variety of industrial applications, including network security, cybersecurity, banking security, smart cards, radio-frequency identification tags, Internet of Things, and Industrial Internet of Things.

Keywords TRNG · FPGA · ADPLL · FIR filter · Keiser window · XOR corrector

1 Introduction

Since the dawn of time, humans have faced the challenge of data protection. With the perception of integrated circuits, many methods have been developed to safeguard them. With the fast growth of the Internet, the need for information security in a variety of sectors is increasing, and security concerns are receiving increased attention throughout the world [1–5]. Generally, this request can be met by integrating multiple software and/or hardware designs able to generate random number combinations and giving the requisite public and private keys for effective data encryption [6]. Random number generators

(RNGs) are critical components of every cryptographic system, as they are utilized in block ciphers, digital signatures [7], and one-time padding [8]. Random numbers may be classified into two types based on the random sequence created, namely pseudo-random numbers generators (PRNGs) and true random numbers generators (TRNGs), as seen in Fig. 1 classification. PRNGs create a sequence of outputs depending on the preceding seed of entropy using a deterministic algorithm and procedure. PRNGs are essentially created using specialized software code and executed via a subroutine [9] that creates a predictable output sequence. In general, the result is a lengthy periodic and repeating series of bitstreams with features

✉ Huitem Bharat Meitei, thinktank453@gmail.com; Manoj Kumar, manoj@nitmanipur.ac.in | ¹Department of ECE, NIT Manipur, Imphal, India.

Fig. 1 Classification random number generators architecture

that roughly match those of the previously produced random numbers. Because the number generated by a PRNG is predictable and likely, it is occasionally vulnerable to attack by such devices if the seed of entropy selected is incorrect, exposing the bitstream to hackers. TRNG, on the other hand, creates unexpected random numbers through the use of a physical process and an asynchronous source. Additionally, the absence of resemblance between the most recent data sequence created and the preceding one [6] makes TRNG more secure than PRNG. TRNGs based on FPGAs enable the usage of a huge matrix of configurable logic blocks (CLBs) coupled through programmable interconnects, therefore enabling the creation of a powerful platform primarily utilized for digital very large-scale integration (VLSI) implementations. TRNG proposed by author Guido Di Patrizio Stanchieri, Andrea De Marcellis, Elia Palange and Marco Faccio [6] are more secured than PRNG proposed by advanced micro devices, Inc (AMD) [9]. The noise source creates an unprocessed bitstream, which frequently requires post-processing to enhance the unpredictability characteristics of the produced random sequences. The purpose of this article is to describe the design, implementation, and analysis of a true random number generators architecture that utilized the jitters generated by an all-digital phase-locked loop (ADPLL) based on FIR filter as a digital low pass filter and the jitters generated by the proposed ring oscillators also with the metastability feature of the Flip-flops (FF) as seeds of entropy. In contrast to other methods, we demonstrate how to get a reliable TRNG architecture using just the onboard resources of the FPGA XC7A35T-CPG236-1 board.

As a result, the proposed solution greatly reduces the number of Configurable Logic Boards required, the intricacy of the architecture, and the total power consumption without compromising the generated throughput. The proposed method of generation of unbiased random data

has been proven to be reliable according to standard criteria. This paper expands on a preliminary study published in [10] by including new findings, experimental results, and analytical methods to provide a more precise and comprehensive explanation of each component of the proposed TRNG architectures. As random numbers generated by RNG devices based on analog noise disturbances are unpredictable, they are vulnerable to attacks that compromise the security of user-provided information [11]. TRNGs based on FPGA digital logic architecture offer greater flexibility, higher speed, and less complexity than analog circuit-based TRNGs [12]. As a result, another ADPLL-based approach for generating highly secure random bitstreams is described. ADPLL-based TRNGs offer several benefits over phase-locked loop (PLL)-based TRNGs, including their simplicity of synthesizability and ability to be fully customized in a short amount of time [10]. Additionally, we used the FIR filter as the ADPLL loop filter to increase the overall performance of the suggested TRNG in terms of area look up table (LUT) and power. We provide a comprehensive overview of the seeds of entropy in the TRNG process, with a particular emphasis on the jitter and metastability effects created by an ADPLL, ring oscillator, and FF. Furthermore, we compare the proposed TRNG's performance to that of existing TRNGs published in the field, demonstrating that the proposed design employs the minimal resources possible, yet retaining a relatively high throughput rate in low power. Subsequently, post-processing actions are conducted on the generated random sequence to increase its statistical distribution and unpredictability characteristics at the expense of throughput. The generated random number sequences with a 200 Mbps output bit rate of both the TRNGs designs meet the requirements of the NIST (National institute of standards and technology) test, making them suitable for use in cybersecurity network systems, smart cards, and IoT applications. The following

sections comprise the paper: Section 1—Introduction, Section 2—Existing work related to TRNG, Section 3—Details description about ADPLL for TRNG implementation, Section 4—TRNG architecture implementation using ADPLL based on FIR as loop filter, Section 5—FPGA implementation of TRNG architecture based on ADPLL, Section 6—Comparisons between proposed TRNG architecture based on ADPLL with other TRNGs, and Section 7—Conclusion.

2 Existing work related to TRNG

Paper [6] describes how a TRNG architecture was design and integrated with the XCKU040 board. Jitter from the PLL and metastability from the Flip-Flop are used as entropy sources. Furthermore, the results show that a TRNG architecture with a throughput of 10 Mbps can be built with only 8 D-Type flip-flops, 17 LUTs, 2 Counters, and less configurable logic blocks (CLB). Vasytsov et al. [13] suggested a TRNG-based solely on the ring oscillator's metastability entropy. Furthermore, the author claims that a digital TRNG based on Meta-RO has a higher value of entropy, resulting in a nominal throughput of 140 Mbps being substantially increased. To demonstrate the efficacy and feasibility of ADPLL architecture, [14] uses an emulation technique. The proposed ADPLL emulation covers a broad frequency spectrum from 120 to 300 MHz, minimizing design time and complexity. In conclusion, as the author claims, the suggested ADPLL is simply transferrable to various processes. Paper [10] proposes the design and construction of a TRNG based on ADPLL jitter along with FF metastability that achieves a whopping 680.7 Mbps throughput. However, due to the absence of post-processing, while this system generates random numbers, the sequence it generates has the potential of being biased. Koyuncu and Turan Özcerit [15] proposed that based on their chaotic architecture, a high-speed chaos-based TRNG be implemented on an FPGA. The design works at a frequency of 293 MHz and has an output bitrate of 58.76 Mbps, and it complies with both NIST 800-22 and FIPS 140-1 requirement.

3 Detail description about ADPLL for TRNG architecture implementation

ADPLL is a digital circuit architecture that utilizes a core digital block that can be effectively replicated on an FPGA [9]. In other words, ADPLL is a fully digital version of a phase-locked loop (PLL) [16]. It consists of three components: phase detectors (PD), loop filters (LF), and digital control oscillators (DCO) [17]. The three units are

connected in a closed-loop feedback system. The basic ADPLL block design is illustrated in Fig. 2.

Third-order broadcast low-pass FIR is used as a loop filter which is used to eliminate noise or undesirable frequency components. ID counters function similarly to DCOs; in that they alter the frequency in response to the output of the LF signal. Figure 3 illustrates an ADPLL's first-order circuit diagram. FIR filter clock is equal to Mf_0 , which is a clock for the FIR filter clock signal. ID-clock, equal to $2Nf_0$, is the DCO clock signal, where M and N are the modulus controls for the FIR filter and DCO, respectively. The XOR gate's output, which is XOR-out, is fed to the input of the FIR filter, along with the clock, which outputs a carry signal (ca) [16]. We chose 50 MHz as the center frequency (f_0) for FAT-1 TRNG design, For ADPLL1 $M=16$ and $N=8$ and for ADPLL2 M equal 8, N equal 4. The ID-clock is the DCO clock signal, which is equivalent to $2Nf_0$. Similarly, for FAT-2, the center frequency (f_0) is 50 MHz for both ADPLL1 and ADPLL2, and the parameters for ADPLL1 are $M=16$, and $N=8$, while for ADPLL2 we used $M=8$ and $N=4$. Table 1 discusses the parameters used in designing ADPLL's of the proposed TRNGs.

3.1 Digital phase detector

The phase detector in this ADPLL design is implemented using an XOR gate [18, 19]. The XOR mechanism is a basic but effective phase detection process. It performs a phase comparison between the entering original signal and the ADPLL output signal and creates an error signal proportionate to the phase difference. The input phase v_1 , output phase v_2 , and frequency are all interlocked using ADPLL. PD is used to reduce the gap between the two signals used in ADPLL [17].

3.2 FIR filter as a loop filter (LF) used in ADPLL's

The filter is a type of basic signal processing circuit that digitally process and generate digital information which is used in a variety of communication system and physical applications. An electronic circuit that permits or transmits the desired band of frequencies while attenuating the undesirable band. Digital filters are composed of adder, multiplier, and delay units. Additionally, a digital filter is

Fig. 2 ADPLL block diagram

Fig. 3 First-order ADPLL circuit diagram

Table 1 Parameter of the ADPLLs design used in proposed TRNGs architecture

(FAT-1) ADPLLs design			(FAT-2) ADPLLs design		
Parameter	(ADPLL) ₁	(ADPLL) ₂	Parameter	(ADPLL) ₁	(ADPLL) ₂
<i>M</i>	16	8	<i>M</i>	16	8
<i>N</i>	8	4	<i>N</i>	8	4
Center frequency (fo)	50 MHz	50 MHz	Center frequency (fo)	50 MHz	50 MHz

a sort of linear time-invariant (LTI) system that acts on a discrete-time sampled signal to diminish or enhance specific characteristics of the signal [20, 21]. The difference equation for a digital low-pass filter in its general form which is used is given below in (1).

$$y(n) = - \sum_{k=1}^N a_k y(n-k) + \sum_{k=0}^M b_k x(n-k) \quad (1)$$

where $y(n)$ denotes the current filter outcomes, $y(n-k)$ denotes the present or previous filter intakes, a_k denotes the feed-forward parameters related to the filter zeros, b_k denotes the feedback coefficient relating to the filter pole, and N defines the orders [22, 23]. Digital filters are categorized primarily into two types: (1) finite impulse response (FIR) or non-recursive filters and (2) infinite impulse response (IIR) or recursive filters. FIR are those whose output is dependent on only the current and previous inputs. They are devoid of feedback from the output. The FIR digital filter is widely used in digital signal processing systems for a variety of applications [24]. Due to the absence of feedback, FIR filters are non-recursive. Since

all the pathways from the source to the destination flow forward, the FIR filter's signal flow is strictly feed-forward. The FIR difference equation, which describes the relationship between the input and output signals, is as follows in Eq. (2):

$$y(n) = b_0 x(n) + b_1 x(n-1) \dots b_N x(n-N) \quad (2)$$

Additionally, it can be expressed as

$$y(n) = \sum_{i=0}^N b_i x(n-i) \quad (3)$$

where $x(n)$ and $y(n)$ signify the input and output signals, respectively, b_i denotes the filter coefficients, and N is the filter order [21].

3.2.1 FIR filter design

Data flow graph for low-pass 3rd-order broadcast FIR filter are shown in Fig. 4. For designing, we are using MATLAB-FDA as the synthesis tool to create these casual low pass filters; the design specifications are shown in Table 2. As

Fig. 4 DFG for low-pass 3rd-order broadcast FIR filter

Table 2 Specification for filter configuration

Properties	Specification
Response	Low pass
Design	FIR
Order	3rd
Window	Kaiser
Cutoff frequency	0.5 (normalized)
Attenuation at a cutoff frequency	6 dB

Table 3 Kaiser window transfer function coefficient

Transfer function	Coefficient
$h(0)$	0.119
$h(1)$	0.380
$h(2)$	0.380
$h(3)$	0.119

(a) Magnitude plot of Kaiser

(b) Phase plot of Kaiser

Fig. 5 **a** Magnitude plot of Kaiser. **b** Phase plot of Kaiser

indicated in Table 3, the transfer function value varies according to the specification. Kaiser's magnitude and phase plots are depicted in Fig. 5a, b, respectively [25].

3.2.2 Kaiser window

Kaiser window creates a prominent central peak. It reduces side lobes and increases the width of the transition band as given by Eq. (4).

$$w_K(n) = \frac{I_0\left[\alpha\sqrt{1-(2n/N-1)^2}\right]}{I_0(\alpha)} \quad \text{for } |n| \leq \frac{N-1}{2} \quad (4)$$

$$= 0 \quad \text{otherwise}$$

where $I_0(x)$ is the modified zeroth-order Bessel function and α is the adjustable parameters [26].

3.3 ADPLL's digital control oscillator (DCO)

DCO is a modified oscillator that modulates the frequency of a signal using the loop-filter output [17]. They adjust their frequency depending on the loop filter's output. The DCO circuit diagram for the ADPLL is shown in Fig. 6. In Fig. 6, the beginning value of T , a TFF input, is "0". Whether the initial value is toggled is determined by the applied Clk signal [10]. The ID Clk, which equals $2Nf_0$, is the clock signal in the DCO circuit, and the 4-bit carry(0) to carry(3) signal is the output of the FIR filter. Furthermore, Divide by N -counter also produces an output that is dependent on the supply original signal. DCO [16] provides IDout as the final output.

Figure 7 illustrates a pulse generator circuit [10] comprised of a chain of 51 inverters that are used to generate pulse signals in our design. Figure 8 illustrates the proposed ring oscillator [10], which contains an odd number of NOR gates and is used in our novel TRNG design.

3.4 Post-processing

The primary property of a True random number generator is its non-bias random output, but this output is often skewed, implying that '0's and '1's don't emerge with the same probability. TRNGs that are cryptographically stable should be completely random. As a result, we propose that all TRNGs built using this technique include some

Fig. 6 DCO circuit diagram

Fig. 7 Pulse generator circuit [10]

kind of bias reduction. Since the generated sequence is not stochastic due to bias factors, post-processing is often needed. The purpose of post-processing is to ensure that the sampler's outcome is unbiased. Post-processing function can be added to the generator's production. There are several well-documented strategies for reducing bias. The development of the post-processing algorithms, on the other hand, necessitates the deployment of extra hardware resources as well as a methodology based on techniques such as [27]: (1) Von Neumann corrector-based comparison is carried out to determine whether or not the two bits are different: If they are distinct, just one of them is used to create the final digital bitstream sequence, with the other bitstream sequence getting rejected; if they are not, both bits are used to create the output bitstream, with the other bitstream being ignored. (2) XOR corrector where two consecutive generated bits are subjected to XOR reduction operation, which reduces the output bit rate while improving the bias of uncorrelated bits. One

of the famous examples is the XOR corrector technique which is used in our design as shown in Fig. 9. Additionally, the XOR corrector eliminates bias and provides tolerance for environmental changes. The disadvantage is that this feature lowers the TRNG's efficiency by a factor of m , in which ' m ' is the distance between the shift registers [28].

4 TRNG architecture implementation using ADPLL based on FIR as loop filter

TRNG based on PLL consumes more power and occupies a larger space [29] than TRNG based on ADPLL. TRNGs may be fully synthesized and optimized in a short amount of time due to their extensive digital design based on ADPLL. Moreover, two ADPLLs provide more seeds of entropy than a single ADPLL-based design, which results in more random and stochastic noise and aids in the creation of highly secure TRNG architecture.

The proposed architectures employing FIR-based ADPLLs, TRNG design are denoted as (FAT-1) as shown in Fig. 10. Here FPGA system clock which is equal to 100 MHz is transmitted to the pulse generator. Now, a 50-MHz center frequency (f_0) output is generated by a pulse generator circuit, whose pulsating output oscillates between two levels of a voltage indicating true and false [10] as the input of the proposed ring oscillator. The output of the proposed ring oscillator is passed as input to ADPLL1 and

Fig. 8 General architecture for the proposed ring oscillator [10]

Fig. 9 A 4 stage XOR corrector

Fig. 10 Block diagram of FAT-1 TRNG design

ADPLL2, resulting in IDout1 of 400 MHz for ADPLL1 and IDout2 of 400 MHz for ADPLL2. Following an XOR operation between the output of ADPLL1 and the feedback loop signal, the result is delivered to DFF1 as a data signal. Simultaneously, IDout2 which is an output of ADPLL2 is used as the clock signal of DFF1 and is also used as input of divide by two counter, whereas the DFF1 is reset using the outcome of the divide by two counter. To achieve the metastability state of FF, the feedback loop control manages the phase difference between the DFF1 clock and data signal appropriately. At the first step, the metastability criterion is satisfied by the DFF1. On the other hand,

the second FF of the two-state shift register, i.e., DFF2, is utilized to prevent metastability propagation, resulting in a consistent logic level of one (1) or zero (0). However, to eliminate slight bias variations, the established raw random bit condition at the output of DFF2 random entropy can be improved further by applying an XOR-corrector post-processing resulting in an unbiased output random bit. Thus, the proposed TRNG has generated a throughput of 200 Mbps. Simultaneously, Fig. 11 illustrates another proposed TRNG architecture with FIR-ADPLL's, TRNG as (FAT-2). In comparison with FAT-1, FAT-2 utilized four DFFs and two XORs which maximized the statistical unpredictability of

Fig. 11 Block diagram of FAT-2 TRNG design

Table 4 FPGA pin details for implemented TRNG based on ADPLL

Symbol	Details	Pin	Input/output
Clk	System clock	W5	Input
t	TFF	V17	Input
Rst	Reset	V16	Input
q3	Output random bit	A14	Output

the proposed TRNGs output bitstream. For FAT-2 design, the center frequency of 50 MHz is fed as an input of both ADPLL1 and ADPLL2. IDout1 and IDout2 both with 400 MHz are produced by ADPLL1, whereas IDout3 with 400 MHz is produced from ADPLL3. Subsequently, after passing to the different metastable states of FF and performing XOR operations, the entropy thus produced is mixed random sequence combination product of ADPLL jitters and metastability state of FF. The raw random bit sequence generated is applied for the post-processing unit in Fig. 9 to improve the overall stochastic nature of the output random bit. An overall throughput of 200 Mbps is generated from the proposed TRNG (FAT-2) as an output random bit with reduced bias properties.

Table 5 Synthesis results of the implemented TRNG based on ADPLL

Parameter	FAT-1	FAT-2
Total on-chip power (W)	0.072 W	0.074 W
Area (LUTs)	n/a	1

5 FPGA implementation of TRNG architecture based on ADPLL's

The experiment is conducted using an Artrix-7 FPGA board (XC7A35T-CPG236-1), and the output waveform is captured using a digital storage oscilloscope (DSO-X3012A). Table 4 gives the FPGA pin details for implemented TRNG based on ADPLL. Figure 12 depicts a block diagram of the pre-processing setup. Figure 13 illustrates the experiments' preparatory setup for an ADPLL-based TRNG experiment. The output is assigned to JB1:A14, which is connected to the DSO's live probe, and to JB5:GND, which is connected to the ground probe. Table 5 represents synthesis results of the implemented TRNG based on ADPLL.

Fig. 12 Pre-processing block diagram **Fig. 13** Experiment setup of FPGA interface with DSO for the TRNG's using ADPLL

Table 6 NIST statistical test results for TRNG (FAT-1)

NIST test	<i>P</i> -value	Result
Frequency	0.7216	Pass
Block frequency	0.9945	Pass
Run	0.0411	Pass
Rank	0.0000*	Fail
DFT	0.0218	Pass
Serial test	0.9945	Pass
Linear complexity test	0.5986	Pass
Longest run test	0.0000*	Fail
Approximate entropy test	0.0300	Pass
Cumulative sum test	0.9996	Pass
Random excursions test	0.0500	Pass

* NIST fail when *p* value is 0.000

Table 7 NIST statistical test results for TRNG (FAT-2)

NIST test	<i>P</i> -value	Result
Frequency	0.7216	Pass
Block frequency	0.9945	Pass
Run	0.0670	Pass
Rank	0.0000*	Fail
DFT	0.0218	Pass
Serial test	0.9945	Pass
Linear complexity test	0.2516	Pass
Longest run test	0.0000*	Fail
Approximate entropy test	0.0440	Pass
Cumulative sum test	0.9996	Pass
Random excursions test	0.0500	Pass

* NIST fail when *p* value is 0.000

5.1 Pre-processing experiment setup used in designing TRNG

The reliability of the random number generated by the proposed TRNG's design is verified using a set of statistical procedures. A digital storage oscilloscope (DSO-X3012A) was used to capture the random bitstream's generated output waveform and to analyze the jitter produced by the various entropy sources. Using (VHDL), VHSIC Hardware Description Language random bits generated by the TRNG architecture are stored in a text file. To assure unpredictability, random numbers from the series are subjected to NIST tests using MATLAB version R2015a. A sequence with a *P*-value $\geq$ of 0.001 is deemed to pass the NIST test [30]. Table 6 for FAT-1 and Table 7 for FAT-2 exhibit the NIST test results for the TRNG's design, proving that the generated sequence is truly a random bit sequence.

The proposed TRNG (FAT-1) is depicted schematically in Fig. 14, while the TRNG (FAT-2) is depicted schematically in Fig. 15. All schematic diagrams were created with Vivado v.2015.2 and simulated on an XC7A35T-CPG236-1 device (Artrix -7) FPGA board. As the number of slice LUTs utilized by proposed FAT-1 TRNG is zero, so the area (LUTs) of FAT-1 designed is n/a as shown in Table 5.

The TRNG output waveforms captured by DSO are depicted in Figs. 16 and 17.

The NIST test evaluates the resulting bitstream's randomness and statistical features, providing critical evidence for the randomness of TRNG's based on ADPLL with FIR as loop filter.

Fig. 14 Schematic diagram of FAT-1 TRNG

Fig. 15 Schematic diagram of FAT-2 TRNG

Fig. 16 TRNG output waveform of FAT-1 design

Fig. 17 TRNG output waveform of FAT-2 design

6 Comparison between proposed TRNG based on ADPLL with existing TRNG's

The performance of various TRNGs is compared in Table 8, as is the synthesis report's comparison of various TRNG architectures in Table 9. Our proposed design made better use of available hardware resources. Additionally, the power consumption of FAT-1 is 0.072 W and FAT-2 is 0.074 W, which is significantly less than that of other literature. Even though fewer hardware resources

and lower power consumption are used, the overall throughput of the design remains unaffected, as shown in Table 9 producing 200 Mbps of both the designs even after post-processing.

The author of [6] discusses a TRNG based on PLL and other FPGA primitives. The primary source of entropy is jitter from PLLs and metastability from FF's. The author claimed that the proposed design would consume less power and take up less space but achieving a throughput of 100 Mbps. However, PLL consumed more power and took up more space [29] than ADPLL, which means

Table 8 Performance comparison among TRNG architecture

Reference	Entropy source	Device	Hardware resource	Throughput	Post-processing
[6]	Metastability/jitter	Xilinx XCKU040	1 PLL 5 Primitives 5 Slices	100 Mbps	Yes
[31]	Metastability	Altera Cyclone III	511 LUTs	133 Mbps	Yes
[32]	Ring oscillator	Xilinx XC5VLX50T	147 LUTs	100 Mbps	Yes
[33]	RS-Latch	Xilinx XC4VFX20	580 Slices	12.5 Mbps	No
[34]	Chaotic ring oscillator	Xilinx XC6SLX16	44 LUTs	125 Mbps	Yes
[35]	PLL jitter	Altera Stratix	120 LE	> 1 Mbps	Yes
[36]	Metastability	Virtex-5	n/a LUT 64 Latches	20 Mbps	No
[37]	Free running ring oscillator	Xilinx Spartan-6 FPGA	10 LUTs and 5 FFs	1.15 Mbps	No
[10]	Metastability/jitter	Xilinx XC7A35T	2 LUTs and 1 Slice	680.7 Mbps	No
[10]	Metastability/jitter	Xilinx XC7A35T	2 LUTs and 1 Slice	676 Mbps	No
FAT-1	ADPLL jitter/metastability	Xilinx XC7A35T-CPG236-1	n/a LUT	200 Mbps	Yes
FAT-2	ADPLL jitter/metastability	Xilinx XC7A35T-CPG236-1	1 LUT	200 Mbps	Yes

Table 9 Synthesis results comparison among different TRNG architecture

Parameter	Paper [38]	Paper [12]	Paper [39]	Paper [40]	Paper [10]	Paper [10]	FAT-1	FAT-2
Speed	10 MHz	125 MHz	1.4 MHz	10 MHz	680.7 MHz	676 MHz	200 MHz	200 MHz
Power consumption	0.0023 W	0.095 W	0.0039 W	0.0036 W	0.076 W	0.078 W	0.072 W	0.074 W
Post-processing	NO	NO	NO	YES	NO	NO	YES	YES
Final throughputs rate	10 Mbps	125 Mbps	1.4 Mbps	10 Mbps	680.7 Mbps	676 Mbps	200 Mbps	200 Mbps

that ADPLL-based designs are far superior when designing a reliable TRNG, while in [32], ring oscillators are used to generate jitters, and an auxiliary source of randomness (ASR) is used to increase the efficiency of the outcome bitstreams, which greatly aids in increasing the degree of unpredictability of TRNG, as the author claims. According to the author of [34], a chaotic ring oscillator-based TRNG can be designed to be simple, resource-efficient, and reliable while still achieving a throughput of 125 Mbps. In comparison with the proposed TRNG architecture, [37] utilizes more number of LUTs and achieves lower throughput.

In [35], a prototype chip with a throughput of 10 Mbps was fabricated using a standard digital 0.018 m *n*-well Complementary Metal Oxide semiconductor (CMOS) process. As described in [12], based on the designed technique in 2-/spl mu/m CMOS technology that generates bit rate sequences up to 1.4 MHz. [39] used typical 0.18-/spl mu/m *n*-well CMOS to fabricate a model with a clock speed of 10 MHz. A prototype has been designed and fabricated in [38] using HHNEC's 0.25 m *e*-flash processes with a double ring oscillator. The proposed fabricated prototype achieved 125 Mbps throughput. Our proposed TRNG architectures are written in VHDL and incorporated on the

(XC7A35T-CPG236-1) FPGA board, resulting in a significant increase in overall speed compared to the existing TRNG architecture. Since our proposed design uses fewer LUTs and FF, the design complexity is reduced compared to existing TRNG architectures.

7 Conclusion

By utilizing all of the seed entropy generated by the ADPLL, flip-flop, and other hardware resources, a more secure and reliable TRNG is generated. Furthermore, a TRNG based on ADPLL generated a more secure random bitstream. Despite their high cost and equipment dependence, TRNGs are preferred and utilized in a wide variety of applications that demand a high level of security, such as secure communication and cryptography. We were able to reduce power consumption and use fewer hardware resources (zero LUT for FAT-1 and one LUT for FAT-2) while maximizing the performance of the FPGA board by introducing FIR-based ADPLL into the TRNG designs, as shown in Table 8. Digital storage oscilloscopes are used to capture and analyze the output waveform (DSO-X3012A). The

implementation and simulation are carried out on a Xilinx artrix-7 (XC7A35T-CPG236-1) FPGA board using Vivado v.2015.2. Following post-processing, both designs generate a throughput of 200 Mbps with a power consumption of 0.072 W for FAT-1 TRNG and 0.074 W for FAT-2 TRNG. The data bitstream generated after post-processing passed the National Institute of Standards and Technology (NIST) test which shows a high degree of randomness. With this work, the future of securing security via the usage of FIR-based ADPLL -TRNG's technologies appears to be bright, making it a more reliable and secure contender for a large range of applications like cybersecurity, banking security, Internet of Things, and Internet of Everything (IOE), etc.

Acknowledgements The author wishes to express his heartfelt appreciation to Dr. Manoj Kumar for his invaluable support and unwavering guidance throughout the writing of this paper.

Author contributions All authors contributed equally.

Funding Not applicable.

Declarations

Conflict of interest The authors have no conflicts of interest to declare. All co-authors have seen and agreed with the contents of the manuscript. We certify that the submission is original work and is not under review at any other publication.

Consent to participate Not applicable.

Consent for publication Not applicable.

Human or animal rights This article does not contain any studies with human participants or animals performed by any of the authors.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Zhou L, Tan F, Yu F (2019) A robust synchronization-based chaotic secure communication scheme with double-layered and multiple hybrid networks. *IEEE Syst J* 14:1–12
- Gu K, Jia W, Wang G, Wen S (2017) Efficient and secure attribute-based signature for monotone predicates. *Acta Inform* 54(5):521–541
- Xia Z, Fang Z, Zou F, Wang J, Sangaiah AK (2019) Research on defensive strategy of real-time price attack based on multiperson zero-determinant. *Secur Commun Netw* 2019:13
- Gu K, Wu N, Yin B, Jia W (2019) Secure data query framework for cloud and fog computing. *IEEE Trans Netw Serv Manag* 17:332–345
- Gu K, Wang Y, Wen S (2017) Traceable threshold proxy signature. *J Inf Sci Eng* 33:63–79
- Stanchieri GDP, 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* 105:15–23
- Buchmann J, Dahmen E, Szydlo M (2009) Hash-based digital signature schemes. *Post Quantum Cryptogr*. https://doi.org/10.1007/978-3-540-88702-7_3
- Tobin P, Tobin L, McKeever M, Blackledge J (2017) On the development of a one-time pad generator for personalising cloud security. In: conference papers <https://doi.org/10.21427/D7VF9V>
- Internet resource: AMD random number generator library. <https://developer.amd.com/wordpress/media/2013/12/AMD-Random-Number-Generator-User-Guide>
- Meitei HB, Kumar M (2021) FPGA implementation of true random number generator architecture using all digital phase-locked loop. *IETE J Res*. <https://doi.org/10.1080/03772063.2021.1963333>
- Handschuh H, Paillier P, Stern J (1999) Probing attacks on tamper-resistant devices. *Cryptogr Hardw Embed Syst*. https://doi.org/10.1007/3-540-48059-5_26
- 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>
- Vasylytsov I, Hambardzumyan E, Kim Y-S, Karpinsky B (2008) Fast digital TRNG based on metastable ring oscillator. *Cryptogr Hardw Embed Syst*. https://doi.org/10.1007/978-3-540-85053-3_11
- Radhapuram S, Yoshihara T, Matsuoka T (2019) Design and emulation of all-digital phase-locked loop on FPGA. *Electronics* 8(11):1307. <https://doi.org/10.3390/electronics8111307>
- Koyuncu İ, Turan Özcerit A (2017) The design and realization of a new high speed FPGA-based chaotic true random number generator. *Comput Electr Eng* 58:203–214. <https://doi.org/10.1016/j.compeleceng.2016.07.005>
- Chaudhary AK, Kumar M (2017) Design and implementation of ADPLL for digital communication applications. *IEEE Conf Convergen Technol*. <https://doi.org/10.1109/12CT.2017.8226159>
- Lata K, Kumar M (2013) ALL digital phase-locked loop (ADPLL): a survey. *Int J Future Comput Commun*. <https://doi.org/10.7763/ijfcc.2013.v2.225>
- Kumar M, Lata K (2012) All digital phase locked loop (ADPLL): a survey. In: proceeding the 4th IEEE international conference on electronics computer technology, Kanyakumari
- Vallabhaneni S, Attri S. Design of an all-digital PLLcore on FPGA, Sci/Engr SF, AISG/AISD, IISU, ISRO India
- Di J, Yuan JS, Demara R, (2003) High throughput power-aware FIR filter design based on fine-grain pipelining multipliers and adders. In: IEEE computer society annual symposium on VLSI (ISVLSI'03) 0-7695-1904-0/03
- Kaur R, Raman A, Singh H, Malhotra J (2011) Design and implementation of high speed IIR and FIR filter using pipelining. *Int J Comput Theory Eng* 3(2):1793–8201
- Sun Y, Kim MS (2011) A high-performance 8-tap FIR filter using logarithmic number system. In: conference communications (ICC) IEEE international conference, pp 5–9

23. Jain R, Yang PT, Yoshino T (1991) A computer-aided design system for high performance FIR filter integrated circuits. *IEEE Trans Signal Process* 39(7):1655
24. Tsao Y-C, Choi K (2012) Area-efficient VLSI implementation for parallel linear-phase FIR digital filters of odd length based on fast FIR algorithm. *IEEE Trans Circuits Syst II Express Br* 59(6):371
25. Bar J, Mukherjee M (2012) A comparative study on direct form -1, broadcast and fine grain structure of FIR digital filter. *Int J Electr Commun Soft Comput Sci Eng* 2(5):14
26. Manoj K, Singh SD (2021) Design and implementation of third order low pass digital FIR filter using pipelining retiming technique. *Int J Adv Technol Eng Explor* 10:178–184. <https://doi.org/10.35940/ijeat.D2446.0410421>
27. Kohlbrenner P, Gaj K (2004) An embedded true random number generator for FPGAs. In: *proc of 12th int symp on field prog gate arrays*, Monterey, pp 71–78
28. Creț O, Orfi T, Suciu A (2012) Implementing true random number generators based on high fanout nets. *Rom J Inf Sci Technol* 15:277–298
29. Kim MS (2012) 0.18μm CMOS low power ADPLL with a novel local passive interpolation time-to-digital converter based on tri-state inverter. *Northeastern edu*. <https://repository.library.northeastern.edu/files/neu:1338>
30. Bassham L, Rukhin A, Soto J, Nechvatal J, Smid M, Barker E, Leigh S, Levenson M, Vangel M, Banks D, Heckert N, Dray J (2010). A statistical test suite for random and pseudorandom number generators for cryptographic applications. <https://csrc.nist.gov/publications/detail/sp/800-22/rev-1a/final>
31. Jun B, Kocher P (1999) Intel random number generator. In: *rambus; cryptography research, inc, white paper prepared for intel corporation*, <https://www.rambus.com/intel-random-number-generator/>
32. Jessa M, Matuszewski L (2011) Enhancing the randomness of a combined true random number generator based on the ring oscillator sampling method. *IEEE*. <https://doi.org/10.1109/ReConFig.2011.35>
33. Hata H, Ichikawa S (2012) FPGA implementation of metastability-based true random number generator. *IEICE Trans Inf Syst* 95(2):426–436. <https://doi.org/10.1587/transinf.e95.d.426>
34. 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>
35. Fischer V, Drutarovský M, Šimka M, Bochar N (2004) High performance true random number generator in altera stratix FPLDs. *Field Progr Log Appl*. https://doi.org/10.1007/978-3-540-30117-2_57
36. Ben-Romdhane M, Graba T, Danger J-L (2013) Stochastic model of a metastability-based true random number generator. *Trust Comput*. https://doi.org/10.1007/978-3-642-38908-5_7
37. Yang B, Rožic V, Grujić M, Mentens N, Verbaauwhede I (2018) ES-TRNG: a high-throughput, low-area true random number generator based on edge sampling. *IACR Trans Cryptogr Hardw Embed Syst*. <https://doi.org/10.13154/tches.v2018.i3.267-292>
38. Bucci M, Germani L, Luzzi R, Trifiletti A, Varanonuovo M (2003) A high-speed oscillator-based truly random number source for cryptographic applications on a smartcard IC. *IEEE Trans Comput* 52(4):403–409
39. Petrie CS, Connelly JA (2000) A noise-based IC random number generator for applications in cryptography. *IEEE Trans Circuit Syst I Fundam Theory Appl* 47(5):615–621. <https://doi.org/10.1109/81.847868>
40. Bucci M, Germani L, Luzzi R, Tommasino P, Trifiletti A, Varanonuovo M (2003) A high-speed IC random-number source for smartcard microcontrollers. *IEEE Trans Circuit Syst I Fundam Theory Appl* 50(11):1373–1380. <https://doi.org/10.1109/tcsi.2003.818610>

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.