Lightweight Side-Channel-Resistant True Random Number Generator Design for FPGA-Based Edge-AI Platforms

Kieu-Nhat-Anh Pham[1,2] and Cuong Pham-Quoc[1,2(✉)]

[1] Ho Chi Minh City University of Technology (HCMUT), Ho Chi Minh City, Vietnam
{anhpkn,cuongpham}@hcmut.edu.vn
[2] Vietnam National University - Ho Chi Minh City (VNU-HCM), Ho Chi Minh City, Vietnam

Abstract. Random number generators are fundamental components in modern cryptographic systems, secure communications, and hardware security modules. Although pseudorandom number generators (PRNGs) are widely used due to their efficiency and low cost, they are inherently deterministic and unsuitable for high-security applications requiring true unpredictability. True random number generators (TRNGs) exploit physical entropy sources to generate non-deterministic random sequences. In addition, side channel analysis (SCA) attack techniques will be introduced to enhance the necessary TRNG in hardware security, especially with IoT devices. This paper aims to describe about the TRNG design for FPGA as a hiding countermeasure technique against SCA, and to simulate again an SCA attack for an unprotected AES algorithm.

Keywords: Field-programmable gate array (FPGA) · True Random Number Generator (TRNG) · Random Number test · Side Channel Analysis (SCA)

1 Introduction

The Internet of Things (IoT) has transformed industries and daily lives through increased connectivity of devices. The expansion of IoT devices has led to increased security concerns, necessitating strong protection measures for sensitive data and critical infrastructure. In contrast to invasive techniques that require a physical breach of a device, contemporary cryptographic attacks often employ side-channel monitoring. This methodology exploits the 'leakage' of physical manifestations such as power consumption, electromagnetic radiation, or timing variations that are produced during computational processes. By analyzing these unintentional emissions, adversaries can non-intrusively reconstruct sensitive cryptographic keys or embedded security parameters without compromising the device's structural integrity [30]. The traditional IoT framework is

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2027
O. Gervasi et al. (Eds.): ICCSA 2026 Workshops, LNCS 16762, pp. 195–212, 2027.
https://doi.org/10.1007/978-3-032-30524-4_13

structured into three layers, namely application, network, and perception layer. Each tier of the IoT architecture faces unique security threats that the application layer is vulnerable to SQL injection [20] and phishing attack [17], the network layer is susceptible to state-based [13] and AI-based [15], and the perception layer is targeted by physical-world disruptions such as RF jamming [9] and side-channel attacks [14]. However, the scope of this paper is limited to the Power Analysis Attacks (PAAs) model, a form of side-channel attack (SCA); in this paper, we will focus on Correlation Power Analysis (CPA).

The main contributions of our paper can be summarized in three folds.

1. We introduced our efficient FPGA-based TRNGs hardware core to enhance security level for edge AI platforms.
2. We investigate on Side-Channel Analysis attack may occur on edge devices
3. We analyze our experiment results based on the NIST and AIS tests for TRNGs quality

In order to increase resistance against side channel attacks, random number generators (RNGs) are essential security tools that leverage physical phenomena to produce randomness, which is vital for cryptographic functions such as authentication and key generation [5]. Beyond encryption, they are utilized in circuit simulation and as masking agents to defend against side-channel attacks [32]. To initiate a challenge-response sequence, an integrated circuit transmits a unique, unpredictable "nonce" to a secure server, typically generated by an on-chip hardware True Random Number Generator (TRNG) that produces a fresh binary combination for every session. An efficiency hardware TRNG design needs to be unpredictability, a modern TRNG is expected to be compact, fast, self-testable, and easily synthesizable. A good design of TRNGs must pass through the statistical test suites of the National Institute of Standards and Technology (NIST) and the BSI. This paper focuses on the BSI statistical test rather than the NIST test.

The rest of the paper is organized as follows. Section 2 discusses related work in the literature. Section 3 discusses the key considerations for designing a TRNG. Section 4 presents the simulation of CPA on unprotected AES algorithms and the hidden countermeasure AES algorithm. We design our own hardware TRNGs with the BIS statistical test in Sect. 5. Finally, we conclude our paper in Sect. 6.

2 Background and Related Work

As these vulnerabilities threaten the integrity of cryptographic systems, the development of robust countermeasures has gained prominence. A crucial component of secure cryptographic implementations is the True Random Number Generator (TRNG), which is essential for generating unpredictable keys and nonces. Designing an efficient and secure TRNG on FPGAs presents unique challenges, particularly in achieving high throughput while maintaining low resource utilization and resilience against side-channel attacks [35]. Effective TRNG designs

must leverage various sources of entropy to enhance randomness and ensure compliance with stringent security standards, such as FIPS 140 and Common Criteria EAL5 [21]. In the context of Edge AI, where devices operate in resource-constrained environments, the need for lightweight and efficient cryptographic solutions becomes even more critical. Edge AI integrates intelligence directly into devices on the network edge, allowing real-time processing while addressing privacy concerns and minimizing latency [10]. As such, the implementation of secure cryptographic mechanisms, including TRNGs, is paramount to protect sensitive data processed in these distributed systems. The evolution of security measures in edge computing environments requires adaptive and proactive strategies to counter emerging cyber threats. This landscape highlights the importance of establishing a secure and reliable foundation for cryptographic applications on FPGA-based Edge AI platforms, balancing the trade-offs between security, resource efficiency, and usability in a dynamic environment [7].

Early FPGA-based True Random Number Generators (TRNGs) predominantly rely on ring oscillator (RO) jitter, where multiple free-running oscillators are combined to extract entropy for cryptographic applications. However, these RO-based designs can suffer from correlation and layout-dependent bias, motivating the use of spatially distributed oscillators and on-chip post-processing such as XOR mixing or hashing to meet standard randomness tests [1].

Metastability-based TRNGs have been proposed as an alternative or complementary entropy source, exploiting unstable latch or flip-flop states to generate high-quality random bits on commercial FPGAs. Recent work demonstrates hybrid entropy sources that combine RO jitter with metastability and use automatic phase tuning or dedicated FPGA primitives (e.g., CARRY chains, clock managers) to enhance throughput and statistical quality for hardware security applications [11,19].

The side-channel vulnerability of FPGA TRNGs has been increasingly investigated, with studies showing that power and electromagnetic analysis can reveal internal behavior or bias in RO-based and coherent-sampling TRNGs. To address this, several authors propose SCA-resistant RO structures such as Galois Ring Oscillators, dynamic partial reconfiguration, and run-time monitoring to detect or mislead ongoing attacks while preserving overall system functionality [2,6,12].

In parallel, FPGAs have emerged as key platforms for edge-AI acceleration, where security primitives must coexist with AI workloads under tight power and resource constraints. Recent industrial and academic reports highlight the need for integrated low-overhead security blocks—including TRNGs—within FPGA-based edge-AI systems, but comprehensive designs that explicitly co-optimize entropy quality, side-channel resistance, and AI performance are still limited, leaving significant room for novel architectural contributions [16,24].

3 TRNG Design: Key Considerations

3.1 Hiding and Masking Countermeasure Against Power Analysis Attack

Investigation of the power consumption of individual AES components reveals that the S-box within the SubBytes stage is the most power-intensive element, accounting for 75% of total power usage in a 1-round-per-cycle loop architecture [30]. Because the S-box is responsible for three-quarters of the power drain, it is the most critical area for applying Hiding or Masking countermeasures. Optimizing the efficiency of the S-box or masking its signature is the most effective way to secure the entire AES engine. Following [30], an AES design without a countermeasure is called "unprotected" to enhance the necessity of countermeasures technique. The countermeasure techniques are defined into two categories that are Hiding and Masking techniques.

The hiding countermeasure technique is used to obscure the relationship between the device's operations and its emissions. Hiding uses two different vertical (amplitude) and horizontal scales (time) to achieve the goal. The vertical scale aims to ensure uniform or randomized power consumption in each clock cycle. In this approach, each operation of the cryptography core consumes an identical level of power in every clock cycle, independent of the input or data values. However, horizontal hiding disrupts the temporal alignment of leakage points. In this approach, the device is designed so that its power consumption varies randomly, meaning that a different amount of power is drawn in each clock cycle. Consequently, the operations of the cryptographic algorithm are executed at varying time instants during each run.

The masking countermeasure technique is to mitigate leakage by dividing sensitive intermediate values into independent shares. This ensures that the power consumption of the device does not depend on the intermediate values being processed. A key advantage of this approach is that its implementation at the algorithm level does not alter the overall power profile of the cryptographic device. Masking countermeasures can be applied at both the algorithmic and circuit levels.

3.2 Random Number Generator Architecture for Countermeasure

In order to design good TRNGs, it is necessary to understand the concept of random numbers. Random numbers are essential for most cryptographic applications, as they are used to generate session keys, signature parameters, nonces, challenges, and blinding or masking values to protect against implementation attacks. If the random number generators (RNGs) used for these purposes are weak, they can severely compromise the security of cryptographic systems. This risk highlights the importance of conducting reliable and reliable security evaluations of RNGs to ensure their suitability for cryptographic use. Therefore, [23] proposed a classification of RNGs to satisfy the hiding or masking countermeasures required in hardware security techniques.

DRNGs, also known as deterministic random number generators or PRNGs (pseudo-random number generators), take a short truly random sequence called a seed—typically obtained from an entropy source—and deterministically extend it into a very long sequence of seemingly random bits. In contrast, PTRNGs (physical true random number generators) rely on genuine physical noise sources, such as dedicated hardware or measurements from physical experiments, to produce randomness. NPTRNGs (non-physical true random number generators) also aim to generate true randomness, but do so without dedicated hardware, instead extracting entropy from non-physical noise sources like system data or user interactions. Figure 1 describes the classification of Random Number Generators.

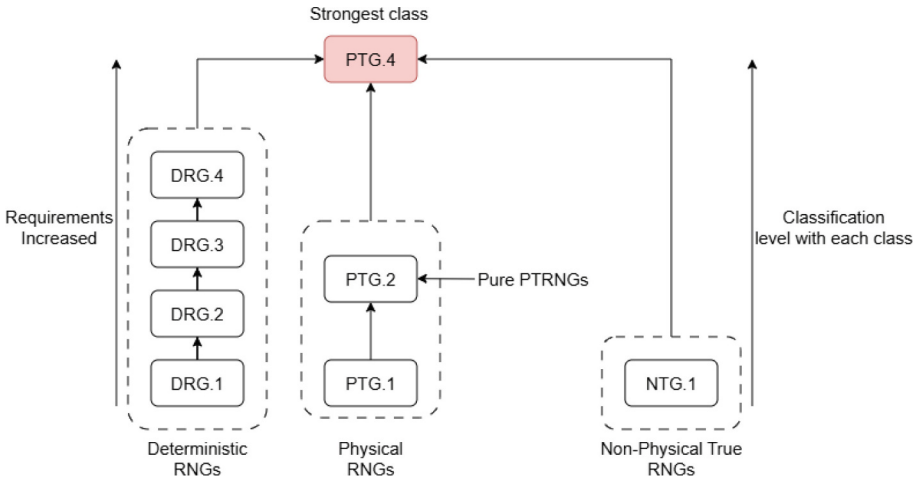

Fig. 1. New AIS 20/31 - Functionality classes of AIS 20/31 [23]

In order to achieve ideal TRNGs, a statistical test suite known as AIS-31 has been developed to evaluate the quality of RNG designs. The AIS-31 standard consists of two procedures, A and B. Procedure A includes six groups of tests designed to verify statistically inconspicuous behavior in the output sequence, while Procedure B includes three groups of tests applied to the internal random bits of the generator. Table 1 presents a detailed description of these tests.

Table 1. AIS-31 statistical tests

No.	Purpose	Requirement
Test 0	Disjointedness test	Subsequent members are pairwise different
Test 1	Monobit test	The number of ones must be between 9,654 and 10,346
Test 2	Poker test	Distribution of 4-bit tuples checked for 15° of freedom
Test 3	Runs test	Runs of 1, 2, 3, 4, 5, and 6 ones (zeros) are checked for expected occurrences
Test 4	Longest run test	No single run can be larger than 34
Test 5	Auto-correlation test	The overlap of the bit stream in the latter half of the sequence is compared to the sequence with the largest overlap in its first half
Test 6	Uniform distribution test	6a: Monobit test to ensure the number of ones is between 25–75% 6b: A special case of Test 7 with a width of 2
Test 7	Homogeneity test	Collect 10,000 occurrences of runs less than the given width and check for the expected transition probabilities. 7a: Test with a width of 3 7b: Test with a width of 3
Test 8	Entropy test	Accumulate the nearest predecessor distance between byte values in a 256,000 + 2,560 bit sequence, and calculate the empirical entropy

3.3 Correlation Power Analysis

[18] proposed Differential Power Analysis (DPA) attacks, which can recover secret keys by analyzing detailed cryptographic operations and statistically examining their power consumption traces. Later, [3] also introduced Correlation Power Analysis (CPA), which uses Pearson's correlation coefficient to improve the performance of power analysis attacks. The most widely used leakage models are Hamming-based, including the Hamming Weight (HW) and Hamming Distance (HD) models. These models often appear in leakage from software running on small microcontrollers and from medium-scale cryptographic implementations on FPGA and ASIC platforms [31].

The power consumption model transition is described in Table 2. The Hamming Weight (HW) model assumes that only bits with a value of 1 contribute significantly to power consumption, meaning that the number of ones in the data determines the leakage. In contrast, the Hamming Distance (HD) model focuses on bit transitions, where both changes $0 \rightarrow 1$ and $1 \rightarrow 0$ result in similar power consumption. Therefore, HW emphasizes the static value of bits, while HD captures dynamic switching activity between states.

Table 2. Power Consumption Models Transition

Transitions	Hamming Weight	Hamming Distance
$0 \rightarrow 0$	0	0
$0 \rightarrow 1$	1	1
$1 \rightarrow 0$	0	1
$1 \rightarrow 1$	1	0

Under a brute-force attack, an attacker would need to try up to $2^{128} \approx 3.4028 \times 10^{38}$ possible keys, which is computationally infeasible. However, in a CPA attack (chosen-plaintext), the attacker only needs $16 \times 2^8 = 4096$ key guesses to recover the secret key. Let N be the number of bytes in plaintext, D the number of measured traces,K the number of hypothetical keys, and T the number of samples. The main steps of CPA are described in Fig. 2.

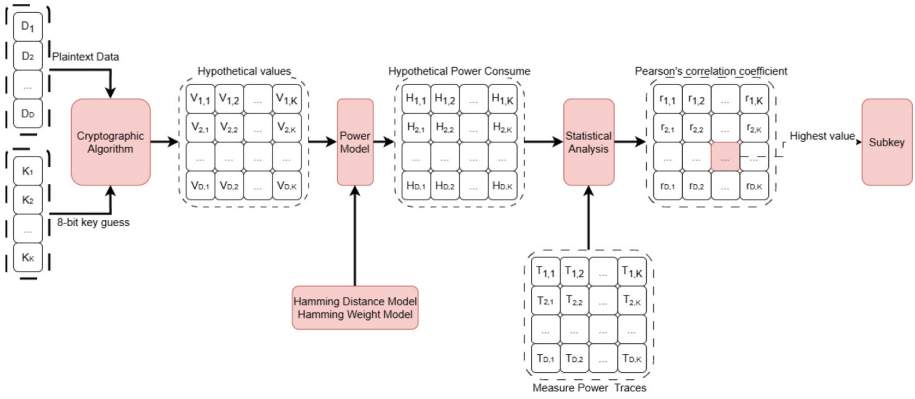

Fig. 2. Main steps of CPA

An intermediate value of the encryption algorithm is first selected. This value must depend on both the secret key and a known variable. Next, a matrix of hypothetical intermediate values V is computed. Let $d_m \in \{0, D-1\}$ denote one byte of the m-th plaintext input, and let $k_i \in \{0, K-1\}$ represent the i-th key hypothesis. The potential output of the S-Box in the first round is then calculated using Eq. 1.

$$v_{m,i} = S(d_m \oplus k_i) \tag{1}$$

Therefore, a matrix of hypothetical intermediate values V is defined. This matrix is then mapped to a hypothetical leakage matrix H using HD or HW power models to estimate the power consumption of the device. If the attacker has additional knowledge about the target device, such as the netlist, more

accurate electromagnetic (EM) simulations of the algorithm can be performed. Improved EM simulations reduce the number of EM traces required to recover the secret key.

Finally, the hypothetical leakage matrix H is compared with the measured power trace matrix T of size $D \times T$. The results of this comparison are stored in a matrix R, which has dimensions $K \times T$. Each comparison between h_i and t_j can be performed using various statistical methods. Each element $r_{i,j}$ in R represents the correlation result between h_i and t_j, and a higher value of $r_{i,j}$ indicates a stronger match.$r_{i,j}$ is calculated using Eq. 2.

$$r_{i,j} = \frac{\sum_{m=0}^{D-1}(h_{m,i} - \overline{h_i})(t_{m,j} - \overline{t_j})}{\sqrt{\sum_{m=0}^{D-1}(h_{m,i} - \overline{h_i})^2 \sum_{m=0}^{D-1}(t_{m,j} - \overline{t_j})^2}} \tag{2}$$

4 Simulation SCA Attack

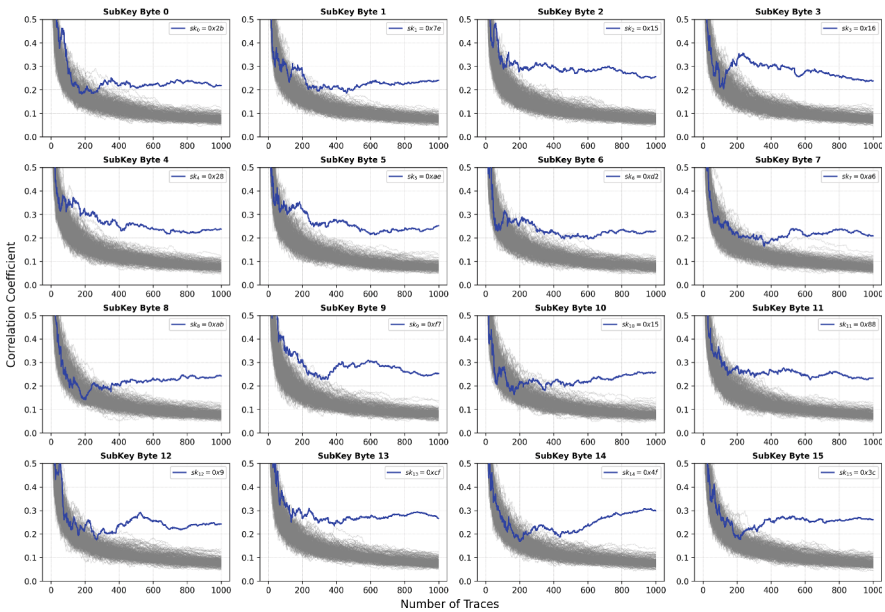

Fig. 3. Attack on simulated "unprotected" AES

Figure 3 illustrates the results obtained from the CPA attack performed on the unprotected implementation of AES. It can be observed that the correct subkey is clearly distinguishable after approximately 1000 traces. This conclusion is drawn based on the Pearson correlation coefficient, where the correct key hypothesis produces the highest correlation value compared to all other candidates. As the number of traces increases, the correlation corresponding to the

true key gradually separates from the incorrect hypotheses, eventually emerging as the dominant peak. This shows that, without any countermeasure, the secret key can be efficiently recovered with a relatively small number of measurements.

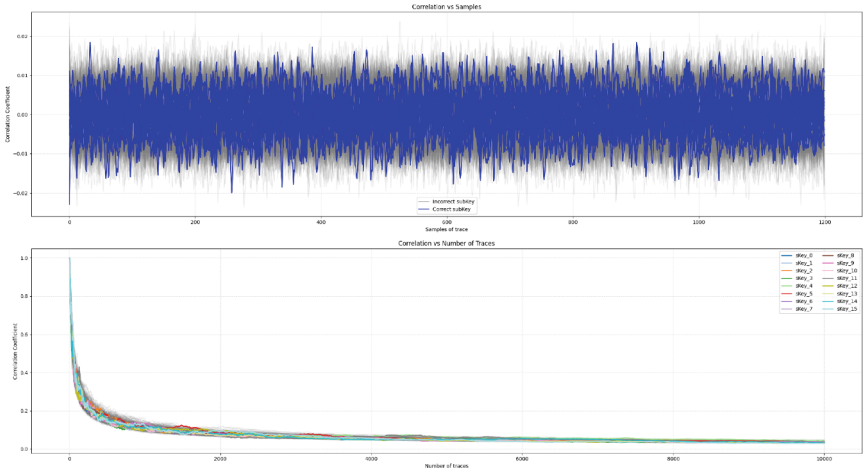

Fig. 4. Attack on simulated hiding countermeasure AES

In contrast, when a hiding countermeasure is incorporated into the simulation, the attack performance is significantly degraded. As shown in Fig. 4, the correlation values of the correct key hypothesis remain indistinguishable from those of the incorrect hypotheses for a much larger number of traces. Even as the number of collected traces increases, the separation between the correct and incorrect key candidates is substantially reduced. The results indicate that the secret key is not reliably exposed until approximately 10,000 traces are used.

5 Lightweight FPGA-Based TRNGs for Edge-AI Platforms

5.1 FPGA Implementation

In this section, we concentrate on the design of TRNG sources intended to hide countermeasures on target devices. Based on our investigation and analysis, we propose a TRNG architecture that combines both a Ring Oscillator (RO) and a Ring Generator (RG). The work in [25] introduced a TRNG jitter sampler that integrates an RO stage as an entropy source with an RG stage. Figure 5 illustrates the overall concept of this TRNG architecture.

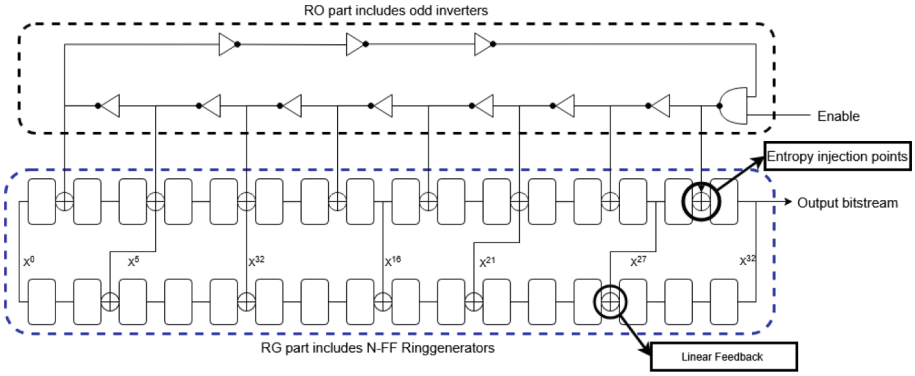

Fig. 5. The core ring-generator-based TRNG with the characteristic polynomial $f(x) = x^{32} + x^{27} + x^{21} + x^{16} + x^{10} + x^5 + 1$

The RG stage is implemented using a Linear Feedback Shift Register (LFSR) structure, while the RO stage consists of a chain of odd-numbered inverters to inject entropy into the RG stage. In our design, an inverter is replaced with a NAND gate, allowing the injection of entropy into the RG stage to be enabled or disabled.

Depending on the XOR gate configuration, LFSR.can be categorized into two types: (a) Fibonacci LFSR and (b) Galois LFSR. In our work, the RG structure is optimized by using a Galois configuration arranged in a ring topology. This optimization is motivated by the critical path limitations associated with the Fibonacci structure and the fan-out issues present in the conventional Galois form.

In this section, a 7-inverter RO generates raw random bits from its jitter, which are then fed into a 64-bit linear feedback shift register (LFSR) defined by the polynomial in Eq. 3.

$$f(x) = x^{64} + x^{52} + x^{39} + x^{26} + x^{14} + x^7 + 1. \tag{3}$$

The LFSR whitens the RO output, spreading entropy across 64 bits and producing a uniform, maximal-length random sequence suitable for TRNG applications. The design is implemented on the Xilinx Artix-7 FPGA chip using the Digilent Arty A7-100T.

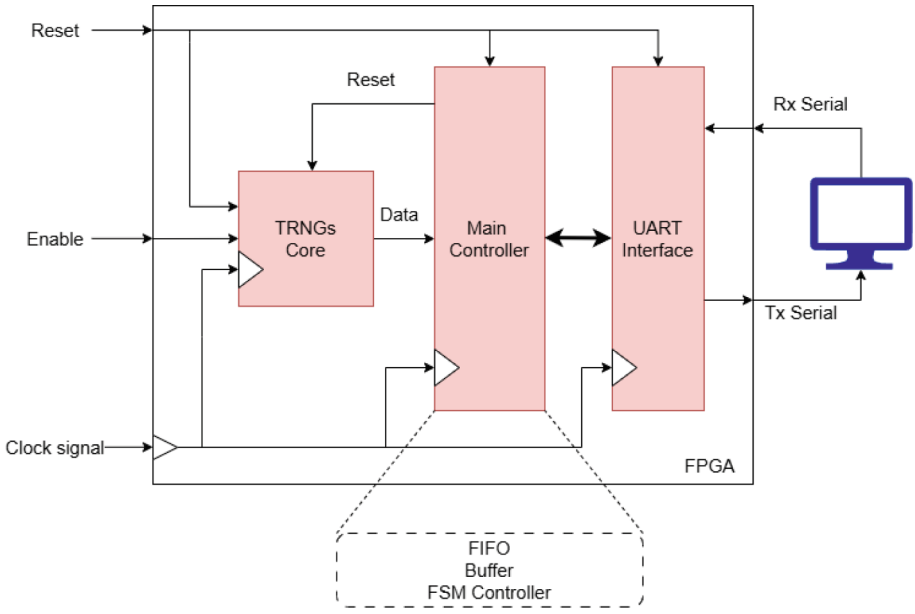

Fig. 6. Hardware implementation

Figure 6 shows the architecture of the design. The implementation uses a total of 10 LUTs and 64 FFs, which uses around 8 SLICEs, making it a lightweight and efficient TRNG source suitable for integration into our next AES core design. We declare the Enable signal as the trigger signal for the RO block to generate entropy, since the first inverter stage is implemented as an AND gate. The oscillation frequency depends on the total number of inverters configured in this experiment, which consists of one NAND gate and ten inverter stages.

The initial validation of the random source involves assessing key statistical properties of the proposed TRNG immediately after start-up, to account for potential correlations in the generated sequences. A preliminary dataset of 1 million bits was analyzed before applying the official standard tests. Among common RNG evaluations, the autocorrelation test is often the most challenging, so following [22], we examined the correlation between each bit and its lagged values. The acceptance threshold for this test is a maximum correlation coefficient of 0.3, indicating the degree of dependence between a variable and its past values [4,33]. As shown in Fig. 7, the observed autocorrelation coefficients are close to zero, indicating that the generated bits are largely independent. This low autocorrelation further suggests that correlation-based attacks would be ineffective against the proposed TRNG upon start-up. Finally, Fig. 8 illustrates the spatial distribution of 1 million bits arranged on a 1000×1000 grid from the start-up dataset.

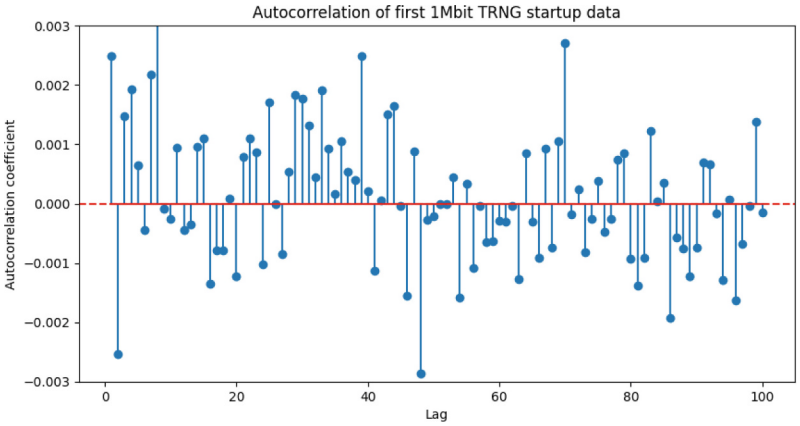

Fig. 7. Auto corelation test

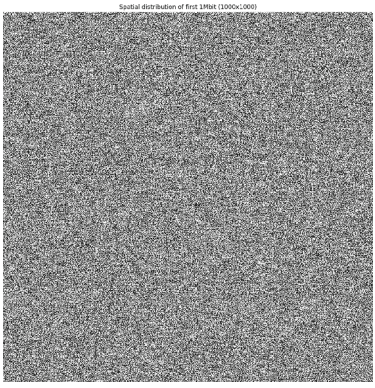

Fig. 8. Distribution of raw data

The results indicate that our TRNG design satisfies all the requirements for a TRNG. In future work, we plan to integrate this design with an AES core, where it can serve as a hiding countermeasure within the system. Table 3 presents the utilization of post-implementation resources of our FPGA design. The LUT usage is higher than in previous designs due to the inclusion of an additional controller and a UART interface.

5.2 Experimental Results

We obtain the bit sequences from the FPGA and then transmit them to the PC through the UART interface. Additionally, the sampling clock rate for the TRNG source was reduced by half to 50 MHz to ensure that the TRNG achieves reliable and stable performance. The results show that we passed all the test from AIS-31 with an ideal entropy TRNG approx 0.998 perbit with the collection data. In

Table 3. FPGA Resource Utilization

Resource	LUT	FF	IO	DSP
Utilization	196	222	7	0
Available	63400	126800	210	0
Utilization %	0.31%	0.18%	3.33%	0%

addition to the AIS-31 tests, there are additional evaluations from NIST, namely SP800-22 and SP800-90B [28].

Table 4 exploits the destination of each test for the TRNG design. According to [26], the latest version of SP800-22 does not provide meaningful results for this evaluation, as it requires a very large input size of approximately 100 million bits. Therefore, in our context, we focus exclusively on the AIS-31 and SP800-90B tests, since the TRNG is intended for cryptographic applications. Table 5 presents the results of the AIS-31 test for the proposed design, while Table 6 summarizes the results of the SP800-90B test.

Table 4. Evaluating test for TRNG

Standard	Focus	Type of Test
AIS-31	TRNG quality	Health and statistical
SP800-22	Statistical randomness	15 statistical tests
SP800-90B	Entropy estimation	Entropy estimation and health test

In Table 5, a more recent NIST suite, SP800-90B, evaluates whether an entropy source is suitable for secure cryptography by estimating its minimum entropy. This assessment includes IID and non-IID tests. Passing the IID test indicates that the source can be reliably used in cryptographic applications. The proposed TRNG exhibits high-quality entropy, confirming its suitability for cryptographic algorithms. In the NIST test, the p-value (Chi square) in a sequence greater than or greater than 0.01 is a good random sequence. Therefore, the results in Table 5 indicate that our TRNGs design achieves the entropy source test as a high quality entropy estimation source.

On the other hand, Table 6, the AIS-31 test, developed by the German BSI, serves as a guide for evaluating RNGs in cryptographic applications and complements the NIST SP800-90 suite. AIS-31 consists of nine tests (T0–T8). To confirm that the sequences are disjoint, preventing repeated cycles in the 16-state ring generator. The tests T1-T5, applied to 257 data segments, achieve a pass rate of 100. Since we use the Python code of AIS-20 provided, the result of T6-T8 only shows pass or fail results. Therefore, T6–T8 assess the entropy of the source and all criteria are satisfied, indicating a high-quality random output. These tests provide good TRNGs development quality.

Table 5. SP800-90B IID test

Test	Default	Value
IID Permutation	Pass	Pass
Chi-square Independence	Pass	0.14021
Chi-square goodness of fit	Pass	0.609907
Length of the longest repeated substring	Pass	Pass
Restart	Pass	Pass
Min. entropy per byte	7.972135	7.972135

Table 6. AIS-31 statistical tests result

No.	Purpose	Results
Test 0	Disjointedness test	257/257
Test 1	Monobit test	257/257
Test 2	Poker test	257/257
Test 3	Runs test	257/257
Test 4	Longest run test	257/257
Test 5	Auto-correlation test	257/257
Test 6	Uniform distribution test	Pass
Test 7	Homogeneity test	Pass
Test 8	Entropy test	7.998013

Table 7 presents the performance of our proposed TRNGs compared to recent FPGA-based designs. By transferring entropy from the RO part to the RG part, we can generate a 64-bit sample in just 64 clock cycles at 50 MHz. The throughput can be calculated using the formula where fmax is our clock system design, n is the number of bit-lengths of the generated random number, and latency is the number of cycles taken to generate the block random number as shown in Eq. 4.

$$\text{throughput} = \frac{n \times f_{\max}}{\text{latency}} \tag{4}$$

Our design achieves higher throughput with lower hardware cost than the approaches reported in [8] and [27], and also has lower hardware cost compared to [29]. Each SLICEs of FPGA contains 4 LUTs and 8 Flip Flops. In [27], the authors only proposed the parameters of SLICEs, therefore we estimate their design based on the Xinlinx Datasheet. Although throughput does not reach the design levels [34], our design successfully passes the AIS-31 test due to the efficient spreading of entropy in the RG part. In edge AI platforms, we focus on lightweight design rather than high-throughput design.

Table 7. Comparison of TRNG Designs on FPGA Platforms

TRNG Design	FPGA Family	Resource Cost	Throughput [Mbps]
TROT [8]	Artix-7	32LUTs/55FFs	25
DC-TRNG [27]	Spartan6	64 SLICEs (50 LUTs)	14.3
DD-cell [29]	Artix-7	256 LUTs/256FFs	225
JL-TRNG [34]	Spartan6	56 LUTs/19FFs	100
This work	Artix-7	10LUTs/64 FFs	50

6 Conclusion

In conclusion, this paper has highlighted the critical role of TRNG in strengthening hardware security, particularly in the context of cryptographic systems and resource-constrained IoT devices. Although PRNG offer efficiency and low implementation cost, their deterministic nature makes them vulnerable in high-security applications where unpredictability is essential. TRNGs, by using physical entropy sources, provide genuine randomness that significantly enhances security robustness. The study demonstrated the design and implementation of a TRNG in FPGA as a hiding countermeasure against side-channel analysis (SCA) attacks. By introducing true randomness into cryptographic operations, the proposed approach reduces information leakage and mitigates the effectiveness of power- and timing-based attacks. Furthermore, the simulation of an SCA attack in an unprotected AES implementation emphasized the practical risks associated with inadequate protection mechanisms. In general, integrating TRNG-based countermeasures into FPGA-based cryptographic systems represents a promising strategy to improve resistance against side-channel threats. Future work may focus on optimizing entropy sources, improving statistical performance, and evaluating the design against advanced SCA techniques to further enhance hardware-level security.

Acknowledgement. This research is funded by Vietnam National University - Ho Chi Minh City (VNU-HCM) under grant number: A2025-20–01.

References

1. Bahadur, V., Selvakumar, D., Sobha, P.: Reconfigurable side channel attack resistant true random number generator. In: 2016 International Conference on VLSI Systems, Architectures, Technology and Applications (VLSI-SATA), pp. 1–6. IEEE (2016)
2. Bommana, S.R., Veeramachaneni, S., Ershad, S., Srinivas, M.: Mitigating side channel attacks on FPGA through deep learning and dynamic partial reconfiguration. Sci. Rep. **15**(1), 13745 (2025)
3. Brier, E., Clavier, C., Olivier, F.: Correlation power analysis with a leakage model. In: Joye, M., Quisquater, J.-J. (eds.) CHES 2004. LNCS, vol. 3156, pp. 16–29. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-28632-5_2

4. Cao, Y., Chang, C.H., Zheng, Y., Zhao, X.: An energy-efficient true random number generator based on current starved ring oscillators. In: 2017 Asian Hardware Oriented Security and Trust Symposium (AsianHOST), pp. 37–42 (2017). https://doi.org/10.1109/AsianHOST.2017.8353992
5. Dang, T.K., Nguyen, K.D., Kieu-Do-Nguyen, B., Hoang, T.T., Pham, C.K.: Realization of authenticated one-pass key establishment on RISC-v micro-controller for Iot applications. Future Internet **16**(5) (2024). https://doi.org/10.3390/fi16050157, https://www.mdpi.com/1999-5903/16/5/157
6. Giechaskiel, I., Eguro, K., Rasmussen, K.B.: Leakier wires: exploiting FPGA long wires for covert-and side-channel attacks. ACM Trans. Reconfigurable Technol. Syst. (TRETS) **12**(3), 1–29 (2019)
7. Goodwill, G., Jaffe, B.J., Rohatgi, J., P.: A testing methodology for side-channel resistance validation. In: NIST Non-invasive Attack Testing Workshop, vol. 7, pp. 115–136 (2011)
8. Grujić, M., Verbauwhede, I.: Trot: a three-edge ring oscillator based true random number generator with time-to-digital conversion. IEEE Trans. Circuits Syst. I Regul. Pap. **69**(6), 2435–2448 (2022). https://doi.org/10.1109/TCSI.2022.3158022
9. Gu, P., Hua, C., Xu, W., Khatoun, R., Wu, Y., Serhrouchni, A.: Control channel anti-jamming in vehicular networks via cooperative relay beamforming. IEEE Internet Things J. **7**(6), 5064–5077 (2020). https://doi.org/10.1109/JIOT.2020.2973753
10. Gyorfi, T., Cret, O., Suciu, A.: High performance true random number generator based on FPGA block rams. In: 2009 IEEE International Symposium on Parallel & Distributed Processing, pp. 1–8. IEEE (2009)
11. Hata, H., Ichikawa, S.: FPGA implementation of metastability-based true random number generator. IEICE Trans. Inf. Syst. **95**(2), 426–436 (2012)
12. Hayashi, K., Minagawa, R., Torii, N.: Side-channel attack on COSO-based TRNG to estimate output bits. In: 2022 Tenth International Symposium on Computing and Networking Workshops (CANDARW), pp. 302–308. IEEE (2022)
13. He, W., Mo, Z.: Secure event-triggered consensus control of linear multiagent systems subject to sequential scaling attacks. IEEE Trans. Cybern. **52**(10), 10314–10327 (2022). https://doi.org/10.1109/TCYB.2021.3070356
14. Heuser, A., Picek, S., Guilley, S., Mentens, N.: Lightweight ciphers and their side-channel resilience. IEEE Trans. Comput. **69**(10), 1434–1448 (2020). https://doi.org/10.1109/TC.2017.2757921
15. Ilahi, I., et al.: Challenges and countermeasures for adversarial attacks on deep reinforcement learning. IEEE Trans. Artifi. Intell. **3**(2), 90–109 (2022). https://doi.org/10.1109/TAI.2021.3111139
16. Jiménez, A.U.: Beyond the GPU: the strategic role of FPGAs in the next wave of AI (2025). arXiv:2511.11614
17. Khonji, M., Iraqi, Y., Jones, A.: Phishing detection: a literature survey. IEEE Commun. Surv. Tutor. **15**(4), 2091–2121 (2013). https://doi.org/10.1109/SURV.2013.032213.00009
18. Kocher, P., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M. (ed.) Advances in Cryptology – CRYPTO' 99, pp. 388–397. Springer, Berlin Heidelberg, Berlin, Heidelberg (1999)
19. Li, C., Wang, Q., Jiang, J., Guan, N.: A metastability-based true random number generator on FPGA. In: IEEE 12th International Conference on ASIC (ASICON), pp. 738–741. IEEE (2017)

20. M, G., H B, P.: Semantic query-featured ensemble learning model for SQL-injection attack detection in Iot-ecosystems. IEEE Trans. Reliabil. **71**(2), 1057–1074 (2022).https://doi.org/10.1109/TR.2021.3124331
21. Mei, F., Zhang, L., Gu, C., Cao, Y., Wang, C., Liu, W.: A highly flexible lightweight and high speed true random number generator on FPGA. In: IEEE Computer Society Annual Symposium on VLSI (ISVLSI), pp. 399–404. IEEE (2018)
22. Ni, T., et al.: Mrco: a multi-ring convergence oscillator-based high-efficiency true random number generator. In: 2022 Asian Hardware Oriented Security and Trust Symposium (AsianHOST), pp. 1–6 (2022). https://doi.org/10.1109/AsianHOST56390.2022.10022291
23. Peter, M., Schindler, W.: A proposal for: functionality classes for random number generators — Version 2.36 — current intermediate document for the AIS 20/31 workshop. Bundesamt für Sicherheit in der Informationstechnik (BSI) (2023). https://www.bsi.bund.de/SharedDocs/Downloads/EN/BSI/Certification/Interpretations/AIS_31_Functionality_classes_for_random_number_generators_e_2023.pdf?__blob=publicationFile&v=2, intermediate workshop document
24. Pham-Quoc, C., Nguyen, X.Q., Thinh, T.N.: Towards an FPGA-targeted hardware/software co-design framework for CNN-based edge computing. Mobile Netw. Appl. **27**(5), 2024–2035 (2022)
25. Rajski, J., Trawka, M., Tyszer, J., Włodarczak, B.: A lightweight true random number generator for root of trust applications. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **42**(9), 2815–2825 (2023). https://doi.org/10.1109/TCAD.2023.3237957
26. Randa, M., Samie, M., Jennions, I.K.: Delay-based true random number generator in sub-nanomillimeter Iot devices. Electronics **9**(5) (2020).https://doi.org/10.3390/electronics9050817, https://www.mdpi.com/2079-9292/9/5/817
27. Rozic, V., Yang, B., Dehaene, W., Verbauwhede, I.: Highly efficient entropy extraction for true random number generators on FPGAs. In: 2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC), pp. 1–6 (2015). https://doi.org/10.1145/2744769.2744852
28. Rukhin, A., et al.: A statistical test suite for random and pseudorandom number generators for cryptographic applications. Tech. Rep. NIST Special Publication 800-22 Rev. 1a, National Institute of Standards and Technology (NIST), Gaithersburg, MD, USA (2010)
29. Sala, R.D., Scotti, G.: Exploiting the dd-cell as an ultra-compact entropy source for an FPGA-based re-configurable PUF-TRNG architecture. IEEE Access **11**, 86178–86195 (2023). https://doi.org/10.1109/ACCESS.2023.3304901
30. Singha, T.B., Palathinkal, R.P., Ahamed, S.R.: Securing AES designs against power analysis attacks: a survey. IEEE Internet Things J. **10**(16), 14332–14356 (2023). https://doi.org/10.1109/JIOT.2023.3265683
31. Tran, T.H., Dao, B.A., Hoang, T.T., Hoang, V.P., Pham, C.K.: Transition factors of power consumption models for CPA attacks on cryptographic RISC-v soc. IEEE Trans. Comput. **72**(9), 2689–2700 (2023). https://doi.org/10.1109/TC.2023.3262926
32. Tran, T.H., Hoang, V.P., Le, D.H., Hoang, T.T., Pham, C.K.: An efficient hiding countermeasure with xilinx mmcm primitive in spread mode. In: 2024 IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1–5 (2024). https://doi.org/10.1109/ISCAS58744.2024.10558293
33. Wang, X., et al.: High-throughput portable true random number generator based on jitter-latch structure. IEEE Trans. Circuits Syst. I Regul. Pap. **68**(2), 741–750 (2021). https://doi.org/10.1109/TCSI.2020.3037173

34. Wang, X., et al.: High-throughput portable true random number generator based on jitter-latch structure. IEEE Trans. Circuits Syst. I Regul. Pap. **68**(2), 741–750 (2021). https://doi.org/10.1109/TCSI.2020.3037173
35. Wold, K., Tan, C.H.: Analysis and enhancement of random number generator in FPGA based on oscillator rings. Int. J. Reconfigurable Comput. **2009**(1), 501672 (2009)