An Efficient Hardware Random Number Generator

D. C. Ranasinghe[*a], Daihyun Lim[b], Srinivas Devadas[b], Behnam Jamali[a], Zeng Zhu[a], Peter H. Cole[a]
[a]Auto-ID Lab, Dept. of Electrical & Electronic Engineering, Univ. of Adelaide, SA, Australia 5005;
[b]Massachusetts Institute of Technology, 77 Massachusetts Ave., Cambridge, MA, USA, 02139

ABSTRACT

A hardware random number generator is different from a pseudo-random number generator; a pseudo random number generator approximates the assumed behavior of a real hardware random number generator. Simple pseudo random number generators suffices for most applications, however for demanding situations such as the generation of cryptographic keys requires an efficient and a cost effective source of random numbers.

Arbiter-based Physical Unclonable Functions (PUFs) proposed for physical authentication of ICs exploits statistical delay variation of wires and transistors across integrated circuits in the fabrication process to build a secret key unique to each IC. Experimental results and theoretical studies show that a sufficient amount of variation exits across IC's. This variation enables each IC to be identified securely.

It is possible to exploit the unreliability of these PUF responses to build a physical random number generator. There exists measurement noise, which comes from the instability of an arbiter when it is in a racing condition. There exist challenges whose responses are unpredictable. Without environmental variations, the responses of these challenges are random in repeated measurements.

Compared to other physical random number generators, the PUF-based random number generators can be a compact and low-power solution. A 64-stage PUF circuit costs less than 1000 gates and the circuit can be implemented using standard IC manufacturing process.

In addition to the development of a fast and an efficient random number generator, we have analysed and described the quality of random numbers produced using an array of tests used by the National Institute of Standards and Technology to evaluate the randomness of random number generators designed for cryptographic applications.

Keywords: PUF, Metastability.

1. INTRODUCTION

Random numbers are required in many applications, such as modelling and simulation applications. However, the most significant of which is their application in cryptography. The wide array of cryptographic applications employs keys that must be generated using a random process to ensure the security of the cryptographic system. Numerous cryptographic protocols also require random or pseudorandom inputs such as in the generation of digital signatures, or the generation of challenges in a challenge-response protocol.

Random number generators can be subdivided into the following categories

1. Pseudorandom number generators
2. True Random number generators

The above types of generators are more specifically referred to as random bit generators when they are used to produce a stream of binary numbers (ones and zeros). This stream may then be subdivided to from blocks of random numbers of required block sizes, such as 256 bits, 512 bits, or 1024 bits.

Pseudorandom number generators are based on a computational algorithm which, given a truly random sequence of length l, outputs a binary sequence of length $x >> l$ which has the appearance of being random [1]. Such a generator may employ a one-way function f and a random seed s to generate a sequence of values

$$s, s+1, s+2, \ldots\ldots.$$

where

$$s+1 = f(s), s+2 = f(s+1), s+3 = f(s+2), \ldots\ldots.$$

There are various algorithms for pseudorandom number generation (PRBG); ANSI X9.17 and FIPS 186 are examples of such generators [1,15]. The output of a PRBG is not random and the number on possible output sequences is at most $2^l / 2^x$, of all possible bit sequences of length *x*.

A true random bit generator is a system whose output consists of completely unpredictable, that is statistically independent and unbiased sequence of bits. In cryptographic applications, the generator must also not be observable and must not provide a means for an attacker to manipulate the output of the generator.

True random numbers cannot be generated using computational algorithms as done in pseudorandom number generators since algorithms are deterministic. Thus true random number generators seek to exploit random physical phenomenon (non deterministic phenomenon), such as the decay of radioactive isotopes to generate random numbers. True random number generators are useful since there may not be equivalent algorithms to simulate and predict the physical phenomenon. However exploiting this random source to produce a bit sequence statistically independent and unbiased is a not effortless. In addition random bit generators based on natural sources of randomness are expose to external influences, failures and possible external attacks to manipulate the random source. However most statistical defects caused by external influences can generally be removed by post processing of the generated bit sequence as shown in figure 1.

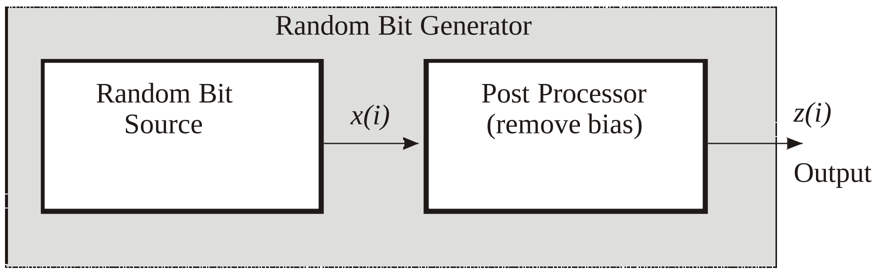

Figure 1: Random bit generator with a post processor

1.1. Sources of Randomness

The random sources can be constructed from dedicated hardware devices (Hardware based generators); such as oscillator with considerable jitter. A number of hardware random number generator can be found in [16], [17] and [18]. Random sources may also be extracted from software procedures using the platform on which the generator is implemented (Software based generators). Event timing of may provide sources for SWG, such as

- mouse clicks,
- key strokes, or
- network access.

Software based generators (SWG) are not based on very ideal sources, and it is difficult to properly evaluate and assess the robustness of the sources in regards to observations and possible manipulations. Thus software based generators use a combination of sources to obtain protection from one of it’s sources being manipulated. Raw bits generated from SWGs most often need to be heavily processed before a random sequence is obtained.

Hardware based generators (HWG) have a number of advantages over software generators. Generally HWG can be implemented using common integrated technologies, they can be fabricated into tamper resistant devices to prevent an adversary from performing observations or manipulating the generator. Hardware based generator are also faster, and are capable of producing generators with high throughput. Commonly used sources for hardware generators are physical phenomena such as

- thermal noise
- shot noise
- avalanche noise
- phase noise.

The HWG presented in this paper is a fully integrable random number generator that exploits the metastability state that arises as a result of a race condition in latches.

1.2. Metastability

When a signal violates a devices signal setup and hold timing requirements, the output from the device becomes unpredictable [19]. The observed output might be that of a high or a low or it may be observed to oscillate between a digital high and a digital low. This, widely undesirable phenomenon is known as metastability [22, 23]. It is a phenomenon that digital circuit designers try to avoid, as the final state of the device is unpredictable.

The reasons behind metastability involve the inability of a latch to lock onto a signal. This can occur when the latch input level changes at about the same time causing an input signal that falls within the invalid region between a logic zero and a logic one. Theoretically a device may remain in a metastable condition indefinitely, thus the device output may continue to oscillate of hang in a region that is neither logic zero nor logic one. However, in reality this is not the case, thermal and induced noise will wobble the meta-stable state causing the device (latch) output to change to a logic one or a logic zero. Once a latch has entered a metastable state, the probability that it will remain in that state a time *t* later has been shown to be negative exponentially distributed [19]. Albeit a small probability that the device may remain in a meta-stable state, there is a considerable chance that the device will, given time, return to a stable state.

Once a device enters a metastable condition, the length of time that it will remain in that state is modelled it is not possible to predict the final outcome of the device thus metastability provides a source of randomness that can be used to construct a simple and efficient true random number generator. A recent attempt at using metastability to generate random numbers can be found in [24], while other attempts to use metastability as a source of randomness can be found in [25, 26] and [27]. While some of these generators used elaborate mechanisms to achieve randomness while [24] failed to account for environmental variations that will effect the functioning of the generator.

The following section describes the details of the implementation.

2. METASTABILITY BASED GENERATOR DESIGN

A technique that exploits the statistical delay variations of wires and transistors across IC were used to build a secret key unique to each IC [20,28] with challenge-response pairs created using a PUF (Physically Unclonable Functions) circuit integrated on silicon.

The PUF circuit is able to uniquely characterise each IC due to manufacturing variations inherent in a fabrication process [20,28]. These individual characteristics then become similar to the secret keys used in a symmetrical encryption scheme. Thus, it is possible to identify and authenticate each IC reliably by observing the PUF response. However for some challenges setup and hold time violation of the arbiter leads to unpredictable response as the result of the arbiter (latch) entering into a metastable condition. It is possible to exploit the unreliability of the PUFs to transform them into PUF Random Number Generators (PUFRNGs).

2.1. Circuit implementation

The block diagram presented in Figure 2 depicts structure of a PUF circuit. This is called an arbiter based PUF [28]. In the implementation presented, two delay paths are excited simultaneously to allow the transitions to race against each other. The arbiter block at the end of the two delay lines determine which rising edge arrives first and sets its output to 1 or 1 depending on which signal arrives first. This circuit used takes a *n* bit challenge where each bit is fed into *b0, b2, b3, …, bn* to configure the delay paths. The PUFRNG used 64 bit challenges (that is *n* is 64).

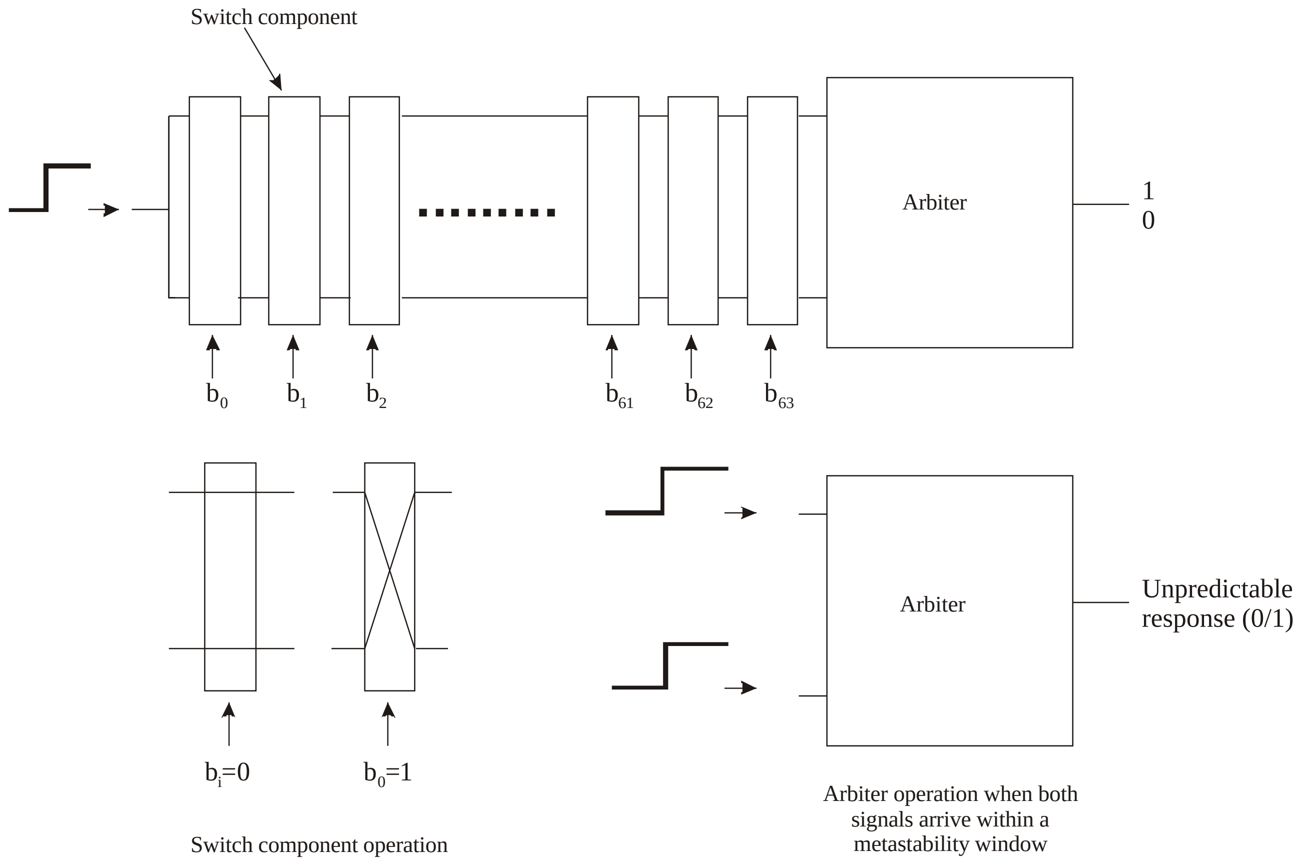
Switch component
Arbiter
1
0
b_0
b_1
b_2
b_{61}
b_{62}
b_{63}
b_i=0
b_0=1
Switch component operation
Arbiter
Unpredictable response (0/1)
Arbiter operation when both signals arrive within a metastability window

Figure 2: PUFRNG based on a arbiter-based PUF circuit.

The details of the switch component can be found in [28] while a logic level diagram of the PUFFNG is given in Figure 3 and 4.

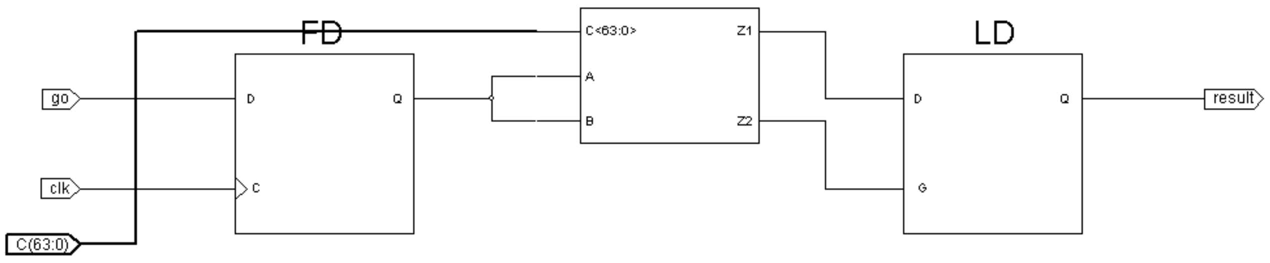

Figure 3: Block Diagram of the PUFRNG. Here FD is a flip-flop initialling a race, which is then fed into a delay circuit along with a 64 bit challenge. LD is the arbiter implemented by a transparent latch, which produces a single bit result at the end of the race.

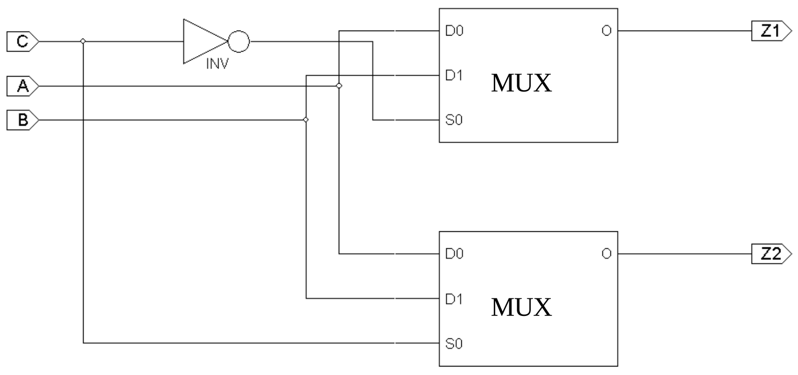

Figure 4: Switch component implemented using two-to-one multiplexers. Here C is the challenge bit while A and B paths contain the transition.

The switch component indicated in Figure 2 is implemented using a pair of two-to-one multiplexers. Depending on the select bit *So* the switch either allows the signal to travel straight through or crosses the paths over so that the signal that was on the top rail is now on the bottom rail and vice versa.

It is thus possible to exploit the unreliability of PUF responses resulting from challenges causing the latch to enter into a meta-stable condition to build a random number generator. Thus our source of randomness is the metastability of the latch where, it is not possible to predict the likelihood of the output of the latch. Figure 2 shows the distribution of the random variable k, which is the number of 1s in 200 repeated measurements for a given challenge. In the middle of the density function, there exist the challenges whose responses consist of 50% logic once and 50% logic zeros. Without much environmental variations, the responses of these challenges can be used in repeated measurements to generate a random bit stream.

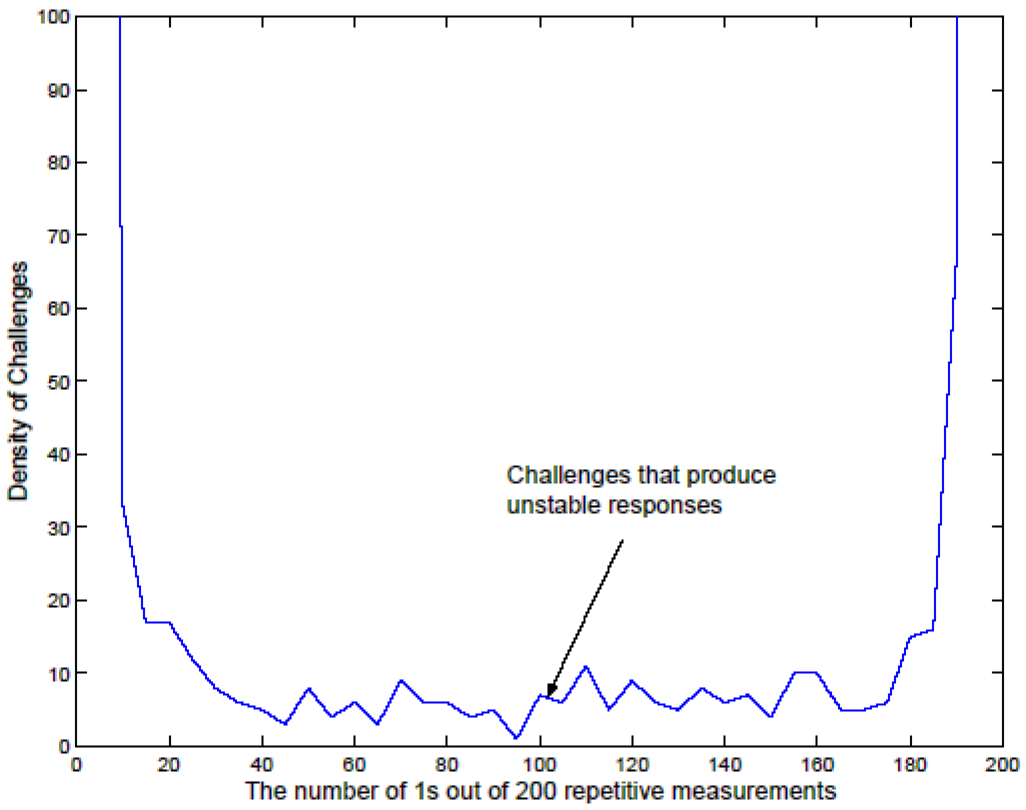

Figure 5: The density function of the random variable k, where k is the number of 1's out of 200 repetitive measurements [20].

The responses from the PUF are sensitive to environmental conditions such as temperature and power supply voltage. In addition fabrication process variations will also influence the responses obtained from one PUF to another for the same challenge. A challenge that generates unreliable responses may not generate an unreliable response if the conditions change. Hence, each time a PUF is used as a source of randomness, a number of random challenges must be used to calibrate the PUF to discover a challenge that produces unstable responses.

Statistically, from 10,000 challenges, there exist approximately 10 challenges whose responses are random, that is 0.1% of the challenges produce unstable responses. Based on the performance of PUF circuits it takes 0.5 seconds to test the randomness of 10,000 challenges by 1000 repeated measurements [20]. Hence it is possible to complete the initialization of a PUF random number generator within a second.

This unique ability of the PUF random number generator allows the generator to adapt to external influences and to fine tune the generator for greater performance. Compared to other physical random number generators, the PUF random number generator can be a compact and a low-power solution. A 64-stage PUF circuit costs only less than 1000 gates and the circuit can be implemented using standard IC manufacturing processes. Additionally, various kinds of low power techniques such as sub-threshold logic design and multi-thresholds CMOS design can be utilized to reduce the power consumption to make it suitable for use in devices sensitive to low power consumption. The following section will examine procedures for evaluating randomness so that the quality of random numbers generated using the PUF random number generator an be analysed.

3. EVALUATING RANDOMNESS

The randomness testing of random number and pseudorandom number generators are required to ensure that these numbers are indeed random. Randomness is a property that is characterized and described in terms of probability. The outcome of statistical tests applied to a random number sequence can be described by in probabilistic terms.

There are a wide number of statistical tests available to test the randomness of random and pseudorandom number generators. Hence it is not possible to define a specific number of statistical tests that are capable of establishing the randomness of a sequence. These statistical tests results do not provide black and white results, however it is possible to interpret these results with care and caution to determine the randomness of a generator.

A set of statistical tests for randomness used by The National Institute of Standard and Technology (NIST) is used to evaluate the randomness of the PUF random number generators (NIST is an agency of the U.S. Commerce Department's Technology Administration [29]). A more detailed discussion of the tests and their interpretations can be found in [1]. It is however important to note that NIST believes that the test suite is suitable for identifying "deviations of binary sequence" from randomness. However factors contributing to these deviations are numerous and it is possible to expect a certain number of failures from a particular generator.

Each NIST statistical test, tests the sequence to establish whether there is significant evidence to suggest that the null hypothesis (H0) should be rejected in favour of the alternative hypothesis. Here the null hypothesis H0 is that the sequence being tested is random, while the alternative hypothesis H1, is that the sequence being tested is not random. Thus for each test applied a decision is derived that accepts or rejects the null hypothesis and hence whether the generator is indeed producing random values. Each test statistic obtained for each test is then used to calculate a P-value that indicates the strength of the evidence against the null hypothesis. Thus for each test, each P-value is the probability that a perfect random number generator would have produced a sequence that is less random than the tested sequence, given the particular non-randomness being gauged by that particular test.

3.1. Hypothesis testing

A statistical hypothesis, commonly denoted as H0 is an assertion about a distribution of one or more random variables [21]. This assertion can then be tested using a method based on the observations made on the random variables. The test can provide evidence to support or reject the hypothesis H0. It is important to note here that the test only provides a measure of the strength of the evidence provided by the data against the hypothesis. Thus the deduction derived from the test is not irrefutable but rather probabilistic.

3.2. Statistical Test Suite

The statistical test suite employed (as detailed in [2]) consists of sixteen test outlined and briefly described in table 1.

No	Test	Description
1	The Frequency Test	The test aims to determine whether proportion of 1's and 0's in a given sequence is that expected from for a random sequence. [1, 2]
2	Frequency Block Test	Similar to the above test but the focus is now the within M-bit blocks within a given sequence. The block size used was 512 bits[2]
3	The Runs Test	The purpose of the test is to determine whether the number of runs (either 0's or 1's) of various lengths in the given sequence is as expected from a random sequence [1,2,3, 4]
4	Test for the Longest Run of Once in a Block	The test examines the length of the longest run of once within the given sequence is consistent with the length of the longest run of once that would be expected in a random sequence [1, 2, 3, 4, 5].
5	The Binary Matrix Rank Test	The purpose of the test is to discover liner dependence among fixed length substrings of the original sequence [2, 6, 7].

6	The Discrete Fourier Transform Test	This test is used to examine the peak heights in the Discrete Fourier Transform of a given sequence. The test is able to depict periodic features in the tested sequence by examining the number of peaks exceeding the 95% peak height threshold value. The number of peaks exceeding this threshold peak height is less than 5% for a random sequence [2, 8].
7 - 154	The Non-overlapping Template Matching Test	This test is designed to search for the number of occurrences of pre-specified bit pattern. The test is aimed at detecting generators that produce large occurrences a certain aperiodic bit pattern. The size of the template used was 9 bits in length, which resulted in a total of 148 templates being applied to each of the sequences. The results from this test is similar to having applied 148 different test on the sequence of numbers provided [2].
155	The Overlapping Template Matching Test	Similar to the above test, with the exception that once a pattern is found the search window is now advanced only one bit instead of advancing the window to the end of the pattern as performed in the non-overlapping template matching test. The length of the once template used was 8 bits in length [2, 9].
155, 156	The Serial Test	The purpose of this test is to establish whether the 2^m m-bit overlapping pattern occurs as many times as expected from a random sequence. In a random sequence every m bit pattern has the same probability of appearing as every other m-bit pattern [1, 2, 10].
157	The Approximate Entropy Test	The purpose of the entropy test is to compare the frequency of overlapping bit patterns of two consecutive lengths of m and m+1 bits with that expected from a random sequence [2, 11, 12, 13].
158, 159	The Cumulative Sums Test	The Cumulative Sums (Cusum) test aims to determine whether the cumulative sum of partial sequences occurring in a given bit string is that expected from a random sequence. The cumulative sum may be considered as a random walk and thus for a random sequence the deviation from the random walk should be near zero. This test is performed once going forward in the sequence and then going in the reverse direction [2, 5].

Table 1: Description of the tests used from the NIST test suite

4. ANALYSIS AND INTERPRETATION OF THE TEST RESULTS

Fabricated PUF generators were mounted on a data acquisition board to feed the PUF generator challenges and then obtain the results. The PUF generator was initialised using 10,000 random challenges to discover a challenge that produced and unstable response at r.t.p. The challenge that forced the latch to enter into a metastable condition was repeatedly presented to the PUFRG to obtain a data set of a 1,500,000 bits. Post processing of the output sequence resulted in a bit string of length 592,000.

Post processing was required to remove bias from the original bit string, the method employed is detailed in [1], and it involves the parsing of the bits generated from the random number generator in groups of two using the transformation given in Table 2.

Groups	Transformation
10	1
01	0
11	Ignore (remove from sequence)
00	Ignore (remove from sequence)

Table 2: Post processing transformations

4.1. Parameters used in the test suite

Table 3 provides the test suite specific parameters used during the testing.

4.2. Evaluating test results

The guidelines in [2] can be used to interpret the test results. Details of these guidelines can be obtained from [2] and [1].

Table 3 gives a summary of the number sequences that passed each of the test performed using a p-value of 0.01 (that is alpha = 0.01) as the significance level to reject or accept the null hypothesis. A pass in a test indicates that there is no significant evidence to reject the null hypothesis and thus the sequence can be considered to be random.

Number	Test	Description
1	The Frequency Tests	100% sequences passed.
2	Frequency Block Test	100% sequences passed.
3	The Runs Test	100% sequences passed.
4	Test for the Longest-Run-of-Once in a Block	100% sequences passed.
5	The Binary Matrix Rank Test	100% sequences passed.
6	The Discrete Fourier Transform Test	100% sequences passed.
7 - 154	The Non-overlapping Template Matching Test (A total of 148 templates are applied to each sequence set)	100% of the sequences passed 121 template-matching tests. 93% of the sequences passed 146 template-matching tests. 87% of the sequences passed 148 template-matching tests.
155	The Overlapping Template Matching Test	100% of the sequences passed.
155, 156	The Serial Test	100% of the sequences passed the first test statistic. 100% of the sequences passed the second test statistic.
157	The Approximate Entropy Test	100% of the sequences passed.
158, 159	The Cumulative Sums Test	100% of the sequences passed the forward cusum. 100% of the sequences passed the backward cusum.

Table 3: Test Result Summary

The test results from the individual tests do not indicate a deviation from randomness. However using two recommended approaches to interpreting the empirical results can be used to further investigate the validity of the null hypothesis that is the sequences are indeed random. The method adopted by NIST include

1. the examination of the proportion of sequences that pass a given statistic test and
2. the distribution of P-values to ensure that they are uniformly distributed.

If either of the above evaluation methods fails, the null hypothesis can be rejected, however further testing on the generator should be performed to ensure that the conclusion derived was not due to a statistical irregularity.

4.2.1. Proportion of sequences passing a Test

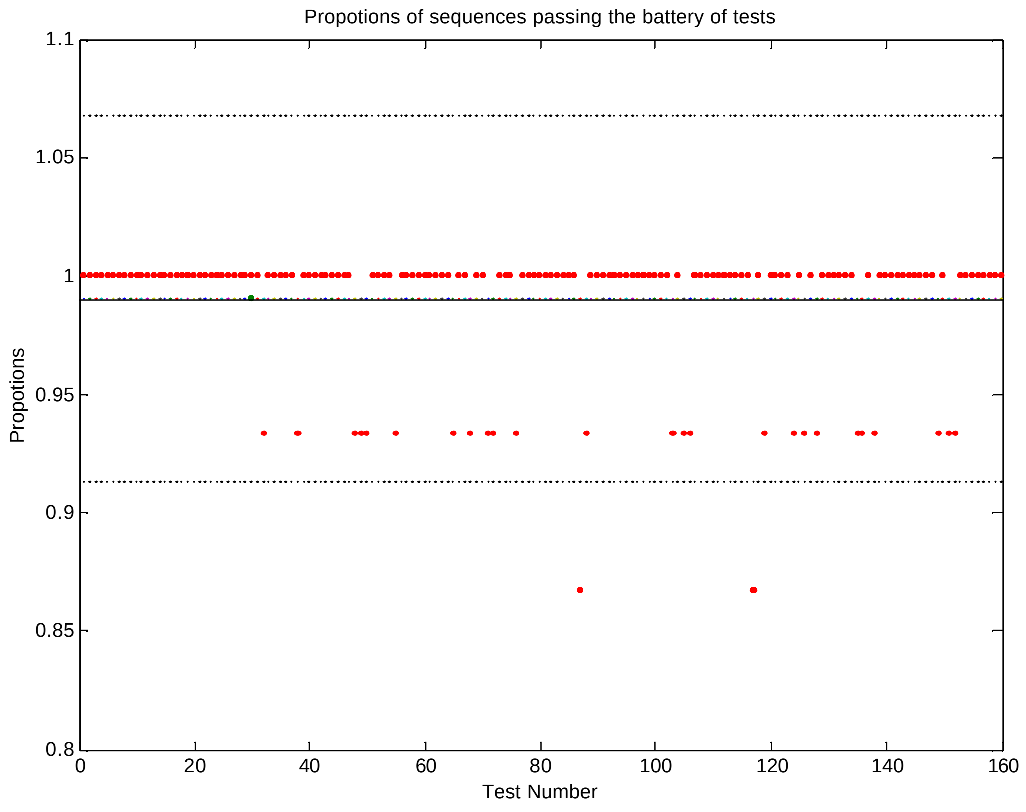

Figure 6: Proportion of sequences passing each test based on their p-value.

4.2.2. Uniform distribution of P-values

Evaluation of the uniformity of the distribution of p-values is discussed in detail in [28]. A p-value calculated on the distribution of p-values of a statistical test can be used to accept the distribution as being uniform or non-uniform. Significance level of $\alpha = 0.001$ was used to assess the uniformity. Thus a p-value calculated on the distribution of p-values ≥ 0.0001 was considered to have a uniform distribution [28]. Table 4 summarises the assessment performed on the p-values obtained for each statistical test.

Number	Test	Uniformity of p-value distribution
1	The Frequency Tests	PASS
2	Frequency Block Test	PASS
3	The Runs Test	PASS
4	Test for the Longest-Run-of-Once in a Block	PASS
5	The Binary Matrix Rank Test	PASS
6	The Discrete Fourier Transform Test	PASS
7 - 154	The Non-overlapping Template Matching Test (A total of 148 templates are applied to each sequence set)	Strong evidence to suggest a uniform p-value distribution for 147 of 148 of the templates. 1 of 148 templates did not have strong evidence suggest a uniform distribution of p-values.
155	The Overlapping Template Matching Test	PASS distribution

155, 156	The Serial Test	PASS (first test statistic) PASS (second test statistic)
157	The Approximate Entropy Test	PASS
158, 159	The Cumulative Sums Test	PASS (forward cusum) PASS (backward cusum)

Table 4: Results of uniform distribution of p-values

REFERENCES

1. A. Menezes, P. van Oorschot and S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996.
2. A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray and S. Vo, *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*, NIST Special Publication 800-22, 2001.
3. J. D. Gibbons, *Nonparametric Statistical Inference*, 2nd edition, New York: Marcel Dekker, 1985.
4. A. P. Godbole and S. G. Papastavridis, (ed), *Runs and patterns in probability*: Selected papers. Dordrecht: Kluwer Academic, 1994.
5. P. Revesz, Random *Walk in Random and Non-Random Environments*, Singapore, World Scientific, 1990.
6. I. N. Kovalenko, *Distribution of the linear rank of a random matrix*, Theory of Probability and its Applications., vol.17, pp. 342-346. 1972.
7. G. Marsaglia and L. H. Tsay, *Matrices and the structure of random number sequences*, Linear Algebra and its Applications, vol. 67, pp. 147-156, 1985.
8. R. N. Bracewell, *The Fourier Transform and Its Applications*., McGraw-Hill, 1986.
9. O. Chrysaphinou and S. Papastavridis, *A Limit Theorem on the Number of Overlapping Appearances of a Pattern in a Sequence of Independent Trials*, Probability Theory and Related Fields, vol. 79 pp. 129-143, 1988.
10. I. J. Good, *The serial test for sampling numbers and other tests for randomness*, Proc. Cambridge Philos. Soc., vol. 47, pp. 276-284, 1953.
11. S. Pincus and B. H. Singer, *Randomness and degrees of irregularity*, Proc. Natl. Acad. Sci. USA, vol. 93, pp. 2083-2088, March 1996.
12. S. Pincus and R. E. Kalman, *Not all (possibly) Random Sequences are created equal*, Proc. Natl. Acad. Sci. USA, vol. 94, , pp. 3513-3518, April 1997.
13. A. Rukhin , *Approximate entropy for testing randomness*, Journal of Applied Probability, vol. 37, 2000.
14. F. Spitzer, *Principles of Random Walk*. Princeton: Van Nostrand, 1964.
15. Cryptography Theory and Practice, CRC Press.. 1995
16. W.T. Holman, J.A. Connelly and A. B. Downlatabadi, *An Integrated Analog/Digital Random Noise Source*, IEEE Trans. Circuits and Systems I, vol. 44, no. 6,pp. 521-528, June 1997.
17. M. Dichtl and N. Janssen, *A High Quality Physical Random Number Generator*, Proc. Sophia Antipolis Forum Microelectronics, pp. 48-53, 2000.
18. C. S. Petrie and J.A. Connelly, *A Noise-Based IC Random Number Generator for Applications in Cryptography*, IEEE Trans. Circutis and Systems I, vol. 47, no. 5, pp.615-621, May 2000.
19. Phillips Semiconductors, *A metastability primer*, Application Note, AN219.
20. D. Lim, *Extracting Secret Keys from Integrated Circuits*, Master thesis, Massachusetts Institute of Technology, May 2004.
21. R. S. Kenett and S. Zacks, Modern *Industrial Statistics: Design and Control of Quality and Reliability*, Duxbury Press, 1998.
22. T.J. Chaney, *Measured Flip-flop Responses to Marginal Triggering*, IEEE Trans. On Comp., vol. 32, no. 12, pp. 1207-1209, December 1983
23. G.R. Couranz and D.F. Wann, *Theoretical and Experimental Behaviour of Synchronizers Operating in the Metastable Region*, IEEE Trans. on Comp., vol 24, no. 6, pp. 604-616, June 1975.
24. M. Epstein, L. Hars, R. Krasinski, M. Rosner and H. Zheng, *Design and Implementation of a True Random Number Generator Based on Digital Circuits Aritifacts*, CHESS 2003.
25. M. J. Bellido, A. J. Acosta, et al., A Simple *Binary Random Number Generator: New Approaches for CMOS VLSI*,35th Midwets Symposium on circuits and Systems, August 1992.

26. M. J. Bellido, A. J. Acosta, M. Valencia, A. Barriga, and J.L. Huertas, *A Simple Binary Random Number Generator,* Electronic Letters, vol. 28, no. 7, pp. 617-618, March 1992.
27. S. Walker and S. Foo, *Evaluating Metastability in electronic Circuits for Random Number Generation,* IEEE Computer Society Workshop on VLSI, pp. 99-102, April 2001.
28. J.W. Lee, D. Lim, B. Gasseend, G.E. Suh, M. van Dijk, S. Devadas, *A Technique to Build a Secret Key in Integrated Circuits for Identification and Authentication Applications,* 2004 Symposium on VLSI circuits, pp 176-179, 2004.
29. NIST home page, http://www.nist.gov.