True Random Number Generation with the Shift-register Reconvergent-Fanout (SiRF) PUF

Nafis Irtija
ECE
University of New Mexico
Albuquerque, USA
nafis@unm.edu

Eirini Eleni Tsiropoulou
ECE
University of New Mexico
Albuquerque, USA
eirini@unm.edu

Cyrus Minwalla
Financial Technology Research Group
Bank of Canada
Ottawa, ON, Canada
cminwalla@bank-banque-canada.ca

Jim Plusquellic
ECE
University of New Mexico
Albuquerque, USA
jimp@ece.unm.edu

***Abstract*—True Random Number Generator (TRNG) is an important hardware security primitive for system security. TRNGs are capable of providing random bits for initialization vectors in encryption engines, for padding and nonces in authentication protocols and for seeds to pseudo random number generators (PRNG). A TRNG needs to meet the same statistical quality standards as a physical unclonable function (PUF) with regard to randomness and uniqueness, and therefore one can envision a unified architecture for both functions. In this paper, we investigate a FPGA implementation of a TRNG using the Shift-register Reconvergent-Fanout (SiRF) PUF. The SiRF PUF measures path delays as a source of entropy within a engineered logic gate netlist. The delays are measured at high precision using a time-to-digital converter, and then processed into a random bitstring using a series of linear-time mathematical operations. The SiRF PUF algorithm that is used for key generation is reused for the TRNG, with simplifications that improve the bit generation rate of the algorithm. This enables the TRNG to leverage both fixed PUF-based entropy and random noise sources, and makes the TRNG resilient to temperature-voltage attacks. TRNG bitstrings generated from a programmable logic implementation of the SiRF PUF-TRNG on a set of FPGAs are evaluated using statistical testing tools.**

***Index Terms*—True Random Number Generator, Physical Unclonable Function, FPGA**

I. Introduction

True random number generators generate a stream of bits from measurements of a physical and stochastic source of entropy. The physical source is often analog in nature and requires an ancillary circuit and post-processing algorithm to convert from analog signal measurements to discrete bit values of 0 and 1. In many cases, the physical source is not truly random, e.g., it exhibits a bias to generate more 0's than 1's or produces sequences with specific patterns, and therefore a post-processing algorithm is required to distill the raw bit sequence until sufficiently high levels of entropy are achieved. The distillation process typically consumes more bits than it emits, which reduces the bit generation rate and increases the size of the implementation. Moreover, most TRNGs provide no protection against adversarial attacks, which attempt to manipulate the system or environment to inject bias and cause non-random bit sequence behaviors.

A physical unclonable function (PUF) is a second, related hardware security primitive responsible for generating and regenerating secret keys and bitstrings for encryption and authentication protocols. PUFs share many of the same attributes as a TRNG from the perspective of the generated bitstrings. In particular, both need to produce bit sequences that are truly random across the emitted stream and both need to produce bit sequences that are unique with respect to the bit sequences generated by other devices.

TRNGs and PUFs differ in two primary ways. First, the bit generation rate requirement for a PUF is typically relaxed (slower) than the rate required from a TRNG. Second, the PUF needs to reproduce a fixed bit sequence on demand and potentially under adverse environmental conditions, while a TRNG is charged with just the opposite, i.e., it should never reproduce a fixed bit sequence. This second difference imposes a fundamental change on the source of entropy leveraged by TRNGs and PUFs. In particular, PUFs generate a random bit sequence from a baked-in (fixed) source of entropy within the device, e.g., from variations in transistor threshold voltages. TRNGs, on the other hand, typically leverage noise as a source of entropy and are not designed to incorporate a fixed source. This is true because the amount of entropy available from a fixed source of entropy is finite and therefore such fixed sources do not have a significant impact on the size of the entropy pool.

In this paper, we propose a unified architecture that incorporates both a PUF and TRNG. The proposed PUF leverages variations in the delay of paths, measured through an engineered network of logic gates, as a source of entropy. The Shift-register, Reconvergent-Fanout (SiRF) PUF (SiRF) PUF utilizes shift registers and MUXs to create structural diversity in the placement and routing of gate-level netlists, and reconvergent-fanout to add uncertainty regarding which logic gates actually define the paths whose delays are utilized in the bitstring generation process. Post-processing techniques including a temperature-voltage calibration method are utilized to significantly improve uniqueness and error-free bitstring regeneration by reducing undesirable environmentally-induced changes in delay, e.g., delay variations introduced by changes in temperature or supply voltage, as well as delay variations that occur globally to all components on the device, a.k.a. chip-to-chip process variations.

The SiRF TRNG is derived directly from the PUF architec-

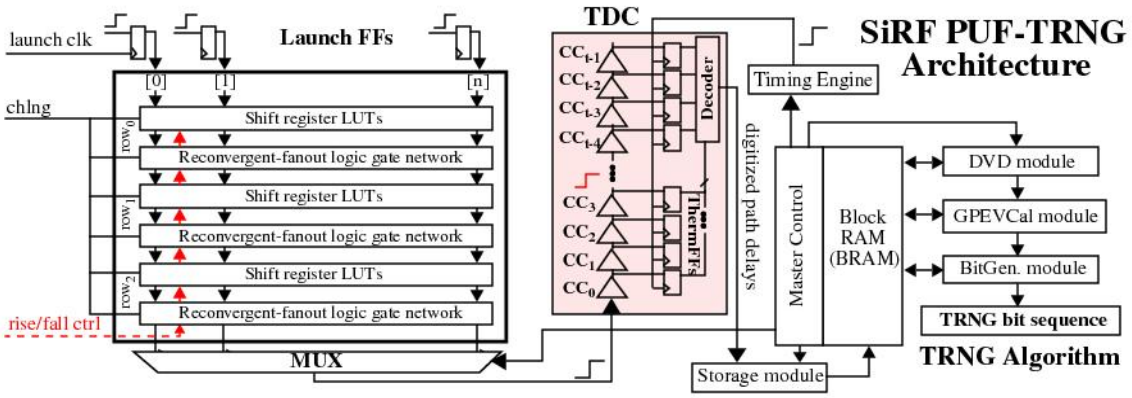

Fig. 1. Block diagram of the shift-register, reconvergent-fanout (SiRF) PUF.

ture and algorithm and is the focus of the analysis in this paper. With the TRNG mode switch enabled, operations related to the transfer of challenges, the collection of multiple path delay samples and other algorithmic processes related to improving reliability for bitstring reproduction are disabled and instead, random challenges are generated and applied to test random paths within the network. A time-to-digital (TDC) is used to obtain high-precision measurements of the path delays. The digitized path delay representations are processed through the SiRF algorithm, which pairs path delays to create differences and calibrates them for temperature-voltage environmental conditions. The low-order bit of the calibrated path delay differences is used as the TRNG bit sequence. Therefore, the SiRF TRNG utilizes both a fixed and random source of entropy, and actively removes temperature-voltage effects on the measured path delays. These characteristics are unique and novel features of the SiRF TRNG.

The following are the main contributions of this paper:

- A compact system architecture implementing both TRNG and PUF hardware security primitives is proposed.
- The TRNG is designed to leverage both a random and fixed source of entropy, increasing the statistical quality of the generated bit sequences.
- The TRNG makes use of the PUF's temperature-voltage calibration algorithm as a means of defending against temperature-voltage attacks.
- Bit sequences from a set of FPGAs are evaluated using the statistical tools.

II. Previous Work

The authors of [1] propose a combined PUF-TRNG architecture that utilizes an array of ring oscillators (ROs) and jitter measurements as a source of entropy. The architecture does not incorporate any protections against temperature-voltage attacks, and uses noise as a sole source of entropy for the TRNG. This work was built on by the authors in [2] who propose a calibration process to improve the performance of the original design. Notably the underlying RO architecture remains susceptible to machine-learning attacks. The authors in [3] designed a unified PUF built on magnetic RAM (MRAM) based on variations in the spin-transfer torque, showcasing zero-bit error rates, however, the PUF itself was not constructed and the proposed error performance not experimentally verified. Khan et al. [4] proposed a similar MRAM construction, where a fabricated version was extensively tested. Larimian et al. [5] proposed a unified PUF and TRNG architecture based on spatial and temporal current variations in embedded flash memory. The PUF was constructed on a (Global Foundries) 55 nm process and successfully thwarted machine learning attacks. In recent work, the authors in [6] developed and constructed an SRAM-based unified PUF and TRNG architecture. While fast and scalable, it has a relatively high LSB BER of 2.0%, which under temperature gradient testing degrades to 4.8%. A PUF-TRNG hybrid architecture is proposed [7] that addresses environmental variations but does so using experimental trials on a 1-bit representation of path delay differences. In contrast, the SiRF PUF computes path delay differences from an exponentially large number of paths in a non-identically designed netlist, leveraging both fixed and random entropy sources, and applying a distribution-based compensation technique to remove environmental variation effects from a set of multi-bit digitized path delays. Moreover, the run time of the algorithm in [7] is unpredictable and bit rates are not reported.

III. SiRF PUF-TRNG Architecture

A block diagram of the SiRF PUF architecture is shown in Fig. 1. The physical design is constructed as a set of modules, arranged as a set of rows as shown on the left side of the figure. The input challenge configures a network of shift-registers, logic gates and MUXs within the modules to create a set of paths between the Launch FFs on the inputs and the MUX on the output. The number of possible paths through the logic gate network (which represents the source of entropy) is more than 10 million, each testable with a rising or falling transition on the inputs, for a total of more than 20 million paths.

Logic signal transitions are introduced by a set of $n = 32$ *Launch FFs* shown along the top and path delays are measured, one-at-a-time, by selecting a path output using the MUX shown along the bottom of the figure. The path selected for timing is directed to the input of a time-to-digital (TDC) converter shown in the center of the figure. The TDC consists

of a sequence of 32 Carry4 elements in the Xilinx FPGA, with each of the 128 carry chain buffers connected to a corresponding FF. The Timing Engine component of the SiRF PUF-TRNG utilizes a state machine to apply a sequence of launch-capture tests to the logic gate network until a valid digital representation of the path delay is obtained from the TDC FFs (similar to the method proposed in [8]).

A. SiRF TRNG Algorithm

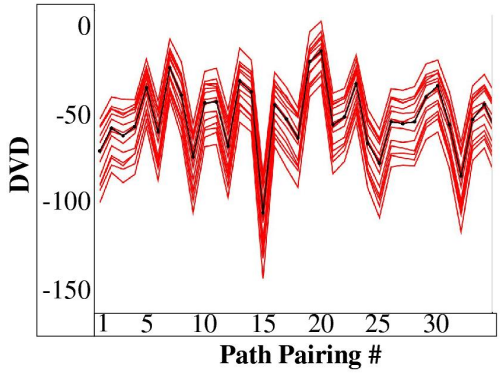

Fig. 2. A set of DVD created by the difference module of the TRNG algorithm.

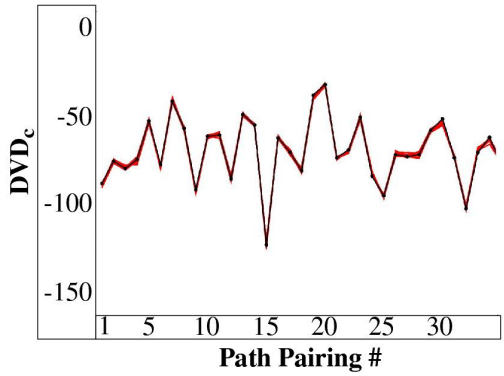

Fig. 3. A set of DVD_c created by the global-process-environmental-variation (GPEV) calibration module of the TRNG algorithm.

The path delays are discretized by the TDC at a resolution of approx. 18 ps and stored as digital values, called delay values or DV, in a Block RAM (BRAM) for subsequent processing into bit sequences by the SiRF TRNG algorithm. The BRAM and post-processing modules associated with the SiRF TRNG algorithm are shown on the right side of Fig. 1. The algorithm selects pairs of DV and creates differences, called DVD, and then applies a global-process-environmental-variation (GPEV) calibration method to generate DVD_c, which reduces undesirable environmentally-induced changes in the path delays.

The impact of the GPEV calibration method is shown for a set of path pairings in Figs. 2 and 3. The DV are measured from an FPGA at a set of temperature-voltage (TV) corners given by all combinations of temperatures 25^oC, 0^oC, -40^oC, 85^oC and 100^oC and voltages $1.00V$, $1.05V$ and $0.95V$, for a total of 15 TV corners. The difference module selects pairs of DV to create DVD which are plotted in Fig. 2. The black curve represents the DVD under nominal conditions, 25^oC, $1.00V$, while the red curves represent the DVD at each of the remaining 14 TV corners. The impact of adverse TV conditions on the DVD is depicted by vertical excursions of the red points around the black points. In contrast, the nearly superimposed DVD_c curves shown in Fig. 3 illustrate that very little delay variation exists after the GPEV calibration process. Although the SiRF TRNG does not require this type of calibration to generate random bit sequences, the application of GPEV defeats any type of temperature-voltage attack that is designed to purposely bias the SiRF TRNG random bitstring generation process.

The last component of the SiRF TRNG algorithm, labeled BitGen module in Fig. 1 generates the bit sequence using the low order bit of DVD_c, which is influenced by both the fixed entropy associated with the calibrated path pairing difference delay and measurement noise.

IV. TRNG Implementation

The SiRF TRNG generates a random bit sequence by carrying out the sequence of steps shown in Fig. 4. The diagram shows an SoC FPGA as the host system but other FPGA-only implementations are also possible. The processor side (PS) of the FPGA is shown along the top of the figure and the programmable logic (PL) side along the bottom. The following sequence of operations are carried out to generate a bit sequence of 5,120 bytes (40,860 bits).

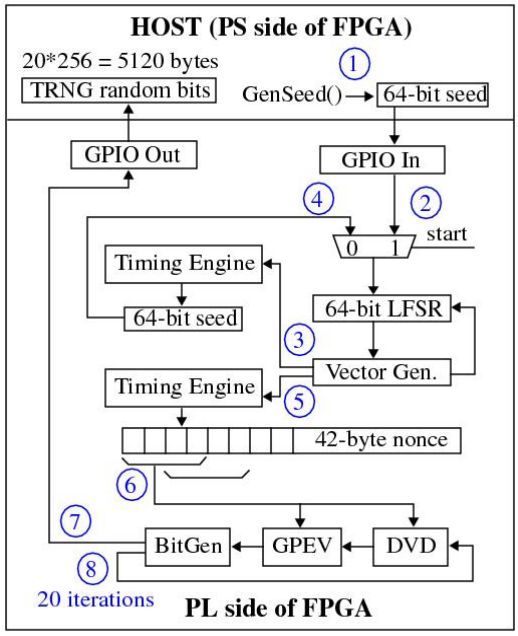

Fig. 4. TRNG flow diagram, showing the sequence of operations carried out to generate 5,120 random nonce bytes.

1) A C program running in the PS side generates a random 64-bit seed, i.e., using srand() or /dev/urandom.
2) The PS side asserts the *start* signal and transfers the seed through a memory-mapped GPIO register to a 64-bit LFSR implemented in the PL side. The *start* signal is then deasserted.

3) The *Vector Gen.* module is started which generates 64-bit pseudo-random numbers that are used as the 2-vector sequences to the SiRF TRNG's Timing Engine. The Timing Engine generates a random 64-bit LFSR seed by XOR-distilling the least significant bit (LSB) of 12 consecutive path delays (DV) to generate each LFSR bit.
4) The new 64-bit LFSR seed is transferred to the LFSR.
5) The Timing Engine is started again to measure the delays of a new set of 4096 paths. The DV are transferred to the BRAM (not shown but see Fig. 1) concurrently with the generation of 42-bytes of a nonce, generated as in the previous step by XOR-distilling the LSBs of DV.
6) The remaining steps of the SiRF TRNG algorithm are run in which a 32-bit chunk of the nonce is used to set parameters of the DVD and $GPEV$ modules. Although the details are omitted, the DVD module incorporates two 11-bit LFSRs to create pairings of DV and the $GPEV$ module uses 10 bits to control the transformation of DVD to DVD_c.
7) The 256-byte output of the $BitGen$ is transferred to the C program through the GPIO Out register.
8) The SiRF TRNG algorithm iterates 20 times using a sequence of overlapping 32-bit chunks from the nonce register as parameters to the DVD and $GPEV$ modules in each iteration.

It should be noted that the seed and LFSR have no bearing on the specific bit sequence produced, and instead are used only as the initial challenge and for uniform challenge selection. The seed is not secret and, in this work, is a fixed constant for all repeated applications of the SiRF TRNG algorithm.

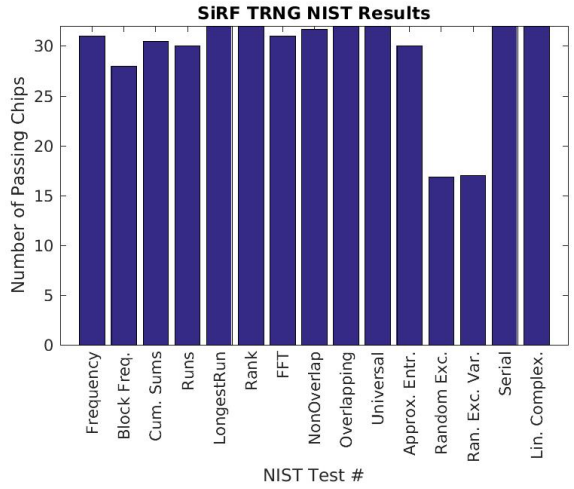

Fig. 5. NIST test results for 32 chips using bit sequences of length 1,024,000 bits. The minimum number of FPGAs that must pass the test before NIST classifies the overall test as a pass is 29 except for Random Exc. and Ran. Exc. Var. which require a value of at least 15 to pass. Only Block Freq. fails but 28 of the FPGAs pass, so the test failed by 1 FPGA.

V. Experimental Results

The bit sequences generated by a SiRF TRNG implementation on a set of 32 Xilinx Zynq 7010 SoC FPGAs are evaluated for statistical quality in this section. The average bit sequence generation rate is approx. 30 KBytes/sec.

The results from applying the NIST statistical test suite [9] to TRNG bit sequences of length 1,024,000 collected from the 32 FPGAs are shown in Fig. 5. The names of the 15 NIST statistical tests are given along the x-axis while the number of passing FPGAs are given along the y-axis. The NIST statistical tools apply the named test to the bit sequence for each FPGA and then compares the computed test statistics to a threshold. The default α value of 0.01 is used as the significance level. Although NIST allows some FPGAs to fail individual tests, it requires a minimum number of 29 FPGAs to pass the test in order to classify the overall test as passed. The TRNG bit sequences pass all tests except for *Block Frequency*, which fails by only one chip with 28 FPGAs passing.

Inter-chip hamming distance (HD) measures the degree of similarity between two bit sequences from different chips by computing the number of bits that are different in the two bit sequences. The ideal value is 50%, i.e., exactly half of the bits are different. We compute the Inter-chip HD for each of the (32*31)/2 = 496 bit sequence pairings, and then compute an overall average. The average value is 49.999% which is very close to the ideal value.

VI. Conclusion

A compact unified system architecture that incorporates both a PUF and TRNG is proposed in this paper. The TRNG is designed to leverage both random noise sources and a persistent (fixed) source of entropy that is associated with the PUF. The bit sequence generation algorithm also carries out temperature-voltage calibration to protect against temperature-voltage attacks designed to add bias to the generated bit sequences. The bit sequences from 32 FPGAs are subjected to a suite of statistical analysis tools, and are shown to exhibit high, cryptographic-level quality.

References

[1] A. Maiti, R. Nagesh, A. Reddy, and P. Schaumont, "Physical unclonable function and true random number generator: A compact and scalable implementation," in *GLSVLSI*, 2009, p. 425–428.

[2] C. Martínez-Gómez and I. Baturone, "Calibration of ring oscillator puf and trng," in *ECCTD*, 2020, pp. 1–4.

[3] E. I. Vatajelu, G. Di Natale, and P. Prinetto, "Security primitives (puf and trng) with stt-mram," in *VTS*, 2016, pp. 1–4.

[4] M. N. I. Khan, C. Y. Cheng, S. H. Lin, A. Ash-Saki, and S. Ghosh, "A morphable physically unclonable function and true random number generator using a commercial magnetic memory," in *ISQED*, 2020, pp. 197–197.

[5] S. Larimian, M. R. Mahmoodi, and D. B. Strukov, "Lightweight integrated design of puf and trng security primitives based on eflash memory in 55-nm cmos," *Trans. on Electron Devices*, vol. 67, no. 4, pp. 1586–1592, 2020.

[6] S. Taneja, V. K. Rajanna, and M. Alioto, "36.1 unified in-memory dynamic trng and multi-bit static puf entropy generation for ubiquitous hardware security," in *ISSCC*, vol. 64, 2021, pp. 498–500.

[7] G. E. S. Charles W. O'Donnell and S. Devadas, "Puf-based random number generation," in *MIT CSAIL CSG TM 481*, 2004, pp. 1–4.

[8] D. Owen Jr., D. Heeger, C. Chan, W. Che, F. Saqib, M. Areno, and J. Plusquellic, "An autonomous, self-authenticating, and self-contained secure boot process for field-programmable gate arrays," *Cryptography*, vol. 2, no. 3, 2018.

[9] (2010) A statistical test suite for random and pseudorandom number generators for cryptographic applications.