**International Journal of Reconfigurable and Embedded Systems (IJRES)**
Vol. 15, No. 1, March 2026, pp. 21~30
ISSN: 2089-4864, DOI: 10.11591/ijres.v15.i1.pp21-30 

# FPGA implementation of a coprocessor architecture for random data generation and encryption

**Manoj Kumar**
Department of Electronics and Communication Engineering, National Institute of Technology Manipur, Imphal, India

**Article Info**

***Article history:***

Received Nov 8, 2025
Revised Jan 12, 2026
Accepted Jan 24, 2026

***Keywords:***

Coprocessor
Encryption
Field programmable gate array
Pseudo-random number generator
True random number generator

**ABSTRACT**

Coprocessors are designed to perform some specific tasks to enhance system performance and speed. Information security is the main focus in internet of thing (IoT), cryptography, and cybersecurity applications. In this work, a coprocessor architecture is designed to generate 4-bits of random data and perform encryption. Coprocessor architecture uses true random number generator (TRNG) and pseudo-random number generator (PRNG) architectures to generate random data. Modified linear feedback shift register (LFSR)-based PRNG and modified transition effect ring oscillator (TERO) and ring oscillator-based TRNG architectures are designed and implemented for performing encryption. A serial-in-parallel-out (SIPO) shift register circuit is used to generate 4-bit random data. A 15-bit instruction word is assigned to coprocessor architecture to perform its task. The coprocessor architecture is designed using VHSIC Hardware Description Language (VHDL) and implemented on an Artix-7 field programmable gate array (FPGA). All simulation and synthesis results of the proposed coprocessor architecture are obtained by the Xilinx Vivado 2015.2 tool. Coprocessor architecture efficiency (throughput (Mbps)/LUTs) is 2.31, and it operates at a 100 MHz clock.



***Corresponding Author:***

Manoj Kumar
Department of Electronics and Communication Engineering, National Institute of Technology Manipur
Langol, Imphal. Manipur, 795004, India
Email: manojara400@gmail.com

## 1. INTRODUCTION

Coprocessors are designed to perform specific tasks more efficiently, such as cryptographic tasks (like encryption and digital signatures), signal processing, I/O data transfer and processing, string processing, and floating-point arithmetic. The coprocessor operates independently of the main processor to improve system speed. Cryptographic coprocessors perform high-speed encryption, digital signing, and key management. Today, cryptographic coprocessors are used in financial services (PIN verification, digital signing), network security, data centres, internet of thing (IoT), and embedded systems for secure transactions and data protection. Field programmable gate array (FPGA)-based embedded systems are capable of achieving high levels of security, flexibility, and adaptability [1]. In the 20th century, public key cryptography was widely used to secure data on the channel. Public key cryptography, such as elliptic-curve cryptography (ECC) [2] and the Rivest–Shamir–Adleman (RSA) [3] algorithm, is based on complex mathematical problems. In the 21st century, quantum computers are widely used to solve mathematical problems in RSA and ECC. Post-quantum cryptography algorithms aim to secure data against quantum computing attacks. Post-quantum cryptography algorithms are inefficient to implement in software [4], so three types of hardware accelerators: i) memory-mapped accelerator, ii) tightly coupled accelerator, and iii) coprocessors, based upon their implementations, have been proposed. Among these hardware accelerator methods, coprocessors are widely used in the modern

central processing unit (CPU) architecture [5], [6]. Coprocessors or accelerators are specialized in executing specific types of instructions. Encryption hides plaintext data by using a cipher to secure the data [7]. Data decryption uses the same cipher to obtain original plaintext data [8]. For cryptographic systems, widely used RNGs are PRNG and TRNG [9], [10]. PRNG output is based on a deterministic algorithm, and it offers a high data rate. TRNG uses high entropy sources to generate true random data. TRNG and PRNG architectures can be used to perform encryption and decryption in cryptographic systems.

Several researchers have been working on designing coprocessor architectures to enhance system performance. Pan *et al*. [11] designed an advanced encryption standard (AES) coprocessor based on the open-source core Hummingbird E203. The proposed AES coprocessor uses RISC-V custom instructions and direct memory access channels to achieve parallel data processing. The Artix-7 FPGA is used to verify the functionality of the AES coprocessor. A programmable crypto processor based on reduced instruction set computer (RISC)-V architecture is proposed by Lee *et al*. [12]. In this architecture, compared to the basic RISC-V architecture, the execution clock cycle and number of executed instructions are significantly improved. The proposed crypto processor is designed using Verilog and synthesized with a 28 nm CMOS process. A secure coprocessor architecture for the TLSv1.2 protocol was designed by Hamilton and Marnane [13] and implemented on Virtex 5 FPGA alongside a Microblaze processor. This design was implemented with TRNG architecture and also includes hardware based on elliptic curve algorithms for signature generation and verification. In this system, all secret key data is kept in the coprocessor, but simulation results for the proposed architecture are not shown. A public key cryptographic coprocessor for developing a unified arithmetic unit is proposed by the authors in 2008 [14]. The proposed cryptosystem supports operations of RSA and ECC and provides functions of dual-field modular multiplication, dual-field modular addition/subtraction, and dual-field modular inversion. In this work, a new adder based on signed-digit (SD) number representation is used for improving system speed. VHSIC Hardware Description Language (VHDL) coding was used to design the cryptosystem architecture, and Xilinx FPGA was used to implement it. A ECC coprocessor over GF($2^m$) implementation on an FPGA was done by Okada *et al*. [15]. For enhancing the speed of an elliptic scalar multiplication, a multiplier over GF($2^m$), which performs multiplication of any bit length, is developed. The proposed coprocessor was designed using Verilog HDL and implemented on EPF10K250AGC599 2 (ALTERA). This coprocessor operates at 3 MHz, and it is suitable for server systems. Leong and Leung [16], a processor was developed to perform an elliptic curve multiplication over the field $F_2^n$ by using FPGA technology. In this work, the control part of the proposed processor is micro coded, and the arithmetic logic unit (ALU) + register file form an $F_2^n$ processor. A reconfigurable cryptographic coprocessor, which consists of several reconfigurable modules, was proposed by Shang *et al*. [17]. The proposed coprocessor was integrated with a 32-bit CPU and fabricated in a 0.18 m complemenatry metal oxide semiconductor (CMOS) process. A cipher coprocessor (RCP) to realize DES, 3DES, AES, IDEA, RC6, and RSA algorithms is designed by Tian *et al*. [18]. To provide security for IoT sensor nodes, a cryptographic coprocessor [19] was designed, and it contains an application-specific instruction set to implement AES and ECC algorithms. For this coprocessor, local instruction and data RAM are integrated. For secure IoT applications, ASCON Cryptographic Coprocessor was developed by Athanasiou *et al*. [20]. In this work, the coprocessor uses AMBA interfaces for communication, and it was realized on both FPGA and ASIC (Application Specific Integrated Circuit) technologies. For cryptographic applications a hardware accelerator for the RISC-V processor is proposed in Athanasiou *et al*. [20]. The proposed accelerator has higher efficiency in computation compared with software execution. Loi and Ko [21] proposed a scalable and unified ECC coprocessor that uses DSP48E slices for evaluating dual-field arithmetic. The proposed coprocessor was implemented on a Xilinx Virtex-5 FPGA, and it consumes 4244 registers, 8316 LUTs, 2291 slices, 5 BRAMs, and 25 DSP48E slices. For this coprocessor also, the simulation output waveform is not shown.

Unlike conventional designs focusing solely on encryption, this work combines optimized random number generation and encryption within a single lightweight coprocessor. The coprocessor uses modified PRNG (LFSR) and TRNG (TERO and ring oscillator) architectures to generate random bits, and a 4-bit SIPO circuit is designed to generate 4-bit random data. A 15-bit instruction word is provided to the proposed coprocessor to perform its task. The coprocessor register file of size 16x4 stores 4-bit plain text data in the source register (Rs), and the destination register stores 4-bit random and encrypted data. On the Artix7 FPGA Board, coprocessor components, including the register file, multiplexer, control logic, shift register circuit, PRNG, and TRNG circuits are implemented. Simulation output waveforms for RNG (PRNG and TRNG) architectures are shown. Also, the simulation output waveform of the complete coprocessor along with register file contents, is shown. The proposed coprocessor achieves a high throughput value while maintaining low hardware resource consumption.

## 2. METHOD

The proposed coprocessor is designed to accelerate plaintext data encryption by using TRNG and PRNG architectures. Figure 1 represents a block diagram of the proposed coprocessor architecture for generating 4-bit random data and performing encryption. All components of the proposed coprocessor architecture are discussed in the subsections below.

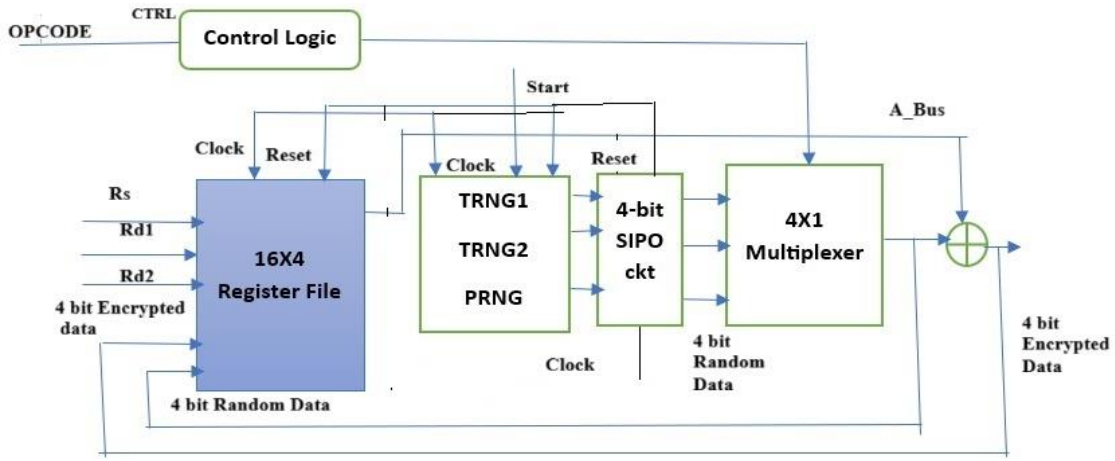


Figure 1. Coprocessor structural diagram

### 2.1. Control logic

Control logic is used to select a particular RNG architecture output in the 4x1 multiplexer architecture.

### 2.2. 16×4 register file

A 16×4 register file is an array of registers that serves as primary memory storage for cryptographic operations. In this register file, selective access to specific registers is done by Rs, Rd1, and Rd2. For the proposed coprocessor architecture, initial values of the register file are listed below.

```
signal REG_FILE: mem_type:=(
0 => "0001",
1 => "1000",
2 => "1001",
3 => "1010",
4 => "1100",
5 => "1011",
6 => "1111",
7 => "0010",
8 => "1010",
9 => "1101",
10 => "0110",
11 => "1110",
12 => "1010",
13 => "0101",
14 => "1011",
15 => "0111",
others => (others => '0')
);
```

### 2.3. Random number generators

TRNG1 architecture is a ring oscillator-based TRNG architecture for generating true random bitstreams. Ring oscillator phase or frequency jitter is used as an entropy source for this TRNG architecture. This architecture was selected due to its simple structure, a detailed and applicable stochastic model and online testability. Figure 2 represents an architectural diagram of the TRNG1 architecture.

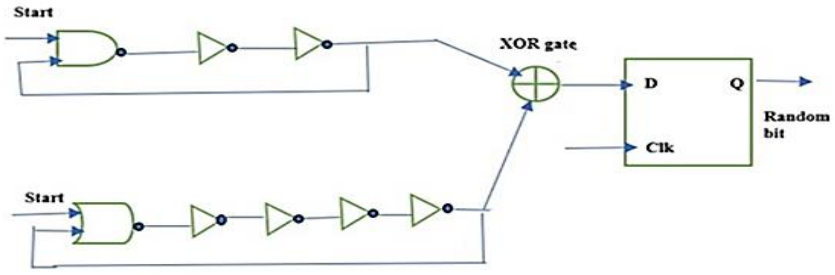


Figure 2. TRNG1 architecture diagram

In this architecture, there are two types of ring oscillators based on NAND, NOR, and NOT gates. The start signal functions as an enable control. The first ring oscillator contains one NAND gate, which uses a constant rise time of value 10 ns [22], and two NOT gates. The second ring oscillator contains one NOR gate, which uses a constant rise time of value 10 ns [22], and four NOT gates. Outputs of both ring oscillators are XORed and sampled by a D FF circuit at a clock frequency of 100 MHz to generate a random bitstream. The proposed transition effect ring oscillator (TERO)-based TRNG2 architecture is shown in Figure 3. The modified TERO based TRNG architecture was selected due to its superior entropy generation and lower susceptibility to bias compared to traditional ring oscillators. Two NAND gates with a constant rise time of value 10 ns and 6 NOT gates form a TERO [23], [24] loop, and this loop is used as an entropy source for the proposed TRNG2 architecture. In this architecture, the start signal is used as an enable signal to trigger the TERO loop. DFF is used to generate random bitstreams.

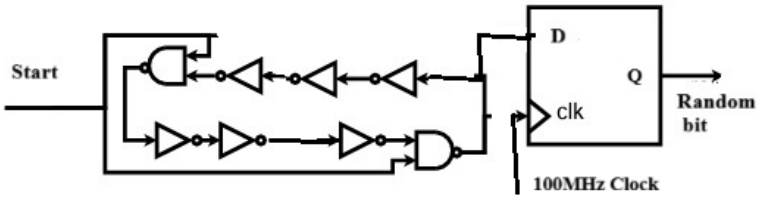


Figure 3. TRNG2 architecture diagram

The proposed 7-bit LFSR-based PRNG architecture consists of 7 DFFs and one XNOR gate, as shown in Figure 4. In this architecture, there is no seed value [25] used, and the output of the last FF (DFF6) is used to generate a random bitstream. This architecture is chosen because it generates random numbers more effectively than traditional LFSR-based architectures that rely on a seed value.

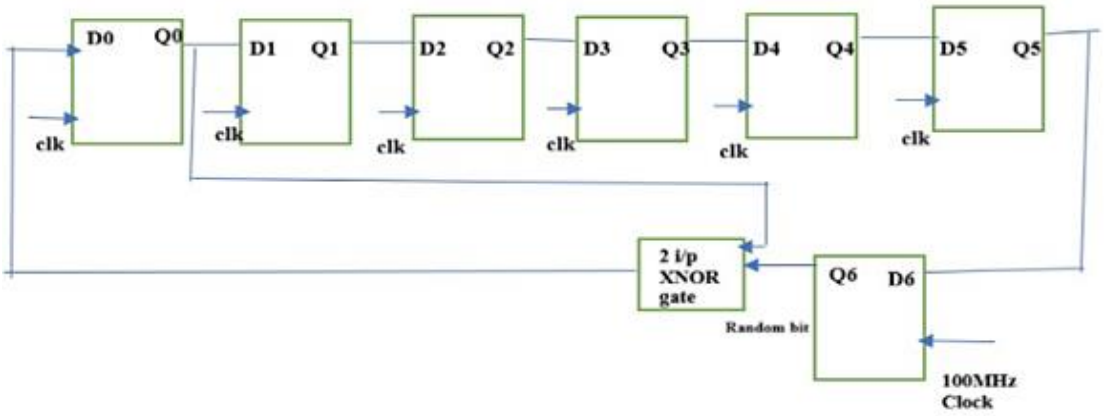


Figure 4. PRNG architecture diagram

To convert the random bitstream obtained from the RNG's architecture into 4-bit random data, a 4-bit DFF-based shift register (SIPO) is used in the proposed coprocessor architecture. In this SIPO circuit, all DFFs operates at 100 MHz system clock. Based on the control logic value, the 4x1 multiplexer circuit produces a particular RNG architecture output of 4-bits of random data.

### 2.4. Encryption logic

A_Bus contains 4-bit plaintext data obtained from the register file by use of the source register operand (Rs). 4-bit encrypted data is obtained by XORING 4-bit random data with 4-bit plaintext data.

### 2.5. Instruction set format

A 15-bit instruction word for the proposed coprocessor architecture is shown in Figure 5. For the proposed coprocessor, mnemonics and instruction functions are shown in Table 1.

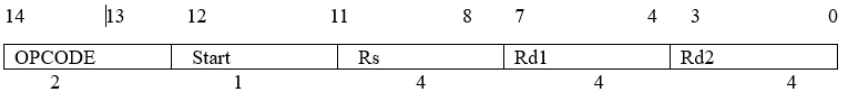


Figure 5. Proposed coprocessor 15-bit instruction word

Table 1. Mnemonics and instruction function details for the proposed coprocessor architecture

| OPCODE | MNEMONIC | Instruction description |
|---|---|---|
| 00 | TRNG1 | Generation of 4-bit random data by using ring oscillator-based TRNG architecture |
| 01 | PRNG | Generation of 4-bit random data by using 7 bit LFSR circuit |
| 10 | TRNG2 | Generation of 4-bit random data by using TERO TRNG architecture |
| 11 | NOP | No operation performed |

## 3. RESULTS AND DISCUSSION

VHDL is used to design a coprocessor architecture. Xilinx ISE Vivado 2015. Two tools are used to synthesize and simulate coprocessor architecture. The proposed coprocessor architecture is implemented on Artix7 FPGA (xc7a35tlcpg236-2L). The individual ring oscillator-based TRNG (TRNG1) architecture simulation output waveform is shown in Figure 6.

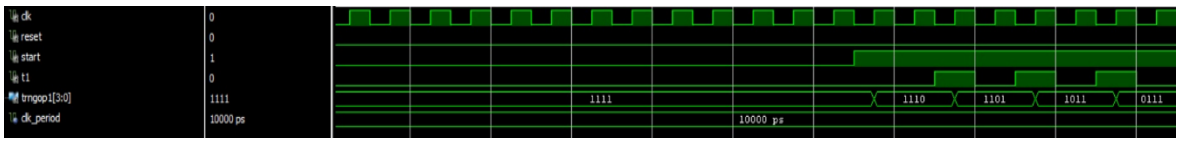


Figure 6. Simulation waveform of TRNG1 architecture

In Figure 6, signal t1 represents the output of the XOR gate in the ring oscillator-based TRNG architecture (TRNG1), signal trngop1 represents 4-bit random data obtained from the SIPO shift register circuit, signal start represents the enable signal for both ring oscillator circuits, and clk represents the 100 MHz system clock. The individual TERO-based TRNG (TRNG2) architecture simulation output waveform is shown in Figure 7.


Figure 7. Simulation waveform of TRNG2 architecture

In Figure 7, signal t1 represents the output of the DFF in the TERO TRNG architecture (TRNG2), signal trngop1 represents 4-bit random data obtained from the SIPO shift register circuit, signal start represents the enable signal for the TERO loop, and signal clk represents the 100 MHz system clock. The individual LFSR-based PRNG architecture simulation output waveform is shown in Figure 8.

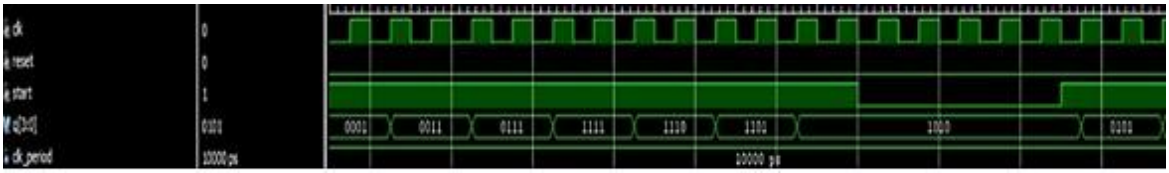

Figure 8. Simulation waveform of LFSR based PRNG architecture

In Figure 8, signal q represents 4-bit random data obtained from the SIPO shift register circuit, signal start represents the enable signal, and signal clk represents the 100 MHz system clock. Figures 6-8 shows that the generated 4-bit random data from the RNG architectures is random in nature. For conducting the NIST SP 800-22 test, 1000 random bits are taken, and if the P-value is≥0.01, then the proposed RNG architecture successfully passes the randomness test. NIST test results are shown in Tables 2-4. The simulation output waveform for the proposed coprocessor architecture for CTRL of value "01" is shown in Figure 9.

Table 2. NIST SP 800-22 Test result for TRNG1 architecture

| NIST test | P-value | Result |
|---|---|---|
| DFT | 0.4024 | Pass |
| Approximate entropy test | 0.0243 | Pass |
| Frequency (monobit)test | 0.5910 | Pass |
| Binary matrix rank test | 0.0391 | Pass |
| Random excursion test | 0.7764 | Pass |
| Frequency test within a block | 0.9971 | Pass |
| Serial | 0.9995 | Pass |
| Non-overlapping template matching test | 0.8772 | Pass |
| Run | 0.0000 | Fail |

Table 3. NIST SP 800-22 Test result for TRNG2 architecture

| NIST test | P-value | Result |
|---|---|---|
| DFT | 0.4024 | Pass |
| Approximate entropy test | 0.0022 | Pass |
| Frequency (monobit) test | 0.6543 | Pass |
| Binary matrix rank test | 0.0391 | Pass |
| Random excursion test | 0.3861 | Pass |
| Frequency test within a block | 0.9209 | Pass |
| Serial | 0.9995 | Pass |
| Non-overlapping template matching test | 0.87727 | Pass |
| Run | 0.6712 | Pass |

Table 4. NIST SP 800-22 test result for LFSR based PRNG architecture

| NIST test | P-value | Result |
|---|---|---|
| DFT | 0.4024 | Pass |
| Approximate entropy test | 0.0196 | Pass |
| Frequency (monobit) test | 0.0000 | Fail |
| Binary matrix rank test | 0.0391 | Pass |
| Random excursion test | 0.9625 | Pass |
| Frequency test within a block | 0.9999 | Pass |
| Serial | 0.9995 | Pass |
| Non-overlapping template matching test | 0.8759 | Pass |
| Run | 0.0119 | Pass |

In Figure 9, signal CTRL represents a 2-bit control signal for selecting a particular RNG architecture output, signal start represents the start signal for the coprocessor architecture, signal reg_file1 shows 16x4

register file content, signal RegA represents source register Rs, signal RegB represents destination register Rd1, signal RegC represents destination register Rd2, signal randomdata represents 4-bit random data of a particular RNG architecture, signal encrypte ddata represents 4-bit encrypted data, and signal clock represents the 100 MHz system clock. In Figure 9, source register Rs at the address of value 0 contains plaintext value 0001, destination register Rd1 at the address of value 4 contains 4-bit random data 0000, destination register Rd2 at the address of value 8 contains 4-bit encrypted data 0001, signal CTRL value is 01 for getting random bitstream from the PRNG architecture, and signal start value is 1.15-bit instruction words read from a process statement in the testbench program. The simulation output waveform for the proposed coprocessor architecture for CTRL of value "10" is shown in Figure 10. In this figure, source register Rs at the address of value 2 contains plaintext value 1001, destination register Rd1 at the address of value 12 contains 4-bit random data 0000→0111→1111, destination register Rd2 at the address of value 9 contains 4-bit encrypted data 0001→0110→1110, signal CTRL value is 10 for getting random bitstream from the TRNG2 architecture, and signal start value is 1. For the instruction word "011000001001000" and "101001011001001 for the coprocessor architecture, register file content is given in Table 5.

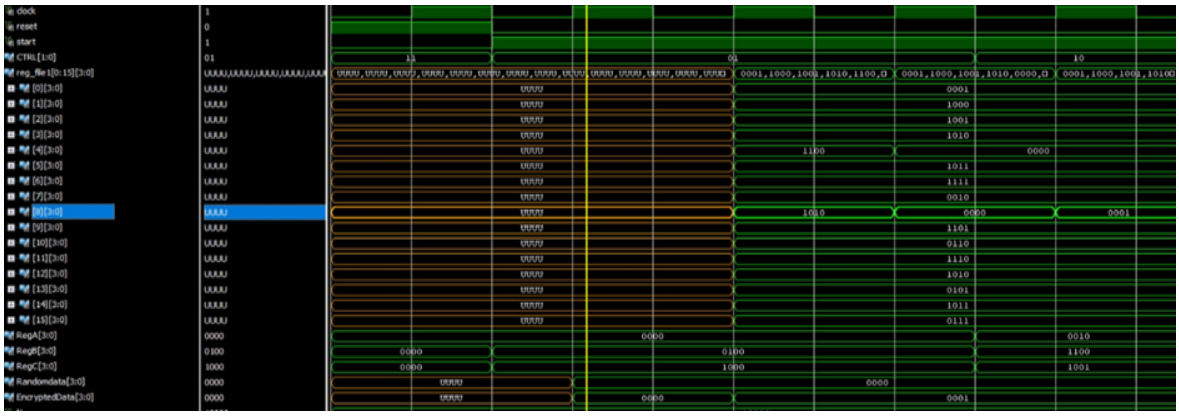

Figure 9. Simulation output waveform for the proposed coprocessor architecture for CTRL="01"

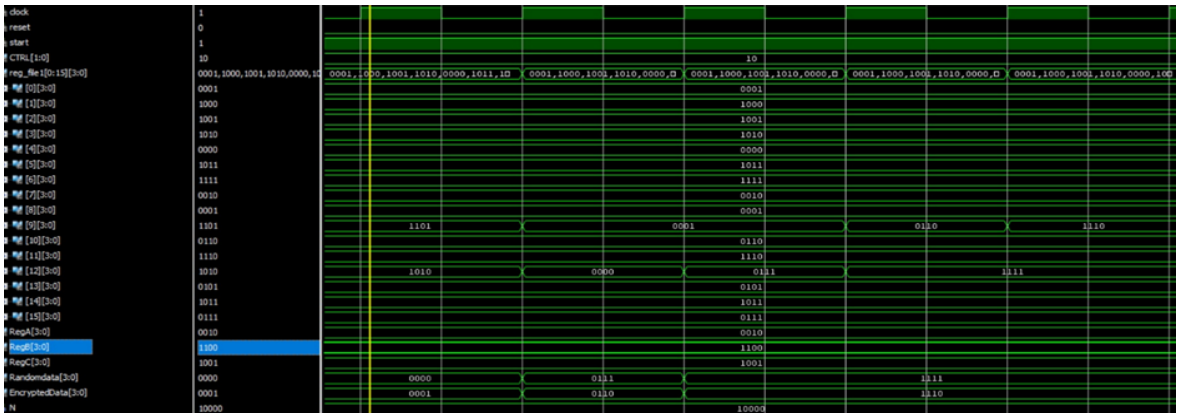

Figure 10. Simulation output waveform for the proposed coprocessor architecture for CTRL="10"

Table 5. Register file content

| MNEMONIC | Rs(source) | Rd1 (destination) | Rd2 (destination) |
|---|---|---|---|
| PRNG | R0(0001) | R4(0100) | R8(1000) |
| TRNG2 | R2(0010) | R12(1100) | R9(1001) |

From Figures 9, 10, and Table 5, it is clear that the proposed coprocessor is working correctly. The expected transition of the register's binary values for the CTRL values "01" and "10" is shown in Table 6. Figures 9 and 10 show that the latency value is 3 clock cycles for getting the register file's (signal reg_file1) content updated.

Table 6. Expected transition of the register's values in the register file

| Register index | Binary values |
|---|---|
| R0 | 0001 |
| R1 | 1000 |
| R2 | 1001 |
| R3 | 1010 |
| R4 | 1100→0000 |
| R5 | 1011 |
| R6 | 1111 |
| R7 | 0010 |
| R8 | 1010→0000→0001 |
| R9 | 1101→0001→0110→1110 |
| R10 | 0110 |
| R11 | 1110 |
| R12 | 1010→0000→0111→1111 |
| R13 | 0101 |
| R14 | 1011 |
| R15 | 0111 |

Figure 11 represents the RTL schematic diagram of the implemented coprocessor architecture. Table 7 represents synthesis results for the implemented coprocessor architecture. From the timing_report command, the data path delay is 4.326 ns, yielding a throughput value [22] of 231.16 Mbps for the coprocessor. Table 8 shows area consumption results in terms of LUTs for the proposed and existing coprocessor architectures. This table shows that the proposed coprocessor consumes a very small number of LUTs in comparison with existing coprocessor architectures. Also, the proposed coprocessor operates at a higher clock frequency in comparison with existing architectures.

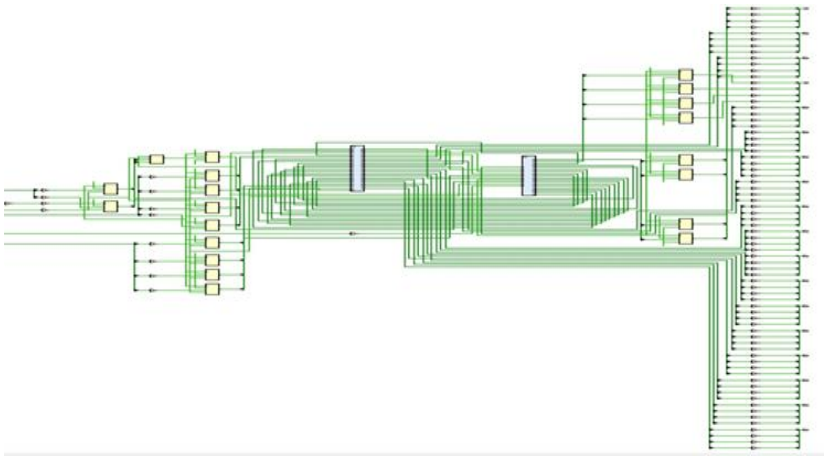

Figure 11. Coprocessor architecture RTL schematic diagram

Table 7. Synthesis results for the proposed coprocessor architecture

| Total on chip power consumption(W) | Speed(ns) | Resources | Utilization |
|---|---|---|---|
| 36.42 | 4.326 | | |
| | | FF | 180 |
| | | LUT | 100 |
| | | Memory LUT | 1 |

Table 8. Area consumption results comparison between proposed and existing coprocessor architecture

| Work | FPGA | Area consumption (LUTs) | Frequency (MHz) |
|---|---|---|---|
| Liu *et al.* [26] | Virtex-4 | 24,003 | |
| Hamilton and Marnane [13] | Virtex-5 | 28,508 | |
| Okada *et al*. [15] | | | 66 |
| Banerjee *et al*. [27] | | | 16 |
| Fritzmann *et al*. [28] | | | 45 |
| Banerjee *et al*. [29] | | | 72 |
| Proposed coprocessor | Artix-7 | 100 | 100 |

## 4. CONCLUSION

In this work, a coprocessor for generating 4-bit random data and performing encryption is presented. Three types of RNG architectures are used to generate random data for performing encryption. TRNG1 architecture is based on modified ring osculators circuits, and TRNG2 architecture based on TERO loop is also modified. A 4-bit SIPO shift register circuit is used to convert 1-bit random data into 4-bit random data. Plaintext data for performing encryption is stored in source register Rs. All coprocessor blocks are designed using VHDL and implemented on Artix-7 FPGA. The proposed coprocessor consumes 100 LUTs, 180 FFs, and 1 memory LUT. The throughput value for the proposed coprocessor is 231.16 Mbps. For the implemented coprocessor architecture simulation output waveform and register file contents are shown. The latency value is 3 clock cycles for updating the register file content. NIST SP 800-22 test results of the proposed TRNG and PRNG architecture validate the randomness quality. The proposed coprocessor architecture consumes less area and operates at a higher clock frequency in comparison with existing coprocessor architecture, and the simulation results validate its functionality. The architecture is applicable to secure embedded systems such as ATMs, routers, and IoT devices. Future research will focus on incorporating decryption functions and scaling the design into a complete system-on-chip (SoC) architecture to support end-to-end cryptographic applications.

## FUNDING INFORMATION

Authors state no funding involved.

## AUTHOR CONTRIBUTIONS STATEMENT

This journal uses the Contributor Roles Taxonomy (CRediT) to recognize individual author contributions, reduce authorship disputes, and facilitate collaboration.

| Name of Author | C | M | So | Va | Fo | I | R | D | O | E | Vi | Su | P | Fu |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Manoj Kumar | ✓ | ✓ | | ✓ | ✓ | ✓ | | | ✓ | ✓ | | | | |

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

## CONFLICT OF INTEREST STATEMENT

Author declares that he has no significant competing financial, professional, or personal interests that might have influenced the performance or presentation of the work described in this manuscript.

## DATA AVAILABILITY

The data used to support the findings of this study are available from the corresponding author upon request.

## REFERENCES

[1] M. K. A. Nassim and Z. Zakarya, "FPGA-based implementation of a substitution box cryptographic co-processor for high-performance applications," *International Journal of Reconfigurable and Embedded Systems (IJRES)*, vol. 14, no. 2, pp. 587–596, 2025, doi: 10.11591/ijres.v14.i2.pp587-596.

[2] V. S. Miller, "Use of Elliptic Curves in Cryptography," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 1986, pp. 417–426, doi: 10.1007/3-540-39799-X_31.

[3] R. L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978, doi: 10.1145/359340.359342.

[4] H. Nejatollahi, N. Dutt, S. Ray, F. Regazzoni, I. Banerjee, and R. Cammarota, "Post-Quantum Lattice-Based Cryptography Implementations," *ACM Computing Surveys*, vol. 51, no. 6, pp. 1–41, Nov. 2019, doi: 10.1145/3292548.

[5] L. Choquin and F. Piry, "Arm Custom Instructions: Enabling Innovation and Greater Flexibility on Arm," Cambridge, UK, 2020.

[6] OpenHW Group, "CORE-VExtension Interface," Ottawa, ON, USA, 2022.

[7] C. J. Ezeofor and A. G. Ulasi, "Analysis of Network Data Encryption & Decryption Techniques in Communication Systems," *International Journal of Innovative Research in Science, Engineering and Technology*, vol. 03, no. 12, pp. 17797–17807, 2014, doi: 10.15680/ijirset.2014.0312008.

[8] V. Shaik and D. N. K., "Flexible and cost-effective cryptographic encryption algorithm for securing unencrypted database files at rest and in transit," *MethodsX*, vol. 9, p. 101924, 2022, doi: 10.1016/j.mex.2022.101924.

[9] H. B. Meitei and M. Kumar, "FPGA design and implementation of TRNG architecture using ADPLL based on fir as loop filter," *Analog Integrated Circuits and Signal Processing*, vol. 122, no. 1, 2025, doi: 10.1007/s10470-024-02295-8.
[10] H. B. Meitei and M. Kumar, "FPGA Implementation of True Random Number Generator Architecture Using All Digital Phase-Locked Loop," *IETE Journal of Research*, vol. 68, no. 3, pp. 1561–1570, 2022, doi: 10.1080/03772063.2021.1963333.
[11] L. Pan, G. Tu, S. Liu, Z. Cai, and X. Xiong, "A Lightweight AES Coprocessor Based on RISC-V Custom Instructions," *Security and Communication Networks*, vol. 2021, p. 9355123, 2021, doi: 10.1155/2021/9355123.
[12] J. Lee, W. Kim, and J. H. Kim, "A Programmable Crypto-Processor for National Institute of Standards and Technology Post-Quantum Cryptography Standardization Based on the RISC-V Architecture," *Sensors*, vol. 23, no. 23, p. 9408, 2023, doi: 10.3390/s23239408.
[13] M. Hamilton and W. P. Marnane, "Implementation of a secure TLS coprocessor on an FPGA," *Microprocessors and Microsystems*, vol. 40, pp. 167–180, 2016, doi: 10.1016/j.micpro.2015.10.009.
[14] Y. Wang, D. L. Maskell, and J. Leiwo, "A unified architecture for a public key cryptographic coprocessor," *Journal of Systems Architecture*, vol. 54, no. 10, pp. 1004–1016, 2008, doi: 10.1016/j.sysarc.2008.04.013.
[15] S. Okada, N. Torii, K. Itoh, and M. Takenaka, "Implementation of elliptic curve cryptographi coprocessor over GF(2 m) on an FPGA," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2000, pp. 25–40, doi: 10.1007/3-540-44499-8_2.
[16] P. H. W. Leong and I. K. H. Leung, "A microcoded elliptic curve processor using FPGA technology," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 10, no. 5, pp. 550–559, 2002, doi: 10.1109/TVLSI.2002.801608.
[17] X. Shang, W. Shan, and X. Liu, "Design and Implementation of a Reconfigurable Cryptographic Coprocessor with Multiple Side-Channel Attacks Countermeasures," *Journal of Circuits, Systems and Computers*, vol. 27, no. 11, 2018, doi: 10.1142/S0218126618501803.
[18] C. Tian, J. Zhu, W. Shan, and X. Fu, "VLSI design of reconfigurable cipher coprocessor supporting both symmetric and asymmetric cryptographic algorithms," in *Advances in Intelligent Systems and Computing*, 2014, pp. 299–307, doi: 10.1007/978-81-322-1759-6_36.
[19] W. Wang, J. Han, Z. Xie, S. Huang, and X. Zeng, "Cryptographie coprocessor design for IoT sensor nodes," in *2016 International SoC Design Conference (ISOCC)*, IEEE, Oct. 2016, pp. 37–38, doi: 10.1109/ISOCC.2016.7799761.
[20] G. S. Athanasiou, D. Boufeas, and E. Konstantopoulou, "A Robust ASCON Cryptographic Coprocessor for Secure IoT Applications," in *2024 Panhellenic Conference on Electronics and Telecommunications, PACET 2024 - Proceedings*, 2024, doi: 10.1109/PACET60398.2024.10497076.
[21] K. C. C. Loi and S. B. Ko, "Flexible elliptic curve cryptography coprocessor using scalable finite field arithmetic blocks on FPGAs," *Microprocessors and Microsystems*, vol. 63, pp. 182–189, 2018, doi: 10.1016/j.micpro.2018.09.003.
[22] H. B. Meitei and M. Kumar, "Design and Implementation of Multiple Ring Oscillator-Based TRNG Architecture by Using ADPLL," *IEEE Access*, vol. 13, pp. 9252–9264, 2025, doi: 10.1109/ACCESS.2025.3527507.
[23] M. Varchola and M. Drutarovsky, "New high entropy element for FPGA based true random number generators," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 351–365, doi: 10.1007/978-3-642-15031-9_24.
[24] M. Varchola and M. Drutarovsk, "New FPGA based TRNG Principle Using Transition Effect with Built-In Malfunction Detection," in *International Workshop on Cryptographic Architectures Embedded in Reconfigurable Devices-CryptArchi*, Prague (Czechia), 2009, pp. 150–155.
[25] M. Kumar, K. Shakyawar, and Navin, "OTP Generation Using LFSR To Validate Applicant Login Account," *Nanotechnology Perceptions*, vol. 20, no. S14, pp. 1313–1326, Nov. 2024, doi: 10.62441/nano-ntp.v20iS14.81.
[26] Z. Liu, D. Liu, and X. Zou, "An Efficient and Flexible Hardware Implementation of the Dual-Field Elliptic Curve Cryptographic Processor," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 3, pp. 2353–2362, 2017, doi: 10.1109/TIE.2016.2625241.
[27] U. Banerjee, A. Wright, C. Juvekar, M. Waller, Arvind, and A. P. Chandrakasan, "An Energy-Efficient Reconfigurable DTLS Cryptographic Engine for Securing Internet-of-Things Applications," *IEEE Journal of Solid-State Circuits*, vol. 54, no. 8, pp. 2339–2352, 2019, doi: 10.1109/JSSC.2019.2915203.
[28] T. Fritzmann, G. Sigl, and J. Sepúlveda, "RISQ-V: Tightly Coupled RISC-V Accelerators for Post-Quantum Cryptography," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, pp. 239–280, 2020, doi: 10.46586/tches.v2020.i4.239-280.
[29] U. Banerjee, T. S. Ukyab, and A. P. Chandrakasan, "Sapphire: A configurable crypto-processor for post-quantum lattice-based protocols," *IACR Transactions on Cryptographic Hardware and Embedded Systems*, vol. 2019, no. 4, pp. 17–61, 2019, doi: 10.13154/tches.v2019.i4.17-61.

## BIOGRAPHY OF AUTHOR

**Manoj Kumar** is currently working as an assistant professor in the Department of Electronics and Communication Engineering, National Institute of Technology, Manipur. Having completed his B.Tech Degree from NIT Calicut, and M.Tech. degree from the Indian Institute of Information Technology (IIIT), Allahabad, he started working as an Assistant Professor in NIT Manipur and received his Ph.D. degree from National Institute of Technology Manipur. He has published several research articles in national and international reputed journals and attended various conferences across India. His research area includes VLSI design, VLSI-DSP, digital electronics, and communications. He has published over 50 scientific articles in international and national journals of repute and at several conferences. He can be contacted at email: manojara400@gmail.com.