HELP: A Hardware-Embedded Delay PUF

Jim Aarestad[1], Philip Ortiz[1], Dhruva Acharyya[2] and Jim Plusquellic[1]
jaarestad@ece.unm.edu, pgortiz@sandia.gov, Dhruva.Acharyya@advantest.com and jimp@ece.unm.edu
[1]Department of ECE, University of New Mexico, [2]Advantest Corporation

Abstract: *The use of embedded secret information such as keys for cryptographic applications, unique identifiers for authentication, and activation of on-chip features is becoming increasingly commonplace in ASICs and FPGAs. The generation of these secret bitstrings using physical unclonable functions, or PUFs, offers distinct advantages over conventional, e.g., EPROM-based methods, in several ways, including eliminating the need to store the bitstring into, and the cost of, a specialized non-volatile memory, and increasing the number of random bits. This paper presents a new PUF called the Hardware-Embedded Delay PUF, or HELP. HELP leverages the natural variations that occur in the path delays of a core macro on a chip to create a unique, stable, and random bitstring of virtually any length. We evaluate several quality statistical metrics of HELP on 29 FPGA boards across a temperature range of 0 to 70°C, and propose an error-avoiding scheme that provides high probability against bit flips.*

Keywords: Hardware security, unique identifier, process variations, physical unclonable function, path delay measurement.

1. Introduction

Physical unclonable functions (PUFs) have emerged as useful mechanisms for generating random numbers for several security-related applications. PUFs utilize the random variations in physical properties of chips to differentiate one chip from another, and are designed to be difficult or impossible to duplicate, even by the manufacturer. While process variations are effectively impossible to control or eliminate, they can be measured. The specific varying properties exploited by the PUF can differ from one PUF design to another. However, common sources of parametric variation include propagation delay, metal resistance, transistor drive strength, and mismatches between complementary transistors. A PUF measures these variations and compares them to generate a bitstring.

The quality of the bitstrings produced by a PUF is measured using several statistical metrics. Three criteria, however, must be met for a PUF to be effective for applications such as encryption: 1) the bitstrings produced for each chip must be sufficiently *unique* to distinguish each chip from every other, 2) the bitstrings must be *random*, making them difficult for an adversary to model and predict, and 3) the bitstring for any one chip must be *stable* over time and across varying environmental conditions.

Process variations are increasingly problematic for semiconductor manufacturers as they move toward more advanced technology nodes and smaller feature sizes. Dopant densities, photolithographic features, and planarization consistency are all examples of manufacturing processes that become harder to control and predict as geometries are reduced. Engineers are working to develop techniques for mitigating these variations. However, even given these improvements, variations are increasing from one technology node to the next. While this creates a challenge for manufacturers, this added variability ensures that PUFs will remain relevant and that advances in PUF designs will continue into the foreseeable future.

In this paper, we present a new PUF, called HELP, that is based upon path delay variations. The novel features that differentiate HELP from other delay-based PUFs include: 1) the capability of comparing paths of widely differing lengths, 2) eliminating the need for specially designed, layout-dependent delay elements that impose a high area cost while providing a relatively small amount of entropy, 3) a minimally invasive design with low area and performance impact, and 4) a hardware-embedded PUF engine requiring no external testing resources. HELP is further differentiated by the large number of paths typically found in logic macros such as the Advanced Encryption Standard (AES). This large source of entropy allows HELP to generate large bitstrings, despite being extremely conservative in the paths selected for bit generation. The large availability of paths also enables unique opportunities for achieving bit stability and avoiding errors.

To prove this PUF concept, and to demonstrate its effectiveness, we have designed a complete, functional FPGA-based implementation of this PUF and validated it on a set of 29 Xilinx Virtex2Pro ("V2Pro") FPGA boards. We present the results of that experimental work, and evaluate the statistical characteristics of the resulting bitstrings.

2. Related Work

The introduction of the PUF as a mechanism for generating secure bitstrings first appeared in [1] and [2], however, the PUF as a chip identifier was introduced earlier in [3]. Since its introduction, there have been many proposed PUF designs, most of which fall into one of several categories: SRAM PUFS [4], ring oscillators [5,6], MOS drive-current PUFs [7], delay line and arbiter PUFs [8], and PUFs based upon variations in a chip's metal wires [9]. The Glitch PUF, a delay-based PUF that relies on the variation in glitch behavior, is presented in [10]. Each of these PUFs attempts to leverage one or more naturally-varying properties, and shares the challenges that arise from a number of sensitivities, such as measurement error and uncertainty, and fluctuations in voltage or temperature.

The HELP PUF proposed in this paper is, to the best of our knowledge, the only delay-based PUF that combines the following features:

- HELP is embedded in the hardware in the sense that the path delays measured in, e.g., an AES core logic macro, are used to generate a bitstring that is later used as the key when AES is run in functional mode. The close proximity of bit generation to the hardware that uses the bitstring improves resilience to invasive, e.g., probing, attacks that are designed to steal the key.
- By using the core logic of AES itself, a large source of entropy is leveraged.
- HELP's bit flip avoidance scheme makes the probability of failure in regenerating the bitstring negligibly small.
- The physical implementation of HELP uses standard logic functions widely available in most commercial chip architectures. In particular, HELP uses standard cell library blocks and an on-chip clock generation scheme, i.e., a digital clock manager (DCM). Our use of the DCM for performing path timing tests is similar to that proposed in [11] for Trojan detection and IC authentication.

In this paper, we analyze very large bitstrings as a means of improving the statistical significance in our reported results. However, we believe that the large number of bits provided by HELP can serve as an enabler for new, yet-to-be-developed, iron-clad security mechanisms that are not possible using a smaller, fixed number of bits provided by current non-volatile memories.

3. Overview

Similar to other PUFs, the bitstring is generated by applying a set of challenges and measuring the corresponding responses, called challenge-response pairs. The challenge component for HELP consists of a randomly selected, two-vector test sequence applied to the inputs of the macro-under-test (MUT), which introduces a set of transitions that propagate through the core logic of the MUT and emerge on its outputs. The responses are the measured path delays on each of the outputs, and are expressed as 8-bit numbers that correspond to path delay. A single MUT output is isolated and measured individually, as explained in this section. A bitstring is generated by comparing pairs of these path delays.

The delay measurement precision has an impact on the stability of HELP. We use an embedded test structure called REBEL to obtain a high-precision, digitized representation of the path delays [12]. REBEL is integrated directly with the scan chain logic and uses the on-chip clock tree network for launch-capture (LC) timing events.

Fig. 1 depicts an overview of the REBEL test structure, consisting of two rows of flip-flops (FFs) that can be connected together into a single scan chain or on separate scan chains (as shown). The top row is the launch row, and is configured to operate in functional mode. A small logic block on the left of the second row of scan cells, labeled RCL for Row Control Logic, allow the scan elements on that row to be configured as follows:

- The second row is the capture row, and is configured in a *mixed* mode, in which a specific FF, called the **insertion point** (IP), is chosen. This scan-FF and each scan-FF to the right of it in the row are placed in a mode called **flush delay** (described below), and form a combinational delay chain, effectively extending the path at the IP.

Flush-delay mode (FD) is a special mode in which a scan chain can be configured as a combinational delay chain. This is depicted in the callout in Fig. 1, which shows two master/slave FFs in which the output of the first master feeds forward into the scan input of the second FF. Any transition that occurs on the IP propagates through the *functional input* and into the first master using logic that selects that path (not shown). In contrast, the logic controlling the scan mux for the second FF (and all FFs to its right) selects the *scan input*, effectively allowing the transition to propagate unimpeded through the masters of these FFs. Details concerning the control logic for the scan chain muxes can be found in [12].

A REBEL path delay test is carried out by scanning in configuration information, which selects the IP and configures the delay chain as shown in Fig. 1. A clock transition is then applied to the launch row FFs which generates transitions that propagate into the MUT. Any transition that occurs on the MUT output at the IP will propagate into the delay chain. By asserting the clock input on the capture row FFs, the master latches revert to storage mode and digitize the time behavior of the transition(s) as a sequence of 1's and 0's. The combined delay of the MUT path and the delay chain can be derived by searching, from right to left, in the binary sequence for the FF that contains the first transition. Note that additional glitching behavior can be present in the sequence to the left of this final transition.

4. Experimental Setup

We've created a complete HELP implementation on an FPGA and carried out experiments on a set of 29 V2Pro-based FPGA boards. Fig.

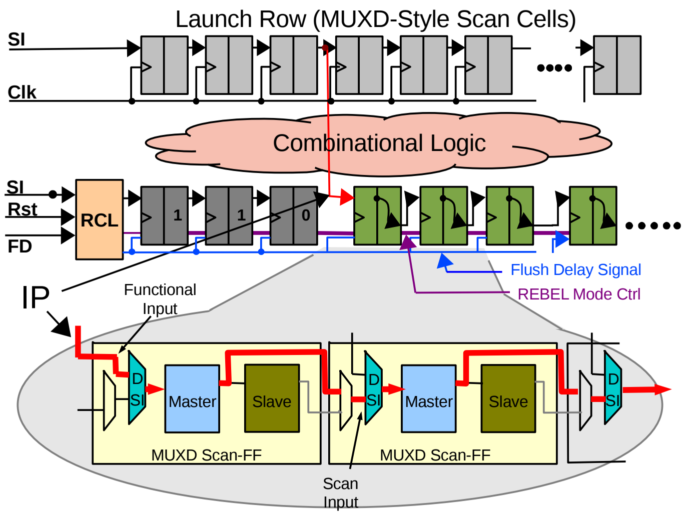

Fig. 1: REBEL embedded test structure.

2 shows a top-level structural diagram of our HELP implementation.

The MUT used in our implementation is the logic defining a single round of a pipelined AES implementation (space limitations prevented inclusion of all 10 rounds of the logic) from OpenCores. The block labeled “Initial Launch Vector (256)” represents the pipeline FFs in the full-blown AES implementation, converted here to MUX-D scan-FFs. A second copy of this block labeled “Final Launch Vector (256)”, is added to emulate the logic from the omitted previous round. In our implementation, two randomly generated vectors that represent the challenge are scan-loaded into the two blocks.

TABLE I. FPGA Resource Utilization

Resource	AES Macro		Full PUF (w/o UART)	
	Used	Util. Pct.	Used	Util. Pct.
Flip-flops	1297	4.7%	1749	6.0%
LUTs	3122	11.4%	7098	25.0%
Slices	2146	15.7%	3986	29.0%
RAMB16	0	0.0%	58	42.0%
BUFGMUX	1	12.5%	5	31.0%
DCMs	0	0.0%	3	37.0%

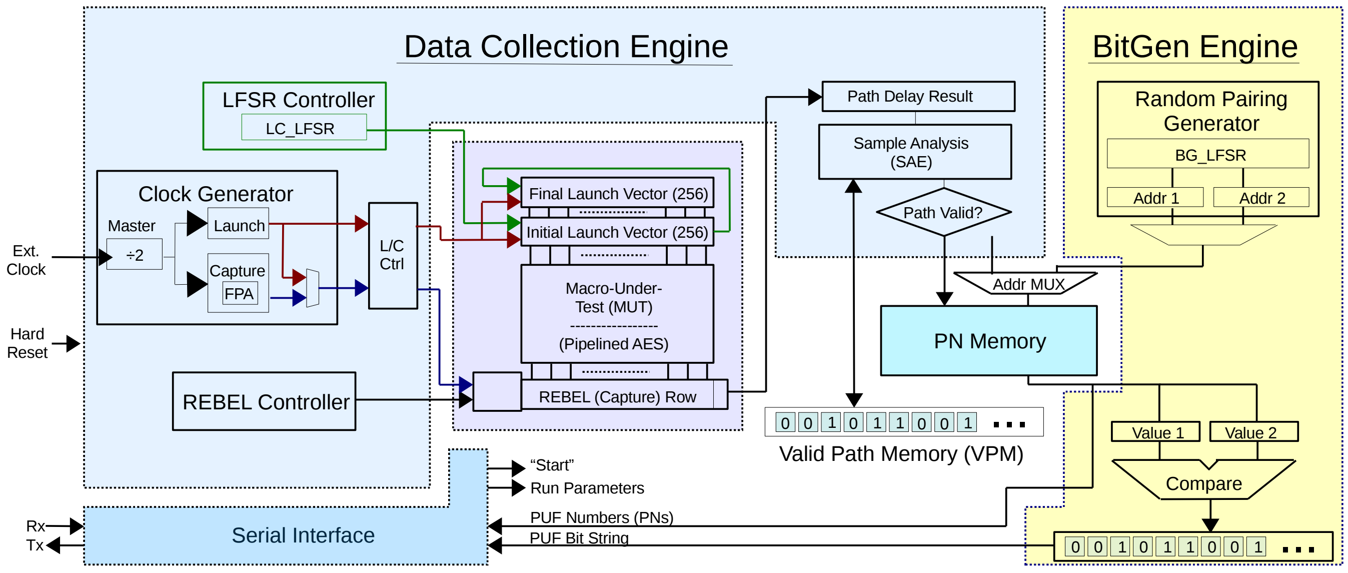

Fig. 2: Top-Level HELP System Diagram

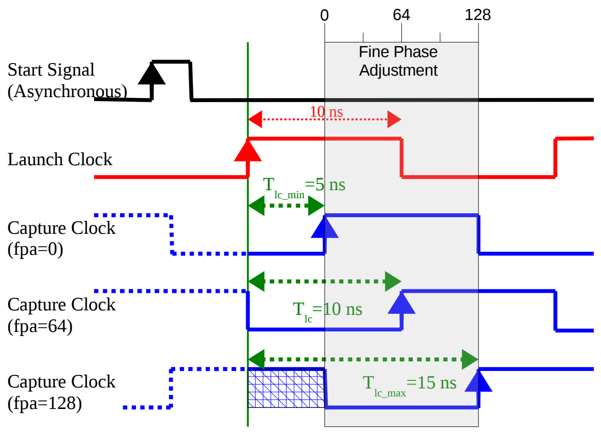

Fig. 3: Launch/Capture Timing Diagram

The block labeled "REBEL (Capture) Row" in Fig. 2 also represents the pipeline FFs between the logic blocks defining the rounds in AES. We modified this row to incorporate REBEL, and designed it to implement the "mixed mode" functionality described previously in relation to Fig. 1. The number of FFs in this row is expanded from 256 to 264 to extend the delay chain for the IPs in the rightmost side of the MUT.

The remaining components in Fig. 2 define the HELP PUF engine, and can be divided into the **Data Collection Engine** (DCE), and the **BitGen Engine** (BGE). One iteration of the whole process produces the bitstring. The engine behaves differently depending on whether a new bitstring is requested (a process called **enrollment**) or whether the bitstring needs to be reproduced (a process called **regeneration**). These scenarios are distinguished between in the following description where needed.

The overhead of HELP is given in TABLE I. The resources under the column "AES Macro" corresponds to a single round of AES. A full pipelined implementation of AES would therefore be 10X larger. Factoring this in reduces the overhead of HELP from 100% as shown in the last column to approx. 10%.

4.1. HELP Components

The DCE in Fig. 2 carries out a sequence of LC tests, measures the path delays, and records the digitized representation of them, called PUF numbers or **PNs**, in block RAM on the FPGA. In our current implementation, the DCE runs to completion before the BGE component is started. Alternatively, the DCE and BGE components can be run simultaneously.

Clock Generator. The clock generator module generates two clock signals: a Launch clock and a Capture clock, and is shown on the left in Fig. 2. In our design, this module contains three digital clock managers, or DCMs. A 'master' DCM is used to reduce the off-chip oscillator-generated 100 MHz clock to 50 MHz. The output of the master DCM drives the Launch and Capture DCMs. We utilize the fine phase adjustment (FPA) feature of the Capture DCM to 'tune' the phase relationship between the Launch and Capture clocks. At 50 MHz, the FPA allows 80-ps increments/decrements in the phase of the Capture clock on the V2Pro FPGA chips.

When the DCE is configuring the scan chains in preparation for the LC test, the phase relationship between the Launch and Capture clocks is set to 0. Just prior to the launch event, the controlling state machine selects the 180° phase-shifted output of the Capture DCM, and the FPA feature is used to tune the phase in an iterative process designed to meet a specific goal (to be discussed).

TABLE II. summarizes the characteristics of the Capture clock, and Fig. 3 illustrates the timing relationship between the Launch and Capture clocks for different values of the 'Phase Adj.' control counter in the DCM. The launch and capture events occur on the rising edge of the corresponding clocks. From the timing diagram, this allows path delays from 5 ns to 15 ns in length to be measured. The 0 to 128 range of values (called PNs) are used as a digital representation of the path delays.

The remaining components of the DCE are as follows:

PN Memory: A block RAM used to store the PNs.

LC LFSR Controller: A 32-bit linear feedback shift register (LFSR) used to produce the randomized launch vectors.

REBEL Controller: Configures the IP in the REBEL row attached to the output of the AES logic block.

Sample Analysis Engine (SAE): Analyzes the digitized results in the delay chain after each LC test for a given path and determines whether the path is 'valid'. **A valid path is defined as one that has a real transition, is glitch-free, and produces consistent results across multiple samples**.

Valid Path Memory: A block RAM used to record a pass/fail flag for each tested path that reflects its validity (as defined under SAE). This memory would normally be stored in non-volatile storage because it represents the *helper data* needed in the regeneration process.

TABLE II. CAPTURE CLOCK PHASE ADJUSTMENT

Phase Adj.	Phase Angle	LC Interval
0	90°	5 ns
64	180°	10 ns
128	270°	15 ns

Random Pairing Generator: Uses a 28-bit LFSR to generate randomized pairings of PNs for bit generation.

The Serial Interface component is used to interact with the HELP engine, and to transfer the results of the path testing and bit generation processes.

4.2. Measuring Path Delays

A sequence of paths are tested by the DCE process to produce the PNs used later in bit generation. The starting point and order in which the paths are tested is completely determined by the LC LFSR. The DCE process begins by loading the LC LFSR with a seed (provided by the user), and instructs the LC LFSR controller to load a random pair of vectors into the launch rows. Simultaneously, the REBEL controller configures the REBEL row with a specific IP and places the REBEL row in FD mode. The same random vector pair is reloaded to test each of the 256 IPs, one at a time, before the LC LFSR generates and loads the next random vector pair.

A key contribution of our technique is the discovery that **path stability** can be used as the basis for random number generation. Path stability is defined as those paths which have a rising or falling transition, do not have temporary transitions or glitches, and that produce a small range of PNs (ideally only one) over multiple repeated sampling. As shown below, the paths that pass the stability test are different for each chip in the population. In generating 4096 paths on 29 boards, less than 100 paths were common to every board, and only 2042 paths had any commonality at all.

A state machine within the DCE is responsible for measuring path delays and for determining the stability of the paths. Our algorithm begins testing a path by setting the FPA to 128, which configures the Capture clock phase to 270°. It then iteratively reduces the phase shift in a series of LC tests, called a **sweep**. For paths that have transitions, the process of 'tuning' the FPA toward smaller values over the sweep effectively 'pushes' the transition backwards in the delay chain, since each successive iteration reduces the amount of time available for the transition to propagate. When the edge is 'pushed back' to a point just before a *target FF* in the delay chain, the process stops (the goal has been achieved). The target FF is an element in the delay chain that is a specific distance (in scan-FFs) from the IP. The value of the FPA at the stop point is saved as the PN for this path, i.e., the PN represents the 'response' to the 'challenge' defined by the launch vector and IP.

Evaluating path stability is accomplished by counting the number of transitions that occurred in the REBEL row by 'XOR'ing' neighboring FFs in the delay chain. The path is immediately classified

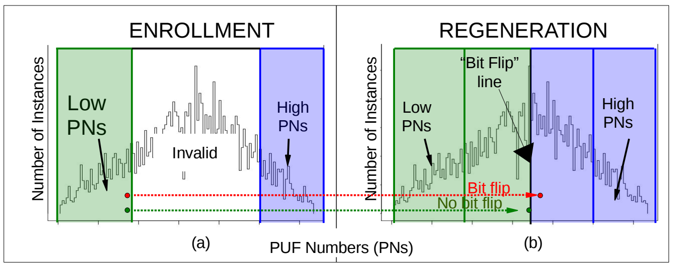

Fig. 4: UNM Bit Generation

as unstable (and the sweep is halted) if the number of transitions exceeds 1 at any point during the sweep. Once the sweep is complete, the whole process is repeated multiple times. If the range of PNs measured across multiple samples varies by more than a user-specified threshold, the path is classified as unstable and is discarded.

Note that path stability evaluation occurs ONLY during enrollment. In order to make it possible for regeneration to replay the valid path sequence discovered during enrollment, the 'valid path' bitstring is updated after testing each path. For paths considered valid, a '1' is stored and for those classified as unstable, a '0' is stored. During regeneration, the exact same sequence of tests can be carried out by loading the LC LFSR with the same seed and using the 'valid path' bitstring to determine which paths are to be tested (a '1' forces the path to be tested, and a '0' forces the path to be skipped).

The usage scenario requires the LC LFSR seed and the 'valid path' bitstring to be stored in publicly accessible non-volatile memory. This enables an adversary to "reverse engineer" the secret bitstring using a simulation model of the MUT and HELP system. Although difficult to accomplish in practice, the only way to completely eliminate this threat is to obfuscate the helper data. Techniques for obfuscation exist, but are beyond the scope of this work.

4.3. The "Universal-No Modulus" (UNM) Techniques

We developed a method called "Universal-No Modulus" (UNM) that is capable of generating $O(n^2)$ bits from n PNs. UNM avoids bit flips by using only the longest and shortest paths in the MUT for comparisons, discarding paths of medium length. It avoids the bias that would normally result under these conditions by exploiting the property that **path stability** is random across chips. In other words, even though the result of comparing a short path with a long path is predictable from the design, the stability, and therefore selection, of short and long paths is random from chip to chip.

Figs. 4(a) and 4(b) show the path distribution from a typical chip, with the PN range plotted along the x-axis against 'number of instances' along the y-axis. During enrollment, UNM uses two thresholds to partition the distribution into 3 regions. The tail regions on the left and right are considered valid PN regions. PNs in the tails represent short (Low PNs) and long (High PNs) paths respectively. The large 'invalid' region between the thresholds, given as 32 and 90 in Fig. 4(a), is a safety zone between the groups designed to prevent 'jumps', and bit flips, between the Low and High PN regions. The placement of the thresholds determines the balance between the number of paths in each tail region, and are established using a process that characterizes the path-length distribution at the start of each enrollment. Jumps, although infrequent, can occur because of the appearance and disappearance of hazards (glitches) on side-inputs of gates along the tested paths. Small temperature variations or power supply noise influence the behavior of these hazards. Examples of tolerable (green line) and intolerable (red line) jumps are shown Fig. 4, wherein the lines indicate PNs that were significantly higher during regeneration than they were during enrollment.

The safety zone is only enforced during enrollment, and is redefined as the midpoint between the margins during regeneration as shown in Fig. 4(b). The DCE process creates a valid path bitstring during enrollment so the same sequence of path tests can be carried out during regeneration. In our experiments, we found that UNM generates a valid PN after approx. every 20 tested paths, depending on the user-specified width of the 'invalid' region. The "XOR-style" bit generation process is carried out by comparing pairs of PNs, where PNs from the same region generate a '0', while those from opposite regions generate a '1'. With n PNs, up to $n*(n-1)/2$ bits can be generated by considering all combinations.

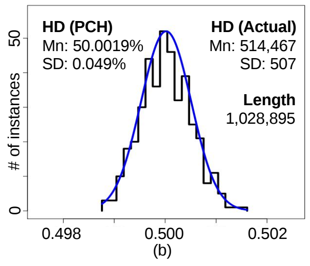

Fig. 5: HD Analysis

5. Experimental Results And Analysis

We collected data on a set of 29 V2Pro boards using a thermoelectric cooler (TEC) apparatus and a programmable power supply. Experiments were carried out at three different temperatures (0C, 25C, 70C) and three different voltages (1.35V, 1.50V, 1.65V). Enrollment data is collected at 25°C and 1.5V. The experiments carried out at the remaining 8 temperature/voltage (TV) corners represent regeneration.

Hamming Distance: Fig. 5 shows the HD distribution as well as the mean (Mn) and standard deviation (SD) of the Gaussian curve fitted to, and superimposed on, the distribution. The inter-chip HD is 50.0019% for bitstrings with a length of 1,028,890. This is very close to the ideal value of 50%. The standard deviation is also very small. Combined with a mean intra-chip HD of **2.74e-7**, these results indicate the bitstrings are highly reliable and unique.

NIST Statistical Analysis of Randomness: The NIST statistical test suite is also applied to the bitstrings from the 29 boards. The bitstrings pass all NIST statistical tests, with no more than 2 boards failing any of the 15 tests. In addition, these bitstrings pass all of the P-value-of-the-P-values tests, even in spite of the fact that the NIST documentation indicates that a minimum of 55 boards is required before this metric can be considered valid.

Running Time Analysis: On average, the number of valid paths tested per second is **30.20** for enrollment and **88.03** for regeneration. This includes the time required to test and discard invalid paths, and the time required to generate the $n(n-1)/2$ bitstrings from the n PNs stored in block RAM.

Probability of Failure: As discussed above, a bit flip occurs when a PN measured during regeneration jumps across the "bit flip line" as shown by the example in Fig. 4(b). The number of bit flips that occurred across the 8 regenerations for the 29 boards is **10**. This yields a probability of failure of **8e-5**, computed as 10 / (29 boards x 4096 PNs per board). Although beyond the scope of this work, we have developed a simple, very low-overhead technique that eliminates all bit flips in our results and improves the probability of failure to **7.25e-11**.

6. Conclusions

A novel PUF, called HELP, is proposed and demonstrated on an FPGA platform. HELP is based on measuring variations in path delays in the core logic macro(s) of the chip. The results of hamming distance and NIST statistical test analyses show the bitstrings are of high quality. In particular, the bitstrings are unique and repeatedly random, and are therefore appropriate for cryptographic applications.

7. Authors

Jim Aarestad received his M.S. in Computer Engineering from the University of New Mexico in 2011. His research interests include hardware security and its role in embedded systems and in the broader context of trusted computing. After completing his PhD., Jim will become a computer scientist at the Cyber Division of the Federal Bureau of Investigation.

Postal Address:
1706 Silver Ave. SE - #28

Albuquerque, NM 87106
Phone: (505) 573-1936
Fax: (505) 277-1439
Email: jaarestad@ece.unm.edu

Philip Ortiz is a Principal Member of the Technical Staff at Sandia National Laboratories, has 17 years' experience in FPGA design, signal processing, and embedded systems. Philip received his B.S. in Electrical Engineering from Stanford University and is completing an M.S. in the same subject from the University of New Mexico.

Postal Address:
7339 Boxwood Avenue NE
Albuquerque, NM 87113
Phone: (505) 845-8156
Email: pgortiz@sandia.gov

Dr. Dhruva Acharyya holds a PhD degree in Computer Engineering from University of Maryland, Baltimore County. His research interests are in the areas of VLSI Design and Testing, Hardware Security, Test Instrumentation and Design for Manufacturability. He is currently employed as a R&D Engineer at Advantest Corporation.

Postal Address:
3061 Zanker Rd.
San Jose, CA 95134
Phone: 410-245-3322
Email: dhruva.acharyya@advantest.com

Dr. Jim Plusquellic received his degree in Computer Science from the University of Pittsburgh in 1997. He is currently an Associate Professor in ECE at the University of New Mexico. His research interests include security and trust in IC hardware, silicon validation, design for manufacturability and delay test methods. He is a member of the IEEE Computer Society's Golden Core and of the IEEE.

Postal Address:
University of New Mexico
ECE 236C
MSC01 1100
1 University of New Mexico
Albuquerque, NM 87131-0001
Phone: (505) 277-0785
Fax: (505) 277-1439
Email: jimp@ece.unm.edu

8. REFERENCES

[1] R.S.Pappu, *et. al*; "Physical One Way Functions", *Science*, 297(6), 2002, pp. 2026-2030.
[2] B. Gassend, *et.al.;* "Controlled Physical Random Functions", *Conf. on Computer Security Applications*, 2002, pp. 149-160.
[3] K. Lofstrom, *et. al.;* "IC Identification Circuits using Device Mismatch", *SSCC*, 2000, pp. 372-373.
[4] P. Simons, *et. al.*; “Buskeeper PUFs, a Promising Alternative to D Flip-Flop PUFs”, *HOST*, 2012, pp. 7-12.
[5] G.E. Suh, S. Devadas; "Physical Unclonable Functions for Device Authentication and Secret Key Generation", *DAC*, 2007, pp. 9-14.
[6] A. Maiti, P. Schaumont; "Improving the Quality of a Physical Unclonable Function using Configurable Ring Oscillators", *Conf. on Field-Programmable Logic and Applications*, 2009, pp. 703-707.
[7] Y. Su, *et. al.*; "A 1.6pj/bit 96% Stable Chip ID Generating Circuit Using Process Variations", *SSCC*, 2007, pp. 406-407.
[8] M. Majzoobi, *et. al.*; "Lightweight Secure PUFs", *ICCAD*, 2008, pp. 670-673.
[9] J. Ju, *et. al.*; “bitstring Analysis of Physical Unclonable Functions based on Resistance Variations in Metals and Transistors”, *HOST*, 2012, pp. 13-20.
[10] D. Suzuki, K. Shimizu; “The Glitch PUF: A New Delay-PUF Architecture Exploiting Glitch Shapes”, *CHES*, 2010, pp. 366-382.
[11] J. Li, J. Lach; “At-Speed Delay Characterization for IC Authentication and Trojan Horse Detection”, *HOST*, 2008, pp. 8-14.
[12] C. Lamech, *et. al.*; "REBEL and TDC: Two Embedded Test Structures for On-Chip Measurements of Within-Die Path Delay Variations", *ICCAD*, 2011, pp. 170-177.