Opto-Electronics Review

journal homepage: https://journals.pan.pl/opelre

Testing an FPGA-based quantum bit emulator as a random number generator

Aleksander Mazur[1], Wawrzyniec Suleja[2], Jacek Długopolski[3], Marcin Sadowski[2], Marek Życzkowski[4*], Samuel Henry[4], Szymon Fiderkiewicz[4], Piotr Sobotka[1]

[1] Faculty of Physics, Warsaw University of Technology, ul. Koszykowa 75, 00-662 Warsaw, Poland
[2] Sonovero R&D, ul. Słoneczna 7, 72-100 Goleniów, Poland
[3] Faculty of Computer Science, AGH University of Krakow, Al. Adama Mickiewicza 30, 30-059 Kraków, Poland
[4] Institute of Optoelectronics, Military University of Technology, ul. gen. Sylwestra Kaliskiego 2, 00-908 Warsaw, Poland

Article info

Article history:
Received 09 Sep. 2025
Received in revised form 29 Oct. 2025
Accepted 31 Oct. 2025
Available on-line 13 Feb. 2026

Keywords:
FPGA;
qubit emulator;
real-time.

Abstract

This study investigates the capability of a field-programmable gate array (FPGA)-based quantum bit (QUBIT) emulator to replicate the quantum superposition state with randomness evaluation serving as the primary verification method. The QUBIT device is built on the FPGA, an integrated circuit that is a massively parallel array of independent logic elements. Using two QUBIT devices, a simple random number generator is created and statistical tests are used to verify randomness. These tests provide a quantitative measure of randomness quality, reflecting the emulator effectiveness at mimicking quantum superposition. The results offer insights into the limitations and potential of using the emulator (QUBIT) as an alternative to a physical quantum system. Randomness of generated numbers (bit strings) was tested with the NIST Statistical Test Suite (SP 800-22), widely regarded as the standard for randomness evaluation. In total, 10^6 bits were generated and tested with different block lengths: 10 blocks of 10^5 bits and 100 blocks of 10^4 bits. The observed deviations indicate that while the emulator can serve as an educational tool, its statistical properties currently limit its applicability in cryptographic contexts.

1. Introduction

Random numbers are essential in various applications such as cryptography [1–3], simulations [4–6], and statistical physics computations [7]. While pseudo-random number generators rely on deterministic algorithms, true randomness can be obtained by exploiting quantum mechanical phenomena. Quantum computing platforms have already been utilized to generate such randomness [8, 9]. Evaluating the quality of these random numbers is critical and standards such as NIST SP-800-90A [10] provide a suite of statistical tests to assess it. These tests target different patterns of potential non-randomness in binary sequences and are based on three key assumptions: uniformity, scalability, and consistency.

Quantum random number generators (QRNGs) exploit the intrinsic indeterminacy of quantum mechanics to generate statistically verifiable randomness. Unlike classical methods, QRNGs use physical quantum processes – such as state measurement – as their entropy source. This allows them to produce genuinely unpredictable sequences, making them well-suited for tasks that demand strong randomness guarantees [11]. However, the practical limitations of quantum systems, such as scalability and control issues, have motivated the development of devices that emulate quantum state-vector dynamics and their evolution to support reliable, scalable random number generation (RNG).

QRNGs can be further categorised by the level of trust placed in the devices involved, with significant implications for both security and performance [12]. Trusted-device QRNGs assume full reliability of the source and measurement apparatus, allowing for relatively high generation rates but relying heavily on hardware integrity. Semi-device-independent QRNGs, on the other hand, operate under specific assumptions – such as a bounded

*Corresponding author at: marek.zyczkowski@wat.edu.pl

https://doi.org/10.24425/opelre.2026.157819
1896-3757/ Association of Polish Electrical Engineers (SEP) and Polish Academic of Sciences (PAS). Published by PAS

Hilbert space dimension – providing a middle ground where specific attacks can be detected without complete trust in all components.

Device-independent QRNGs offer the strongest security guarantees by relying solely on fundamental quantum principles, such as Bell inequality violations, to certify randomness even in fully untrusted devices. However, these systems are currently limited by technical complexity and low bit generation rates. Selecting an appropriate QRNG type depends on the balance between practical feasibility and the desired level of trust [13]. Previous works on hardware-based QRNGs implemented on photonic systems [14], superconducting qubits [15], or cloud-accessible quantum processors [8, 9] have demonstrated the ability to produce random bit streams that pass the majority of NIST tests, although typically at limited generation rates. Notably, some RNGs implementations exploit field-programmable gate array (FPGA)-specific features, such as jitter, or FPGA primitives as entropy sources – illustrating alternative approaches to generating randomness in hardware devices [16, 17].

In this work, by focusing on an FPGA-based device designed primarily for educational and research purposes, our work highlights the degree to which such emulators can replicate quantum statistical properties, while also exposing their inherent limitations. In this study, an FPGA-based hardware emulator of a two-level quantum system (quantum bit) was tested. The emulator (QUBIT) was tested only from a user perspective, without any knowledge of the device internal architecture. The device internal architecture and usage are described in more detail in [18]. There, a description of the quantum state measurement process can be found. Offered library functions were used in building a QRNG using Hadamard gates and state measurement. Two physical emulators were used simultaneously, which enabled the generation of two random bits per iteration. The QUBIT emulator should mimic the superposition state and the randomness of the measurement result. The degree to which the emulator reproduces this behaviour determines its usefulness as a QRNG. QUBIT primary purpose is not generating true random numbers, but emulating a qubit state, so the process of preparing a qubit state, applying the Hadamard gate, and measuring takes time, which strongly limits the speed of random bit generation. Due to QUBIT internal architecture and operating principles, accelerating random-bit generation is not feasible. In [18], the authors emphasise that the goals of the QUBIT emulator are research and educational applications, such as developing efficient noise-reduction algorithms for quantum systems. The idea of creating a simple RNG using two QUBIT emulators was motivated by this goal, as stressed by the authors.

2. Materials and methods

Both qubits were placed in identical circuits consisting of a Hadamard and a measurement (Fig. 1). Two devices (QUBITs) were used and quantum circuits were coded using library functions as presented in Fig. 2. The experimental setup is shown in Fig. 3. Qubits were initialised in the $|0\rangle$ state. Applying Hadamard gates to both qubits resulted in two single-qubit initial states of the following form:

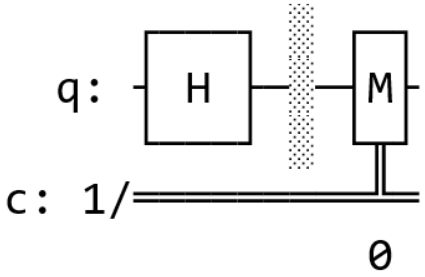

Fig. 1. Single quantum circuit for RNG, using one qubit. Two such circuits were created, and both qubits were prepared in the $|0\rangle$ state. The Hadamard gate was then applied and the quantum state was measured. The result was a two-bit random number every loop iteration. The concatenated bit string was tested using the NIST test suite.

```
for i in range(500000):
    set_device_name("genmet0")
    setup_0_state()
    H()
    m0 = measure()


    set_device_name("genmet1")
    setup_0_state()
    H()
    m1 = measure()

    q_rand_nums.append(m0)
    q_rand_nums.append(m1)
```

Fig. 2. A fragment of a Python code in which two bits are generated in each iteration of the loop.

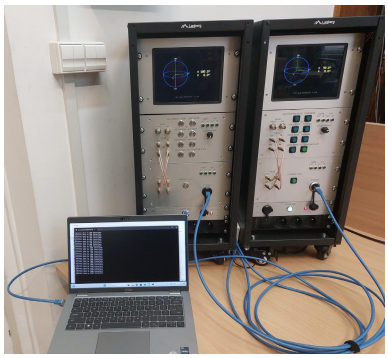

Fig. 3. Experimental setup, two QUBIT devices connected to a computer via an Ethernet interface.

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle). \tag{1}$$

The $|0\rangle$ and $|1\rangle$ states could be measured with the same probability of $\left(\frac{1}{\sqrt{2}}\right)^2 = \frac{1}{2}$, mapping a quantum mechanical projection measurement and giving a theoretically uniform distribution of possible bit register outcomes, both in each single register treated independently (giving two sets of $\{q_0^{(1)}, q_0^{(2)}, \ldots, q_0^{(n)}\}, \{q_1^{(1)}, q_1^{(2)}, \ldots, q_1^{(n)}\}$), as well as in a set made out of pair measurements (a set of the form $\{(q_0^{(1)}, q_1^{(1)}), (q_0^{(2)}, q_1^{(2)}), \ldots, (q_0^{(n)}, q_1^{(n)})\}$, where $q_i^{(j)}$ denotes the result of the *j*-th measurement performed on the *i*-th qubit

and n is the number of performed pairs of single-qubit measurements). These uniform distributions imply maximum entropy of bitstrings obtained by concatenating the single register bitstrings $\left\{q_0^{(1)},q_0^{(2)},\ldots,q_0^{(n)}\right\},\left\{q_1^{(1)},q_1^{(2)},\ldots,q_1^{(n)}\right\}$, as well as the maximum entropy of, for example, a sequence created by XORing the pairs in the pair measurements set and combining them into a bitstring $\left\{q_0^{(1)}\oplus q_1^{(1)},q_0^{(2)}\oplus q_1^{(2)},\ldots,q_0^{(n)}\oplus q_1^{(n)}\right\}$.

Each loop iteration resulted in two generated bits. The bits were combined, and after 500 000 iterations, a sequence of one million bits was obtained. Then, the randomness of the bit string was tested with the NIST test suite with two different block lengths (10^5 and 10^4 bits). Additional measurements were performed with generated bit strings of lengths 2×10^6 and 3×10^6, tested using the same NIST test suite.

The experiments were conducted on two independent QUBIT devices connected to a workstation via Ethernet interface. The emulators were built on commercially available FPGA chips (Xilinx Kintex family) and operated at a clock frequency of 100 MHz. Generating a sequence of 10^6 bits required approximately 35 minutes, reflecting the relatively slow bit rate of the emulator compared to that of physical QRNGs. This limitation is consistent with the QUBIT device design goal, which prioritises faithful emulation of state-vector dynamics over generation speed.

3. Results

The NIST tests were conducted for two cases: 10 blocks of 10^5 bits and 100 blocks of 10^4 bits. In both cases, the test results are comparable, particularly with regard to the tests the bit sequence passed. The tables present the results along with the corresponding p-values and proportions. The null hypothesis is that the bitstring is a random sequence, and the threshold used as a reference point in the NIST tests is p-value > 0.01. All NIST suite tests were performed, but due to the significant imbalance between zeros and ones in the tested bitstring, the number of cycles in the random excursions and random excursions variant tests was too small to meet the criteria for these tests. For this reason, no p-values were reported for random excursions and random excursions variant tests. In Table 1, one can find p-values and proportions for bit strings of length 10^6 and block length 10^5. In Table 2, data are presented for 106-bit strings with a block length of 10^4. A non-overlapping test was performed for two block lengths using the suggested NIST standard criteria ($m = 9$) and yielded 148 results across different templates. Of these, 108 ended with p-value > 0.01 (block length 10^5) and 106 with p-value > 0.01 (block length 10^6). Most p-values were evenly distributed in the range (0.1–0.9), but the occurrence of results with very low p-values (< 0.001) may suggest the presence of strong, local deviations from randomness. The serial test was performed for block length $m = 16$, resulting in two p-values: one checking the frequency of occurrence of blocks of length m and the other checking the frequency of occurrence of blocks one bit shorter (m−1). The cumulative sums test also yields two p-values for the forward and reverse cumulative sum modes.

Figure 4 presents a 1000×1000 bitmap with pixels corresponding to generated bits. Visual inspection of the 1000×1000 binary bitmap reveals regions of local uniformity, with no obvious large-scale periodic patterns. However, subtle clustering of certain bit values can be noticed, which aligns with the statistical evidence of global bias obtained from the frequency test. Such visual artifacts, although not sufficient for rigorous evaluation, provide complementary insight into the presence of non-random structures within the generated sequence.

Table 1.
Results of NIST statistical tests with p-values and proportions for 10 blocks of 10^5 bits.

p-value	Proportion	Statistical test
0.000000 *	5/10 *	Frequency
0.000000 *	4/10 *	BlockFrequency
0.000000 *	5/10 *	CumulativeSums(Forward)
0.035174	4/10 *	CumulativeSums(Reverse)
0.066882	10/10	Runs
0.000000 *	3/10 *	LongestRun
0.213309	10/10	Rank
0.911413	10/10	FFT
0.000000 *	0/10 *	Universal
0.000000 *	0/10 *	ApproximateEntropy
0.000001 *	6/10 *	Serial (m)
0.534146	10/10	Serial (m−1)
0.350485	9/10	LinearComplexity
0.000000 *	5/10 *	OverlappingTemplate

(*) the test conditions for which the blocks did not pass the NIST test, i.e., the required p-value or proportion criteria were not satisfied.

Table 2.
Results of statistical tests with p-values and proportions for 100 blocks of 10^4 bits.

p-value	Proportion	Statistical test
0.003996 *	94/100 *	Frequency
0.000000 *	91/100 *	BlockFrequency
0.000017 *	93/100 *	CumulativeSums(Forward)
0.035174 *	94/100 *	CumulativeSums(Reverse)
0.191687	98/100	Runs
0.000105 *	93/100 *	LongestRun
0.000040 *	99/100	Rank
0.883171	99/100	FFT
0.000000 *	100/100	Universal
0.000000 *	23/100 *	ApproximateEntropy
0.262249	97/100	Serial (m)
0.181557	99/100	Serial (m−1)
0.719747	97/100	LinearComplexity
0.236810	96/100	OverlappingTemplate

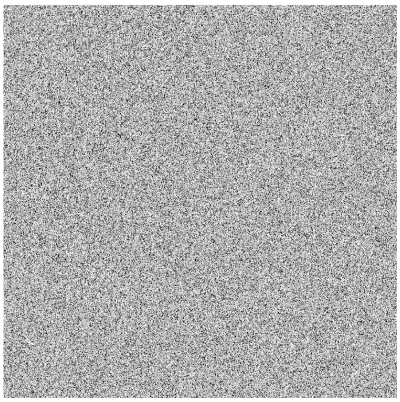

Fig. 4. Visualisation of a 1 000 000-bit sequence represented as a 1000 × 1000 binary bitmap, generated using a QUBIT emulator. Each pixel corresponds to a single bit: white indicates a logical 0 and black indicates a logical 1.

4. Conclusions

A sequence of 10^6 bits was generated using two QUBIT emulators and evaluated with the NIST Statistical Test Suite. The sequence was divided into two configurations: 100 blocks of 10^4 bits and 10 blocks of 10^5 bits. In both cases, the sequences did not meet the full set of NIST test requirements across most tests. These outcomes suggest the presence of non-ideal statistical features that may arise from subtle implementation-specific effects in the quantum-state measurement procedures used in the QUBIT emulator.

For example, a closer examination of the frequency test reveals that it failed with an extremely low *p*-value in the 10-block case. With 100 smaller blocks, even though the test still failed, we achieved a significantly better result. This observation suggests that the generator exhibits global statistical bias. In shorter blocks, natural statistical fluctuations are relatively large, which makes minor systematic deviations from uniformity difficult to detect. In contrast, longer blocks reduce the relative impact of such fluctuations, thereby revealing the generator underlying global bias.

A closer examination of the frequency test algorithm reveals potential sources of the generator global bias. Each bit in the NIST test suite is represented as $X_i = 2 - 1 = \pm 1$, where ε_i is the actual value of the *i*-th bit and the sum over a block of *n* bits is defined as $S_n = \sum_{i=1}^{n} X_i$. The expected value is $E[S_n] = 0$ and the standard deviation is $\sigma_{S_n} = \sqrt{n}$. Normalising to the proportion of one's $p = \frac{S_n}{n}$, gives $\sigma_p = \frac{\sqrt{n}}{n} = \frac{1}{\sqrt{n}}$. This shows that for larger blocks, the natural statistical fluctuations σ_p decrease as $\frac{1}{\sqrt{n}}$, while any small systematic bias remains constant. Therefore, the same underlying bias that is hidden by fluctuations in short blocks becomes statistically significant in longer blocks, resulting in much lower *p*-values. This indicates that the QUBIT emulator passes local uniformity checks but fails to maintain this property over long sequences.

It is also instructive to consider the approximate entropy test, which the emulator failed in both the 10-block and 100-block cases. In contrast to the frequency test, which reveals global imbalance between zeros and ones, the approximate entropy test provides insight into the diversity of local patterns [10, 19]. It evaluates how the block-level entropy of the sequence changes as it moves from m-bit to (m + 1)-bit overlapping patterns. For an ideal random sequence, all *m*-bit and $(m + 1)$-bit patterns occur with probabilities consistent with uniform randomness, resulting in a predictable increase of entropy. Any systematic preference for certain patterns or correlations between successive bits reduces the observed approximate entropy relative to the expected value.

The observation that the emulator performs worse on this test suggests that the bias is not limited to global imbalance but also involves structural dependencies in the bitstream. These dependencies may arise from architectural features of the FPGA, such as routing asymmetries, coupling between logic elements, or deterministic components of the noise process that influence the frequency of repeating local patterns.

When compared with previously reported results for hardware-based QRNGs, such as those implemented on IBM cloud quantum processors, clear differences emerge. In that study, the generated bit streams successfully passed the majority of the NIST test suite, with failures limited to entropy-related tests. In contrast, our emulator-based generator failed a larger fraction of the tests, most notably those sensitive to global statistical bias. This comparison highlights the fundamental distinction between physical QRNGs, where randomness originates from genuine quantum measurement outcomes, and FPGA-based emulators, where residual architectural asymmetries play a decisive role.

Randomness in the QUBIT emulator originates from physical noise sources inherent to the FPGA fabric. The system can be viewed as deterministic chaos modulated by physical noise, rather than a purely stochastic process, therefore making the bias observed in the frequency test as likely rooted in architectural features of the emulator, reflecting systematic asymmetries in how measurement outcomes are sampled. Future work should therefore aim to pinpoint these architectural sources of bias and develop countermeasures to reduce their effects.

The presented results must be interpreted with certain limitations in mind. First, only two physical emulators were tested, limiting the generalisability of the findings. Second, the QUBIT device internal architecture is not fully transparent to the user, hindering precise identification of the hardware-level mechanisms responsible for statistical bias. Third, the evaluation was limited to sequences of up to 3×10^6 bits, which, while sufficient for initial testing, remains small compared to typical datasets used in benchmarking physical QRNGs.

Despite these limitations, the study underscores the value of FPGA-based emulators as accessible platforms for research and education in quantum information science. Their ability to mimic quantum state preparation and measurement makes them suitable for testing algorithms, teaching concepts of quantum randomness, and developing post-processing strategies such as randomness extraction. However, their current statistical shortcomings render them unsuitable for direct use in cryptographic or other high-security applications.

The results point to potential directions for further optimising the emulation and measurement processes. Improvements in these areas may enhance the statistical properties of the output sequences, aligning them more closely with the criteria for truly random bit streams set by the NIST.

Future work should aim to integrate post-processing techniques, such as Toeplitz hashing or Trevisan extractors, to mitigate statistical bias and increase the effective entropy of generated sequences. Longer experimental runs will also allow for a more robust characterisation of the emulator statistical behaviour. Finally, a direct comparison with physical QRNGs operating under similar conditions could provide further insights into the strengths and limitations of FPGA-based emulation for randomness generation. In addition to hardware-level improvements, practical approaches to mitigating statistical bias can be pursued through software-based post-processing. Widely used randomness extractors, such as von Neumann correctors, Toeplitz hashing, and Trevisan extractors, are already available as open-source implementations in environments such as Python and MATLAB. Systematic benchmarking of these extractors on bit streams generated by the QUBIT emulator could provide an efficient way to increase effective entropy and bring the output closer to the requirements for cryptographic-grade randomness.

References

[1] Kelsey, J., Schneier, B., Wagner, D. & Hall, C. Cryptanalytic Attacks on Pseudorandom Number Generators. in *Fast Software Encryption* (ed. Vaudenay. S.) **1372**, 168–188 (Springer Berlin Heidelberg, 1998). https://doi.org/10.1007/3-540-69710-1_12

[2] Ferguson, N., Schneier, B. & Kohno, T. *Cryptography Engineering: Design Principles and Practical Applications*. (John Wiley & Sons, Chichester, 2010). https://doi.org/10.1002/9781118722367

[3] Shannon, C. E. Communication theory of secrecy systems. *Bell Syst. Tech. J.* **28**, 656–715 (1949). https://doi.org/10.1002/j.1538-7305.1949.tb00928.x

[4] L'Ecuyer P. Random numbers for simulation. *Commun. ACM* **33**, 85–97 (1990). https://doi.org/10.1145/84537.84555

[5] Landau, D. P. & Binder, K. *A Guide to Monte Carlo Simulations in Statistical Physics*. *Third Edition*. (Cambridge University Press, 2011). https://doi.org/10.1017/cbo9780511994944

[6] Metropolis, N. & Ulam, S. The Monte Carlo method. *J. Am. Stat. Assoc.* **44**, 335–341 (1949). https://doi.org/10.1080/01621459.1949.10483310

[7] Binder, K. & Heermann, D. W. *Monte Carlo Simulation in Statistical Physics: An Introduction*. *Sixth Edition*. (Springer Nature, 2019). https://doi.org/10.1007/978-3-030-10758-1

[8] Kumar, V., Rayappan, J. B. B., Amirtharajan, R. & Praveenkumar, P. Quantum true random number generation on IBM's cloud platform. *J. King Saud Univ. Comput. Inf. Sci.* **34**, 6453–6465 (2022). https://doi.org/10.1016/j.jksuci.2022.01.015

[9] Karthick, V., Haritha, S. & Janani, K. True Random Number Generation on IBM Real-Time Quantum Computer for Secure and Unpredictable Cryptographic Applications. in *2024 Second International Conference on Advances in Information Technology (ICAIT)* **1**, 1–6 (IEEE, 2024). https://doi.org/10.1109/icait61638.2024.10690780

[10] Rukhin, A. *et al*. A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. *US Department of Commerce. NIST Special Publication 800-22 Revision 1a* https://doi.org/10.6028/NIST.SP.800-22r1a (2010).

[11] Eastlake, D. E., Crocker, S. & Schiller, J. I. Randomness Requirements for Security. Preprint at https://doi.org/10.17487/RFC4086

[12] Mannalatha, V., Mishra, S. & Pathak, A. A comprehensive review of quantum random number generators: concepts, classification and the origin of randomness. *Quantum Inf. Process.* **22**, 439 (2023). https://doi.org/10.1007/s11128-023-04175-y

[13] Herrero-Collantes, M. & Garcia-Escartin, J. C. Quantum random number generators. *Rev. Mod. Phys.* **89**, 015004 (2017). https://doi.org/10.1103/RevModPhys.89.015004

[14] Gabriel, C. *et al.* A generator for unique quantum random numbers based on vacuum states. *Nat. Photon.* **4**, 711–715 (2010). https://doi.org/10.1038/nphoton.2010.197

[15] Tamura, K. & Shikano, Y. Quantum Random Number Generation with the Superconducting Quantum Computer IBM 20Q Tokyo. in *Proceedings of Workshop on Quantum Computing and Quantum Information (QCQI)* (eds. Hirvensalo, M. & Yakaryilmaz, A.) 1–13 (Cryptology ePrint Archive, 2019). https://eprint.iacr.org/2020/078.pdf

[16] Petrica, L. FPGA optimized cellular automaton random number generator. *J. Parallel Distrib. Comput.* **111**, 251–259 (2018). https://doi.org/10.1016/j.jpdc.2017.05.022

[17] Di Patrizio Stanchieri, G., De Marcellis, A., Palange, E. & Faccio, M. A true random number generator architecture based on a reduced number of FPGA primitives. *AEU-Int. J. Electron. Commun.* **105**, 15–23 (2019). https://doi.org/10.1016/j.aeue.2019.03.006

[18] Długopolski, J., Sadowski, M. & Suleja, W. A physical model of quantum bit behavior based on a programmable FPGA integrated circuit. *Comput. Sci.* **25**, (2024). https://doi.org/10.7494/csci.2024.25.4.6289

[19] Good, I. J. The serial test for sampling numbers and other tests for randomness. *Math. Proc. Camb. Philos. Soc.* **49**, 276–284 (1953). https://doi.org/10.1017/S030500410002836X