electronics

MDPI

Article

Hardware-Accelerated Cryptographic Random Engine for Simulation-Oriented Systems

Meera Gladis Kurian * and Yuhua Chen *

Department of Electrical and Computer Engineering, University of Houston, Houston, TX 77204, USA
* Correspondence: mgladiskurian@uh.edu (M.G.K.); yuhuachen@uh.edu (Y.C.)

Abstract

Modern computing platforms increasingly rely on random number generators (RNGs) for modeling probabilistic processes in simulation, probabilistic computing, and system validation. They are also essential for cryptographic operations such as key generation, authenticated encryption, and digital signatures. Deterministic Random Bit Generators (DRBGs), as specified in the National Institute of Standards and Technology (NIST) Special Publication (SP) 800-90A, provides a standardized method for expanding entropy into cryptographically strong pseudorandom sequences. This work presents the design and Field Programmable Gate Array (FPGA) implementation of a hash-based DRBG using Ascon-Hash256, a lightweight, quantum-resistant hash function from the NIST-standardized Ascon cryptographic suite. It implements hash-based derivation, instantiation, generation, and reseeding of the generator via iterative hash invocations and state updates. Leveraging Ascon's sponge-based structure, the design achieves efficient entropy absorption and diffusion while maintaining an area-efficient FPGA architecture, making it well suited for resource-constrained platforms. The diffusion properties of the proposed DRBG are evaluated through avalanche and reproducibility analyses, confirming strong sensitivity to input variations and secure, repeatable operation. Moreover, Monte Carlo and stochastic-diffusion evaluation of the generated bitstreams demonstrates correct convergence and statistically consistent behavior. These results confirm that the proposed hash-based DRBG provides reproducible, hardware-efficient, and cryptographically secure random numbers suitable for next-generation neuromorphic, probabilistic computing systems, and Internet of Things (IoT) devices.

Keywords: hash-based DRBG; Ascon-Hash256; lightweight cryptography; hardware acceleration; Monte Carlo simulation; probabilistic computing; FPGA

Academic Editors: Shaogang Hu and Guanchao Qiao

Received: 23 February 2026
Revised: 16 March 2026
Accepted: 17 March 2026
Published: 20 March 2026

1. Introduction

Random number generation is fundamental to a wide range of computing workloads, enabling probabilistic algorithms, brain-inspired computing architectures such as neuromorphic systems, large-scale scientific simulations, and cryptographic applications. In all these domains, the quality of the random number generator (RNG) significantly affects both security and correctness. In cryptographic systems, inadequate randomness can undermine security by enabling predictable keys, compromised session nonces, or broken authentication. In scientific and engineering simulations, hidden correlations, clustering, or distributional effects in random streams can bias results, slow algorithmic convergence, or mask true system behavior [1]. This influence becomes even more critical in large-scale parallel environments, where stream independence and correlation control are essential for accuracy and reproducibility [2]. Emerging paradigms such as neuromorphic systems

and probabilistic accelerators also rely on controlled stochasticity, further increasing the demand for scalable and hardware-efficient random number generation [3].

Secure platforms generally use a combination of physical entropy sources and deterministic random bit generators (DRBGs) to guarantee both unpredictability and throughput. Physical entropy sources generate true randomness from physical phenomena, such as electronic thermal noise, radioactive decay, photon emission, and fluctuations in chaotic systems [4–6]. However, their output rates are often constrained by validation requirements, sensitivity to environmental conditions, aging vulnerabilities, and post-processing to remove bias or non-uniformity. DRBGs overcome this limitation by expanding a finite amount of entropy into an extended, deterministic pseudorandom sequence while maintaining robust security guarantees. According to National Institute of Standards and Technology (NIST) Special Publication (SP) 800-90A, a DRBG is an algorithm that produces a bitstream from an initial value determined by a seed. The seed is derived from a randomness source, and its selection directly influences the security of the DRBG. When the seed remains confidential within the algorithm, the DRBG output remains unpredictable up to the algorithm's security strength. NIST SP 800-90A recommends DRBG mechanisms based on hash functions and block ciphers [7]. Among these, hash-based DRBGs are particularly attractive because of their efficiency, conceptual simplicity, and strong security properties.

Beyond security applications, high-quality deterministic randomness plays a key role in scalable computation, including stochastic optimization, probabilistic inference, and hardware-accelerated simulation [8]. Many modern computing platforms rely on controlled and reproducible randomness to support sampling-based algorithms, hardware verification, and simulation-driven system design [9]. In such systems, DRBG functionality is often implemented in software on embedded processors due to ease of deployment and flexibility. Some examples include software implementations of CTR-DRBG using Advanced Encryption Standard (AES) and Hash-DRBG using Secure Hash Algorithm (SHA), as defined in NIST SP 800-90A [7]. These DRBGs are widely deployed in embedded cryptographic libraries such as OpenSSL and mbedTLS to support secure boot, key generation, and communication protocols [10–12]. However, software-based DRBG implementations may introduce performance overheads, increased latency, and limited throughput, particularly in large-scale simulation and stochastic computing environments where other components are hardware-accelerated [13]. Furthermore, software execution may expose sensitive internal state transitions to side-channel leakage, particularly in systems lacking dedicated hardware isolation mechanisms such as Trusted Execution Environments (TEEs) [14,15]. Hardware-based DRBG implementations address these challenges by providing predictable timing, improved isolation, and efficient integration into hardware accelerators and embedded processing systems.

Existing DRBG implementations commonly rely on SHA-family hash functions. While efficient pipelined FPGA implementations of SHA have been widely studied, the hash-based DRBG requires multiple hash evaluations, which can increase computational costs [16,17]. The Ascon family of cryptographic algorithms, selected through the NIST Lightweight Cryptography (LWC) standardization process, was specifically designed for low-power Internet ofThings (IoT) nodes and embedded systems. Ascon is based on a sponge-based duplex construction that enables compact Field Programmable Gate Array (FPGA) and ASIC implementations. It provides strong diffusion and functional reuse across multiple cryptographic modes, including authenticated encryption, hashing, and eXtendable-Output Functions (XOFs) [18]. Among these, the hash variant Ascon-Hash256 is an appropriate choice for DRBGs deployed in always-on sensing nodes, edge neuromorphic accelerators, and battery-powered IoT devices.

Despite the standardization and extensive evaluation of Ascon, its integration into a fully NIST-compliant hash-based DRBG and its efficient hardware implementation remain largely unexplored. Motivated by this gap, this work presents a complete FPGA implementation and evaluation of a hash-based DRBG using Ascon-Hash256 as the underlying hash function. The proposed architecture adheres to the NIST SP 800-90A specification and adopts a modular design suitable for FPGA implementation. The generated pseudorandom bitstreams are evaluated through empirical analysis, including deterministic reproducibility, diffusion analysis, and statistical testing, as well as simulation-oriented benchmarks such as Monte Carlo-based computations [19]. These results demonstrate that the proposed Ascon-Hash256-based DRBG provides a secure, statistically robust, and resource-efficient randomness generation solution suitable for cryptographic applications, neuromorphic computing architectures, probabilistic algorithms, and low-power embedded platforms. The main contributions of this work include:

- An architectural design of a NIST SP 800-90A-compliant hash-based DRBG using the NIST-standardized Ascon-Hash256,
- Implementation and evaluation of the proposed design on Xilinx Artix-7 FPGA, demonstrating efficient hardware realization for embedded and edge computing platforms, and
- Comprehensive validation through statistical testing, deterministic reproducibility analysis, and Monte Carlo-based simulation benchmarks, confirming the cryptographic strength of the proposed DRBG for stochastic computation and simulation applications.

The remainder of this paper is organized as follows. Section 2 reviews the background of hash-based DRBGs and the Ascon-Hash256 function. Section 3 describes the construction of the proposed Ascon-Hash-based DRBG, including the instantiation, random bit generation, and reseeding procedures, as well as the internal state organization and data formatting in accordance with the NIST SP 800-90A framework. Section 4 presents the proposed architecture and FPGA implementation. Section 5 evaluates implementation results as well as statistical analysis of the Ascon-Hash256 based DRBG. Section 6 discusses the key design insights of the proposed DRBG. Section 7 concludes this paper.

2. Background and Related Work

IoT and edge computing platforms deploy massive sensor networks that continuously collect and transmit data for monitoring, control, and automation. Ensuring the confidentiality, integrity, and authenticity of this data requires cryptographic mechanisms such as encryption, authentication, and secure communication protocols, all of which rely on secure random bit generation [17]. While many IoT nodes are implemented using low-cost microcontrollers, advanced IoT gateways and industrial edge nodes increasingly adopt hybrid architectures in which a microcontroller manages networking and system control, while dedicated hardware accelerators, such as FPGAs, handle performance-critical data processing or cryptographic tasks.

Beyond secure communication, sensor data is increasingly used to construct digital twins, which are virtual replicas of physical systems. They require high quality RNG with strong statistical properties to support monitoring, predictive maintenance, and system-level analysis [20]. Similarly, neuromorphic computing systems incorporate stochastic behavior into neuronal dynamics by injecting randomness into parameters such as synaptic currents, membrane potentials, and refractory intervals, enabling neural sampling and probabilistic inference [21]. Integrating randomness directly into hardware provides an alternative approach that enables parallel sampling while reducing computational and memory overhead. Such hardware-level stochastic behavior is particularly valuable for

applications such as Bayesian neural networks, generative models, and Monte Carlo methods, which require frequent sampling from complex probability distributions [22].

Recent neuromorphic processors, including Intel Loihi, IBM TrueNorth, and BrainChip Akida, exemplify the growing deployment of stochastic neural computation in edge environments [23,24]. Many neuromorphic learning and inference algorithms rely on probabilistic mechanisms such as stochastic synaptic updates and random sampling, which require reliable sources of randomness [21]. Reliable random bit generation is therefore essential to ensure both the correctness of probabilistic computation and the security of edge-connected neuromorphic systems. Cryptographically secure DRBGs provide a standardized and hardware-efficient solution for generating high-quality random bitstreams with strong statistical properties, making them well suited for secure, reproducible, and scalable operations in edge, IoT, and neuromorphic computing platforms.

According to NIST, RNGs are broadly classified into non-deterministic random bit generators (NRBGs) and deterministic random bit generators. NRBGs produce output directly from physical entropy sources whose behavior is inherently unpredictable. These sources include thermal noise, oscillator jitter, metastability, and emerging quantum and nanoscale phenomena [25,26]. In contrast, DRBGs generate pseudorandom sequences algorithmically from an initial seed derived from a randomness source. Although deterministic in operation, properly constructed DRBGs provide outputs that are computationally indistinguishable from true randomness up to their claimed security strength [7,27]. Algorithmic generators lacking cryptographic guarantees are widely used as pseudorandom number generators (PRNGs), such as the Permuted Congruential Generator (PCG) and the Mersenne Twister [28,29], which are fast and space-efficient but do not provide cryptographic guarantees.

As summarized in Table 1, RNGs support critical functions ranging from cryptographic key and nonce generation to physical modeling and neural sampling, directly influencing both system security and computational correctness. Unlike conventional PRNGs, DRBGs are stateful cryptographic mechanisms designed to provide prediction resistance, backtracking resistance, and controlled reseeding. These properties ensure that past or future outputs cannot be inferred even if partial internal state information is exposed. NIST SP 800-90A specifies standardized DRBGs based on hash functions and block ciphers [7]. Hash-based DRBGs are well suited for hardware implementation due to their structural simplicity, non-requirement of complex key scheduling, and flexibility in supporting variable-length output. Prior implementations of hash-based DRBGs have predominantly relied on SHA-family hash functions or AES-based counter-mode constructions, which often incur significant area, power, and latency overheads on constrained platforms [16,30].

Beyond cryptographic security, DRBGs play a vital role in stochastic computation, simulation frameworks, and accelerated probabilistic processing. Monte Carlo methods, probabilistic modeling, and hardware verification frameworks rely on statistically independent and uniformly distributed random inputs to ensure accurate convergence and unbiased estimation [8]. Hardware accelerators, including FPGA-based simulation engines and embedded processing systems, increasingly integrate dedicated RNG modules to support high-throughput sampling and stochastic computation. Compared to software-based RNG implementations, hardware-oriented DRBGs provide deterministic timing behavior, improved reproducibility, and reduced performance overhead, enabling scalable and efficient probabilistic computation [2].

Table 1. Applications of random number generation across modern computing domains.

Application Domain	Role of Random Numbers
Cryptography and Secure Communication	Key generation, nonce generation, encryption protocols, authentication, and secure boot mechanisms [7,18].
IoT Security and Embedded Systems	Secure firmware updates, authentication protocols, and trusted execution environments [6,31].
Neuromorphic and Probabilistic Computing	Stochastic neuron firing, probabilistic inference, and neural sampling in hardware accelerators [32,33].
Monte Carlo and Scientific Simulation	Numerical estimation, stochastic modeling, uncertainty analysis, and probabilistic simulation [8,34].
Machine Learning and Artificial Intelligence	Random initialization, stochastic optimization, and probabilistic model training [35].
Hardware Testing and Verification	Randomized test pattern generation, fault injection, and system validation [36].
Blockchain and Distributed Systems	Nonce generation, consensus protocols, and cryptographic hashing [37].
Probabilistic Data Structures	Bloom filters, hash-based indexing, and probabilistic membership testing [38,39].
Computer Graphics and Visualization	Ray tracing, procedural generation, and realistic rendering effects [40].

The NIST LWC initiative has accelerated the development of cryptographic primitives specifically optimized for resource-constrained environments. Following a rigorous multi-round evaluation process, Ascon was selected as the NIST standard for lightweight authenticated encryption and hashing due to its sponge-based design, efficient bitsliced hardware implementation, and strong security guarantees [18]. Several studies have demonstrated efficient FPGA and embedded implementations of Ascon, highlighting its low resource overhead, scalability, and suitability for secure edge and IoT platforms [31,41]. Ascon's permutation core serves as the fundamental computational component and supports multiple cryptographic functionalities, including authenticated encryption, hashing, and extendable-output generation, enabling core reuse and thereby simplifying system integration.

Sponge-based constructions provide a natural foundation for DRBGs because they can securely absorb entropy inputs and generate arbitrarily long pseudorandom outputs through iterative permutation of an internal state [42,43]. This aligns closely with the structure of hash-based DRBGs specified in NIST SP 800-90A, where a cryptographic hash function is used to derive and update the internal state while preserving unpredictability and diffusion. Ascon-Hash256 inherits these properties while maintaining a lightweight hardware footprint, making it particularly well suited for FPGA implementation. Its sponge-based design enables efficient state transformation without complex arithmetic or lookup tables, enabling secure and resource-efficient random bit generation. These characteristics make Ascon-Hash256 a strong choice for implementing hash-based DRBG targeting edge computing, IoT, and hardware-accelerated stochastic systems. The following section presents the architecture and FPGA implementation of the proposed Ascon-Hash256-based DRBG.

3. Analysis of Ascon-Hash-Based DRBG Construction

NIST SP 800-90A allows hash-based DRBG constructions based on an approved cryptographic hash function, provided that the selected hash supports the required security strength and sufficient entropy for the seed during instantiation [7]. In a hash-based DRBG, entropy is obtained from an external source to initialize the internal state, after which successive hash function invocations deterministically expand and update the state, ensuring

forward security and resistance to state recovery. In accordance with this framework, the proposed DRBG design adopts Ascon-Hash256, standardized in NIST SP 800-232, as the underlying hash function. Ascon-Hash256 produces a 256-bit digest and offers 128-bit security against collision, preimage, and second-preimage attacks under the standardized claims [18]. The algorithm follows a sponge construction with a 320-bit internal state composed of a 64-bit rate and a 256-bit capacity, offering strong diffusion and robustness against state-recovery and forgery-style attacks. Recent cryptanalytic studies suggest that the practical preimage resistance of Ascon-Hash256 may approach higher security margins under certain attack models, reinforcing confidence in its suitability for DRBG implementation [44].

At the algorithmic level, Ascon-Hash256 operates in three stages: initialization, absorption, and squeezing. The internal state is first initialized using a fixed initialization vector, after which message blocks are absorbed into the state with permutation-based diffusion applied between blocks. Following padding and finalization, the squeezing phase repeatedly applies the permutation to extract output blocks and form the final digest [18]. This sponge construction naturally supports variable-length inputs and repeated output generation, aligning well with the needs of hash-based DRBG procedures such as seed derivation, state update, and pseudorandom bit generation.

The Ascon-Hash-based DRBG is essentially a stateful construction in which an internal state is initialized, periodically refreshed, and deterministically evolved to produce pseudorandom output. As illustrated in Figure 1, the DRBG consists of three primary stages: Instantiate, Reseed, and Generate. Two internal procedures, namely, `Hash_df` and `Hashgen`, are employed to support these stages. The `Hash_df` function is responsible for compressing entropy and optional inputs into a fixed-length seed suitable for initializing or updating the internal state, while `Hashgen` governs the expansion of the internal state into variable-length output during the Generate phase. Together, these procedures enable deterministic expansion of the internal state between Reseed events.

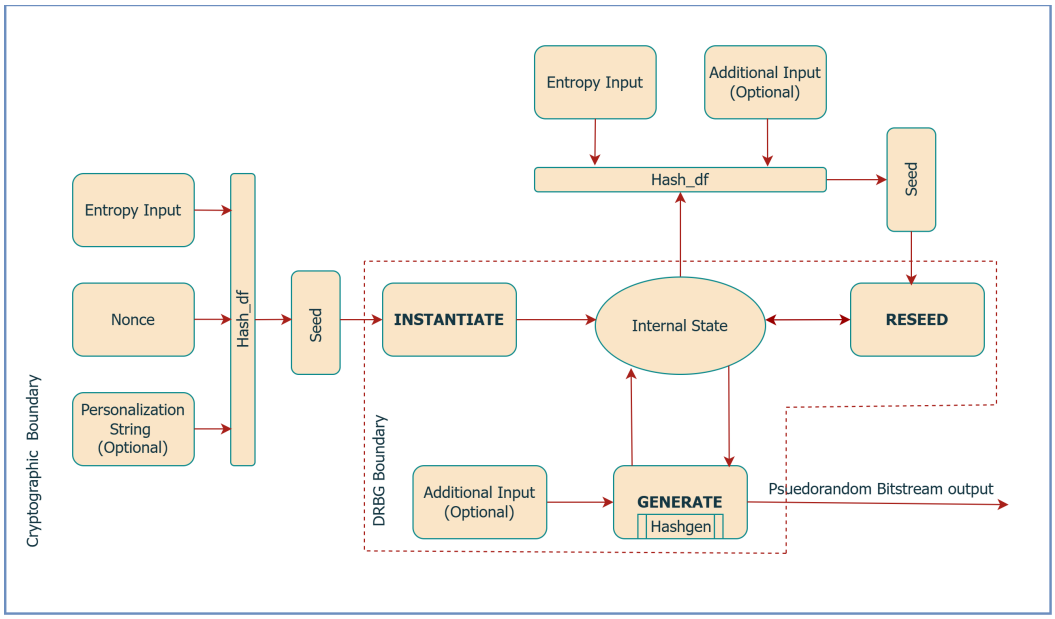


Figure 1. High-level overview of the hash-based DRBG mechanism.

The internal state of the DRBG consists of a seed value (V), a constant value (C), and a reseed counter (*reseed_counter*). Both V and C are derived through the `Hash_df` procedure during Instantiate and Reseed operations. State evolution during the Generate operation follows the NIST-defined update rule, ensuring that each output depends on both the prior

internal state and fresh hash evaluations. All hash invocations use domain-separation constants to distinguish between Instantiate, Reseed, and Generate operations.

4. Proposed Hardware Architecture and FPGA Implementation

The proposed design follows a modular, FPGA-oriented architecture that implements a hash-based DRBG using the NIST-standardized Ascon-Hash256 function. Figure 2 illustrates the high-level FPGA organization of the proposed system. In this design, the architecture consists of four main modules.

- A reusable Ascon-Hash256 core;
- A seed derivation function module, `Hash_df`;
- A Generate Control module, `Hashgen`;
- A DRBG control engine managing instantiation, reseeding, and generation.

This layered structure separates cryptographic computation from control logic, enabling flexible configuration while simplifying verification and design reuse.

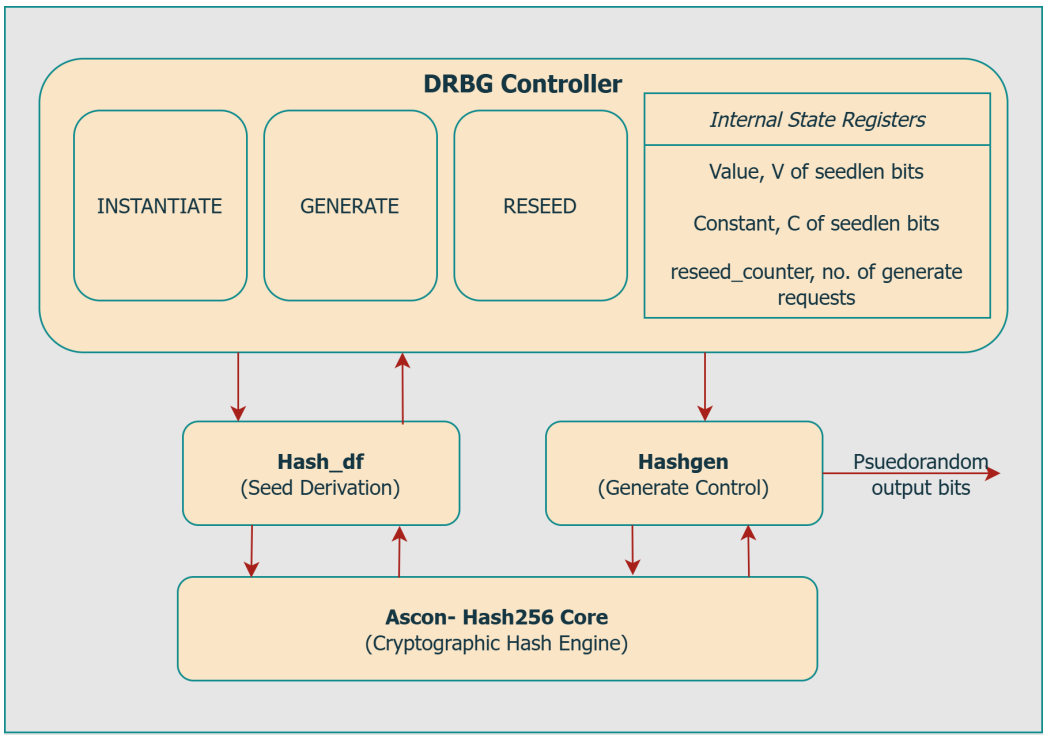


Figure 2. FPGA-based architecture of the proposed Ascon-Hash-based DRBG.

4.1. Ascon-Hash256 Core

At the lowest level, the Ascon-Hash256 core implements the sponge-based hashing mode defined in the Ascon specification. The permutation operates on a 320-bit internal state, which is divided into a 64-bit rate portion and a 256-bit capacity portion. The permutation is realized as a sequence of rounds, each consisting of three layers: a constant addition layer, which injects round-dependent constants to ensure asymmetry between rounds; a nonlinear substitution layer, implemented using a bit-sliced S-box to provide cryptographic confusion; and a linear diffusion layer, which ensures rapid entropy distribution and a strong avalanche effect [18].

The hash function follows a three-stage operation:

1. Initialization: The internal state is initialized using a fixed initialization vector defined by the Ascon-Hash256 specification, followed by application of the p^{12} permutation to establish a secure initial state.

2. Absorption: Input message blocks are sequentially XORed into the 64-bit rate portion of the state. After each block is absorbed, the p^{12} permutation is applied to mix the input data into the full 320-bit state, ensuring strong diffusion and resistance to cryptographic attacks.
3. Squeezing: After message padding and final permutation, the hash output is extracted from the rate portion of the state. Additional permutation applications are performed if more output bits are required, consistent with the sponge construction.

A `ready/valid` handshake interface enables streaming absorption of input blocks without requiring full message buffering. This streaming architecture is particularly well suited for FPGA implementations, as it minimizes on-chip memory requirements, supports continuous data flow, and enables efficient integration with higher-level cryptographic modules such as the derivation function and DRBG control engine.

4.2. Hash Derivation Function

The derivation function, `Hash_df`, repeatedly invokes the underlying Ascon-Hash256 core to derive a fixed-length seed from a variable-length input. In the proposed design, the derivation function generates a 440-bit seed, consistent with NIST recommendations for hash-based DRBG. The input to the derivation function includes entropy input, a nonce, and optional personalization data. Internally, the derivation function performs multiple hash invocations and concatenates their outputs until the required seed length is reached. In this design, two sequential Ascon-Hash256 invocations are sufficient to generate the 440-bit seed, with the final output truncated to the required length following the expansion strategy defined in SP 800-90A.

4.3. Generate Control

The Generate Control module implements the `Hashgen` procedure defined in NIST SP 800-90A, which is responsible for producing pseudorandom output bits from the internal DRBG state. This module handles iterative invocations of the Ascon-Hash256 core and manages intermediate state updates. During a Generate request, the control block reads the current internal state value V, which is a 440-bit register maintained by the DRBG controller. The Hashgen procedure operates by repeatedly applying the hash function to the evolving state value and concatenating the resulting outputs until the requested number of pseudorandom bits has been generated. Once sufficient output has been generated, the leftmost required number of bits is returned as the pseudorandom output.

4.4. DRBG Control Engine

The DRBG control module orchestrates instantiation, random number generation, and optional reseeding operations in accordance with the hash-based DRBG mechanism. The control logic is implemented as a Finite State Machine (FSM), as illustrated in Figure 3, which coordinates entropy processing, hash execution, output generation, and internal state updates.

The Reseed operation, implemented as a dedicated sub-FSM, refreshes the internal state using new entropy input and optional additional data. This process invokes the derivation function, `Hash_df`, to compute a new internal state and resets the reseed counter, thereby restoring full cryptographic strength and preventing state compromise. The FSM interfaces directly with the Ascon-Hash256 core using a handshake mechanism, ensuring proper sequencing of hash operations and deterministic timing behavior. Overall, this architecture provides a resource-efficient and lightweight FPGA implementation of a hash-based DRBG for secure random number generation. The following subsections describe the

implementation of the Instantiate, Reseed, and Generate operations within the proposed FSM-based design.

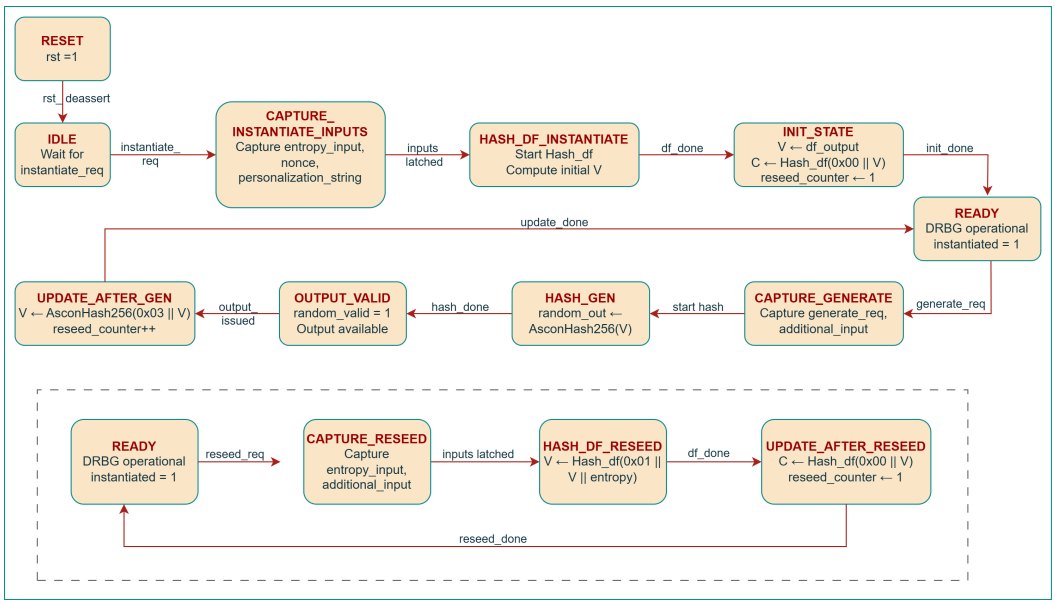


Figure 3. FSM of the Ascon-Hash256-based DRBG control engine. The main FSM handles initialization, random number generation, and state updates. The Reseed procedure is shown as a separate FSM (inside the dotted box) to refresh the internal state using fresh entropy input without disrupting normal operation.

4.4.1. Instantiate

The Instantiate procedure initializes the internal state of the Hash-based DRBG using an entropy input, a nonce, and an optional personalization string. These values are first concatenated to form the seed material:

$$seed\_material = entropy\_input \| nonce \| personalization\_string. \tag{1}$$

The derivation function, `Hash_df`, is then applied to compress this variable-length input into a fixed-length seed value:

$$seed = \texttt{Hash\_df}(seed\_material, seedlen). \tag{2}$$

where *seedlen* denotes the internal state length of the DRBG. The internal working state is initialized directly from this seed,

$$V \leftarrow seed, \tag{3}$$

and a constant value is derived by hashing the state prefixed with a single zero byte,

$$C = \texttt{Hash\_df}((0x00 \| V), seedlen). \tag{4}$$

Finally, the reseed counter is initialized as

$$reseed\_counter \leftarrow 1. \tag{5}$$

The procedure returns the initialized internal state $(V, C, reseed\_counter)$ and is executed during initial setup or after a full reseeding of the generator.

4.4.2. Generate

The Generate procedure produces pseudorandom output from the current internal state and updates the DRBG state accordingly. Before output generation, the reseed counter is checked. If the reseed counter exceeds the predefined reseed interval, the procedure returns an indication that reseeding is required. If additional input is provided, it is first incorporated into the internal state by computing

$$w = \texttt{Ascon-Hash256}(0x02 \| V \| additional\_input), \tag{6}$$

and updating the state as

$$V \leftarrow (V + w) \bmod 2^{seedlen}. \tag{7}$$

Pseudorandom output bits are then generated using the output expansion function `Hashgen`, which iteratively hashes the evolving state to produce the requested number of output bits.

$$returned\_bits = \texttt{Hashgen}(requested\_number\_of\_bits, V). \tag{8}$$

After output generation, a hash of the current state is computed,

$$H = \texttt{Ascon-Hash256}(0x03 \| V), \tag{9}$$

and the internal state is updated according to

$$V \leftarrow (V + H + C + reseed\_counter) \bmod 2^{seedlen}. \tag{10}$$

Finally, the reseed counter is incremented as

$$reseed\_counter \leftarrow reseed\_counter + 1, \tag{11}$$

and the procedure returns the generated output together with the updated internal state $(returned\_bits, V, C, reseed\_counter)$.

4.4.3. Reseed

The Reseed procedure updates the internal state of the hash-based DRBG using fresh entropy and optional additional input. A new seed material string is first constructed by prefixing the current state with a domain-separation byte and concatenating the external inputs:

$$seed\_material = 0x01 \| V \| entropy\_input \| additional\_input. \tag{12}$$

The derivation function `Hash_df` is then applied to compress this input into a new fixed-length seed:

$$seed = \texttt{Hash\_df}(seed\_material, seedlen). \tag{13}$$

The internal working state is updated from this new seed,

$$V \leftarrow seed, \tag{14}$$

and a constant value is recomputed by hashing the state prefixed with a single zero byte,

$$C = \texttt{Hash\_df}(0x00 \| V, seedlen). \tag{15}$$

Finally, the reseed counter is reset as

$$reseed\_counter \leftarrow 1. \tag{16}$$

The procedure returns the updated internal state $(V, C, reseed\_counter)$ and may be invoked periodically or whenever fresh entropy becomes available.

In the proposed architecture, the primary computational workload is handled by the Ascon-Hash256 core. Each hash evaluation requires several rounds of permutation on the 320-bit internal state. DRBG operations such as `Hash_df` and `Hashgen` invoke the hash core multiple times for seed derivation and pseudorandom bit generation. The remaining modules implement the algorithmic steps of the DRBG procedures, including *Instantiate*, *Generate*, and *Reseed*, while coordinating state updates and data flow between the internal registers and the hash module. These components introduce relatively small additional hardware complexity. Furthermore, the Ascon-Hash256 core can be reused across different DRBG operations.

5. Experimental Evaluation

This section evaluates the proposed Ascon-Hash DRBG from two complementary perspectives. First, FPGA implementation results are reported to assess the resource footprint, timing, and feasibility on resource-constrained devices. Second, a Python 3.13.9 -based evaluation is used to study the statistical characteristics of the pseudorandom bits generated by the proposed DRBG. This includes Monte Carlo π estimation to assess practical usefulness in stochastic workloads, determinism and reproducibility checks to ensure that identical seed and entropy inputs consistently produce identical output sequences, diffusion and avalanche sensitivity analysis to observe how small input changes influence the generated output, and lightweight statistical sanity checks to identify any structural bias or statistical dependence in the generated bitstream.

5.1. FPGA Implementation Results

The proposed Ascon-Hash-based DRBG was implemented using Xilinx Vivado 2024.2, targeting Artix-7 FPGA. The design operates at 100 MHz, corresponding to the default on-board clock. Table 2 summarizes the FPGA resource utilization of the complete DRBG architecture, including the Ascon-Hash256, derivation function, and DRBG control logic. The design occupies only 7.6% of the available slice LUTs and 4.8% of the available slice registers, demonstrating the efficiency achieved by adopting the lightweight Ascon-Hash256 as the underlying hash function for the DRBG. This low resource usage enables integration with other system components, such as communication interfaces, control logic, and accelerators, without incurring significant area overhead.

Table 2. FPGA resource utilization of the Ascon-Hash-based DRBG at 100 MHz.

Resource Type	Used
Slice LUTs	4835
Slice Registers	6118
Slices	1764

The estimated total on-chip power consumption of the design is 0.179 W at 100 MHz. This low power profile reflects the efficiency of the Ascon permutation logic and the absence of large memory blocks or complex arithmetic units.

The architecture is designed as an FSM, with each DRBG stage progressing through a sequence of states. Table 3 shows the measured cycle counts for the Instantiate, Reseed, and Generate procedures of the DRBG.

Table 3. Cycle count of DRBG operations measured from the FSM-based RTL implementation.

Operation	Cycles
Instantiate	262
Reseed	303
Generate (per 256-bit output block)	276

Table 4 compares FPGA implementations of hash-based DRBGs compliant with the NIST SP 800-90A specification. Prior works based on SHA-256 require 20,694 logic elements on an Altera Cyclone IV FPGA and 7327 LUTs on a Xilinx Virtex UltraScale+, respectively. The reduction in logic utilization in the proposed design can be attributed to Ascon-Hash256, the underlying hash function. The Ascon permutation primarily relies on simple bitwise operations such as XOR and shift/rotation, together with a compact bit-sliced S-box that maps efficiently to FPGA logic resources. As a result, the overall architecture can be implemented with a smaller hardware footprint while preserving full DRBG functionalities.

Table 4. Comparison of hash-based DRBG implementations across different FPGA platforms.

DRBG Architecture	Hash Function	Logic Resources	Device	Freq. (MHz)
Hash-DRBG [17]	SHA-256	20,694 Logic Elements	Altera Cyclone IV	68.43
Hash-DRBG [45]	SHA-256	7327 LUTs	Xilinx Virtex UltraScale+	260
Hash-DRBG (this work)	Ascon-Hash256	4835 LUTs	Xilinx Artix-7	100

Moreover, the DRBG operations are controlled by the FSM, which calls the Ascon-Hash256 core whenever needed. Because of these multiple hash invocations for output generation and internal state updates, the computational cost of the DRBG is determined by the number of permutation executions within the FSM flow. Future optimizations may explore higher-throughput permutation architectures, such as multi-round-per-cycle implementations, which could improve DRBG performance without modifying the overall control structure [31]. This flexibility makes the proposed design adaptable to various performance needs in FPGA-based cryptographic and stochastic computing systems.

5.2. Statistical Analysis

A Python-based evaluation framework was used to analyze the statistical characteristics of the pseudorandom sequences generated by the proposed Ascon-Hash-based DRBG. The Ascon-Hash256 implementation was verified against official known-answer test (KAT) vectors to ensure correct operation of the hash primitive. The DRBG was then instantiated to generate large pseudorandom bitstreams for subsequent statistical analysis.

The generated sequences were examined for possible bias, serial correlation, avalanche diffusion behavior, and convergence characteristics in Monte Carlo-based simulations. For controlled validation experiments, fixed entropy, nonce, and personalization inputs were used so that the reported figures and statistical measurements could be reproduced. In addition, separate multi-seed instantiations were performed to observe how the generator behaves across different entropy sources.

5.2.1. Monte Carlo π Estimation

Monte Carlo π estimation is used as an application-level stochastic model to assess the convergence behavior and practical suitability of the proposed Ascon-Hash256-based DRBG. In this approach, a large number of random bitstreams generated by the DRBG

are used to estimate π through repeated probabilistic trials. Convergence characteristics indicate how closely the generated samples approximate an ideal uniform distribution. Even minor correlations or structural bias in the output sequence may result in slower convergence or measurable deviation in the estimated value of π [34]. Thus, this estimation provides a practical means to evaluate whether the generated bitstreams behave as expected in simulation workloads. It is important to note that the Ascon permutation, the core structure of the Ascon cryptographic suite, has already undergone extensive cryptographic and statistical evaluation during the NIST LWC standardization process [46]. Therefore, the objective of this experiment is not to reevaluate the intrinsic statistical properties of Ascon but rather to assess the correctness and practical behavior of the proposed DRBG construction.

For Monte Carlo simulation, random pairs (x, y) were sampled uniformly from a unit square. A sample point was classified as lying inside the unit circle if it satisfies

$$x^2 + y^2 \leq 1. \tag{17}$$

Since the ratio of the area of a quarter circle to that of the unit square is $\pi/4$, the value of π was estimated as

$$\hat{\pi} = 4 \cdot \frac{N_{\text{in}}}{N}, \tag{18}$$

where N_{in} denotes the number of points inside the quarter circle and N is the total number of samples.

Three RNGs were evaluated: (i) the proposed Ascon-Hash256-based DRBG, (ii) PCG64 as provided by NumPy, and (iii) the operating system entropy interface `os.urandom`. All experiments used $N = 2{,}000{,}000$ sample pairs. To observe convergence behavior, intermediate estimates of $\hat{\pi}$ were logged periodically and plotted as a function of the sample count.

As the number of samples increases, a high-quality RNG is expected to produce estimates that converge toward the true value of π. Figure 4 illustrates the convergence behavior for all three RNG sources. At lower sample counts, visible fluctuations are observed due to inherent statistical variance. As N approaches 2×10^6, the estimates stabilize and converge toward π, indicating uniform sampling without observable bias.

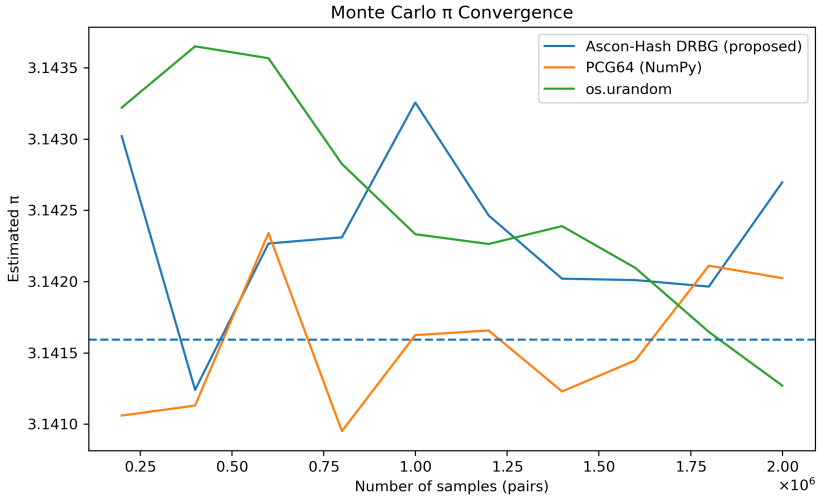


Figure 4. Monte Carlo π convergence over increasing sample counts (N up to 2×10^6 pairs).

Table 5 summarizes the final estimated value $\hat{\pi}$ and the absolute estimation error $|\pi - \hat{\pi}|$.

Table 5. Monte Carlo π estimation results for $N = 2{,}000{,}000$ sample pairs.

RNG	$\hat{\pi}$ (Final)	$\lvert\pi - \hat{\pi}\rvert$
Ascon-Hash DRBG (proposed)	3.142696	1.103×10^{-3}
PCG64 (NumPy)	3.142024	4.313×10^{-4}
os.urandom	3.142848	1.255×10^{-3}

All three RNGs achieve absolute errors on the order of 10^{-3}–10^{-4}, consistent with the expected $O(N^{-1/2})$ convergence rate of Monte Carlo methods. Minor differences in the final error across RNG sources reflect normal statistical variation rather than intrinsic differences in randomness quality. PCG64 exhibits the smallest final error in this particular run, while the proposed Ascon-Hash DRBG demonstrates comparable convergence behavior without persistent drift across logged batches.

5.2.2. Multi-Seed Statistical Validation

To verify that the observed statistical characteristics are inherent to the DRBG design rather than due to a particular seed, the generator was evaluated across 100 independent instantiations. Each instantiation was initialized with randomly generated entropy (256 bits) and a nonce (128 bits), conforming to the NIST SP 800-90A instantiation framework. For each instantiation, independent output sequences were generated and analyzed using multiple statistical and probabilistic metrics, including monobit fraction, serial correlation, avalanche diffusion behavior, Monte Carlo integration accuracy, and random-walk diffusion characteristics [19].

For statistical balance and correlation analysis, 200 KB of output data (1.6 Mbits) were generated per instantiation. Avalanche diffusion was evaluated by flipping a single entropy bit and computing the Hamming distance ratio between two independent output streams of equal length (200 KB each). Monte Carlo integration experiments used 50,000 uniformly distributed samples derived from 64-bit output words to estimate the integral $\int_0^1 e^{-x^2} dx$. Random-walk diffusion was evaluated using a 50,000-step two-dimensional random walk driven by DRBG output bits. The random-walk experiment reflects practical simulation workloads in which successive random decisions accumulate over time, making it a useful indicator of long-term diffusion and stability.

Table 6 summarizes the mean and standard deviation of the evaluated metrics across all instantiations. These metrics collectively evaluate statistical balance, independence, diffusion strength, and probabilistic correctness, which are essential properties for cryptographic security as well as stochastic and simulation-driven computing applications.

Table 6. Statistical properties across 100 independent DRBG instantiations.

Metric	Mean	Standard Deviation
Fraction of ones (Monobit)	0.500047	0.000374
Serial correlation (lag-1)	-4.0×10^{-5}	7.53×10^{-4}
Serial correlation (lag-8)	-8.9×10^{-5}	8.24×10^{-4}
Avalanche ratio	0.499990	0.000433
Monte Carlo integration estimate	0.746747	0.000791

Monte Carlo integration experiments provide an additional validation of statistical correctness across independent entropy instantiations. Unlike the Monte Carlo π estimation presented in Section 5.2.1, which evaluates convergence behavior in a stochastic simulation setting and compares multiple RNGs, this experiment focuses on assessing statistical stability and absence of seed-dependent bias within the proposed DRBG itself. Using DRBG-generated random samples to estimate the integral $\int_0^1 e^{-x^2} dx$, each instantiation

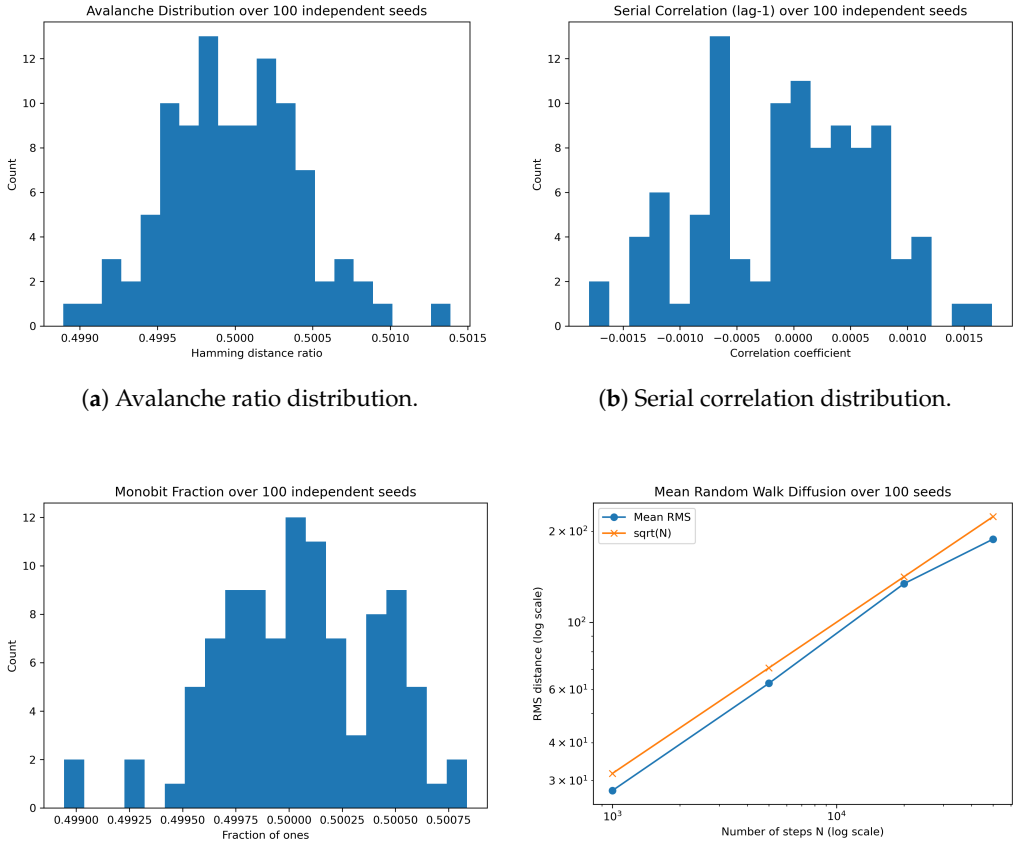
produced 50,000 uniformly distributed samples derived from 64-bit output words. The mean estimate of 0.746747 closely matches the theoretical value of approximately 0.746824. The extremely small variance across independent instantiations confirms stable probabilistic behavior and absence of seed-dependent sampling bias.

6. Discussion

The experimental results demonstrate that the proposed Ascon-Hash256-based DRBG, implemented on a moderate-capability FPGA platform, achieves efficient resource utilization and low power consumption. Minor architectural optimizations can further improve throughput without significantly increasing design complexity. The statistical analysis confirms that the DRBG preserves the strong diffusion and randomness properties of the underlying Ascon-Hash256. Therefore, from an implementation perspective, the design appears promising for a wide range of platforms and applications, ranging from simulation workloads to secure cryptographic protocols, while accommodating requirements in terms of performance and hardware efficiency.

The distributions of statistical properties across independent instantiations are illustrated in Figure 5. The avalanche ratio distribution shown in Figure 5a remains tightly centered around the theoretical ideal value of 0.5, indicating that small changes in entropy inputs consistently produce widespread and uniform changes in output bits. This behavior provides empirical evidence that the diffusion properties of Ascon-Hash256 are preserved within the DRBG construction. The serial correlation distributions shown in Figure 5b are symmetrically centered around zero with extremely low variance, indicating no detectable linear dependence between adjacent output bits across independent instantiations. This behavior demonstrates the absence of measurable short-range correlation and supports the statistical independence required for unbiased sampling and unpredictability in both cryptographic and simulation-oriented applications. The monobit fraction distribution shown in Figure 5c confirms excellent statistical balance, with the fraction of ones consistently converging to the ideal value of 0.5. The limited variation across independent seeds suggests that no measurable seed-dependent bias is introduced by the DRBG construction. The random-walk diffusion behavior illustrated in Figure 5d provides a higher-level validation of stochastic correctness. The mean displacement magnitude closely follows the theoretical $\sqrt{N}$ scaling relationship expected for ideal random sequences. The inference is that the generator exhibits correct diffusion characteristics, ensuring reliable performance in probabilistic simulation workloads.

These results indicate that the proposed DRBG preserves the statistical properties expected from the underlying Ascon-Hash256 function. The consistent behavior observed across independent instantiations also highlights the stability of the design. From a system design perspective, these findings suggest that lightweight cryptographic primitives such as Ascon-Hash256 can be incorporated into standardized DRBG constructions without affecting the statistical quality of the generated outputs. Unlike NRBG or Physical Unclonable Function (PUF) based approaches, which depend on device-specific randomness, DRBG constructions expand a small amount of physical entropy into long pseudorandom sequences while producing deterministic outputs. This property is particularly useful in applications such as system validation, simulation, and cryptographic protocols.



(**a**) Avalanche ratio distribution.

(**b**) Serial correlation distribution.

(**c**) Monobit fraction distribution.

(**d**) Random-walk diffusion behavior.

Figure 5. Statistical behavior of the proposed Ascon-Hash-based DRBG across 100 independent instantiations.

7. Conclusions

This work presented a lightweight hash-based DRBG architecture based on the NIST-standardized Ascon-Hash256 and demonstrated its implementation on an FPGA platform. The implementation results show that the design can be realized with low FPGA resource usage. Statistical evaluation, including Monte Carlo π estimation, multi-seed validation across independent entropy instantiations, avalanche analysis, and probabilistic diffusion experiments, indicated consistent statistical balance, the absence of detectable correlation, and strong diffusion properties inherited from the underlying Ascon-Hash256 primitive. These observations suggest that the generated bitstreams are suitable for applications requiring reproducible stochastic behavior, including simulation workloads, secure embedded systems, machine learning acceleration, and neuromorphic computing platforms. Future work will investigate DRBG constructions based on Ascon-XOF128, utilizing its variable-length output capability to produce extended pseudorandom streams from a single internal state for large-scale artificial intelligence and probabilistic computing systems.

Author Contributions: Conceptualization, M.G.K. and Y.C.; methodology, M.G.K. and Y.C.; software, M.G.K.; validation, M.G.K. and Y.C.; formal analysis, M.G.K.; investigation, M.G.K.; resources, M.G.K.; data curation, M.G.K.; writing—original draft preparation, M.G.K.; writing—review and editing, M.G.K. and Y.C.; visualization, M.G.K.; supervision, Y.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding authors.

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

Abbreviations

The following abbreviations are used in this manuscript:

DRBG	Deterministic Random Bit Generator
NIST	National Institute of Standards and Technology
SP	Special Publication
IoT	Internet of Things
FPGA	Field Programmable Gate Array
RNG	Random Number Generator
SHA	Secure Hash Algorithm
TEE	Trusted Execution Environment
NRBG	Non-Deterministic Random Bit Generator
PRNG	Pseudorandom Number Generator
FSM	Finite State Machine
PUF	Physical Unclonable Function
AES	Advanced Encryption Standard
LWC	Lightweight Cryptography

References

1. Crocetti, L.; Nannipieri, P.; Di Matteo, S.; Fanucci, L.; Saponara, S. Review of methodologies and metrics for assessing the quality of random number generators. *Electronics* **2023**, *12*, 723. [CrossRef]
2. Howes, L.; Thomas, D. Efficient Random Number Generation and Application Using CUDA. In *GPU Gems 3*; Nguyen, H., Ed.; Addison-Wesley Professional: Upper Saddle River, NJ, USA, 2007; Chapter 37.
3. Indiveri, G.; Liu, S.C. Memory and information processing in neuromorphic systems. *Proc. IEEE* **2015**, *103*, 1379–1397. [CrossRef]
4. Wold, K.; Tan, C.H. Analysis and Enhancement of Random Number Generator in FPGA Based on Oscillator Rings. In Proceedings of the 2008 International Conference on Reconfigurable Computing and FPGAs, Cancún, Mexico, 3–5 December 2008; pp. 385–390. [CrossRef]
5. Li, C.; Wang, Q.; Jiang, J.; Guan, N. A metastability-based true random number generator on FPGA. In Proceedings of the 2017 IEEE 12th International Conference on ASIC (ASICON), Guiyang, China, 25–28 October 2017; pp. 738–741.
6. Kalanadhabhatta, S.; Kumar, D.; Anumandla, K.K.; Reddy, S.A.; Acharyya, A. PUF-based secure chaotic random number generator design methodology. *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.* **2020**, *28*, 1740–1744. [CrossRef]
7. Barker, E.; Kelsey, J. *NIST Special Publication 800-90A Rev. 1: Recommendation for Random Number Generation Using Deterministic Random Bit Generators*; Technical Report; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2015. [CrossRef]
8. Gentle, J.E. *Random Number Generation and Monte Carlo Methods*; Springer: New York, NY, USA, 2003.
9. Vijaykumar, V.; Sekar, S.R.; Jothin, R.; Diniesh, V.; Elango, S.; Ramakrishnan, S. Novel light weight hardware authentication protocol for resource constrained IoT based devices. *IEEE J. Radio Freq. Identif.* **2024**, *8*, 31–42. [CrossRef]
10. OpenSSL Project. OpenSSL Random Number Generation. 2023. Available online: https://www.openssl.org/docs/man3.0/man7/RAND.html (accessed on 16 February 2026).
11. ARM Ltd. mbedTLS Random Number Generation. 2023. Available online: https://mbed-tls.readthedocs.io/en/latest/kb/how-to/add-a-random-generator.html (accessed on 16 February 2026).
12. Ge, Q.; Yarom, Y.; Cock, D.; Heiser, G. A survey of microarchitectural timing attacks and countermeasures on contemporary hardware. *J. Cryptogr. Eng.* **2018**, *8*, 1–27. [CrossRef]
13. Diehl, W.; Farahmand, F.; Yalla, P.; Kaps, J.P.; Gaj, K. Comparison of hardware and software implementations of selected lightweight block ciphers. In Proceedings of the 2017 27th International Conference on Field Programmable Logic and Applications (FPL), Ghent, Belgium, 4–8 September 2017; pp. 1–4. [CrossRef]
14. Bahadur, V.; Selvakumar, D.; Sobha, P.; Bharathi, M.; Rajasekaran, R.; Mohan, S. Reconfigurable side channel attack resistant true random number generator. In Proceedings of the 2016 International Conference on VLSI Systems, Architectures, Technology and Applications (VLSI-SATA), Bengaluru, India, 10–12 January 2016; pp. 1–6.

15. Hoang, V.T.; Shen, Y. Security analysis of nist ctr-drbg. In *Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 16–20 August 2020*; Springer: New York, NY, USA, 2020; pp. 218–247.
16. Baldanzi, L.; Crocetti, L.; Falaschi, F.; Bertolucci, M.; Belli, J.; Fanucci, L.; Saponara, S. Cryptographically secure pseudo-random number generator IP-core based on SHA2 algorithm. *Sensors* **2020**, *20*, 1869. [CrossRef]
17. Chen, S.; Li, B.; Zhou, C. FPGA implementation of SRAM PUFs based cryptographically secure pseudo-random number generator. *Microprocess. Microsystems* **2018**, *59*, 57–68. [CrossRef]
18. Turan, M.S.; McKay, K.; Kang, J.; Kelsey, J.; Chang, D. *NIST Special Publication 800-232: Ascon-Based Lightweight Cryptography Standards for Constrained Devices*; Technical Report; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2025. [CrossRef]
19. Rukhin, A.; Soto, J.; Nechvatal, J.; Smid, M.; Barker, E. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. Technical Report. 2010. Available online: https://nvlpubs.nist.gov/nistpubs/legacy/sp/nistspecialpublication800-22r1a.pdf (accessed on 20 January 2026).
20. Sasikumar, A.; Vairavasundaram, S.; Kotecha, K.; Indragandhi, V.; Ravi, L.; Selvachandran, G.; Abraham, A. Blockchain-based trust mechanism for digital twin empowered Industrial Internet of Things. *Future Gener. Comput. Syst.* **2023**, *141*, 16–27.
21. Davies, M.; Srinivasa, N.; Lin, T.H.; Chinya, G.; Cao, Y.; Choday, S.H.; Dimou, G.; Joshi, P.; Imam, N.; Jain, S.; et al. Loihi: A Neuromorphic Manycore Processor with On-Chip Learning. *IEEE Micro* **2018**, *38*, 82–99. [CrossRef]
22. Misra, S.; Bland, L.C.; Cardwell, S.G.; Incorvia, J.A.C.; James, C.D.; Kent, A.D.; Schuman, C.D.; Smith, J.D.; Aimone, J.B. Probabilistic neural computing with stochastic devices. *Adv. Mater.* **2023**, *35*, 2204569. [CrossRef]
23. Merolla, P.A.; Arthur, J.V.; Alvarez-Icaza, R.; Cassidy, A.S.; Sawada, J.; Akopyan, F.; Jackson, B.L.; Imam, N.; Guo, C.; Nakamura, Y.; et al. A million spiking-neuron integrated circuit with a scalable communication network and interface. *Science* **2014**, *345*, 668–673. [CrossRef]
24. Silva, D.; Shymyrbay, A.; Smagulova, K.; Elsheikh, A.; Fouda, M.E.; Eltawil, A.M. End-to-end edge neuromorphic object detection system. In Proceedings of the 2024 IEEE 6th International Conference on AI Circuits and Systems (AICAS), Abu Dhabi, United Arab Emirates, 22–25 April 2024; pp. 194–198.
25. Kavuri, G.A.; Palfree, J.; Reddy, D.V.; Zhang, Y.; Bienfang, J.C.; Mazurek, M.D.; Shalm, L.K. Traceable Random Numbers from a Non-Local Quantum Advantage. *Nature* **2025**, *642*, 916–921. [CrossRef]
26. Phan, N.T.; Prasad, N.; Hakam, A.; Sidi El Valli, A.; Anghel, L.; Benetti, L.; Madhavan, A.; Jenkins, A.S.; Ferreira, R.; Stiles, M.D.; et al. Unbiased Random Bitstream Generation Using Injection-Locked Spin-Torque Nano-Oscillators. *Phys. Rev. Appl.* **2024**, *21*, 034063. [CrossRef]
27. Barker, E.; Kelsey, J.; McKay, K.; Roginsky, A.; Turan, M.S. *NIST Special Publication 800-90C: Recommendation for Random Bit Generator (RBG) Constructions*; Technical Report; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2024. [CrossRef]
28. O'neill, M.E. PCG: A family of simple fast space-efficient statistically good algorithms for random number generation. *ACM Trans. Math. Softw.* **2014**, *204*, 1–46.
29. Matsumoto, M.; Nishimura, T. Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Trans. Model. Comput. Simul. (TOMACS)* **1998**, *8*, 3–30. [CrossRef]
30. Hayati, N.; Ramli, K.; Windarta, S.; Suryanegara, M. A novel secure root key updating scheme for LoRaWANs based on CTR_AES DRBG 128. *IEEE Access* **2022**, *10*, 18807–18819. [CrossRef]
31. Gladis Kurian, M.; Chen, Y. Ascon on FPGA: Post-Quantum Safe Authenticated Encryption with Replay Protection for IoT. *Electronics* **2025**, *14*, 2668. [CrossRef]
32. Brown, S.D.; Chakma, G.; Adnan, M.M.; Hasan, M.S.; Rose, G.S. Stochasticity in neuromorphic computing: Evaluating randomness for improved performance. In Proceedings of the 2019 26th IEEE International Conference on Electronics, Circuits and Systems (ICECS), Genoa, Italy, 27–29 November 2019; pp. 454–457.
33. Billaudelle, S.; Stradmann, Y.; Schreiber, K.; Cramer, B.; Baumbach, A.; Dold, D.; Göltz, J.; Kungl, A.F.; Wunderlich, T.C.; Hartel, A.; et al. Versatile emulation of spiking neural networks on an accelerated neuromorphic substrate. In Proceedings of the 2020 IEEE International Symposium on Circuits and Systems (ISCAS), Seville, Spain, 12–14 October 2020; pp. 1–5.
34. Mooney, C.Z. *Monte Carlo Simulation*; Number 116; Sage: Thousand Oaks, CA, USA, 1997.
35. Alloun, Y.; Azzaz, M.S.; Kifouche, A. A new approach based on artificial neural networks and chaos for designing deterministic random number generator and its application in image encryption. *Multimed. Tools Appl.* **2025**, *84*, 5825–5860. [CrossRef]
36. Turan, M.S.; Barker, E.; Kelsey, J.; McKay, K.; Baish, M.; Boyle, M. *NIST Special Publication 800-90B: Recommendation for the Entropy Sources Used for Random Bit Generation*; Technical Report; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2018. [CrossRef]
37. Simić, S.D.; Šajina, R.; Tanković, N.; Etinger, D. A Review on Generating Random Numbers in Decentralised Environments. In Proceedings of the 2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO), Opatija, Croatia, 25–29 May 2020; pp. 1668–1673. [CrossRef]

38. Gladis Kurian, M.; Chen, Y. The Untapped Potential of Ascon Hash Functions: Benchmarking, Hardware Profiling, and Application Insights for Secure IoT and Blockchain Systems. *Sensors* **2025**, *25*, 5936. [CrossRef]
39. Dillinger, P.C.; Manolios, P. Bloom filters in probabilistic verification. In *Proceedings of the International Conference on Formal Methods in Computer-Aided Design, Boston, MA, USA, 29 November–2 December 2004*; Springer: New York, NY, USA, 2004; pp. 367–381.
40. Pharr, M.; Jakob, W.; Humphreys, G. *Physically Based Rendering: From Theory to Implementation*; MIT Press: Cambridge, MA, USA, 2023.
41. Mirigaldi, M.; Piscopo, V.; Martina, M.; Masera, G. The Quest for Efficient ASCON Implementations: A Comprehensive Review of Implementation Strategies and Challenges. *Chips* **2025**, *4*, 15. [CrossRef]
42. Kim, Y.B.; Youn, T.Y.; Seo, S.C. Chaining optimization methodology: A new sha-3 implementation on low-end microcontrollers. *Sustainability* **2021**, *13*, 4324. [CrossRef]
43. *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions*; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2015.
44. Nageler, M.; Schmid, L.; Eichlseder, M. Preimage-Type Attacks for Reduced Ascon-Hash. Technical Report, Cryptology ePrint Archive, Report 2025/1259, 2025. Available online: https://eprint.iacr.org/2025/1259 (accessed on 12 February 2026).
45. Crocetti, L.; Di Matteo, S.; Nannipieri, P.; Fanucci, L.; Saponara, S. Design and test of an integrated random number generator with all-digital entropy source. *Entropy* **2022**, *24*, 139. [CrossRef] [PubMed]
46. Bellini, E.; Huang, Y.J. Randomness Testing of the NIST Lightweight Cipher Finalist Candidates. In *Proceedings of the Lightweight Cryptography Workshop, Virtual, 9–11 May 2022*; National Institute of Standards and Technology (NIST): Gaithersburg, MD, USA, 2022.