INTEGRATION the VLSI journal xxx (xxxx) xxx–xxx

Contents lists available at ScienceDirect

INTEGRATION, the VLSI journal

journal homepage: www.elsevier.com/locate/vlsi

Reliable techniques for integrated circuit identification and true random number generation using 1.5-transistor flash memory

Lawrence T. Clark*, James Adams, Keith E. Holbert

School of Electrical, Computer and Energy Engineering, Arizona State University, PO Box 875706, Tempe, AZ 85287, USA

ARTICLE INFO

Keywords:
True random number generation
Physically unclonable functions
Flash memory
Systematic mismatch

ABSTRACT

Flash memory bits, like other integrated circuit (IC) devices, are prone to random variability in their actual versus nominal characteristics. We present the use of 1.5-transistor flash memory cells in physically unclonable functions leveraging their erase speed variability. This type of memory is interesting for the internet of things due to its wide availability as intellectual property at foundries. Using experimentally measured results, we show simple methods that provide high reliability with no or limited need for helper data and error correction. High quality fingerprints for IC identification are demonstrated. Moreover, techniques to remove systematic variations from the array response are shown, allowing the resulting binary strings to pass all National Institute of Standards and Technology tests for randomness. Consequently, with low complexity helper functions, true random numbers can be readily produced.

1. Introduction

Using physical variations in integrated circuit (IC) constituent components, such as transistors, is an increasingly popular technique for IC identification and generating true random numbers. The latter circuits are known as true random number generators (TRNGs). Since the resulting random numbers are based on the as-fabricated characteristics of a specific IC, any two will have different random fabrication process induced variations. Thus, they are unclonable—the resulting random values cannot be guessed. Hence, the generating circuits are also known as physically unclonable functions (PUFs) [1–3]. Given a sufficient number of bits, two ICs are distinguishable based on the resulting fingerprint (ID) which differs for each IC.

The use of PUFs as IC fingerprints allows secure identification of devices and users, as well as the ability to trace ICs, e.g., to determine grey market devices. In this section we evaluate the resulting IC fingerprint quality. The key to a reliable fingerprint is that it must be repeatable and its randomness (the PUF entropy) must ensure that other fingerprints are sufficiently far from the repeatable matching fingerprint [4].

1.1. Physically unclonable functions

PUFs are derived from underlying physical differences in fabricated ICs. This leverages the intrinsic manufacturing variations that have been increasing over time.

Delay racing (arbiter) based PUFs have been extensively analyzed and generate one bit at a time based on the outcome of racing paths through multiplexer paths selected by the input code or its result through a helper function [5–7].

Memories can also be used as a PUF, based on a number of characteristics [8–21]. Holcomb et al. used SRAM power-up state [20]. Chellappa et al. showed improved circuits that work by destabilizing continuously powered SRAM cells [10]. DRAM was effective as a PUF in [14]. Fingerprints based on the order that flash cells program, when programming is interrupted has been suggested as a PUF [15]. An important issue with memories is systematic mismatch, which was shown to affect both of these designs. Systematic mismatch can be due to poly misalignment in older technologies, and threshold implant screening by masks, especially for halo implants in more modern technologies. Layout, e.g., location of word line tap cells and array borders, can also affect the nominal local matching.

Memory based PUFs are referred to as weak [6,7], while delay based, e.g., arbiter, PUFs are considered strong. This strength is not due to the latter having greater variability or reliability. The concept derives from the ability to completely read a memory because it has finite locations, while trying all combinations of an arbiter based PUF can be impossible in finite time. Delay based PUFs are based on linear superposition of the underlying delays. As a consequence, machine learning algorithms have been shown effective at determining underlying delay PUF values and are aided when some bits are not reliable, which is common [5].

* Corresponding author.
E-mail address: Lawrence.clark@asu.edu (L.T. Clark).

https://doi.org/10.1016/j.vlsi.2017.10.001

1.2. True random number generation

Because a PUF should exhibit near perfect randomness, it can also be used as a TRNG [22–24]. The latter is easier, since repeatability is not of concern and may in fact be undesirable. TRNGs may then be used to produce high quality seeds for pseudo-random number generators or keys for encryption codes. Random telegraph noise (RTN) has been an increasing source of instability in flash and SRAM memories, as the current drive has diminished with scaling [25–28]. Depending on the location of the trapped electron in the oxide and channel, greater than 100 mV threshold voltage shifts have been observed [29]. Wang et al. used standard flash memory to produce random strings based on the RTN variability of the flash cells [15]. A random binary string is generated using those bits that are found to be unstable.

1.3. Reliability

One significant problem with PUFs is reliability and repeatability. In the latter, the same sequence may not be produced by the PUF for the same input in all cases. For instance, when delay or mismatch values are very close, the result is a function of noise rather than the mismatch. These errors can require that error detection and correction (EDAC) be applied to the result so that it is consistent, e.g., see Fig. 1. The former can manifest as a time-varying response, for instance due to circuit aging changing circuit delays.

When using a PUF for authentication, the response of the same IC to the same challenge, e.g., accessing the same physical characteristics, is an intra-class access and should produce an identical (or identical post-helper corrected) response. Conversely, a challenge to a different location (inter-class) or the same challenge to a different IC should produce a different response. The most common method of determining response similarity is the Hamming distance, which is given as

$$D_H = \sum_{i=1}^{k} |x_i - y_i|, \tag{1}$$

where x_i and y_i are the ith bits of the length k sequence being compared. The result is the number of mismatching bits for binary sequences. The ideal value for intra-class sequences is zero and for inter-class comparisons mean should be $k/2$ as half the bits in purely random sequences mismatch on average.

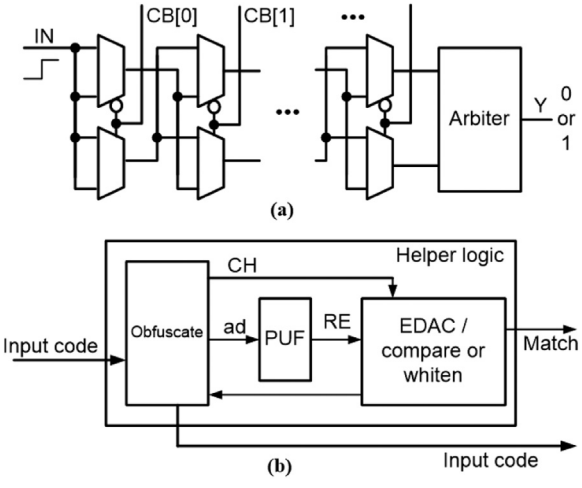

Fig. 1. An arbiter PUF (a) that compares racing timing paths generates one output bit per challenge. Any PUF is generally enclosed in helper logic (b) that protects access to the PUF and corrects unreliable bits, potentially responding to the challenge with a Boolean match signal and otherwise obfuscating access to the actual PUF challenge (CH) and response (RE).

1.4. Testing randomness

The PUF randomness properties are characterized by the min entropy, which is defined for a binary source stream i as

$$H_{min} = -\log_2(\max(p_0^i, p_1^i)), \tag{2}$$

where p_0 and p_1 are the 0 and 1 probabilities of the ith bit, respectively. The maximum function ensures that values above 0.5 are used.

The max entropy H_{max} is generally defined by the compressibility of a binary sequence. Thus, a truly random sequence should not compress to a smaller sequence of dictionary symbols. A common measure of H_{max} is the Unix *gzip* utility. Accessing the header allows determination that the file is left uncompressed. Ideally, min entropy H_{min} approaches the max entropy and both approach one.

The U.S. National Institute of Standards and Technology (NIST) has published tests to determine the randomness of TRNG circuits [30]. The NIST tests provide a measure of the randomness of binary strings. Each test quantifies a different randomness aspect for the binary string under analysis.

The tests are based on determining a P-value that provides confidence in whether or not the null hypothesis for that test should be accepted, i.e., that the data are in fact random. The alternative hypothesis is that the sequence is not random. The goal of the tests is to minimize the probability of type II error, i.e., that the binary strings are not random but pass. For each test, the bit sequence is analyzed by the appropriate algorithm and the output X is input to either a complementary error function or upper incomplete gamma function. The result is compared with the significance level α, which is a measure of the type I error probability. The tests make two primary assumptions. First is uniformity: the number of 0's and 1's should be approximately equal at any point in the string. The second is scalability: randomly extracted sub-sequences should not be identical.

1.5. Helper functions

Because both weak and strong PUFs are susceptible to attack and cloning if subject to repeated interrogations, a PUF is usually protected by a circuit that obfuscates the actual response from the applied interrogations. The helper logic can also preclude direct access completely. Moreover, since PUFs may not return the exact expected response, some form of EDAC is often needed. These functions are commonly referred to as helper data algorithms [6–8] and help the encapsulated PUF to meet the required reproducibility, entropy, and control specifications.

PUFs often produce grey or “dark” bits that are not repeatable (many TRNG are based purely on these sporadic bits). For an arbiter PUF (Fig. 1(a)) this is due to the two racing paths having nearly identical delays, so that noise can affect the outcome. Mitigation of these grey bits is often accomplished by temporal replication, i.e., multiple sampling and voting to determine their preferred direction. EDAC can subsequently repair bits that are still inconsistent. The entropy requirement must be primarily met by the physical mechanism controlling the PUF—it cannot be added by the helper, which by definition must be systematic. The control function generally includes shielding the raw PUF data from the user to mitigate attacks. In the memory case, it can for instance, disallow continuous read out, as otherwise a memory based PUF can be trivially copied.

A post-helper function value would be stored elsewhere. It comprises the challenge or the code required to create the challenge and therefore also includes the location to be accessed and the appropriate response, ideally obfuscated since it is publicly accessible (Fig. 1). Determining a challenge is referred to as the enrollment operation [1,2,6].

Fig. 1(b) illustrates a PUF enclosed in helper data algorithm logic. Initially, the PUF produces a response RE without or with a partial

input code. This is obfuscated or otherwise combined with the partial input code to produce a full output code for later use. Some form of this response is output so that it can be used in the future as a challenge to the PUF. In an interrogation or check for a specific device, the input code is used to interrogate the PUF. After the helper function performs EDAC and compares the challenge CH to the PUF response RE, an output is provided. Since helper functions may include correction bits, etc., they can leak information out of the PUF and thus be used in attacks [6,7]. An effective means of avoiding such information leakage is to provide a simple binary match/non-match response to a challenge. Given a sufficient number of bits, which must be guessed, such a scheme makes even weak PUFs immune to simple attacks.

The problem with EDAC in PUFs is that the error correction code bits must be part of the challenge and thus leak information about which bits of the underlying response are sporadic. Additionally, a relatively large percentage of the bits may be grey, so large and sophisticated EDAC is required. Ideally, the helper logic is small, so this affects the size of the PUF adversely.

An often applied helper function is the eponymous Von Neumann correction, which he designed to convert biased into fair sampling. In this scheme, pairs are taken—in this context inputs can be sampled from either two PUF/RNGs or bits can be taken two at a time. Only when they differ, i.e., a transition in the latter case, is a result registered. A 0-1 is 0 and 1-0 is a 1 for instance. A drawback is the rejection rate varies between 0.5 and 1, so many bits are lost.

2. Prior memory PUFs

Memory PUFs have been proposed for nearly all types of memory. Flash and fuses have long been used for IC device IDs [31–33]. These are programmed at the factory in a program once fashion, but require specialized processes.

2.1. SRAM memory

SRAM does not require specialized processes, making it very appealing in a PUFs application. However, SRAM PUF bit density is relatively poor and the large number of grey bits can be problematic [8,10,12]. Power-up state, based on the patterns produced by mismatch within the cells has been shown usable as a PUF [20]. Ideally, SRAMs have excellent matching characteristics, which enhance their stability. However, the best matching bits are those most subject to being grey bits, because their mismatch is less than the noise [9]. Moreover, they are subject to variability due to temperature as the P and N ratio changes [10]. They are also subject to aging effects, particularly negative bias temperature instability (NBTI), which has also been suggested as a means to increase mismatch and reliability [13].

2.2. Nonvolatile memory

Standard flash memory is ubiquitous in modern portable devices, exactly where the security afforded by a PUF is very desirable. Wang et al. used standard flash memory to produce random strings based on the variability of the flash cells [15]. Specifically, they purposely programmed cells to a point where the cell could flip either way based on whether or not trapping, in the form of RTN, caused the cell to flip either way when sensed. A string is then generated based on those bits that are found to be unstable. Unlike prior flash RTN work that required access to the transistor currents [34] they showed that weak programming could take cells to their failure cliff, so that bits were unstable. They suggest IDs based on the order that the cells program, where programming was interrupted so that it was incremental.

Similarly, Rose et al. proposed using memristor write time as a PUF [18]. Magnetic tunnel junction (MTJ) memory has also been suggested as a TRNG [19]. In this scheme, a single MTJ state is perturbed by an optimal perturb pulse designed to have a 50% probability of switching the cell. Using 10 bit counters and with feedback adjustment, after Von Neumann correction of the output, all NIST tests pass.

Flash characteristics are to first order temperature independent and not subject to NBTI. However, conventional one transistor (1-T) flash memory is subject to strain induced leakage current (SILC) and anode hole injection (AHI) [27,35]. The former causes charge loss, so SILC should not affect PUFs operation. AHI however causes erratic erase, i.e., unpredictable variation of the resulting threshold voltage after consecutive erase/program operations [35].

2.3. DRAM memory

Rosenblatt et al. used DRAM memory to produce fingerprints [14]. Here, the rate that cells lost their state due to leakage is used to determine the unique ID. While exponentially dependent on temperature, cells leakage rates are affected proportionally.

3. SST 1.5-T flash memory

The 1.5-T flash memory marketed as SuperFlash® by Silicon Storage Technologies (SST) is used in this work. It is interesting for internet of things (IoT) applications as it is available for many technology nodes at many foundries as intellectual property (IP).

3.1. Memory cell architecture

The split-gate flash cell (Fig. 2) is referred to as a 1.5-T cell since it has separate select gate (word line, WL) and floating gate devices configured as a composite transistor between the source and drain [36]. The cell transistor pairs share a common source line (source in Fig. 2(a)). Fig. 2 shows the third generation cell—there are many variations, but they share common behaviors. From a circuit point of view, the design is two series devices as in Fig. 2(b)—the floating gate (FG) at the source line (SL) provides programmability. Programming is accomplished by source-side injection. To program a cell, a high voltage (approx. 10 V) is applied to the control gate (CG) and a moderate bias is applied to the source. If the drain is at V_{DD}, no current flows so the cell is not programmed. When the drain is near 0 V, hot electrons are injected across the WL to FG gap to discharge the FG. Programming is thus self-limiting since as the FG collects electrons injected across the gap, its potential becomes increasingly unfavorable to further electron capture. Additionally, this process is very efficient as compared to conventional flash memory.

To read, V_{DD} is applied to the WL and $V_{CG} = V_{DD}$ to couple the FG up slightly. The source line is at ground and the bit line (BL) is biased to about 1 V, producing the read current. A large IoT benefit of the 1.5-T cell is high voltage is not needed for read.

The FG is erased by removing electrons via poly to poly Fowler-Nordheim (F-N) tunneling, raising the FG potential. To erase, the erase gate (EG) is driven to a high voltage (~ 11.5 V) while the BL, WL and CG are grounded (Fig. 2(b)). The WL and CG couple the FG to a low potential, increasing the tunneling probability to the EG. The F-N tunneling current density is

$$J = C_1 \ E^2 \ \exp(-E_0/E) \tag{3}$$

where E is the electric field, and C_1 and E_0 are constants related to the effective mass and barrier height [37]. Since E is inversely proportional to the inter-poly erase tunnel oxide equivalent thickness, the rate at which a cell erases is a measurement of the effective inter-poly oxide thickness t_{ox}. Thus, the erase rate is an exponential function of the effective as-fabricated floating gate to erase gate inter-poly dielectric thickness. While other variations affect the erase speed [35,38], this variation is the first order basis of the 1.5-T flash memory based PUF proposed in subsequent sections.

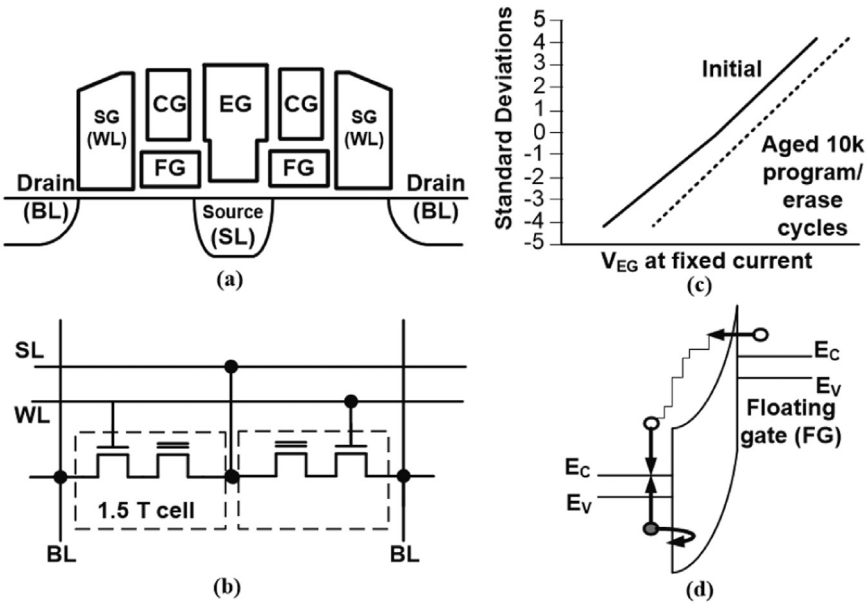

Fig. 2. 1.5-T third generation cell cross-section (a) and circuit configuration (b). Aging response remains almost perfectly Gaussian as shown in the Q-Q plot cartoon (c) adapted from []. The lack of AHI is shown by the corner enhanced field at the FG side but not at the erase gate side (d) adapted from [].

The source line is parallel to the WLs as evident in Fig. 2(b). Consequently, the memory can be erased in small sections of 256 to 1k bits, which is basically the array width. This allows small granularity PUF sizes.

3.2. 1.5-T erase aging and reliability

Like any flash memory, the SST cells have been extensively studied with respect to their physics and reliability [35,38]. The corner evident in Fig. 2(a) enhances the erase speed. While the earlier generations have more pronounced corners, such *E* enhancement is common across SST cell generations [38]. The corner geometry enhancement of the erase field at the FG is evident in Fig. 2(d) as the apparent thinner erase oxide. The erase tunneling area is thus about 100 times smaller than in an equivalent geometry (1-T) flash cell [38]. Greatly enhanced field at the tip allows thicker erase oxide while maintaining good erase speed [36]. A byproduct of a thicker oxide is that the probability of SILC is greatly reduced. SILC and AHI are the primary causes of reliability degradation in conventional stacked gate flash memories. The latter causes erase time acceleration with aging. In the SST cells used here, AHI is essentially eliminated since the anode has a lower electric field than the corner enhanced FG cathode. This lower apparent electric field prevents hole tunneling that produces AHI as shown in Fig. 2(d) [38].

The primary aging mechanism in the SST cell is electron trapping in the inter-poly oxide near the FG corner. This electron trapping diminishes the peak electric field and thus after repeated stress, the erase rate degrades. The cells show a predictable degradation of erase efficiency and with up to 300 K cycles, no erase enhancement has been observed [38]. The response of an aggregate of cells is shown in Fig. 2(c).

4. Experimental design and results

The experiments use 512k-bit blocks in Microchip Technology Inc. 16 Mbit SST39VF1601C devices. We begin by fully programming a block to all 0's. The 1.5-T flash cells have very high program and erase efficiency [36]. The former takes only one clock and so cannot be interrupted. Erase requires many cycles, so we interrupt the erase operations as fast as our test system can (in one memory instruction).

Consequently, we interrupt the erase to provide a partial erase of the device. At nominal voltages, most array cells erase in the single cycle even when interrupted as fast as possible. Consequently, to slow the erase further, we reduce the flash V_{DD} during erase. Lower IC V_{DD} reduces the internal charge pump output voltage, diminishing the erase gate to floating gate potential. Since F-N tunneling current density is exponentially related to the field (Eq. (2)) the erase rate is slowed significantly.

4.1. Slow erase

Fig. 3 shows the number of 1 state (erased) cells vs. partial erase cycles for one device under test (DUT) array. We use the array states that are closest to, but over 50% 1's vs. 0's to determine the codes. Note the saturation in the values after approximately 15 cycles—many of the remaining un-erased cells will not erase at the low voltages used. A bit map shows that the erase characteristics appear to be random (Fig. 4). The operations are highly repeatable and the erase operations are monotonic with time, until the saturation point where further erase operations have little effect (see Fig. 3).

4.2. Erase behavior

We observed no bits changing from 1 to 0 with subsequent erase operations until the point was reached where we had saturated the

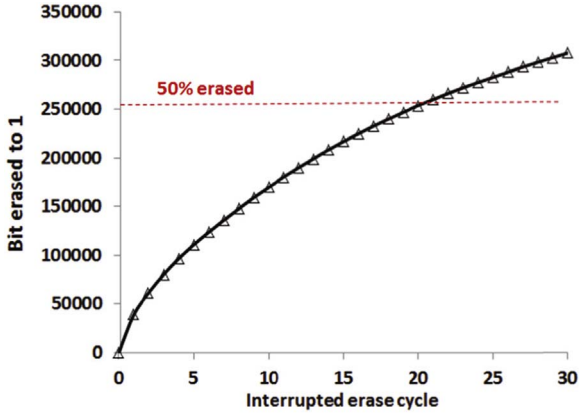

Fig. 3. Flash block low V_{DD} erase characteristics for multiple trials. After each erase cycle, the block is read out to determine the number of cells erased to the logic 1 state.

Fig. 4. A small sub-section of the array partial erase response at about 50% erased bits. There is no apparent systematic variability.

number of erases [21]. At this point we are subject to thermal and RTN noise, as well as the sensing offsets. Use of this mode of operation was demonstrated for conventional flash based TRNG as described above in Section 2.2.

Unlike prior work, we directly use the array state as a random binary string for input to helper functions for both PUF and TRNG applications. Since maximizing entropy requires approximately 50:50 ones and zeros, we use the points at or near 50% 1's (half erased flash bits) as the basis.

Fig. 3 shows that a particular device has consistent behavior at a given reduced erase voltage. However, between devices we observed significant differences. The experiments thus tuned V_{DD} by DUT. About 2 V was usually effective, but too low a voltage would cause the devices to become inoperable. In addition, we repeated the experiments with embedded SST flash array IP. However, the embedded flash failed to erase at any out of specification V_{DD}.

4.3. Impact of programming conditions

Experiments were run with different multiple programming scenarios, e.g., 1x, 2x, 4x and 8x program to 0 operations on the entire array. Additionally, beginning at different states, i.e., weak or full erase before the programming, were used. In all cases, these changes had no impact, reflecting the self-limiting program nature of the cell described in Section 3.1.

4.4. Systematic offsets

SRAMs can show systematic variations that come from identically balanced drawn cells having geometric imbalances due to their surroundings, which affect the actual processing, i.e., lithography, implant, or etch. The result is a greater overall propensity to have 1s or 0s in the power-up state. The offset can of course be spatially varying [3].

A closer examination of the spatial variation by mapping multiple 512k-bit arrays (raw data) shows systematic erase rate variability across a block, as shown in Fig. 5. Rows read in address order from left to right show a systematic bias towards 0's (un-erased) at the edges. This was revealed by summing the ones per array column. Thus, each 2048 bit substring has a lower likelihood of a one at its ends. All blocks on a device and across devices show very similar behavior, so we attribute it to the specific array design. For instance, the slower erase at outer columns may be due to IR drop in the erase voltage across the array. We speculate as we do not have access to the specific design, but simulations show such a droop if driven only at the middle. Similar systematic variation has been observed in other flash memories from multiple vendors, shown by block and page number offsets using aborted programming [17].

5. Novel IC fingerprint approach

Following other memory based fingerprint analyses, we compared every 256-bit string generated in the flash memory with each other. We generated the same fingerprints in ten separate trials on multiple DUTs. The intra-class keys have 25–65 mismatches and the inter-class keys had 92–172 mismatching bits for cases with up to 10% fewer

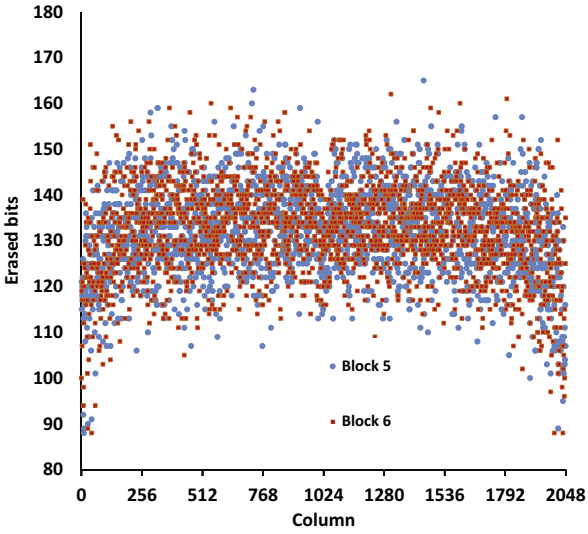

Fig. 5. Variability in the number of 1's by column across two DUTs. Some exhibited strong variability, remarkably similar to that observed in planar flash [17] while some did not.

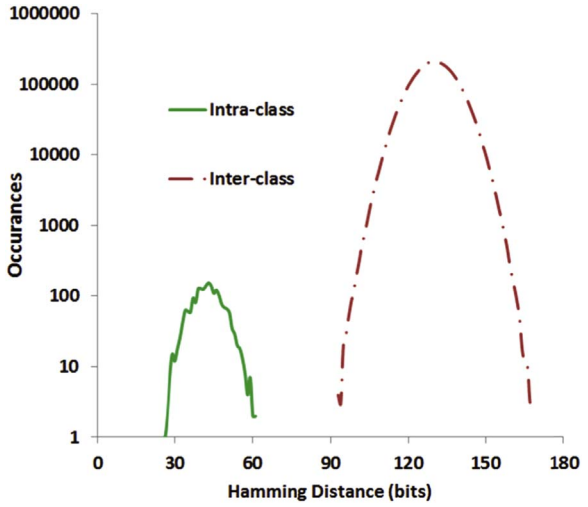

Fig. 6. Hamming distances (log ordinate scale) of intra-class and inter-class challenge (CH) vs. response (RE) codes for a sample DUT. The CH is generated by greater than 50% bits programmed in the first slow erase trial and the RE is from a subsequent slow erase trial.

erased bits in the response vs. the challenge, a representative is shown in Fig. 6. Consequently, directly comparing bits has to allow some mismatches, as is the case in most memory PUF schemes, due to grey bits. Here we note that all of the intra-class mismatching bits are a 1 in the challenge, as they have not yet erased in the response. This suggests the helper scheme described in the next section.

Moreover, EDAC is very expensive when there is a large number of mismatching bits. A key issue is the ability to stop the erase at the same point. Precisely producing a repeatable stop point may be impossible, particularly with aging, so there will always be some mismatches. In these experiments the erase resolution is purposely crude to avoid optimistic results, but they are quite encouraging.

5.1. MON1: a novel lightweight helper function

Fig. 7 shows the bits for a fingerprint subset over multiple partial erase cycles. Erase is monotonic so a code generated with for instance 45% 1's as the response can be effectively compared with a challenge code generated with 55% 1's. The challenge code (CH) is produced by stopping the erase when the flash array has more than 50% 1's and the

Fig. 7. Partial fingerprint compare based on the monotonicity of the erase operations (scheme MON1). The challenge (CH) is chosen as codes after the memory block reaches 50% ones content. The monotonicity of cells becoming 1 with more erase cycles (EC) is evident in the top (Case A) codes. This also shows the time evolution of the number of allowed mismatching bits (elaborated in green) at far right by erase cycle. The response (RE) is reproduced by the flash as a cycle before 50% ones content on demand. A matching challenge bit that is a 0 cannot be a 1 in the response as this indicates a mismatch. Conversely, mismatching challenges (intra-code on Cases B and C) have many 0's that are 1's in the response (red). This experiment purposely has fast (coarse) erasing. The CG output refers to the column gate outputs in the circuit of Fig. 8. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article).

result code (RE) has been erased to less than 50% 1's. In Fig. 7 at top (labeled Case A), challenge CH A is compared with the time evolution of its intra-class response RE A over 4 different erase cycles (the fifth is used as CH). The matching code subset with lesser erase and thus fewer bits erased, has a few mismatching bits. However, in the mismatches the challenge bits are 1 in this intra-class case. This suggests a simple scheme whereby for each (RE[x]!= CH[x]) if CH[x] == 1, the mismatch is ignored or conversely, if any CH[x] == 0 on a mismatch, the codes do not match. As more of the response bits are erased, there are fewer allowed mismatches where the CH is 1. Since this scheme is based on erase monotonicity, we call it MON1.

At the bottom of Fig. 7 it is evident when using this scheme that many mismatching bits, i.e., with CH[x] == 0, are 1's in the inter-class response. Two inter-class representatives labeled Case B and Case C are shown.

The circuit to implement this match determination helper function is indeed simple, as shown in Fig. 8 for 256 bit codes. Our initial approach was a logical OR of the CG bit compare results (the far right aggregation circuit in Fig. 8). Any CG = 1 would indicate a mismatch. This was suggested by experiments on “virgin” devices that never had an intra-class code CH 0 to RE 1 response. However, aging showed some mismatching bits (see Section 5.3) and subsequent experiments with other devices that had been used extensively, but not purposely aged, show cases that violate this rule. This leads to summing the CG bits and comparing the result to a predefined threshold as shown in Fig. 8. While 8 bits are required for 256 bits, 3 or 4 should be adequate, as we have not seen more than 4 bit mismatches in this MON1 scheme.

5.2. Code distances and their evolution

For the measurements in this section, the total number of erased bits is captured for a 512 Kb block. While it is possible to erase sectors, and thus tune each code word, we purposely keep the experiments at the large block count as the coarseness makes the likelihood of breaking the scheme greater (it was also expedient). We ensure the monotonicity of erased bits by avoiding the shallow portion of the curve in Fig. 3 by choosing the voltage to be high enough that the 50% point is below the erase saturation. While bits may still fluctuate if they are at the input referred offset of the sense circuits [14], we have not seen such fluctuations until the flat portion of the curve in over 4 M bits. In the shallow region, only a few hundred fluctuating bits have been observed on multiple DUTs.

The distribution of intra-code CG sums comprises Fig. 9. The values are increased by erasing more response bits. A threshold of 8–10 is sufficient to avoid an incorrect match even with early termination. Fig. 10 shows time evolution of the intra-class code Hamming distances for 256-bit codes over multiple erase cycle, this time labeled by the number of erased bits, since we skipped cycles for brevity. In the cases of 170k, 200k, and 227k erased bits, the mismatching bits are a 1 in the challenge and a 0 in the response. As more bits are erased, the Hamming distances decrease since more mismatching 0s in the response are erased to 1s. Obviously, continuing until more bits are erased in the response has a positive effect. However, as the number of bits erased approaches, and certainly as it surpasses that of the challenge, the MON1 scheme begins to fail.

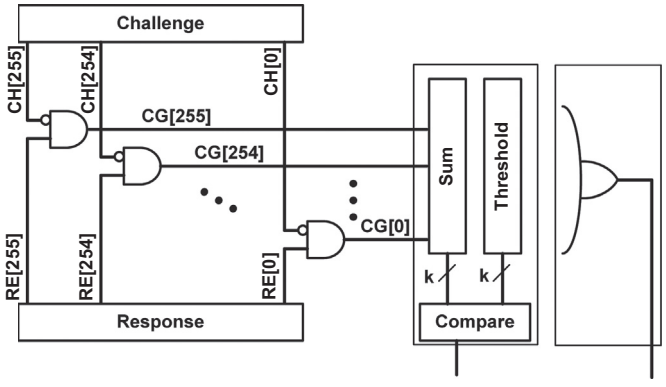

Fig. 8. Simple compare circuit to implement MON1. At right is the initial simple approach simply ORing the CG bit compares [21] but a population count (labeled Sum) is required as non-virgin and aged devices can produce a few mismatching bits with a 0 in the challenge.

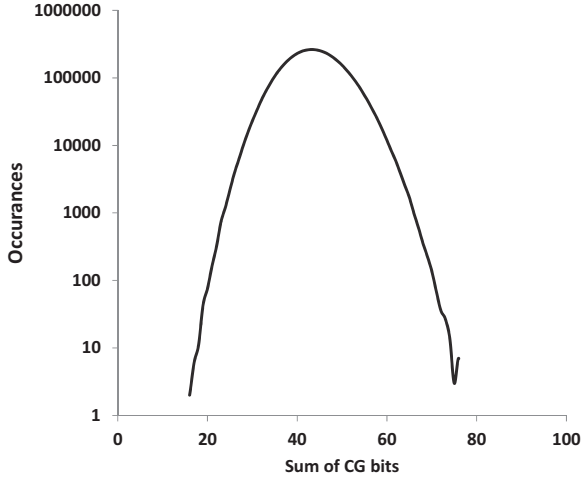

Fig. 9. Distribution (log ordinate scale) of intra-code CG bit sums at about 20% fewer erased bits in the response. The minimum count is 15 for this DUT.

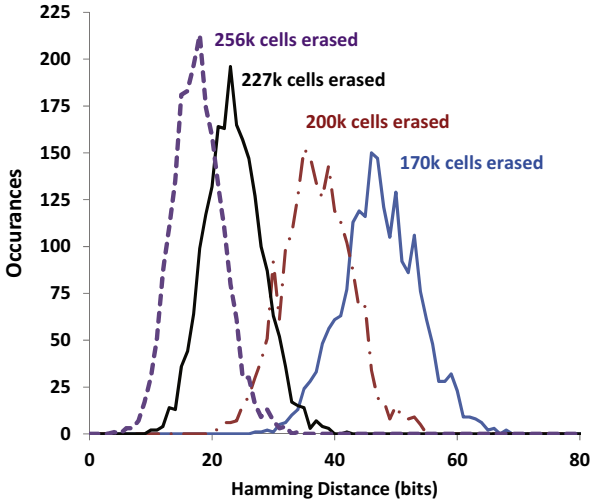

Fig. 10. Time evolution of the intra-code Hamming distances with the count of erased bits as a parameter. This DUT is heavily used but not aged.

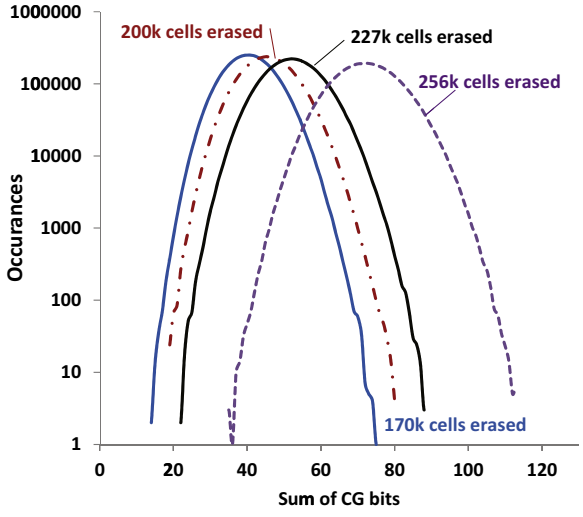

Fig. 11. Time evolution of the inter-code CG sum with the count of response total erased bits as a parameter. This is the same DUT as in Fig. 10.

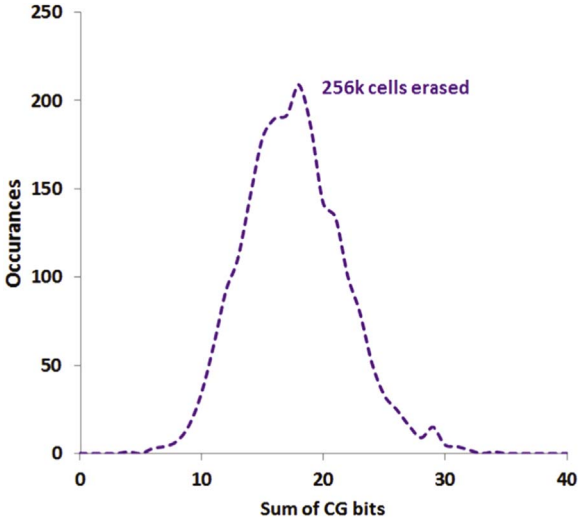

Fig. 12. Time evolution of the intra-code CG sum with the count of response erased bits as a parameter. This is the same DUT as in Fig. 10.

Fig. 11 shows the CG bit count by erased bits in the response. Recall that the challenge is generated by stopping the erase as close to 50% 1's (erased bits) as possible, but always greater than 50%. Maximizing the CG sum helps avoid a false match, so continuing seems to be useful. However, as Fig. 12 shows, at roughly equal challenge and erased response bits, the MON1 scheme has broken down. While at lower erase levels the sum of CG bits for intra-class codes is identically 0, it is not at 256k cells erased. The maximum of intra-class has increased to essentially reach the inter-class minimum. This underscores the importance of not over-erasing the response. If the response is generated at a small erase granularity, it may be possible to store a number of them as the 50% point is approached and then pick the appropriate one.

5.3. Aging impact

Delay PUFs and some SRAM PUFs are substantially impacted by process, voltage and temperature (PVT) variations [7,17,39]. Temperature greatly affects delay, some SRAM, and of course DRAM based designs [10,14]. This is important since many IoT and certainly automotive applications require wide temperature range. The F-N tunneling rate is not significantly impacted by temperature and we saw no significant temperature impact in experiments varying it.

We also investigated how the codes (IDs) withstood repeated program/erase cycles. Embedded flash allows saving the generated code post-helper algorithm directly, this violates the premise of a PUF. Ideally, if program/erase stress does not impact the response, then any flash sector (the minimum erase block) can be used effectively. This would maximize the possible codes. Of course for a TRNG, small changes are not important.

We stressed two DUTs with 10k full high voltage program/erase cycles. The post-stress MON1 code distances are very similar to the responses shown earlier. However, as Fig. 13 shows, there are some intra-code bits that deviate from our MON1 scheme, i.e., some bits exhibit non-monotonic behavior when compared to the pre-stress challenge code increasing the CG sum above zero for about 10% of the code words. This can be dealt with by measuring misses to be a count above some threshold as suggested in Section 5.1. We observed similar behavior when generating the challenge from a well-used or stressed DUT too.

5.4. Analysis

The behavior of the SST flash memory allows a simple helper function since nearly all of the mismatching bits are necessarily un-erased 0's. This follows from the fact that the erase is monotonically, albeit discretely, slowed but never accelerated by the aging, as pointed out in Section 3.2. However, whether or not trapping near the high field EG tip occurs is random. Consequently, a given bit may be slowed

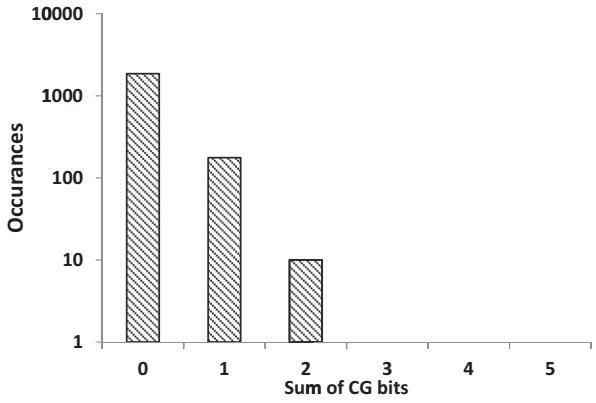

Fig. 13. Aged intra-code CG sum. Some bits exhibit early erase.

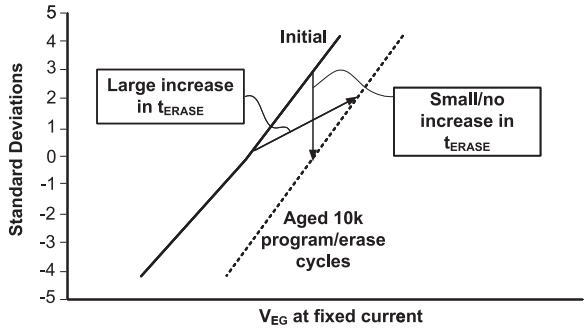

Fig. 14. Bits can erase earlier since a challenge bit may not slow down significantly in erase speed, while another may. The post-stress distribution is nearly Gaussian, but bits do not uniformly degrade in erase speed.

significantly, while another may not. Assuming the original oxide thickness is normally distributed—it is close as shown in Fig. 2(c) and the likelihood of trap formation is also normally distributed then the result of their addition is normally distributed, as also apparent from Fig. 2(c). The reason for some fails in the original scheme is due to bits passing each other. This is illustrated in Fig. 14 and is a consequence of bits not aging uniformly. That some bits may not produce a trap with aging is evident by the overlap of the pre- and post-erase distributions.

Fig. 15 shows the correlation between the pre-aged time to erase and that post-aged for 8 kb. Since the erase rates of two trials are not calibrated, we guard band by three partial erase operations. There are 8153 data points above the red line, which represents the passing zone for the simple MON1 (ORing CGs) and 39 below. Note also that the 50% point is greatly exceeded. Stopping before the 50% erase point allows only 16 points below the failure line, i.e., 32 codes with a 'bad' bit.

6. True random number generation

In this section the use of the SST flash for TRNG is investigated.

6.1. Raw erase data

The max-entropy was determined using the Unix *gzip* program, where the compression ratio estimates the max-entropy. In all cases, no compression was provided, and analysis of the zip file header showed

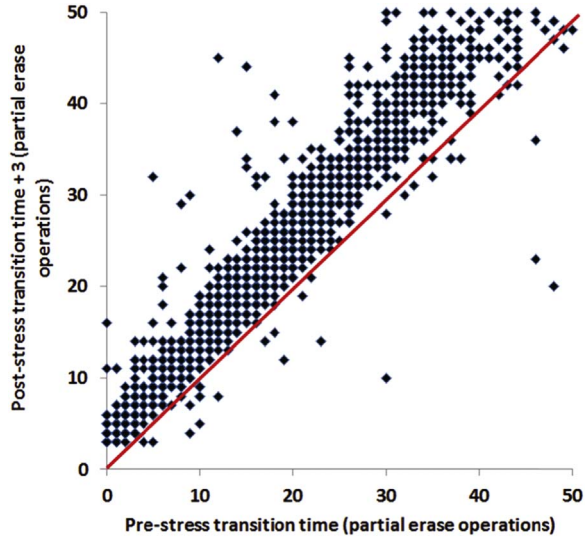

Fig. 15. Pre and post-stress erase time correlation for 8k bits. While less than 1% of bits erase faster post-stress than at pre-stress, many 256 bit code words are affected.

that the resulting file was uncompressed. Thus, we estimate the max-entropy to be unity. The min-entropy (Eq. (2)) was calculated for the raw flash data. The min-entropy for all blocks on all DUTs tested was greater than 0.9. This min-entropy is considerably higher than the 0.75–0.76 reported for SRAM power-up [8–10].

We used the fifteen NIST tests to evaluate the raw output as true random numbers. The tests evaluate deviations from expected randomness and search for patterns, which systematic variations might produce if the bits are chosen in row order, as we did here. Most importantly, nearly 50:50 ones and zeros are required, specifically for the Frequency test, but also for others. This would require very small steps in the erase cycles. We have, with very careful V_{DD} selection for each device, achieved values within 2k matching in 500k bits (0.4%) but most experiments here use much coarser granularity (as mentioned) to ensure worst-case analysis. Using raw data as read from the 1.5-T memory by row, very few of the tests pass (Table 1 labeled *Raw*). Tests that were not run are left blank. For the 150 non-overlap template tests, we considered anything with less than 5 failures a pass (the fail counts are indicated in Table 1).

6.2. Helper functions to alleviate systematic variation

The strong column dependency of erased bits in Fig. 5 is evident to the eye, and the NIST tests detect as a pattern in the FFT tests causing them to fail. As mentioned, not all DUTs exhibited this strong dependency, though it was common. Interestingly, the NIST tests also failed on raw data for DUTs that did not exhibit the strong column dependence. While this column dependence did not adversely affect the MON1 IC fingerprint scheme it does indicate that the entropy is less than perfect, and that use as a true random number generator will require helper data algorithms to ensure better one vs. zero distribution and counteract such systematic artifacts. We investigated a number helper functions to improve the 1.5-T flash response with the goal of passing the NIST tests for TRNG application. Our secondary goal is to ensure that the helper functions can be implemented in synthesized logic.

The cause of most of the raw data NIST test failures is attributable to there being quite unequal numbers of ones and zeros, failing unless the erase could be stopped before the array was 51% ones. The use of RTN instability for a flash memory TRNG above used Von Neumann conditioning to avoid long strings of ones or zeros. As mentioned Von Neumann conditioning also naturally whitens the distribution to be an even number of ones and zeros. However, we were uncomfortable with the fact that this conditioning approach necessarily throws out at least half of the available bits.

Given the good initial entropy from the flash memory itself, our initial scheme inverts every other bit to whiten the string distribution. Since an input bit has a 50% likelihood of being inverted, the resulting string has very nearly 50:50 one vs. zero distribution. This did remove the obvious column dependencies (Fig. 16 red squares). The resulting strings, labeled *EOinvert* in Table 1, pass more tests, but not all. Specifically, FFT still fails. Only every other column is affected by our scheme, so the outer edge columns may retain more variability. Every other row at the ends now being either more 0's or more 1's in an even/odd row predictable fashion is also problematic.

Our next scheme Scramble uses a pseudo-random number generator to change the bit order of each *EOinvert* generated 256 bit code word. This passes all the tests but is not a simple hardware solution, since it requires a large buffer memory. *Scramble16* performs the same function with 16 bit groups. We surmise that the Runs test fails this helper algorithm since the edge 0–1 frequency slopes are retained with this scheme, as the first and last 70 or so bits show strong roll off (see Fig. 5). The *Complex* helper data algorithm drops the problematic columns at the outer edges (and some in the array middle columns) and interleaves two bits at a time from two 512k-bit arrays with bits from the second array inverted. It passes all of the tests. The H_{min} of

Table 1
Results from applying the National Institute of Standards randomness tests on the 1.5-T TRNG generated strings with different helper functions.

	Raw		EOinvert		Scramble16		Complex		RandomInvert		LFSRinvert	
Test	Result	P-value	Result	P-value	Result	P-value	Result	P-value	Result	P-value	Result	P-value
Approximate Entropy	Fail	0.0000	Fail	0.0008	Pass	0.1535	Pass	0.0821	Pass	0.4311	Pass	0.3343
Block Frequency	Fail	0.0000	Fail	0.0000	Pass	0.8944	Pass	0.4511	Pass	0.0974	Pass	0.3426
Cumulative Sums	Fail	0.0000	Pass	0.0534	Pass	0.0769	Pass	0.0180	Pass	0.2572	Pass	0.5643
FFT	Fail	0.0008	Fail	0.0051	Pass	0.0193	Pass	0.9345	Pass	0.9957	Pass	0.6605
Frequency	Fail	0.0000	Pass	0.0621	Pass	0.0579	Pass	0.0405	Pass	0.2631	Pass	0.5857
Linear Complexity	Pass	0.2571	Pass	0.5396	Pass	0.8514	Pass	0.7571	Pass	0.8769	Pass	0.0921
Longest Run	Fail	0.0000	Pass	0.6835	Pass	0.1212	Pass	0.9080	Pass	0.2281	Pass	0.9058
Non-overlapping Template	Fail	0.0000	Pass	0.2971	Pass	0.5293	Pass	0.6034	Pass	0.6427	Pass	0.3379
Tests failing	78		4		1		0		2		2	
Overlapping Template	Fail	0.0000	Pass	0.8619	Pass	0.4643	Pass	0.7182	Pass	0.2598	Pass	0.3623
Random Excursions	N/A		Pass	0.3373	Pass	0.5350	Pass	0.7597	Pass	0.1826	Pass	0.9087
Rand. Excursions Variant	N/A		Pass	0.0999	Pass	0.9253	Pass	0.4287	Pass	0.6224	Pass	0.7488
Rank	Pass	0.9844	Pass	0.0639	Pass	0.2826	Pass	0.8733	Pass	0.3946	Pass	0.7498
Runs	N/A		Fail	0.0000	Fail	0.0000	Pass	0.0816	Pass	0.3379	Pass	0.3006
Serial	Pass	0.2779	Pass	0.2936	Pass	0.2065	Pass	0,1945	Pass	0.5551	Pass	0.3489
Universal	Fail	0.0000	Pass	0.7176	Pass	0.7527	Pass	0.3336	Pass	0.6597	Pass	0.2716

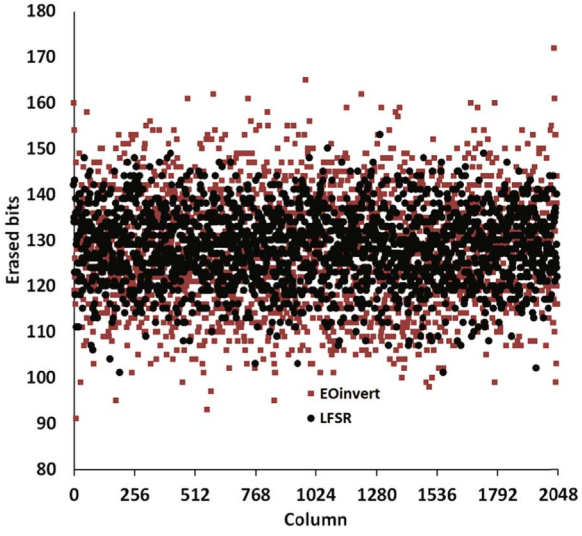

Fig. 16. Variability in the number of 1's by column after data whitening. *EOinvert* removes the overall column dependency but does not pass the NIST test. *LFSRinvert* exhibits a tighter spread and does pass. (For interpretation of the references to color in this figure, the reader is referred to the web version of this article).

this result is over 0.99 and the Q-Q plot of the intra-class codes is nearly ideal [21]. While not retaining all bits, it preserves over 90%.

Realizing that the logic to implement scrambling functions is potentially large and wire limited, we turned attention to which bits are inverted. *RandomInvert* uses a pseudo-random number generator (a Perl call to Unix *Rand*) to decide which bits to invert (with 50:50 behavior). It passes all the tests as evident in the table. This result suggested the final very simple procedure *LFSRinvert*. This scheme inverts bits in the raw string based on the LSB of a 127 cycle linear feedback shift register (LFSR). This relatively simple helper function provides good column by column 1:0 uniformity as evident in Fig. 16 (the black dots). It also passes all NIST tests, demonstrating the good intrinsic 1.5-T flash entropy requires only a minimal helper data algorithm for nearly ideal results. The H_{min} of the resulting string is over 0.99 and the max-entropy is 1, again based on no compression by the *gzip* program.

6.3. Helper function circuit complexity

The *LFSRinvert* can also operate in parallel (at the length of the LFSR) to limit the number of cycles required to whiten the flash TRNG data. Longer LFSRs are effective and may be more useful in that manner. Implementations of the simpler data helpers, i.e., *EOinvert* and *Complex* require only 632 μm^2 and 9192 μm^2 of cell area on a 90-nm low standby power process, respectively. The interface is designed to match the embedded SST flash macro described earlier. The latter is large as it must keep track of where it is at in the flash row so columns can be discarded. The *EOinvert* can be so small since it also need not store more than the 32-bit word output from the macro. In either case, the helper logic is less than 1% of the 3.2 Mb flash macro size on the same process. Thus, we believe the helper functions can be readily implemented without significant IC cost.

7. Conclusions

This paper demonstrated that 1.5-T erase rate produces effective device IDs. The resulting codes, based on the as-fabricated tunnel oxide variability, are highly repeatable. Thus, repeated erase cycling is an effective PUF approach. The min-entropy of the 1.5-T flash memory used as a PUF as described is quite good, varying between 0.90 and 0.91 across DUTs. We believe that further work truly embedding it as a PUF with better voltage and timing control could improve this to near ideal. As it is, the large Hamming distance avoids bit-aliasing causing false-positives. Relatively poor SRAM PUF reliability has suggested soft-decision encoding [8]. Moreover, prior flash PUF schemes have been based on bit program ordering, but required complex bit conditioning and helper data logic. This paper has demonstrated a straightforward helper data algorithm for the 1.5-T flash memory PUF. High reliability has been demonstrated even after aging. The physical bases for the PUF mechanisms and good aging characteristics have also been described.

Weak PUFs have been suggested for choosing keys in cryptographic blocks [22]. Despite the good entropy characteristics, systematic variation causes the raw data from the 1.5-T flash memory to fail the NIST randomness tests. However, we have shown that multiple simple data helper (whitening) functions effectively alleviate these systematic offsets so the resulting codes pass the NIST tests. Consequently, with minimally sized additional logic, the memories can be effective TRNG circuits. Finally, we believe that the whitening schemes presented here should be effective for other memory based TRNG circuits, assuming they begin with good entropy characteristics.

Acknowledgment

This work was sponsored by the U.S. Dept. of Energy (Award Number DE-NE0000679).

References

[1] B. Gassend, D. Clarke, M. van Dijk, S. Devadas, Controlled physical random functions, in: Proceedings of the Computer Security Application Conference, 2002, pp. 149–160.

[2] G. Suh, S. Devadas, Physical unclonable functions for device authenticationand secret key generation, in: Proceedings of the Design Automation Conference, 2007.

[3] K. Lofstrom, W.R. Daasch, D. Taylor, IC identification circuit using device mismatch, in: Proceedings of the ISSCC, Feb. 2000, pp. 372–373.

[4] A. Vijayakumar, V.C. Patil, S. Kundu, On testing physically unclonable functions for uniqueness, in: Proceedings of the ISQED, May 2016, pp. 368–373.

[5] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, J. Schmidhuber, Modeling attacks on physical unclonable functions, in: Proceedings of the ACM Conference on Computer and Communications Security, 2010, pp. 237–249.

[6] J. Devaux, D. Gu, D. Schellekens, I. Verbauwhede, Helper data algorithms for PUF-based key generation: overview and analysis, IEEE Trans. CAD Int. Circ. Syst. 34 (6) (2015) 889–902.

[7] S. Eiroa, I. Baturone, A.J. Acosta, J. Dávila, Using physical unclonable functions for hardware authentication: a survey, in: Proceeings of the XXV Conference on Design of Circuits and Integrated Systems, 2010, pp. 204–209.

[8] R. Maes, P. Tuyls, I. Verbauwhede, A soft decision helper data algorithm for SRAM PUFs, in: Proceeings of the IEEE International Sympopsium on Information Theory, 2009, pp. 2101–2105.

[9] M. Hofer, C. Boehm, An alternative to error correction for SRAM-like PUFs, in: Proceeings of the International Workshop on Cryptographic Hardware and Embedded Systems, 2010, pp. 335–350.

[10] S. Chellappa, A. Dey, L.T. Clark, SRAM-based unique chip identifier techniques, IEEE Trans. VLSI Syst. 24 (4) (2016) 1213–1222.

[11] M. Claes, V. van der Leest, A. Breaken, Comparison of SRAM and FF PUF in 65 nm technology, in: Proceeings of the Nordic Conference on Secure IT Systems, Oct. 2011, pp. 47–64.

[12] J. Jang, S. Ghosh, Design and analysis of novel SRAM PUFs with embedded latch for robustness, in: Proceedings of the ISQED, May 2015, pp. 298–303.

[13] S.K. Mathew, S.K. Satpathy, M.A. Anders, H. Kaul, S.K. Hsu, A. Agarwal, G.K. Chen, R.J. Parker, R.K. Krishnamurthy, V. De, A 0.19 pJ/b PVT-variation-tolerant hybrid physically unclonable function circuit for 100% stable secure key generation in 22 nm CMOS, in: Proceedings of the ISSCC, Feb. 2014, pp. 278–279.

[14] S. Rosenblatt, S. Chellappa, A. Cestero, N. Robson, T. Kirihata, S.S. Iyer, A self-authenticating chip architecture using an intrinsic fingerprint of embedded DRAM, IEEE J. Solid-State Circ. 48 (11) (2013) 2934–2943.

[15] Y. Wang, W. Yu, S. Wu, G. Malysa, G.E. Suh, E.C. Kan, Flash memory for ubiquitous hardware security functions: true random number generation and device fingerprints, in: Proceedings of the IEEE Security and Privacy Symposium, 2012, pp. 33–47.

[16] P. Prabhu, A. Akel, L.M. Grupp, W.-.K.S. Yu, G.E. Suh, E. Kan, S. Swanson, Extracting device fingerprints from flash memory by exploiting physical variations, in: Proceedings of the International Conference on Trust and Trustworthy Computing, 2011, pp. 188–2010.

[17] S.Q. Xu, W. Yu, G.E. Suh, E.C. Kan, Understanding sources of variations in flash memory for physical unclonable functions, in: Proceedings of the IEEE International Memory Workshop, 2014.

[18] G.S. Rose, N. McDonald, L.-.K. Yan, B. Wysocki, A write-time based memristive PUF for hardware security applications, in: Proceedings of the International Conference on Computer-Aided Design, 2013, pp. 830–833.

[19] W.H. Choi, Y. Lv, J. Kim, A. Deshpande, G. Kang, J.-.P. Wang, C.H. Kim, A magnetic tunnel junction based true random number generator with conditional perturb and real-time output probability tracking, in: Proceedings of the IEEE International Electron Devices Meeting (IEDM), 2014, pp. 12.5.1–12.5.4.

[20] D. Holcomb, W. Burleson, K. Fu, Power-up SRAM state as an identifying fingerprint and source of true random numbers, IEEE Trans. Comp. 58 (9) (2009) 1198–1210.

[21] L.T. Clark, J. Adams, K.E. Holbert, Integrated circuit identification and true random numbers using 1.5-transistor flash memory, in: Proceedings of the International Symposium on Quality Electronic Design (ISQED), 2017.

[22] A. Maiti, I. Kim, P. Schaumont, A robust physical unclonable function with enhanced challenge-response set, IEEE Trans. Inf. Forensics Secur. 7 (1) (2015) 333–345.

[23] R. Maes, V. van der Leest, E. van der Sluis, F. Willems, Secure key generation from biased PUFs, in: Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems, Sept. 2015, pp. 517–534.

[24] I. Cicek, A.E. Pusane, G. Dundar, A novel design method for discrete time chaos based true random number generators, Integration 47 (1) (2014) 38–47.

[25] R. Micheloni, L. Crippa, M. Sangalli, G. Campardo, The flash memory read path: building blocks and critical aspects, Proc. IEEE 91 (4) (2003) 537–553.

[26] G. Molas, D. Deleruyelle, B. De Salvo, G. Ghibaudo, M. Gély, L. Perniola, D. Lafond, S. Deleonibus, Degradation of floating-gate memory reliability by few electron phenomena, IEEE Trans. Electr. Dev. 53 (10) (2006) 2610–2619.

[27] A. Scarpa, G. Tao, J. Dijkstra, F.G. Kuper, Reliability implications in advanced embedded two-transistor-Fowler–Nordheim-NOR flash memory devices, Solid-State Electron. 46 (11) (2002) 1765–1773.

[28] P. Cappelletti, A. Modelli, Flash memory reliability, in: P. Cappelletti, C. Golla, P. Olivo, E. Zanoni (Eds.), Flash Memories, Springer, New York, 1999, pp. 399–441.

[29] N. Amarasinghe, Z. Celik-Butler, P. Vasina, Characterization of oxide traps in 0.15 μm^2 MOSFETs using random telegraph signals, Microelectron. Reliab. 40 (11) (2000) 1875–1881.

[30] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, S. Vo, A statistical test suite for random and pseudorandom number generators for cryptographic applications, NIST Special Publication 800-22 Revision 1a, Apr. 2010.

[31] M. Alavi, M. Bohr, J. Hicks, M. Denham, A. Cassens, D. Douglas, M.-.C. Tsai, A PROM element based on salicide agglomeration of poly fuses in a CMOS logic process, in: Proceedings of the IEDM, Dec. 1997, pp. 855–858.

[32] N. Robson, J. Safran, C. Kothandaraman, A. Cestero, X. Chen, R. Rajeevakumar, A. Leslie, D. Moy, T. Kirihata, S. Iyer, Electronically programmable fuse (eFUSE): from memory redundancy to autonomic chips, in: Proceedings of the IEEE CICC, Sept. 2007, pp. 799–804.

[33] R. Glidden, C. Bockorick, S. Cooper, C. Diorio, D. Dressler, V. Gutnik, C. Hagen, D. Hara, T. Hass, T. Humes, J. Hyde, R. Oliver, O. Onen, A. Pesavento, K. Sundstrom, M. Thomas, Design of ultra-low-cost UHF RFID tags for supply chain applications, IEEE Commun. Mag. 42 (8) (2004) 140–151.

[34] Y. Tkachev, X. Liu, A. Kotov, V. Markov, A. Levi, Observations of single electron trapping/detrapping events in tunnel oxide of SuperFlash memory cell, in: Proceedings of the Non-Volatile Memory Technology Symposium, 2004.

[35] X. Liu, V. Markov, A. Kotov, T.N. Dang, A. Levi, I. Yue, A. Wang, R. Qian, Endurance characteristics of SuperFlash® memory, in: Proceedings of the International Conference on Solid-State and Integrated Circuit Technology, 2006, pp. 763–765.

[36] A. Kotov, Three generations of embedded SuperFlash split gate cell: scaling progress and challenges, in: Proceedings of the Memory Workshop, June 2013.

[37] S.M. Sze, Physics of Semiconductor Devices, John Wiley and Sons, New York, 1981.

[38] Y. Tkachev, X. Liu, A. Kotov, Floating-gate corner-enhanced poly-to-poly tunneling in split-gate flash memory cells, IEEE Trans. Electr. Dev. 59 (1) (2012) 5–11.

[39] G. Kömürcü, A.E. Pusane, G. Dündar, Effects of aging and compensation mechanisms in ordering based RO-PUFs, Integration 52 (2016) 71–76.