On-chip Lightweight Implementation of Reduced NIST Randomness Test Suite

Vikram B. Suresh, Daniele Antonioli* and Wayne P. Burleson
Dept. of Electrical and Computer Engineering, University of Massachusetts, Amherst, USA
*Dept. of Electronical, Electronic and Information Engineering,University of Bologna, Italy
{vsuresh, burleson}@ecs.umass.edu, daniele.antonioli@studio.unibo.it

Abstract— **On-chip Random Number Generators (RNGs) are critical components in lightweight ubiquitous devices like RFIDs and smart cards. These devices require low cost test methodologies and security against cryptanalytic and invasive attacks. In this work we propose an on-chip implementation of a reduced set of NIST-SP-800-22 randomness test suite to provide on-line RNG testing for low cost security devices along with run-time monitoring of RNG performance. The on-chip NIST module monitors the effect of dynamic variation of operating condition and time dependent wear-out on RNG circuits. It indicates invasive attacks on RNG and allows the secure system to take protective measures. Six NIST tests are optimized to a hardware design friendly format, but in compliance with the NIST standard. The lightweight implementations reduce complex statistical and arithmetic operations of conventional NIST tests to a series of bit stream count and compare operations. A cycle-to-cycle serial test of incoming bits from RNG eliminates need for additional storage. A partial re-configurable feature is designed to set the pass/fail threshold for each test depending on the system requirements. The on-chip NIST module, although not exhaustive, is an effective layer of validation and security for RNG circuits. The six 128-bit tests implemented in 45nm NCSU PDK have a total synthesized area of ~1926.sq.µm for an optimized frequency of 2GHz. The total dynamic power is 3.75mW and leakage power is 10.5µW. At 2Gbps, the NIST module consumes 1.87pJ/bit. The lightweight ultra-low power implementation is scalable for larger input bit samples.**

Keywords- PRNG, TRNG, NIST, Statistical Test

I. INTRODUCTION

Random Number Generators (RNGs) are cryptographic primitives used to generate random keys, IDs and nonces. RNGs are generally classified into two categories, Pseudo Random Number Generators (PRNG) and True Random Number Generators (TRNG). PRNG are deterministic systems based on algorithms and generate bit sequences with a random distribution over a large sample of bits. TRNGs, on the other hand, harness random physical phenomena to generate random bits. On-chip noise, clock jitter and stray electromagnetic field are some of the sources of randomness for a TRNG. TRNG circuits are often used to periodically seed a PRNG, generate nonces for security protocols and in data encryption/decryption. The most commonly used TRNG circuits are based on Ring Oscillators which sample and digitize on-chip jitter noise to generate random bits [1, 2]. Ring Oscillator based circuits are also popular for FPGA based implementations [3, 4].

Noise amplifiers and Analog-to-Digital Converters (ADC) are used to sample on-chip thermal and shot noise to obtain random samples [5, 6]. Digital circuits for sampling thermal noise use metastable elements like a pair of cross coupled inverters [7, 8, 9]. Power up state of SRAM [10], read-refresh collision in DRAM [11] and Random Telegraph Noise (RTN) in Contact Resistive RAM [12] are examples of memory-cell based TRNG circuits.

The conventional metrics for evaluating an analog/digital circuit are area, power and performance. However, RNG circuits need a fourth metric which measures the degree of randomness. The most basic metric of randomness is the Shannon bit entropy which measures the proportions of bit 0's and 1's in a sequence to give an entropy value in the range of 0 to 1. However, equal proportions of 0's and 1's do not validate all possible weaknesses of an RNG. Sophisticated statistical tests are required to quantify the various aspects of randomness like correlation, run length and random distribution of bits. The most popularly used test suites are the RNG test suite by American National Institute of Standards and Technology (NIST) [13] and the DIEHARD Tests [14]. These test suits evaluate large sequences of bit streams for a pre-defined null hypothesis to predict the randomness of the source. The NIST test suits perform an exhaustive statistical analysis and hence are computation intensive. In [15], approximations in the test suite are proposed for a computation friendly implementation. Byte-wise implementation [16] and parallelizing the tests [17] further improve run time of NIST test suite. A test methodology for TRNG circuits in compliance with NIST standards is proposed in [18], while a run-time hardware implementation for NIST tests on FPGA is proposed in [19]. Researchers have also highlighted approximation errors in frequency tests in NIST suite [20] and have proposed second level tests for a more complete validation for randomness of bit sequences [21, 22].

In this work, we propose an on-chip lightweight implementation of a reduced set of NIST Test suite. The implementation is targeted to provide a run-time cycle-to-cycle evaluation of on-chip TRNG circuits at very low area and energy cost. The rest of this paper will discuss the motivation behind this work in section II, an introduction to reduced NIST test suite in section III, optimization and digital logic implementation in section IV, implementation results in section V, conclusion and future work in section VI.

II. MOTIVATION

Random Number Generators are critical blocks in a variety of cryptographic systems. They provide keys or IDs to initiate an encryption algorithm or a secure communication

This work is funded by Task.ID. 1836.074, Texas Analog Center of Excellence, Semiconductor Research Corporation.

protocol. As a result, weak RNGs can be a single point of failure of the entire system. The weakness of an RNG can be due to a weak algorithm (PRNG), a bad source of randomness or process induced variation in *sample and digitize* circuit (TRNG). The algorithm used in a PRNG can be validated before implementing in a system through regression tests or statistical and empirical analysis. However, TRNG circuits depend on physical implementation and local sources of randomness. Therefore, the statistics of these circuits have to be validated post device fabrication. Unlike conventional VLSI testing methodologies, the statistics of TRNG circuits cannot be tested using fault models. Large sets of data have to be generated and validated using statistical test suite. Due to increasing variation in fabrication process, testing a larger sample of chips does not necessarily guarantee a good TRNG in every chip. An exhaustive post-fabrication test will increase the test time and hence the cost per chip. In this work we propose a lightweight implementation of six (out of fifteen) statistical tests in the NIST test suite. The proposed implementation is not a replacement to the exhaustive testing provided by the NIST standards. However, they encompass the mandatory tests listed in FIPS 140-1 standard [30] to provide a layer of security for lightweight TRNG implementations and detect abnormal behavior due to variation in fabrication process, dynamic variation in environmental conditions during run-time and malicious attacks on TRNGs.

RNG circuits are used in ubiquitous devices like smart cards, RFID tags and sensor nodes. These devices are extremely low-cost and cannot afford to go through an extensive post-fabrication testing process to validate the TRNG circuits. They are also highly constrained in area and may be passively powered or battery operated. Therefore, a lightweight on-chip NIST module provides efficient statistical validation for the generated random bit stream. A reduced test implementation operating cycle-to-cycle on fewer samples eliminates the need for additional storage elements. With advancing CMOS technologies, transistors are increasingly sensitive to dynamic variation in operating temperature and power supply noise. Further, time dependent wear-out like Negative Bias Temperature Instability (NBTI) and Hot Carrier Injection (HCI) also affect circuit performance. A one-time statistical test performed during the chip testing phase does not provide constant monitoring of circuit behavior. A good TRNG could degrade intermittently due to temperature or voltage variation and permanently degrade over time due to wear out. An on-chip test module can be used to continuously monitor the output of RNG. The cryptographic system using the RNG can make an informed decision about the randomness of inputs in run-time. TRNG in Intel's Ivy Bridge Microprocessor incorporates a health check by counting empirically arrived bit patterns [29]. The health check only detects extreme cases of bias with Shannon entropy less than 0.5. The TRNG module depends on AES based conditioning for entropy improvement. In [19], a complete NIST test suite implementation has been proposed for real-time statistical test. However, such an implementation adds tremendous area overhead to the system. The authors indicate that only 2 NIST tests could be ported completely on a Xilinx Virtex II Pro FPGA V2P30. The test suite implementation itself consumes a large area of the order of 1000's of flip flops.

As discussed earlier, RNG could serve as a single point of failure and hence provides an avenue for malicious attack on cryptographic systems. In [23], J. Kelsy, *et. al* discuss in detail the various cryptanalytic attacks on PRNG. A practical attack on ring oscillator based TRNG is demonstrated in [24]. The authors present a real attack on EMV card using frequency injection. The randomness of the TRNG is compromised and key space of the secure micro controller is reduced from 2^{64} to 3300. Other practical attacks have also been reported in literature, like breaking weak RNG in MIFARE card [25], eavesdropping on a weak PRNG on EPC Gen2 compliant RFID tag [26] and contactless electromagnetic attack on ring oscillator based TRNG [27]. Commercial microprocessor manufacturers use on-chip sensors (temperature, voltage) to detect invasive attacks. An on-chip NIST module can detect variation in randomness, which is the symptom of an attack, thereby protecting the device against all sources of attack. The randomness information can be used by the secure module (Crypto module/ Micro controller) to take necessary evasive action, like a Denial of Service to the attacker and thereby prevent economic and personal identity loss. In a shared key protocol for secure communication, the system receiving a key can use on-chip NIST module to get a measure of the randomness. Based on the NIST test results, the received key can be used or a request can be made for a new key. All systems using a TRNG employ some form of calibration or post-processing to mitigate the effect of physical variation on the statistics of RNG. The post-processing techniques could vary from very lightweight XOR function or von Neumann corrector [28], to robust entropy extraction using HASH functions or AES [29]. While this is a fault tolerant approach to entropy extraction, an on-chip NIST module allows multiple post-processing units to be implemented and then select the appropriate one based on the quality of the TRNG. The other blocks can be powered OFF to reduce leakage power.

Thus, on-chip statistical test suite is imperative for secure computation on lightweight devices with scaling technology and increasing invasive attacks. We propose a lightweight implementation of 6 statistical tests from the NIST test suite. The tests are chosen based on the minimum sample set recommended by NIST and the complexity of storage and computation in terms of ultra-low power implementation. Each of the 6 tests is reduced to obtain a hardware design friendly structure. However, the tests comply with the NIST standard and do not result in a deviation in accuracy of the results. The reduced implementations do not require any complex arithmetic or statistical computation and result in a very low area and power overhead. Although the reduced test set is not as exhaustive as using the entire NIST test suite, it provides an effective layer of security and monitoring for lightweight RNG implementations.

III. REDUCED NIST TEST SUITE

Testing randomness of a bit sequence is as challenging as generating them. Randomness is a relative term and there is no definite metric to quantify it. Depending on the field of application, tools from probability theory are used to develop Statistical Hypothesis Testing (SHT) or Statistical Testing (ST). These tests provide a hypothesis of whether the bit sequence can be considered random or not, with a specific level

of confidence. The Statistical Test Suite standardized by the American National Institute of Standards and Technology (NIST) and the DIEHARD statistical test suite developed by George Marsaglia are two of the most popular statistical test suites used for randomness testing. In this work, we adapt the NIST tests to design a lightweight randomness test block.

The NIST test suite consists of a collection of fifteen statistical tests. Each of these tests hypothetically quantifies a certain aspect of randomness. Given a bit sequence *a* of length *n*, an assumption of randomness, defined as null hypothesis *H(0)*, is fixed. A counter assumption is defined as alternate hypothesis *H(a)* along with a significance level alpha (α). The value of α, also called critical value, is fixed *apriori* in the range of [0.001, 0.01]. The bit sequence from an RNG is analyzed using a specific data processing technique for each test. The output of the data processing stage, variable *X*, is used as one of the inputs to either Complimentary Error Function (*erfc*) or the Upper Incomplete Gamma Function (*igamc*) depending on the test. The output of this stage is called the P-value. This value is compared with the constant significance level α. If the P-value is greater or equal to α, the null hypothesis is accepted with a confidence of 1-α, else is rejected with a confidence of 1-α, Fig. 1.

A summary of the tests in the NIST suite with the null hypothesis for each test is shown in Table 1. Each test in the NIST suite requires a pre-defined minimum sample set of size ranging from 100 bits to 100,000 bits. The size of hardware for computation and data storage required for a test is proportional to the size of the data sample. Since the focus of this work is to realize a test module for ultra-lightweight applications, we choose six of the fifteen NIST tests which are viable for lightweight implementation. The reduced set of NIST tests consist of Frequency Monobit Test, Frequency Block Test, Runs Test, Test for Longest Runs of Ones, Binary Matrix Rank Test and Non Overlapping Template Matching. The reduced set includes the four tests mandated by FIPS 140-1 standard.

As described, the data processing stage of NIST tests consists of either the Complementary Error Function (*erfc*) or the Upper Incomplete Gamma Function (*igmac*). These two functions are the bottleneck in computing the P-value for each test. For a given sample size n, the test is considered to be passed if the P-value is greater or equal to target critical value α. To reduce the computation required, we fix the value of n and compute the range of input *X* to the *erfc* or *igmac* which lead to a P-value greater than the critical value of α. This completely eliminates the need for complex computation of error function or gamma function, thereby significantly reducing the complexity of hardware required to implement the tests. In the proposed lightweight implementation, all computations and test results are quantified in terms of the value of *X* instead of the P-value. Since the value of *X* and the corresponding P-value have a one-to-one mapping, the accuracy of the results is not lost. The partial reconfigurable feature allows the user to set the critical value α depending on the need of the application to make the tests more stringent. Modifying the critical value only varies the bounds for the input to the complex function. The bounds can be set one-time by blowing fuses or can be configured using registers to store the values. Further, all computations are performed serially on the incoming bits from the RNG. This eliminates the need for additional storage devices except for byte-wide shift registers.

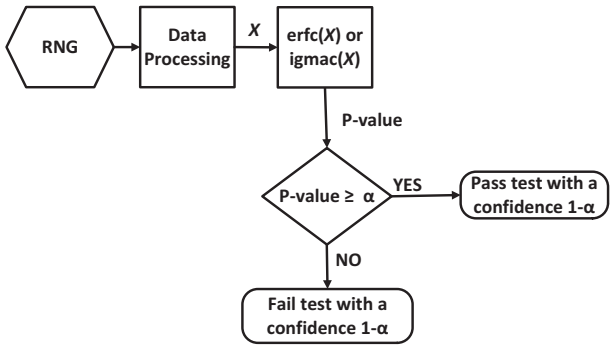

Figure 1: Generic Flow of NIST Tests

TABLE 1: SUMMARY OF NIST SP 800-22 TEST SUITE

Test	Null Hypothesis
Frequency Monobit Test	**Good proportion of 1s and 0s**
Frequency Block Test	**Good proportion of 1s and 0s within N blocks of M-bit**
Runs test	**No clusters of 1s and/or 0s in a sequence.**
Test for the Longest Run of 1s in a Block	**No clusters of 1s and/or 0s in N blocks of M-bit**
Binary Matrix Rank Test	**Low linear dependence between sub-strings, sub-matrices with high rank**
Discrete Fourier Transform (Spectral) Test	Balanced amount of peaks in the frequency domain
Non Overlapping Template Matching Test	**Not too many non-overlapping equal sequences**
Overlapping Template Matching Test	Not too many overlapping equal sequences
Maurer's Universal Statistics Test	Sequence cannot be significantly compressed
Linear Complexity Test	Sequence with long LFSRs
Serial Test	Every m-bit template has equal probability to arise
Approximate Entropy Test	Non-regular occurrence of the same overlapping template
Cumulative Sums Test	Cumulative sum excursion near zero
Random Excursion Test	Random distribution of visits among cycles to eight states
Random Excursion Variant Test	Random distribution of visits among cycles to eighteen states

IV. LIGHTWEIGHT IMPLEMENTATION OF REDUCED NIST TEST SUITE

In this section a detailed description of the lightweight implementation for Frequency Block Test (FBT) is presented. The implementation of the other five tests is also based on similar techniques. For a detailed statistical analysis of each test, interested readers may refer to [13].

The FBT checks for a good proportion of 1's and 0's in blocks of random bits. Given a sample bit stream of length n, the test breaks the bit stream into N blocks of size M bits. For each block the distance of 1's proportion (or 0's) over M-bit from 0.5, the expected mean value is computed. The partial results are squared and accumulated; the resulting number is multiplied by a constant and used as one input to igmac. The gamma function returns the P-value that is compared with the pre-defined significance level, α. If the P-value is greater or equal to α, the bit stream is predicted to be random with a confidence of 1-α. The above steps can be reduced to accommodate a lightweight hardware implementation as follows:

$$Let\ n = 128\ (Total\ number\ of\ bits)\ and$$

$$M = 8\ (Number\ of\ bits\ per\ block)$$

$$N = \frac{n}{M} = 16\ is\ the\ Number\ of\ non-overlapping\ blocks$$

$$Assuming\ \alpha = 0.01, the\ test\ passes\ for\ \ P-value \geq 0.01,$$

$$In\ the\ Frequency\ Block\ Test, P-value = \ igamc\left(\frac{N}{2}, \frac{\chi^2}{2}\right)$$

$$Hence, igamc\left(\frac{N}{2}, \frac{\chi^2}{2}\right) \geq 0.01$$

$$where\ \chi^2\ is\ the\ output\ of\ the\ data\ processing\ stage$$

$$Computing\ the\ inverse\ of\ igmac(),\quad \boldsymbol{\chi^2 \leq 16}$$

$$In\ the\ Frequency\ Block\ Test, X = \chi^2 = 4 \times M \times \sum_{i=1}^{N}\left(\pi_i - \frac{1}{2}\right)^2$$

$$Hence,\quad 4 \times M \times \sum_{i=1}^{N}\left(\pi_i - \frac{1}{2}\right)^2 \leq 16$$

$$where\ \pi_i\ is\ the\ ratio\ of\ 1's\ in\ each\ of\ the\ N\ blocks.$$

If a counter c is used to count the number of 1's in each block,

$$4 \times M \times \sum_{i=1}^{N}\left(\frac{c}{8} - \frac{1}{2}\right)^2 \leq 16$$

$$\boldsymbol{\sum_{i=1}^{N}(c-4)^2 \leq 32\ ,\ is\ the\ condition\ for\ passing\ FBT}$$

The complex computation to estimate the P-value and hence deem a bit stream to pass/fail the Frequency Block Test can be reduced to a series of counter, offset calculation and squaring operations. Since the number of bits in the sample, n and the block size M are positive whole numbers; all computations involve whole numbers allowing a simpler combinatorial logic based implementation. A similar calculation is performed for the other five NIST tests to reduce the computation and design a lightweight implementation. Further, the NIST module can be turned ON intermittently to reduce power overhead.

By calculating the bounds of input to *igmac* function, the FBT is optimized to a series of count, accumulate and compare operations as shown in Fig. 2. The complex *igmac* function is avoided enabling a lightweight implementation. The hardware implementation for each stage of FBT is as show in Fig. 3.

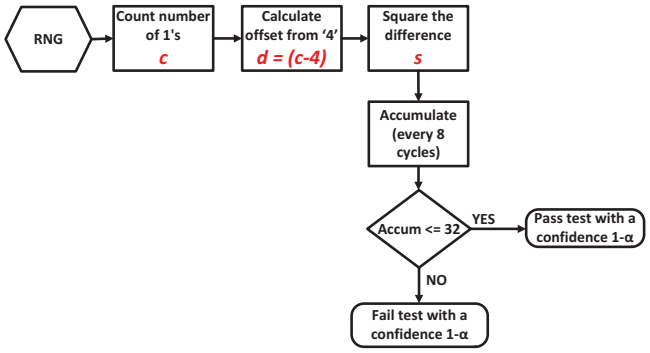

Figure 2: Flow of optimized Frequency Block Test

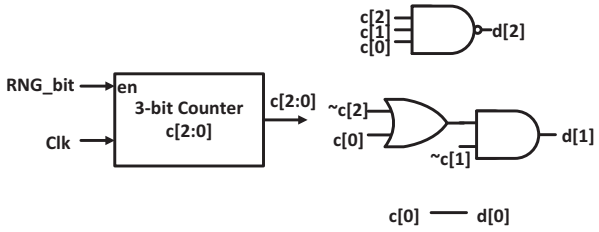

Figure 3a: Enable counter (*c*) and Variance Calculator (*d*)

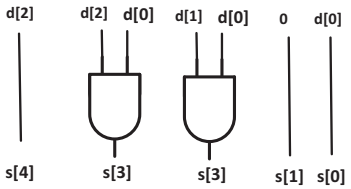

Figure 3b: Squaring the variance (*s*)

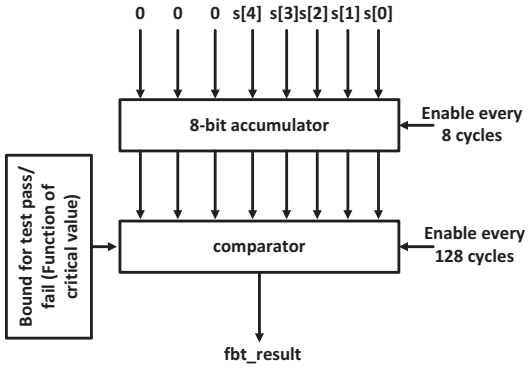

Figure 3c: Accumulator and Comparator

Figure 3: Digital logic for lightweight FBT implementation

TABLE 2: INPUT TO ERFC/IGMAC FOR DIFFERENT CRITICAL VALUE SETTINGS

Test	α = 0.001	α = 0.0025	α = 0.0040	α = 0.0055	α = 0.0070	α = 0.0085	α = 0.010
Frequency Mono-bit Test	2.3268	2.1378	2.0352	1.9631	1.9070	1.8608	1.8214
Frequency Block Test (*M=8)*	19.6262	18.2279	17.4898	16.9802	16.5888	16.2698	16.0000
Runs Test	2.3268	2.1378	2.0352	1.9631	1.9070	1.8608	1.8214
Test for Longest Runs of Ones	8.1331	7.1602	6.6582	6.3168	6.0574	5.8481	5.6724
Non Overlapping Template Matching Test $(m = 9)$	13.0622	11.8872	11.2726	10.8507	10.5280	10.2660	10.0451
Binary Matrix Rank Test $(n = 608\ M = Q = 4)$	6.9078	5.9915	5.5215	5.2030	4.9618	4.7677	4.6052

Similar to Frequency Block Test, other NIST tests are also reduced to facilitate optimum hardware implementation. All the tests operate on each incoming bit from the RNG serially, thereby minimizing additional hardware to store the bits. Only for the Non Overlapping Template Matching test and the Binary Matrix Rank test, a 10 bit and 8 bit shift registers are used respectively to store the previous bits. The counters and control logic is shared across different tests to further optimize area and power.

The partial reconfigurable feature of the reduced NIST test module provides the flexibility of choosing a different critical value α for each test based on the requirement of the platform/application. The reconfigurable *bound registers* in each test module allows appropriate value of α to be set as the threshold critical value for test pass/fail. The calculated upper bounds for the inputs to *erfc*/*igmac* functions for each test for different critical values are shown in Table 2. Although the calculated input bounds have fractional values, the final hardware counter/comparator bounds will be whole numbers.

V. LOGIC IMPLEMENTATION AND RESULTS

The proposed lightweight implementation for the reduced NIST test suite consisting of six tests was designed in Verilog and verified for functionality using ModelSim. The designs were synthesized in Synopsys Design Compiler using the 45nm SOI NCSU/OSU Open Source Standard Cell Library. The synthesized designs were optimized for a cycle time of 0.5ns (2GHz).

The area and power numbers for each 128 bit test are as shown in Fig. 4 and Fig. 5 respectively. The lightweight implementation results in a synthesized area ranging from 240μm^2 to 460μm^2 for each test. The shared control logic and counters reduce the overall implementation area. The common counter and control logic consume an area of around 200um^2 resulting in the overall NIST module area of 1926μm^2. This translates to 1026 NAND gates equivalent in 45nm technology. The active power for each test is of the order of 0.4mW to 0.8mW. All the tests are designed to operate in parallel, resulting in an overall active power of 3.75mW for the NIST module operating at 2GHz. The overall cell leakage is ~10.5μW which is 0.28% of the total power.

Since the target applications include passively powered and battery operated devices like RFIDs and smart cards, energy/bit is an important metric. The 128-bit reduced NIST module operating on 2Gbps consumes 1.87 pJ/bit.

The proposed implementation is scalable to larger number of bit samples as well. Depending on the number of bits *n* and block sizes, the bounds for each test varies.

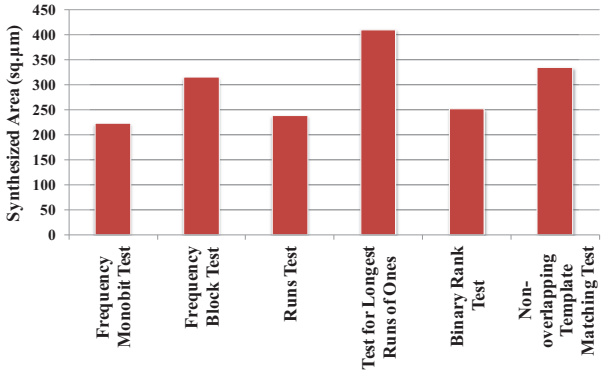

Figure 4: Synthesized area of lightweight NIST test implementation

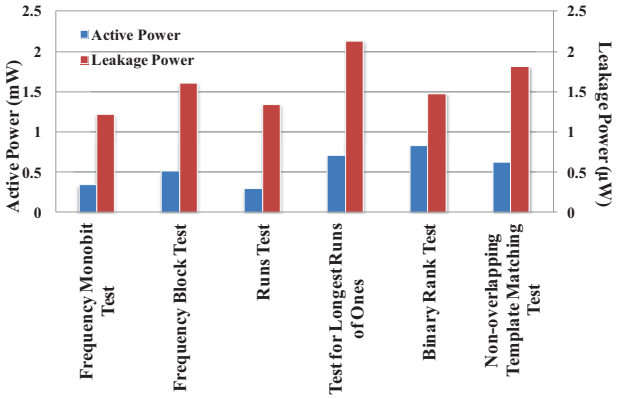

Figure 5: Active and leakage power of lightweight NIST test implementation

The Binary Matrix Rank Test and Non Overlapping Template Test are implemented with a minimum bit sample size greater than 512 bits. The optimized tests can be scaled for larger bit sequence range at the cost of increased area and power. The area, power and energy/bit for 256-bit and 512-bit implementations are as shown in Table 3.

TABLE 3: AREA, POWER AND ENRGY/BIT FOR 256 BIT AND 512 BIT IMPLEMENTATIONS IN 45NM TECHNOLOGY

Bit length	256 bits	512 bits
Area (μm^2)	2394	2787
Power (mW)	4.03	4.37
Energy/bit @ 2Gbps (pJ/bit)	2.01	2.18

VI. CONCLUSIONS AND FUTURE WORK

A lightweight implementation of reduced set of NIST randomness test suite is proposed in this work. The implementation of the six NIST tests is based on optimized calculations to bypass the computation intensive *erfc* and *igmac* functions. The modified critical value is used as a bound for setting the test pass/fail threshold. The complex statistical computations are converted to a series of count, add and compare operations implemented using combinatorial logic. The design is further optimized for a fixed bit sample size and share global counters and control signals. A partial reconfiguration feature allows the critical value to be changed in the form of modified bounds for each test. The tests operate serially on each incoming bit from the RNG, thereby avoiding additional hardware for storage. The design for 128-bit tests has a synthesized area of $1926\mu m^2$ for an optimized cycle time of 0.5ns. The six tests operate in parallel consuming an active power of 3.75mW and leakage power of 10.5µW. As part of the future work, we propose to explore lightweight implementation of other statistical tests from NIST and DIEHARD test suites and expand to second level tests for analyzing the distribution of P-values.

REFERENCES

[1] B. Sunar, W. J. Martin, and D. R. Stinson, "A Provably Secure True Random Number Generator with Built-In Tolerance to Active Attacks," *IEEE Transactions on Computers*, vol. 56, no. 1, Jan. 2007.

[2] T. Amaki, M. Hashimoto, and T. Onoye, "An oscillator-based true random number generator with jitter amplifier," in *2011 IEEE International Symposium on Circuits and Systems (ISCAS)*, 2011.

[3] N. Bochard, F. Bernard, and V. Fischer, "Observing the Randomness in RO-Based TRNG," in *International Conference on Reconfigurable Computing and FPGAs, 2009. ReConFig '09*, 2009, pp. 237 –242.

[4] M. Jessa and L. Matuszewski, "Enhancing the Randomness of a Combined True Random Number Generator Based on the Ring Oscillator Sampling Method," in *2011 International Conference on Reconfigurable Computing and FPGAs (ReConFig)*, 2011,.

[5] F. Pareschi, G. Setti, and R. Rovatti, "Implementation and Testing of High-Speed CMOS True Random Number Generators Based on Chaotic Systems," *IEEE Transactions on Circuits and Systems* Dec. 2010.

[6] W. Chen, *et. al*, "A 1.04 #x00B5;W Truly Random Number Generator for Gen2 RFID tag," in *Solid-State Circuits Conference, 2009. A-SSCC 2009. IEEE Asian*, 2009, pp. 117 –120.

[7] S. Srinivasan, *et. al*, "2.4GHz 7mW all-digital PVT-variation tolerant True Random Number Generator in 45nm CMOS," in *IEEE Symposium on VLSI Circuits (VLSIC)*, 2010.

[8] C. Tokunaga, D. Blaauw, and T. Mudge, "True Random Number Generator With a Metastability-Based Quality Control," *IEEE Journal of Solid-State Circuits*, vol. 43, no. 1, pp. 78 –85, Jan. 2008.

[9] V. B. Suresh and W. P. Burleson, "Robust metastability-based TRNG design in nanometer CMOS with sub-vdd pre-charge and hybrid self-calibration," in *Intn'l Symposium on Quality Electronic Design* 2012.

[10] D. E. Holcomb, W. P. Burleson, and K. Fu, "Power-Up SRAM State as an Identifying Fingerprint and Source of True Random Numbers," *IEEE Transactions on Computers*, vol. 58, no. 9, pp. 1198 –1210, Sep. 2009.

[11] C. Pyo, S. Pae, and G. Lee, "DRAM as source of randomness," *Electronics Letters*, vol. 45, no. 1, pp. 26 –27, 2009.

[12] C.-Y. Huang, W. C. Shen, Y.-H. Tseng, Y.-C. King, and C.-J. Lin, "A Contact-Resistive Random-Access-Memory-Based True Random Number Generator," *IEEE Electron Device Letters*, Aug. 2012.

[13] National Institute of Standards and Technology, "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications." Apr-2010.

[14] G. Marsaglia, "DIEHARD: a battery of tests of randomness", http://www.stat.fsu.edu/pub/diehard/.

[15] F. Pareschi, R. Rovatti, and G. Setti, "On Statistical Tests for Randomness Included in the NIST SP800-22 Test Suite and Based on the Binomial Distribution," *IEEE Transactions on Information Forensics and Security*, vol. 7, no. 2, pp. 491 –505, Apr. 2012.

[16] A. Suciu, K. Marton, I. Nagy, and I. Pinca, "Byte-oriented efficient implementation of the NIST statistical test suite," in *IEEE International Conference on Automation Quality and Testing Robotics*, 2010.

[17] A. Suciu, I. Nagy, K. Marton, and I. Pinca, "Parallel implementation of the NIST Statistical Test Suite," in *2010 IEEE International Conference on Intelligent Computer Communication and Processing (ICCP)*, 2010.

[18] K. Udawatta, *et. al*, "Test and validation of a non-deterministic system - True Random Number Generator," in *High Level Design Validation and Test Workshop, 2008.*

[19] D. Ho oleanu, O. Cre , A. Suciu, T. Gyorfi, and L. V cariu, "Real-Time Testing of True Random Number Generators Through Dynamic Reconfiguration," in *2010 13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools (DSD)*, 2010.

[20] F. Pareschi, R. Rovatti, and G. Setti, "On the approximation errors in the frequency test included in the NIST SP800-22 statistical test suite," in *IEEE Asia Pacific Conference on Circuits and Systems, 2008.*

[21] F. Pareschi, R. Rovatti, and G. Setti, "Second-level NIST Randomness Tests for Improving Test Reliability," in *IEEE International Symposium on Circuits and Systems, 2007. ISCAS 2007*, 2007, pp. 1437 –1440.

[22] F. Pareschi, R. Rovatti, and G. Setti, "Second-level testing revisited and applications to NIST SP800-22," in *18th European Conference on Circuit Theory and Design, 2007. ECCTD 2007*, 2007, pp. 627 –630.

[23] J. Kelsey, *et. al*, "Cryptanalytic Attacks on Pseudorandom Number Generators," in *Fast Software Encryption, Fifth International*, 1998.

[24] A. T. Markettos and S. W. Moore, "The Frequency Injection Attack on Ring-Oscillator-Based True Random Number Generators," in *Proceedings of the 11th International Workshop on Cryptographic Hardware and Embedded Systems*, Berlin, Heidelberg, 2009.

[25] T. Kasper, M. Silbermann, and C. Paar, "All You Can Eat or Breaking a Real-World Contactless Payment System," in *Financial Cryptography and Data Security*, R. Sion, Ed. Springer Berlin Heidelberg, 2010.

[26] J. Melià-Seguí, J. Garcia-Alfaro, and J. Herrera-Joancomartí, "A Practical Implementation Attack on Weak Pseudorandom Number Generator Designs for EPC Gen2 Tags," *Wireless Pers Commun*, 2011.

[27] P. Bayon, *et. al*, "Contactless Electromagnetic Active Attack on Ring Oscillator Based True Random Number Generator," in *Constructive Side-Channel Analysis and Secure Design*, W. Schindler and S. A. Huss, Eds. Springer Berlin Heidelberg, 2012, pp. 151–166.

[28] V. B. Suresh and W. P. Burleson, "Entropy extraction in metastability-based TRNG," in *2010 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)*, 2010, pp. 135 –140.

[29] M. Hamburg, *et.al*, "Analysis of Intel's Ivy Bridge Digital Random Number Generator." Cryptography Research, Inc., Mar-2012.

[30] A. J. Menezes, S. A. Vanstone, and P. C. V. Oorschot, *Handbook of Applied Cryptography*, CRC Press, Inc., 1996.