engineering proceedings

MDPI

Proceeding Paper

Prime Number Generator Based on Chaotic System and FPGA Implementation †

Chang-Ming Wu *, Yuan-Shuo Yu *, Hung-Ru Lin and Chih-Hau Chang

Department of Electronic Engineering, Chung Yuan Christian University, Taoyuan 32023, Taiwan; 0509andylin@gmail.com (H.-R.L.); cupid92929@gmail.com (C.-H.C.)

* Correspondence: cmwu@cycu.edu.tw (C.-M.W.); mark1218mark12@gmail.com (Y.-S.Y.)

† Presented at the 7th Eurasia Conference on IoT, Communication and Engineering 2025 (ECICE 2025), Yunlin, Taiwan, 14–16 November 2025.

Abstract

With the growing importance of personal information security, numerous methods have been proposed for data encryption. To ensure system safety, ciphers must be unpredictable and robust. In modern Rivest–Shamir–Adleman (RSA) encryption systems, two prime numbers are required for key generation, and their randomness and unpredictability are essential for security. In this study, we propose a secure system for generating the prime numbers used in RSA encryption. The inherent properties of chaotic systems are employed as a Pseudo Random Number Generator (PRNG), while a Ring Oscillator is utilized as a True Random Number Generator (TRNG). The Miller–Rabin algorithm is further applied to verify the primality of the numbers produced by the PRNG. The entire design is implemented on a Field Programmable Gate Array (FPGA) to achieve a fully hardware system.

Keywords: chaotic system; RSA; FPGA; encryption; prime number

1. Introduction

In rapid digitalization and network development, there are many encryption systems protecting our data and data transfer. Among these encryption methods, the Rivest–Shamir–Adleman (RSA) algorithm remains one of the most important cryptosystems. However, the security of the RSA algorithm depends not only on the bit count but also on the unpredictability and reliability of the random numbers used in key generation. Therefore, ensuring stable and secure prime number acquisition is crucial for practical applications.

This study aims to design a secure prime number generator system for generating the prime numbers used in the RSA algorithm. To ensure the randomness of the prime numbers, we apply Random Number Generators to generate sequences and verify the prime numbers. To satisfy the security requirements, we combine both the True Random Number Generator (TRNG) and the Pseudo Random Number Generator (PRNG).

A Ring Oscillator (RO) is used to produce a True Random Number Generator (TRNG) capable of generating high-entropy random seeds for the Pseudo-Random Number Generator (PRNG). The chaos theory is deterministic, nonlinear, and unpredictable. Thus, we can take Chua's chaotic system as the PRNG, which we refer to as the Chaos-based Pseudo Random Number Generator (CPRNG). After that, the candidates generated by the CPRNG are tested by the Miller–Rabin primality test algorithm to screen out the prime numbers

Check for updates

Academic Editors: Teen-Hang Meen, Chi-Ting Ho and Cheng-Fu Yang

Published: 9 April 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

that the RSA algorithm needs. Finally, we use the prime numbers generated to generate the key for the RSA algorithm and undertake the RSA encryption process.

For implementation, we employ the Field Programmable Gate Array (FPGA) platform—a fully hardware system that can simplify the system environment and reinforce the system security.

This article is structured as follows: in Section 2, we provide the necessary background of the important parts of our system; in Section 3, we describe the FPGA implementation and our system architecture; in Section 4, we present and discuss the experimental results; and in Section 5, we present our conclusions.

2. Background Knowledge

2.1. TRNG-RO

An RO is a circuit composed of an odd number of inverters or other logic switches with inverting characteristics, connected end to end to form a loop. Typically, jitter is accumulated in the free-running ROs consisting of an odd number of inverters or delay elements connected in ring configuration. This causes the digital value of the oscillator's output to change with a period of approximately 2DL, where D is the delay of a single inverter and L is the number of inverters in an oscillator. These oscillation periods vary from cycle to cycle, causing the rising and falling edges of the generated RO clocks to jitter, as shown in Figure 1. Such oscillations—and hence jitters—in digital circuits can occur due to power supply variations, crosstalk, semiconductor noise, temperature variations, and propagation delays [1]. Therefore, jitter can be utilized as entropy for true random generation, enabling the ring oscillator to function as a True Random Number Generator (TRNG).

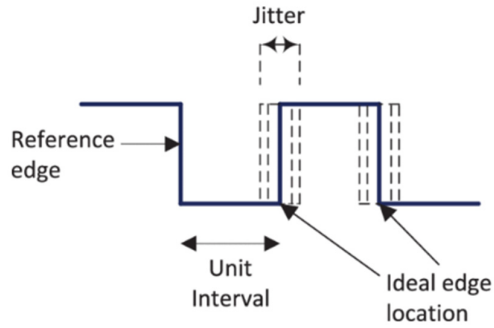


Figure 1. Jitter in clock signals [1].

2.2. Pseudo Random Number Generator—Chaotic System

The PRNG deterministically expands one or more seed values into a sequence of outputs; the seed must possess sufficient randomness and unpredictability and is therefore typically obtained from a TRNG. PRNG outputs may empirically exhibit superior statistical properties and higher throughput than some physical sources. Since each term is computed from preceding terms via deterministic transformations, the sole source of entropy remains the seed, and retaining that seed is sufficient to reproduce or validate the sequence [2]. In this study, we utilize Chua's chaotic system as the PRNG according to the characteristics of chaos theory; i.e., it is the CPRNG.

2.2.1. Chaos Theory

The fundamental concepts of chaos theory—including deterministic models, sensitivity to initial conditions, strange attractors, and fractal dimensions—are intrinsic to its development [3]. Chaos is characterized by a stretching and folding mechanism in which nearby trajectories of a dynamical system are repeatedly separated exponentially and subsequently folded back together.

https://doi.org/10.3390/engproc2026134039

To exhibit chaotic behavior, an electronic circuit composed of resistors, capacitors, and inductors must include at least one nonlinear element, at least one locally active resistor, and a minimum of three energy-storage components. Chua's circuit (Figure 2) represents the simplest electronic configuration that satisfies these requirements. Moreover, it can be constructed at low cost using standard electronic components and is known to display a wide range of bifurcations and chaotic phenomena [4].

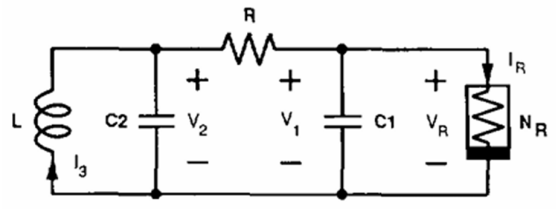


Figure 2. Chua's electronic circuit [4].

The system of equations representing the dynamics of Chua's circuit are based on Equations (1) and (2):

$$\begin{cases} \tau_x \frac{dx}{dt} = \alpha[y - x - f(x)] \\ \tau_y \frac{dy}{dt} = x - y + z \\ \tau_z \frac{dz}{dt} = -\beta y \end{cases}, \quad (1)$$

where $f(x)$ is given by the following:

$$f(x) = bx + \frac{a-b}{2}\{|x + B_p| - |x - B_p|\}, \quad (2)$$

and τ_x, τ_y, τ_z , α, β, a, b and B_p are real parameters [5].

2.2.2. Discrete Chaotic System

Numerical solutions of ordinary differential equations are essential techniques in the study of time dynamics. Because most ordinary differential equations cannot be solved analytically, numerical integration provides the only means of obtaining information about system trajectories. Numerous methods have been proposed and applied to achieve accurate solutions for different classes of ordinary differential equations [6]. Among these, the Runge–Kutta family of methods is recognized for its superior accuracy compared to alternative approaches. Accordingly, the fourth-order Runge–Kutta method was selected to discretize Chua's chaotic circuit, with the governing equations derived from Equations (3)–(5) [7]. For an ordinary differential equation with initial value X_0, the following is defined:

$$dX_t = F(t, X_t)dt, \quad (3)$$

the following equation is also used:

$$X(t_{n+1}) = X(t_n) + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4), \quad (4)$$

where the k_i, $i = 1, 2, 3, 4$.

$$\begin{cases} k_1 = h\mathrm{F}(t_n,\ x_n) \\ k_2 = h\mathrm{F}\left(t_n + \frac{h}{2},\ X_n + \frac{h}{2}\,k_1\right) \\ k_3 = h\mathrm{F}\left(t_n + \frac{h}{2},\ X_n + \frac{h}{2}\,k_2\right) \\ k_4 = h\mathrm{F}(t_n + h,\ X_n + hk_3) \end{cases}, \quad (5)$$

where $t_n = nh$, and the step size, h, is constant for all steps.

2.2.3. Dynamic Characteristics of Chaotic System

Due to chaotic system being sensitive to initial conditions, the Lyapunov function, named after the Russian mathematician Aleksandr Lyapunov, is widely used in nonlinear dynamical systems and control theory. The Lyapunov function is employed to analyze the stability of equilibrium points in a system. It is a scalar function used to prove whether the equilibrium point of a system is stable.

This method provides a powerful tool for determining the stability of a system without solving the equations directly, relying solely on the properties of the function. To analyze the dynamic characteristics according to initial conditions in a chaotic system, the Lyapunov exponents must be calculated. If the maximum exponent is positive, this indicates that the system exhibits chaotic behavior; if it is negative, the system is stable; and if it is zero, the system may be in a boundary state or neutral. To determine the magnitude of the Lyapunov exponent, we first consider a dynamic system in Equation (6):

$$\dot{x} = f(x, t), \tag{6}$$

where x is the state vector, and f is a time-dependent and differentiable function. Considering an infinitesimal n-dimensional sphere of initial conditions, the evolution of the sphere becomes an n-dimensional ellipsoid. Then, the LEs of the system is calculated as Equation (7).

$$\lambda_i = \lim_{t \to \infty} \frac{1}{t} \ln \frac{\|\delta x_i(t)\|}{\|\delta x_i(t_0)\|}, \quad i = 1, \cdots, n, \tag{7}$$

where λ_{imax} denotes the magnitude of the Lyapunov exponent [8].

2.3. RSA Algorithm

RSA is an asymmetric encryption algorithm that uses two prime numbers to generate a public key for encryption and a private key for decryption.

2.3.1. RSA Methodology

The following is the process of generating RSA keys [9]. The encryption and decryption methods are shown in Figure 3.

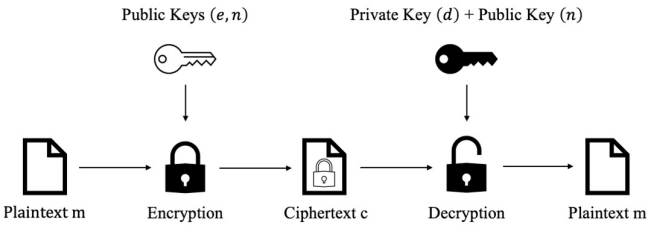


Figure 3. RSA encryption and decryption methods.

- Choose two large prime numbers p and q (choose random numbers and make sure they are prime);
- Count $n = p \times q$;
- Calculate Euler function φ, $\varphi(n) = (p-1) \times (q-1)$;
- Choose the public key e $(1 < e < \varphi(n))$, prime relative to $\varphi(n)$; here, we choose 65537 as the public key;
- Generate a private key d, $d \equiv e^{-1} \, mod \, \varphi(n)$;
- Generate public keys (e, n) and a private key (d);
- Encryption method: $c = m^e mod \, n$;
- Decryption method: $m = c^d mod \, n$.

2.3.2. Primality Test

For the large amount of pseudo-random numbers generated by the CPRNG, the Miller–Rabin algorithm is a probabilistic primality test used to determine whether an integer is prime. The Miller–Rabin test is an improvement over Fermat's Little Theorem. According to the theorem, if p is a prime number and a is any number that is not divisible by p, then the theorem can be expressed as Equation (8).

$$a^{p-1} \equiv 1 \ (mod \ p) \tag{8}$$

$a^p \equiv a(\text{mod } p)$. As 5 is prime, $3^4 \equiv 1(\text{mod}5)$ and $8^4 \equiv 1(\text{mod}5)$. This theorem always returns true if p is prime. The Miller–Rabin test can be made deterministic. This is achieved by trying all possible values of x below a certain limit in the algorithm given. For example, if $n < 2047$, then it is sufficient to test for only $x = 2$; if $n < 3474749660383$, then it is sufficient to test for $x = 2, 3, 5, 7, 11, 13$. It uses a particular property such that multiples of 2 can be represented by $2a \cdot b$, where a and b are positive integers [10].

3. FPGA Implementation

3.1. System Architecture

For the implementation, our system combines the Prime Number Generator with a 64-bit RSA encryption framework. The Prime Number Generator comprises three components: the TRNG, the CPRNG, and a primality testing module. The RSA encryption framework includes a key generation module and an encryption module. The overall system architecture is illustrated in Figure 4.

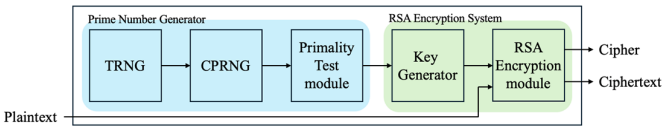


Figure 4. Block diagram of our system.

When plaintext is provided as input, the Prime Number Generator automatically produces prime numbers for the Key Generator. The RSA encryption module then uses the generated key to encrypt the plaintext. Upon completion, the system outputs both the cipher (including the public and private keys) and the ciphertext.

3.2. Implementation and Platform

The system was implemented in hardware using Verilog Hardware Description Language to describe the individual modules. The Terasic DE0-Nano-SoC FPGA Development Kit was selected as the hardware platform. FPGAs offer higher performance and lower power consumption compared to microprocessors. In addition, when compared with application-specific integrated circuits, FPGAs provide lower non-recurring engineering (NRE) costs, shorter development time, easier debugging, and reduced design risk [11]. Owing to these characteristics, FPGAs are well-suited for implementing the system entirely in hardware, thereby reinforcing both security and stability.

4. Results and Discussion

4.1. Lyapunov Exponent

Figure 5 shows a simulation plot using MATLAB_R2023b to calculate the range of initial conditions for Chua's chaotic system based on the Lyapunov exponent. Therefore, we extracted a cubic region in the middle of the plot to ensure that the initial conditions

within this range would reliably display chaotic behavior. The range of initial values is shown in Equation (9).

$$\begin{cases} -1.61 \leq x \leq 1.61 \\ -0.365 \leq y \leq 0.365 \\ -1.05 \leq z \leq 1.05 \end{cases} \quad (9)$$

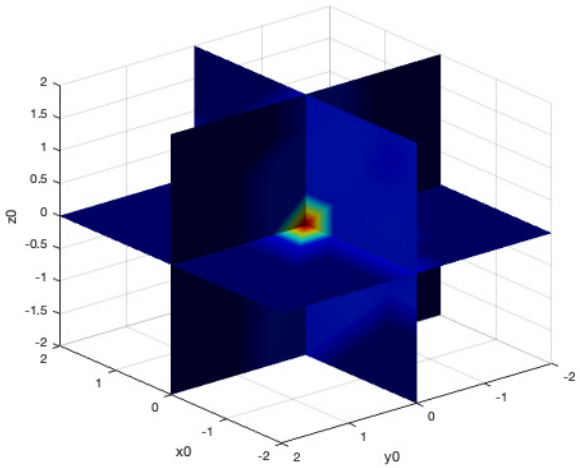
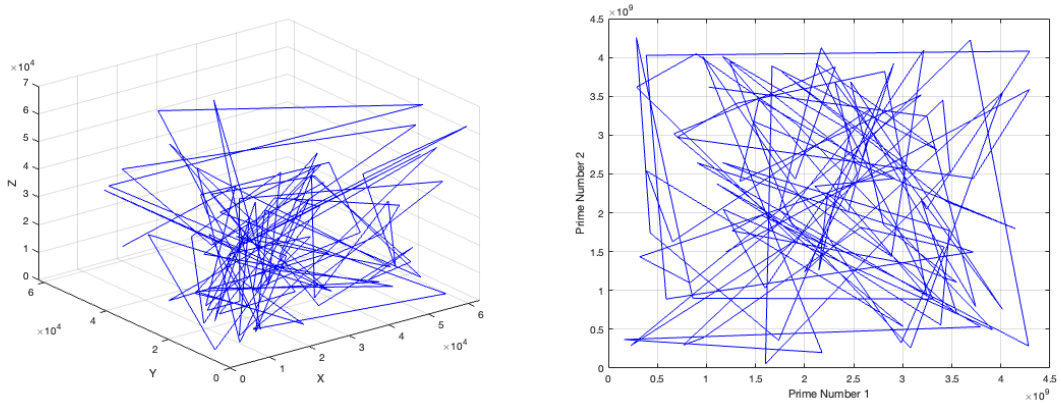


Figure 5. Lyapunov exponent 3D simulation diagram.

4.2. Random Number Generators

The results of the TRNG are used as the initial conditions for the CPRNG. Because the TRNG must produce truly random values, 100 outputs were collected, each consisting of three-dimensional data constrained within the range defined by Lyapunov analysis. As illustrated in Figure 6a, these results exhibit genuine randomness without discernible patterns. Consequently, the TRNG outputs can be employed to generate the random seeds required by the CPRNG.



(**a**) (**b**)

Figure 6. (**a**) Results of the TRNG. (**b**) Results of the CPRNG.

Furthermore, by using these random seeds, the CPRNG can generate a large amount of pseudo random numbers. After applying the Miller–Rabin primality test to the CPRNG, we collect 100 results, each containing two prime numbers, as required by the RSA algorithm. As can be seen in Figure 6b, the results obtained using the random seeds are also random and have no regular pattern.

4.3. RSA Encryption on FPGA

By utilizing the generated prime numbers, the RSA keys are produced following the procedures described in Section 2.3.1. Subsequently, we can implement our system (as described in Section 3.1) on an FPGA and encrypt the plaintext. To verify the correctness of the encryption process, the results are compared with those obtained by the RSATool v1.18 [12] software. The encryption results and comparison results are summarized in Table 1.

Table 1. RSA encryption results.

Plaintext (ASCII)	n	d	Ciphertext	RSATool Comparison Result
HELLO (48454C4C4F)	FE0791DE8B4CA0E5	A2AD54526A42174D	8A44C27351D478CC	Correct
CYCUEL (43594355454C)	FEF5CC961506FE61	1D6234A6A8787001	477C7CA35B97F3DB	Correct
RSA_FPGA (5253415F46504741)	FC477C1535F50C01	A8AA8261B827E781	601DCF5AB52F1B98	Correct
chaotic (6368616F746963)	FA5FADD4D9D124A9	D5E79E8335859631	803B46BBAD38B9C4	Correct
Taiwan (54616977616E)	FE5C081E19301A0B	749DFF3DDB733891	4E57F887395C72D2	Correct
AndyruSB (416E647972755342)	FDA36E6F207900BB	4BA95B219748FD31	C6EF051FDB529A25	Correct
baseball (6261736562616C6C)	FE9F90EBAB66022D	10F773018F28C145	3A0432E5863D5B95	Correct
starwars (7374617277617270)	FDACBCDA62DB0D79	7A16096908014791	2CD5CCEE442F9C2D	Correct
fireball (6669726562616C6C)	FCCCD63174726DDD	5EBBA7FBEBBB8141	289B5A096261241D	Correct
meatball (6D65617462616C6C)	FEFB0A3373BD2FD1	00C53564ACDBA181	8BADE770555886F6	Correct

5. Conclusions

We propose a prime number generator that uses a TRNG to supply genuine entropy, a CPRNG to expand and decorrelate that entropy into a high-throughput stream of candidate values, and the Miller–Rabin primality test algorithm to select the prime numbers. We combined the prime number generator based on the chaotic system with the RSA encryption algorithm. To reinforce the system security and stability, we implement the system in FPGA to build a fully hardware environment. The hardware architecture must be established to enhance entropy efficiency, increase the prime numbers' bit length to 2048 to satisfy more safety requirements, and explore the integration of this RSA generation framework into practical cryptographic and communication systems.

Author Contributions: Conceptualization, C.-M.W. and Y.-S.Y.; data curation, Y.-S.Y., H.-R.L. and C.-H.C.; formal analysis, Y.-S.Y. and H.-R.L.; investigation, Y.-S.Y., H.-R.L. and C.-H.C.; methodology, C.-M.W. and Y.-S.Y.; software, Y.-S.Y. and C.-H.C.; supervision, C.-M.W.; writing—original draft, Y.-S.Y., H.-R.L. and C.-H.C.; writing—review and editing, C.-M.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: No new data were created.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Anandakumar, N.N.; Sanadhya, S.K.; Hashmi, M.S. FPGA-Based True Random Number Generation Using Programmable Delays in Oscillator-Rings. *IEEE Trans. Circuits Syst. II Express Briefs* **2019**, *67*, 570–574. Available online: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8726130 (accessed on 14 July 2025). [CrossRef]
2. Rukhin, A.; Soto, J.; Nechvatal, J.; Smid, M.; Barker, E.; Leigh, S.; Levenson, M.; Vangel, M.; Banks, D.; Heckert, A.; et al. *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications, Rev. 1a*; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2010; p. 14. Available online: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=906762 (accessed on 18 July 2025).
3. Oestreicher, C. A history of chaos theory. *Dialogues Clin. Neurosci.* **2007**, *9*, 279–289. Available online: https://pmc.ncbi.nlm.nih.gov/articles/PMC3202497/ (accessed on 10 October 2025). [CrossRef] [PubMed]
4. Kennedy, M.P. Three Steps to Chaos-Part II: A Chua's Circuit Primer. In *IEEE Transactions on Circuits and Systems-I: Fundamental Theory and Applications, 10 October 1993*; IEEE: Piscataway, NJ, USA, 1993; Volume 40, pp. 657–673. Available online: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=246141 (accessed on 10 October 2025).
5. Rodríguez-Vázquez, A.; Delgado-Restituto, M. CMOS Design of Chaotic Oscillators Using State Variables: A Monolithic Chua's Circuit. In *IEEE Transactions on Circuits and Systems-11: Analog and Digital Signal Processing, 31 October 1993*; IEEE: Piscataway, NJ, USA, 1993; Volume 40. Available online: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=246161 (accessed on 11 October 2025).
6. Cartwright, J.H.E.; Piro, O. The Dynamics of Runge-Kutta Methods. *Int. J. Bifurc. Chaos* **1992**, *2*, 27–449. Available online: http://www.physics.ntua.gr/konstant/ph36/doc/CartwrightAndPiroRKpaper.pdf (accessed on 14 October 2025). [CrossRef]
7. Yang, X.; Shen, Y. Runge-Kutta Method for Solving Uncertain Differential Equations. *J. Uncertain. Anal. Appl.* **2015**, *3*, 17. Available online: https://juaa-journal.springeropen.com/counter/pdf/10.1186/s40467-015-0038-4.pdf (accessed on 14 October 2025). [CrossRef]
8. Amiri, M.; Dehghani, M.; Khayatian, A.; Mohammadi, M. Lyapunov Exponent based Stability Assessment of Power Systems. In *2019 6th International Conference on Control, Instrumentation and Automation (ICCIA), 30–31 October 2019*; IEEE: Piscataway, NJ, USA, 2019. Available online: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=9030854 (accessed on 14 October 2025).
9. Aufa, F.J.; Endroyono; Affandi, A. Security System Analysis in Combination Method: RSA Encryption and Digital Signature Algorithm. In *2018 4th International Conference on Science and Technology (ICST), 15 November 2018*; IEEE: Piscataway, NJ, USA, 2018. Available online: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8528584 (accessed on 1 September 2025).
10. Tarafder, A.K.; Chakroborty, T. A Comparative Analysis of General, Sieve-of-Eratosthenes and Rabin-Miller Approach for Prime Number Generation. In *2019 International Conference on Electrical, Computer and Communication Engineering (ECCE), 7–9 February 2019*; IEEE: Piscataway, NJ, USA, 2019. Available online: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8679358 (accessed on 1 September 2025).
11. Leong, P.H.W. Recent Trends in FPGA Architectures and Applications. In *2008 IEEE International Symposium on Electronic Design, Test and Applications, Hong Kong, China, 23–25 January 2008*; IEEE: Piscataway, NJ, USA, 2008. Available online: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4459527 (accessed on 2 September 2025).
12. RSATool. RSA Public Key Encryption Algorithm Tool [Online]. Available online: https://kocianjiri.com/rsatool/index.html (accessed on 15 September 2025).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.