Cryptographically Secure Random Number Generator utilising Environmental Radiation

Pan Ziyue  
Engineering Product Development  
Singapore University of Technology & Design  
ziyue\_pan@mymail.sutd.edu.sg  
  
Chua Kevin Subong  
Engineering Product Development  
Singapore University of Technology & Design  
kevin\_chua@mymail.sutd.edu.sg  
Ng Au Hern Wesley  
Engineering Product Development  
Singapore University of Technology & Design  
wesley\_ng@mymail.sutd.edu.sg  
Kaung Khant Htet  
Engineering Product Developmen  
Singapore University of Technology & Design  
khanthtet\_kaung@mymail.sutd.edu.sg  
   
Shum Hei Lam  
Engineering Product Development  
Singapore University of Technology & Design  
heilam\_shum@mymail.sutd.edu.sg

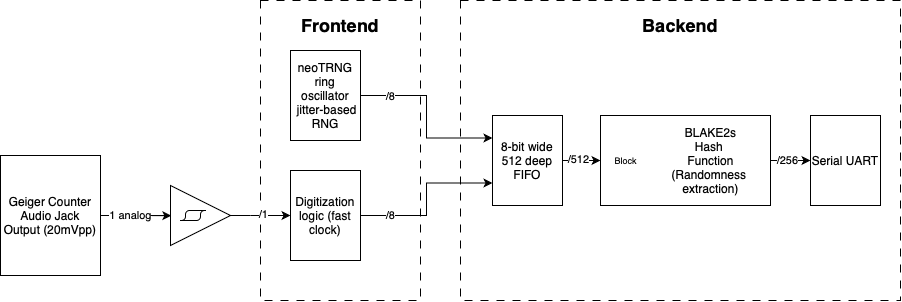


Diagram of our proposed design

*Abstract* — Generating random numbers that are cryptographically secure can be expensive with dedicated hardware. This paper describes the process of creating a cost-effective Cryptographically Secure Random Number Generator on an FPGA coupled to a Geiger counter. The frontend consists of a Geiger counter system that outputs pulses in response to environmental radioactivity, as well as a neoTRNG-based ring oscillator RNG. The backend uses a BLAKE2s hash function to remove any potential biasness created by the frontend. The paper will provide an insight into the conceptualization and implementation processes when creating the CSRNG, as well as highlighting challenges faced.

# Introduction

From computing to cryptography, random numbers are essential for various algorithms used in our computer systems. However, computers, being deterministic machines, can never generate truly random numbers on their own. Instead, they rely on Pseudo-random Number Generators (PRNGs), algorithms designed to produce sequences of numbers that each simulate the properties of a random number.

PRNGs use a seed value to generate a sequence of numbers through mathematical formulas or state machines which can appear randomly but are entirely determined by the seed value. However, this makes PRNGs highly predictable, which can make them unviable for cryptographic uses as they may be vulnerable to cyber-attacks. As such, creating a True Random Number Generator (TRNG) is key to ensuring information security.

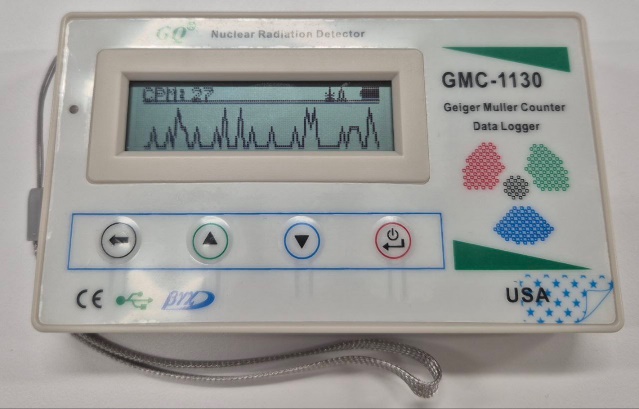
TRNGs, unlike PRNGs, derive their randomness directly from entropy sources, which are inherently unpredictable processes, such as electronic noise and radioactive decay. By seeding PRNGs through such unpredictable phenomena, non-deterministic outputs can be output, making them highly secure for generating encryption keys and truly random numbers for cryptographic algorithms. The paper focuses on an implementation of such an RNG that utilizes radioactive decay detected in the environment, as well as ring oscillator jitter noise as input data to be used as a cryptographically secure random number generator (CSRNG).

# Background and Motivation

Ensuring true randomness is especially critical in cryptography, where deterministic patterns may compromise key generation and encryption schemes. Recent works have demonstrated the viability of hybrid RNG systems utilizing physical entropy sources. Inspired by radiation-based RNG systems such as the one in Bonacini's implementation [1], and supplemented by techniques described in quantum RNG literature [2], this project aims to combine high quality physical entropy with cryptographically secure post-processing, limiting the possibilities of side-channel attacks.

# Frontend Circuitry

## Geiger Counter



1. Geiger counter GQ GMC-1130 used as radiation detection

To detect and retrieve information regarding radioactive decay in the environment, a Geiger counter was used. Geiger counter model GQ GMC-1130 was used to accurately retrieve this information. The counter has a 3.5mm analog data output port and a USB port, where the former is used to retrieve analog signals used as a data input.

## Radioactive Source



1. Americium-241 ion chamber used as radioactive source

A smoke detector ion chamber containing Americium-241 was originally to be used as the radioactive source. However, due to the shielding around the Americium for safety reasons, there is a negligible change in readings on the counter. Throughout this paper, the Americium was not removed from this shielding, to minimize harm towards people in the environment. As such, background radioactivity is used for detection instead. Future implementations could include a properly designed module which hermetically seals the Americium while allowing alpha particles to pass through (e.g. mica window) to the Geiger counter.

## Schmitt trigger for Geiger counter data input

The Geiger counter outputs a pulse signal which must be converted to a digital signal to be read by the Field-Programmable Gate Array (FPGA). To do this, a circuit consisting of resistors and op amps was used, forming a Schmitt Trigger that converts the pulse signal of the Geiger counter into square waves that adhere to the FPGA’s static discipline.

A screenshot of a computer

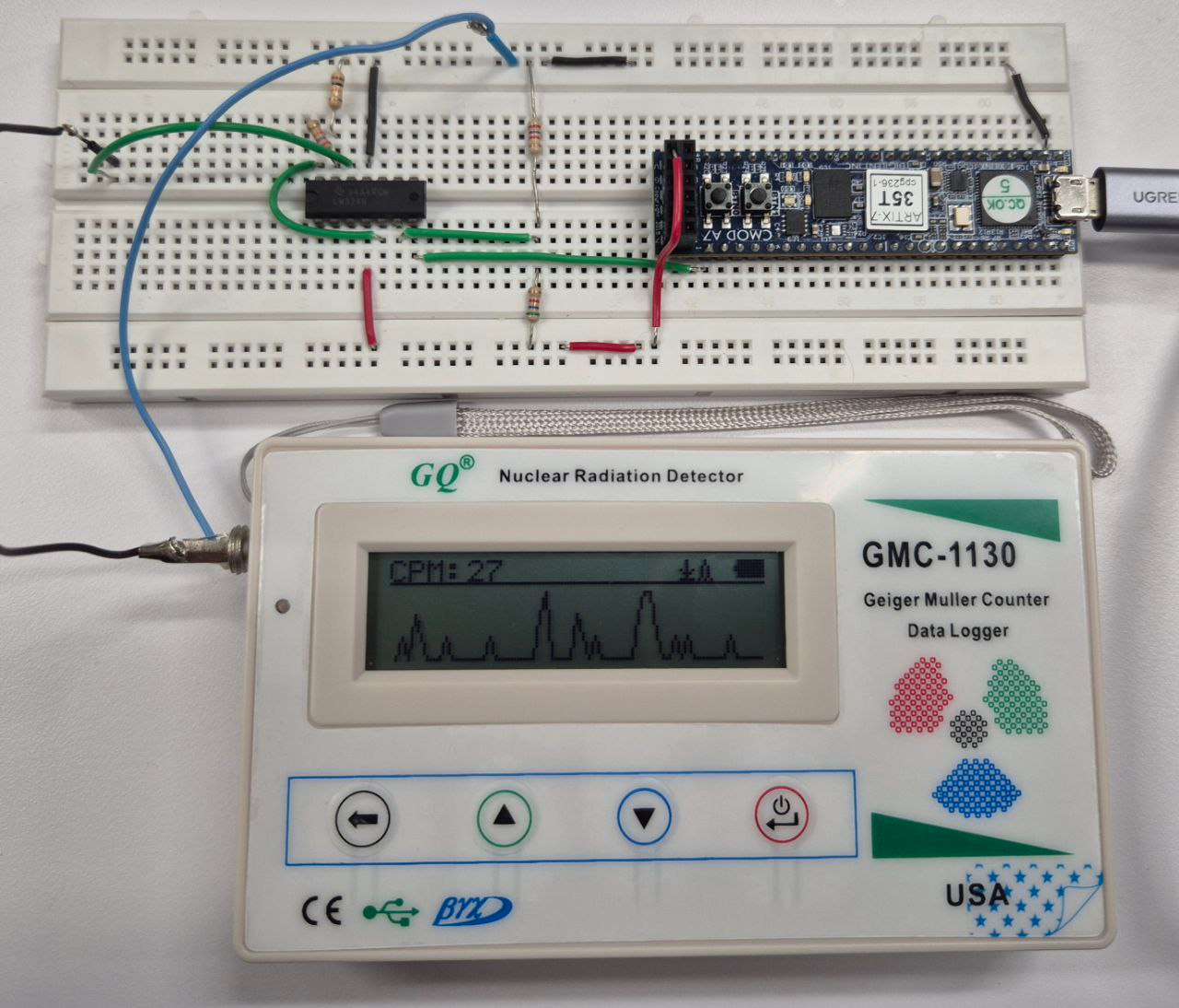
AI-generated content may be incorrect.

1. Analog output direct from Geiger counter

A diagram of a circuit

AI-generated content may be incorrect.

1. Non-Inverting Gain and Comparator amplifier circuitry



1. Physical circuitry with Geiger counter and FPGA

After the Schmitt Trigger, the signal adheres outputs a low value of 0, and a high value of 2.26V, which adheres to the and of the FPGA I/O pins. is the voltage at which an input voltage above this value is recognized as high, while is the voltage at which an input voltage below this value is recognised as low.

## Digitalization through fast clock method

To digitize the signal, the fast clock method is used. This approach converts the temporal spacing between radiation events into 8-bit numerical values

A diagram of numbers and lines

AI-generated content may be incorrect.

1. Fast clock method: A fast clock (down) is used to increase a counter. Whenever a detection is made (up), the counter is read and reset, generating one random number. Image retrieved from [1].

This method was chosen as opposed to the slow clock method and time difference method due to the low number of counts generated by environmental radioactivity. The fast clock method will be able to generate a higher deviation of numbers that are non-zero compared to the slow clock method, and the generation of bits will also be faster compared to the time difference method.

A screenshot of a video game

AI-generated content may be incorrect.

1. Testbench of fastclock module in Vivado. Gpio reflectes incoming rising edges from the Geiger counter circuit.

## Additional Entropy Source

To increase available entropy of the seeding system, a ring oscillator jitter-based RNG, based off the neoTRNG module, is implemented. A ring oscillator is a chain of inverters connected in a loop, forming a feedback circuit that toggles rapidly due to propagation delays. By extracting timing jitter between toggles due to electronic shot noise in the circuit, a source of entropy is introduced.

# Backend Circuitry

## FIFO Buffer

The 8-bit FIFO buffer mixes and caches the 2 incoming sources of entropy, from both the Geiger counter and the neoTRNG, and releases the data to the randomness extractor module once it has stored 32 bytes (512 bits). This ensures a fair mix of incoming data and steady data input into the next stage.

## Randomness Extractor

Given the source of radiation, the environment, is constant, there is a high likelihood that the data retrieved is biased towards a certain range of numbers. To remove the potential bias, a randomness extractor is used, in the form of a BLAKE2s hash function, like how Linux kernel RNG uses BLAKE2s as its entropy extractor [3]. The BLAKE2s hash function mixes and scrambles the input bits in such a way that the output doesn’t reflect the pattern. Additionally, this ensures that the TRNG system is preimage resistant and collision resistant, generating outputs that are uniformly distributed and free of observable bias.

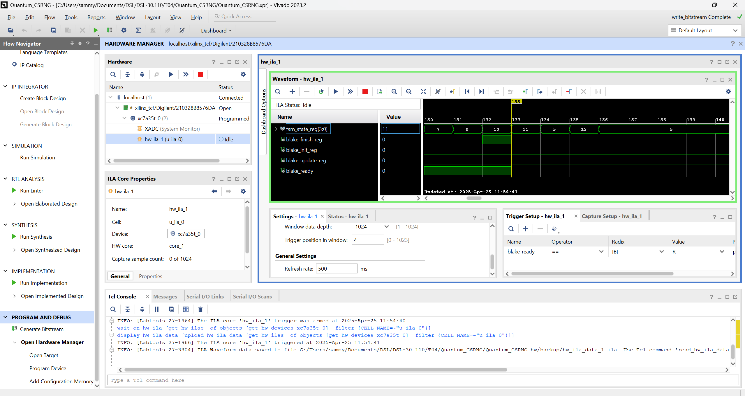
## Serial UART Interface

The PRNG output is transmitted over a Universal Asynchronous Receiver-Transmitter (UART) interface for external use. This module is equipped with its own FIFO to accommodate bursty data transmission. The output is then logged and stored in a text file.

# Validation and Tesing

## Debugging

Debugging is conducted through the Xilinx Integrated Logic Analyzer (ILA), which greatly reduced debugging time and allowed us to identify issues rapidly.



1. Debugging with Vivado Integrated Logic Analyzer

## Quality Testing

The NIST Statistical Test Suite (based on the NIST Special Publication 800-22 specification) [5] was used to evaluate whether the outputs from the number generator is suitable for cryptographic grade randomness. The test suite includes tests such as Frequency Test, Approximate Entropy Test, Maurer’s “Universal Statistical” Test.

The input parameters for the test are:

* Length of tested bitstream: 1,600,000 bits
* Number of bitstreams: 50
* Tested binary file size: 1,600,000 bits×50=10MByte

The results obtained were all above the recommended threshold of 47/50 and 33/36 for statistical tests and random excursion tests respectively. For the full results, please refer to Appendix A.

# Conclusion

In conclusion, this project presents a practical and cost-effective approach to generating cryptographically secure random numbers by leveraging both environmental radiation and electronic jitter as entropy sources. Through the integration of a Geiger counter and a ring oscillator-based neoTRNG, the system produces high-entropy, bias-free random outputs suitable for cryptographic applications.

The implementation on an FPGA not only ensures hardware-level security but also demonstrates the feasibility of building secure random number generators with accessible components. Validation through the NIST statistical test suite further confirms the reliability and effectiveness of the design.

##### Acknowledgment

We would like to express our heartfelt gratitude to all those who have guided and supported us throughout the course of this project.

First and foremost, we extend our deepest appreciation to Dr. Teo Tee Hui, our mentor from SUTD, for his invaluable insights, guidance, and encouragement at every stage of this project. His expertise and constructive feedback have been instrumental in shaping our work.

##### References

1. G. Bonacini, *Nuclear RNG Generator*, GitHub Repository: <https://github.com/gbonacini/nuclear_rng_generation2>
2. Herrero-Collantes, M., & Garcia-Escartin, J. C. (2017). Quantum random number generators. *Reviews of Modern Physics*, *89*(1). https://doi.org/10.1103/revmodphys.89.015004
3. Saarinen, M. O., & Aumasson, J. (2015). The BLAKE2 Cryptographic Hash and Message Authentication Code (MAC). *RFC*, *7693*, 1–30. https://doi.org/10.17487/RFC7693
4. Nir, Y., & Langley, A. (2015). *CHACHA20 and Poly1305 for IETF protocols*. <https://doi.org/10.17487/rfc7539>
5. Rukhin, A., Sota, J., Nechvatal, J., Smid, M., Barker, E., Leigh, S., Levenson, M., Vangel, M., Banks, D., Heckert, A., Dray, J., & Vo, S. (2000). *A statistical test suite for random and pseudorandom number generators for cryptographic applications*. https://doi.org/10.6028/nist.sp.800-22

##### **Appendix A**

##### NIST Test Suite Run Results

------------------------------------------------------------------------------

RESULTS FOR THE UNIFORMITY OF P-VALUES AND THE PROPORTION OF PASSING SEQUENCES

------------------------------------------------------------------------------

generator is <serial\_dump.bin>

------------------------------------------------------------------------------

C1 C2 C3 C4 C5 C6 C7 C8 C9 C10 P-VALUE PROPORTION STATISTICAL TEST

------------------------------------------------------------------------------

2 5 5 7 3 5 4 7 8 4 0.699313 50/50 Frequency

3 6 5 6 3 2 5 8 6 6 0.739918 50/50 BlockFrequency

5 1 7 5 5 5 6 7 2 7 0.574903 49/50 CumulativeSums

3 4 7 8 7 5 2 6 5 3 0.616305 50/50 CumulativeSums

2 7 3 2 5 6 7 7 4 7 0.534146 50/50 Runs

5 5 4 4 3 4 7 8 9 1 0.319084 48/50 LongestRun

4 11 6 6 5 0 5 4 5 4 0.153763 49/50 Rank

7 8 10 3 6 2 3 3 5 3 0.171867 48/50 FFT

7 5 2 6 7 1 3 7 7 5 0.419021 50/50 NonOverlappingTemplate

4 4 3 6 4 6 6 5 9 3 0.739918 50/50 NonOverlappingTemplate

5 7 10 2 3 3 2 8 5 5 0.171867 49/50 NonOverlappingTemplate

3 6 8 3 5 7 4 3 4 7 0.699313 50/50 NonOverlappingTemplate

11 4 3 5 7 5 3 7 0 5 0.075719 50/50 NonOverlappingTemplate

10 4 5 4 7 3 3 5 4 5 0.534146 49/50 NonOverlappingTemplate

4 7 1 2 11 3 6 7 5 4 0.085587 49/50 NonOverlappingTemplate

11 5 2 3 2 6 2 7 7 5 0.085587 48/50 NonOverlappingTemplate

5 5 6 3 2 4 6 5 9 5 0.699313 50/50 NonOverlappingTemplate

6 3 5 8 6 8 3 3 5 3 0.616305 50/50 NonOverlappingTemplate

5 6 3 5 6 4 10 3 4 4 0.574903 50/50 NonOverlappingTemplate

8 4 11 5 2 6 5 3 2 4 0.122325 49/50 NonOverlappingTemplate

7 4 4 3 2 6 5 2 10 7 0.236810 49/50 NonOverlappingTemplate

5 6 6 4 8 3 9 2 3 4 0.419021 50/50 NonOverlappingTemplate

5 8 6 4 5 3 7 3 8 1 0.383827 49/50 NonOverlappingTemplate

6 5 8 2 3 4 6 7 5 4 0.739918 49/50 NonOverlappingTemplate

4 3 3 5 10 2 7 6 5 5 0.383827 50/50 NonOverlappingTemplate

5 7 4 1 3 4 6 9 3 8 0.262249 50/50 NonOverlappingTemplate

4 4 8 6 6 2 4 5 4 7 0.779188 50/50 NonOverlappingTemplate

4 8 1 4 5 10 5 5 3 5 0.262249 49/50 NonOverlappingTemplate

7 3 4 10 3 4 7 1 4 7 0.171867 49/50 NonOverlappingTemplate

5 5 5 5 4 4 4 4 9 5 0.911413 50/50 NonOverlappingTemplate

5 5 4 6 7 3 6 7 2 5 0.851383 48/50 NonOverlappingTemplate

6 6 2 5 4 6 6 5 4 6 0.955835 50/50 NonOverlappingTemplate

5 8 3 5 3 6 6 5 6 3 0.851383 49/50 NonOverlappingTemplate

6 5 3 3 5 8 5 5 3 7 0.816537 50/50 NonOverlappingTemplate

2 6 7 6 3 5 5 4 7 5 0.851383 50/50 NonOverlappingTemplate

5 4 3 8 3 2 3 8 6 8 0.350485 50/50 NonOverlappingTemplate

4 3 8 4 7 2 6 3 7 6 0.574903 50/50 NonOverlappingTemplate

7 5 4 5 5 6 4 5 5 4 0.996335 49/50 NonOverlappingTemplate

7 6 5 3 5 5 3 5 3 8 0.816537 49/50 NonOverlappingTemplate

8 1 4 9 4 6 7 6 1 4 0.153763 50/50 NonOverlappingTemplate

2 7 2 4 6 5 6 6 7 5 0.739918 50/50 NonOverlappingTemplate

6 9 5 6 6 5 4 1 5 3 0.534146 50/50 NonOverlappingTemplate

10 5 5 4 4 5 4 5 5 3 0.699313 50/50 NonOverlappingTemplate

6 7 3 6 0 6 4 3 6 9 0.236810 49/50 NonOverlappingTemplate

2 3 6 6 6 1 8 8 7 3 0.236810 50/50 NonOverlappingTemplate

6 9 6 2 5 3 6 3 6 4 0.574903 50/50 NonOverlappingTemplate

2 8 4 3 3 7 6 4 4 9 0.350485 50/50 NonOverlappingTemplate

4 2 3 10 3 7 2 8 5 6 0.153763 49/50 NonOverlappingTemplate

4 2 4 6 6 4 10 5 4 5 0.534146 50/50 NonOverlappingTemplate

6 2 8 6 7 6 1 3 5 6 0.419021 48/50 NonOverlappingTemplate

4 4 4 5 7 6 5 4 4 7 0.971699 50/50 NonOverlappingTemplate

4 8 5 5 4 7 1 11 3 2 0.066882 49/50 NonOverlappingTemplate

5 3 7 5 7 3 5 5 6 4 0.935716 50/50 NonOverlappingTemplate

6 7 5 7 4 4 6 1 6 4 0.739918 50/50 NonOverlappingTemplate

6 5 3 4 6 4 3 8 5 6 0.883171 50/50 NonOverlappingTemplate

8 3 1 5 5 6 7 5 4 6 0.616305 49/50 NonOverlappingTemplate

7 2 4 6 8 2 6 4 8 3 0.383827 50/50 NonOverlappingTemplate

3 4 11 2 3 3 9 5 7 3 0.058984 50/50 NonOverlappingTemplate

3 1 5 5 6 5 9 10 3 3 0.122325 50/50 NonOverlappingTemplate

8 7 3 8 3 6 3 1 5 6 0.319084 48/50 NonOverlappingTemplate

7 7 6 5 2 4 6 2 8 3 0.494392 49/50 NonOverlappingTemplate

7 2 7 7 7 1 4 6 7 2 0.262249 48/50 NonOverlappingTemplate

8 8 6 6 3 3 5 2 5 4 0.574903 49/50 NonOverlappingTemplate

5 5 6 4 2 6 0 10 9 3 0.058984 49/50 NonOverlappingTemplate

5 2 3 4 6 6 9 4 6 5 0.657933 49/50 NonOverlappingTemplate

6 2 7 3 8 7 5 6 2 4 0.494392 50/50 NonOverlappingTemplate

5 4 7 2 5 6 4 5 8 4 0.816537 49/50 NonOverlappingTemplate

4 11 7 7 2 4 1 3 5 6 0.085587 50/50 NonOverlappingTemplate

6 5 6 2 3 7 8 6 5 2 0.574903 49/50 NonOverlappingTemplate

2 0 6 7 4 5 5 6 7 8 0.289667 50/50 NonOverlappingTemplate

9 6 2 4 6 3 2 8 5 5 0.350485 49/50 NonOverlappingTemplate

5 6 5 5 8 1 8 4 6 2 0.419021 50/50 NonOverlappingTemplate

3 5 7 4 5 4 6 7 6 3 0.911413 50/50 NonOverlappingTemplate

8 1 3 7 6 4 2 7 7 5 0.319084 49/50 NonOverlappingTemplate

7 9 5 3 4 7 3 2 6 4 0.455937 50/50 NonOverlappingTemplate

9 5 2 5 2 4 3 6 3 11 0.066882 50/50 NonOverlappingTemplate

3 8 6 5 5 4 5 7 5 2 0.779188 48/50 NonOverlappingTemplate

5 2 1 4 4 10 6 4 4 10 0.066882 50/50 NonOverlappingTemplate

4 8 2 6 6 2 5 9 2 6 0.262249 49/50 NonOverlappingTemplate

7 1 2 8 4 6 8 2 4 8 0.137282 50/50 NonOverlappingTemplate

5 7 2 3 6 4 3 6 8 6 0.657933 49/50 NonOverlappingTemplate

6 4 9 4 2 2 4 7 4 8 0.319084 50/50 NonOverlappingTemplate

7 5 2 6 7 1 3 7 7 5 0.419021 50/50 NonOverlappingTemplate

8 4 5 8 5 4 4 2 5 5 0.739918 49/50 NonOverlappingTemplate

7 6 5 8 5 2 3 5 5 4 0.779188 50/50 NonOverlappingTemplate

3 3 3 6 5 6 6 2 6 10 0.350485 50/50 NonOverlappingTemplate

4 3 5 6 9 5 2 4 6 6 0.657933 50/50 NonOverlappingTemplate

5 1 5 4 7 3 6 8 6 5 0.616305 50/50 NonOverlappingTemplate

2 6 4 2 2 7 9 5 4 9 0.153763 50/50 NonOverlappingTemplate

5 4 7 7 5 1 7 5 6 3 0.657933 50/50 NonOverlappingTemplate

6 4 5 2 7 7 5 6 4 4 0.883171 49/50 NonOverlappingTemplate

7 11 3 3 4 9 1 6 2 4 0.030806 50/50 NonOverlappingTemplate

9 4 2 4 8 4 7 2 7 3 0.236810 50/50 NonOverlappingTemplate

10 2 6 2 6 1 5 6 8 4 0.108791 50/50 NonOverlappingTemplate

12 2 6 6 6 6 5 2 3 2 0.051942 49/50 NonOverlappingTemplate

8 5 4 2 8 3 5 8 4 3 0.419021 50/50 NonOverlappingTemplate

6 5 2 9 4 3 8 3 6 4 0.419021 49/50 NonOverlappingTemplate

8 3 3 3 4 3 7 6 9 4 0.383827 49/50 NonOverlappingTemplate

5 8 4 4 4 6 3 8 3 5 0.739918 49/50 NonOverlappingTemplate

3 4 5 4 6 9 3 5 8 3 0.534146 50/50 NonOverlappingTemplate

2 6 7 8 5 6 3 7 5 1 0.383827 50/50 NonOverlappingTemplate

6 5 5 3 7 7 4 5 3 5 0.935716 50/50 NonOverlappingTemplate

8 4 7 1 5 2 6 7 3 7 0.319084 48/50 NonOverlappingTemplate

3 6 2 5 4 7 6 7 4 6 0.816537 49/50 NonOverlappingTemplate

2 8 6 6 5 3 6 5 5 4 0.816537 50/50 NonOverlappingTemplate

8 9 4 4 7 1 5 3 5 4 0.319084 49/50 NonOverlappingTemplate

6 6 4 2 3 5 5 5 8 6 0.816537 49/50 NonOverlappingTemplate

6 7 5 5 3 5 5 2 5 7 0.883171 50/50 NonOverlappingTemplate

4 7 7 2 3 5 6 5 6 5 0.851383 50/50 NonOverlappingTemplate

6 3 8 2 7 6 4 4 6 4 0.699313 50/50 NonOverlappingTemplate

6 4 5 6 6 2 4 2 7 8 0.616305 50/50 NonOverlappingTemplate

4 2 4 10 6 5 6 5 4 4 0.534146 50/50 NonOverlappingTemplate

5 6 6 8 7 2 3 2 7 4 0.494392 50/50 NonOverlappingTemplate

9 1 5 5 4 6 5 4 5 6 0.616305 50/50 NonOverlappingTemplate

2 4 5 5 8 4 9 4 5 4 0.574903 50/50 NonOverlappingTemplate

8 4 3 4 8 4 6 3 6 4 0.699313 49/50 NonOverlappingTemplate

1 9 4 6 5 5 5 5 2 8 0.319084 50/50 NonOverlappingTemplate

6 6 8 6 4 7 6 3 2 2 0.534146 50/50 NonOverlappingTemplate

6 6 5 2 7 4 3 8 4 5 0.739918 47/50 NonOverlappingTemplate

5 5 5 5 5 6 7 3 5 4 0.991468 50/50 NonOverlappingTemplate

5 3 8 3 3 8 7 3 4 6 0.534146 49/50 NonOverlappingTemplate

3 1 5 3 7 7 3 8 6 7 0.350485 50/50 NonOverlappingTemplate

4 6 6 2 5 1 6 9 8 3 0.236810 50/50 NonOverlappingTemplate

2 1 4 10 3 10 4 7 6 3 0.035174 50/50 NonOverlappingTemplate

5 4 6 5 7 1 7 3 4 8 0.534146 50/50 NonOverlappingTemplate

8 3 6 3 2 10 4 5 4 5 0.289667 50/50 NonOverlappingTemplate

5 6 4 7 7 2 3 3 8 5 0.616305 50/50 NonOverlappingTemplate

5 2 5 10 4 5 1 9 6 3 0.108791 50/50 NonOverlappingTemplate

5 3 2 7 3 6 5 6 4 9 0.534146 49/50 NonOverlappingTemplate

6 9 7 4 5 3 3 4 6 3 0.616305 50/50 NonOverlappingTemplate

5 9 4 8 4 2 7 6 1 4 0.236810 50/50 NonOverlappingTemplate

4 4 6 8 4 2 6 6 4 6 0.816537 49/50 NonOverlappingTemplate

5 12 3 4 5 4 4 2 5 6 0.153763 50/50 NonOverlappingTemplate

4 6 6 4 3 6 5 5 7 4 0.971699 50/50 NonOverlappingTemplate

1 4 3 8 5 8 6 8 4 3 0.289667 50/50 NonOverlappingTemplate

1 7 3 7 5 7 3 4 7 6 0.494392 50/50 NonOverlappingTemplate

3 2 6 9 5 5 4 4 5 7 0.616305 48/50 NonOverlappingTemplate

3 5 6 6 7 8 5 2 6 2 0.574903 50/50 NonOverlappingTemplate

4 5 8 4 8 4 5 4 4 4 0.851383 49/50 NonOverlappingTemplate

6 4 5 4 3 5 5 9 5 4 0.851383 50/50 NonOverlappingTemplate

4 4 8 6 4 3 5 4 8 4 0.779188 50/50 NonOverlappingTemplate

4 8 4 6 8 0 2 4 5 9 0.108791 50/50 NonOverlappingTemplate

6 6 2 9 4 6 5 4 3 5 0.657933 50/50 NonOverlappingTemplate

8 6 2 4 3 8 6 6 5 2 0.455937 50/50 NonOverlappingTemplate

4 5 3 4 6 6 3 3 8 8 0.657933 50/50 NonOverlappingTemplate

6 5 4 4 1 4 4 6 8 8 0.534146 49/50 NonOverlappingTemplate

3 5 9 3 2 7 2 5 10 4 0.108791 50/50 NonOverlappingTemplate

3 6 7 5 2 5 8 2 5 7 0.534146 50/50 NonOverlappingTemplate

5 3 3 3 8 3 10 5 6 4 0.319084 50/50 NonOverlappingTemplate

6 4 7 5 3 9 7 4 3 2 0.455937 49/50 NonOverlappingTemplate

1 7 5 7 4 4 8 3 6 5 0.534146 50/50 NonOverlappingTemplate

5 5 4 4 8 5 3 6 6 4 0.935716 49/50 NonOverlappingTemplate

5 4 6 5 9 7 3 3 6 2 0.534146 50/50 NonOverlappingTemplate

3 6 7 6 8 5 5 4 3 3 0.779188 50/50 NonOverlappingTemplate

3 7 4 5 7 6 3 8 4 3 0.699313 50/50 NonOverlappingTemplate

6 4 9 4 2 2 4 7 4 8 0.319084 50/50 NonOverlappingTemplate

4 6 8 4 4 6 3 4 4 7 0.851383 50/50 OverlappingTemplate

6 4 8 2 6 5 2 4 7 6 0.616305 49/50 Universal

4 3 8 6 4 8 5 5 2 5 0.657933 50/50 ApproximateEntropy

5 6 3 3 2 5 4 4 1 3 0.534146 35/36 RandomExcursions

5 9 2 1 3 4 4 2 4 2 0.054199 34/36 RandomExcursions

5 2 6 5 0 5 5 0 5 3 0.066882 36/36 RandomExcursions

5 3 6 2 4 4 6 1 2 3 0.350485 36/36 RandomExcursions

4 3 2 3 2 4 5 7 1 5 0.299251 36/36 RandomExcursions

3 2 3 5 5 4 3 4 3 4 0.911413 36/36 RandomExcursions

4 2 4 4 4 4 5 4 3 2 0.911413 36/36 RandomExcursions

2 3 0 2 3 3 7 3 10 3 0.002624 35/36 RandomExcursions

4 3 6 3 3 8 2 4 1 2 0.122325 36/36 RandomExcursionsVariant

3 2 7 6 1 5 2 6 3 1 0.066882 36/36 RandomExcursionsVariant

4 2 9 3 2 2 2 3 5 4 0.082177 36/36 RandomExcursionsVariant

3 4 6 5 3 1 5 3 1 5 0.350485 36/36 RandomExcursionsVariant

3 4 7 5 6 2 1 3 3 2 0.213309 35/36 RandomExcursionsVariant

3 6 7 3 4 3 2 2 3 3 0.407091 35/36 RandomExcursionsVariant

6 3 3 2 5 4 8 3 0 2 0.054199 35/36 RandomExcursionsVariant

4 9 0 1 4 5 5 4 2 2 0.014216 35/36 RandomExcursionsVariant

4 5 3 4 5 3 0 8 0 4 0.035174 35/36 RandomExcursionsVariant

2 4 0 6 4 6 6 5 1 2 0.066882 36/36 RandomExcursionsVariant

2 3 5 5 3 2 3 7 4 2 0.407091 36/36 RandomExcursionsVariant

4 4 4 2 2 2 2 5 4 7 0.407091 35/36 RandomExcursionsVariant

4 3 2 3 2 1 2 6 6 7 0.122325 36/36 RandomExcursionsVariant

5 1 1 3 3 2 3 3 5 10 0.008879 35/36 RandomExcursionsVariant

3 4 4 1 1 5 2 5 4 7 0.213309 35/36 RandomExcursionsVariant

5 4 0 4 3 7 2 1 4 6 0.082177 35/36 RandomExcursionsVariant

6 3 2 3 2 5 4 4 3 4 0.739918 35/36 RandomExcursionsVariant

6 5 3 2 4 3 1 4 2 6 0.350485 34/36 RandomExcursionsVariant

5 5 4 4 6 5 7 6 4 4 0.991468 50/50 Serial

1 10 6 1 8 5 6 6 5 2 0.075719 50/50 Serial

9 8 2 4 5 5 5 3 6 3 0.455937 50/50 LinearComplexity

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -

The minimum pass rate for each statistical test with the exception of the

random excursion (variant) test is approximately = 47 for a

sample size = 50 binary sequences.

The minimum pass rate for the random excursion (variant) test

is approximately = 33 for a sample size = 36 binary sequences.

For further guidelines construct a probability table using the MAPLE program

provided in the addendum section of the documentation.

- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -