True Random Number Generation Using Latency Variations of FRAM

Md Imtiaz Rashid, *Member, IEEE*, Farah Ferdaus, *Student Member, IEEE*, B. M. S. Bahar Talukder, *Graduate Student Member, IEEE*, Paul Henny, *Student Member, IEEE*, Aubrey N. Beal, *Member, IEEE*, and Md Tauhidur Rahman, *Member, IEEE*

***Abstract*— True random number generation (TRNG) plays an important role in security applications and protocols. In this article, we propose an effective technique to generate a robust true random number using emerging, energy-efficient, nonvolatile, consumer-off-the-shelf (COTS) ferroelectric random access memory (FRAM) chips. In the proposed method, we extract inherent randomness from internal ferroelectric capacitors by exploiting latency variation across cells within FRAM. Hardware results and subsequent National Institute of Standards and Technology (NIST) statistical test suite (STS) testing indicate that the proposed latency-based TRNG is robust over a wide range of operating conditions at speeds of 6.23 Mb/s using COTS, silicon FRAM chips from Cypress Semiconductor Corporation.**

***Index Terms*— Ferroelectric random access memory (FRAM), true random number generation (TRNG), memory-based TRNG.**

I. Introduction

TRUE random number generation (TRNG) translates random physical phenomena into random digits and plays an important role in both cryptographic and noncryptographic applications [1]–[11]. The TRNG output has to be acceptably fast, nonuniform, unpredictable, and robust against a wide range of operating conditions. Existing TRNGs rely on different physical entropy sources, such as random telegraphic noise [5], thermal noise on resistors and capacitors [6], [7], phase jitter of oscillating signals [8], chaos [9], and others [10]. However, due to the high complexity of implementation and the requirement for additional circuitry, these proposals are less viable for resource-constrained embedded systems. Furthermore, some of these proposed methods are vulnerable to external effects, such as power supply voltage and temperature [10]. A feasible solution to these problems requires a robust and simple implementation of random number generation for low-cost and energy-efficient systems. Pseudorandom number generators (PRNGs) are alternatively used in modern systems because of their comparatively better feasibility of implementation and high generation speed. However, since they cannot provide true randomness, it possesses relatively reduced unpredictability and makes the system more vulnerable to security threats [4].

Manuscript received April 3, 2020; revised July 14, 2020; accepted August 4, 2020. Date of publication September 10, 2020; date of current version December 29, 2020. *(Corresponding author: Md Imtiaz Rashid.)*

The authors are with the Department of Electrical and Computer Engineering, The University of Alabama in Huntsville, Huntsville, AL 35805-1911 USA (e-mail: mr0119@uah.edu).

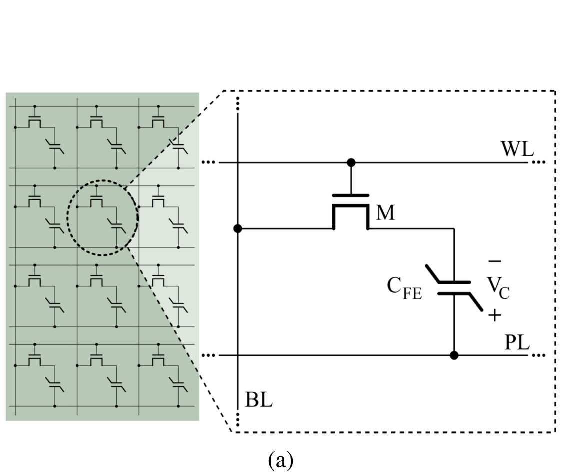
Color versions of one or more of the figures in this article are available online at https://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2020.3018998

The memory chip is an essential part of an embedded system, and FRAM, as a nonvolatile and fast memory, is best suited for low power systems. FRAM chip is an emerging memory technology due to its comparatively lower power usage, faster write performance, and high data retention time. Its usage is increasing in many applications such as MSP430 ultralow-power microcontrollers manufactured by Texas Instruments [43], MB95R203A general-purpose microcontrollers manufactured by Fujitsu Semiconductor Ltd [44], and so on. That is why FRAM is targeted to be used as a platform to generate true random numbers. There have been several techniques of generating a true random number from semiconductor memory chips, including both volatile memory and nonvolatile memory (NVM) [5], [11]–[18]. All these approaches exploit different sources of entropy to generate random numbers. However, most of the TRNGs using NVM (nonvolatile) memory chips are either not energy efficient or not implemented in commercially available memory chips or based on simulation [19], [20]. Ray and Milenkovic [5] proposed a Flash-based TRNG which has a comparatively low throughput. Huang *et al.* [14] proposed a contact-resistive random access memory-based TRNG that requires additional circuitry. Jiang *et al.* [15] presented TRNG based on stochastic diffusive memristor, which is not commercially available yet.

Techniques for creating random numbers by exploiting ferroelectric properties have been demonstrated [21]–[23]; however, the strategy that Mulaosmanovic *et al.* [21] described was not implemented on FRAM chip. Peeters *et al.*'s [22] proposed method that generates a random number from virgin cells. Importantly, Rodriguez *et al.* [23] showed the generation of a random number using 2T-2C FRAM memory.

These previous contributions provide the foundation for FRAM-based TRNGs, but there is some need for designs that: 1) are readily available through consumer-off-the-shelf (COTS) components; 2) require little-to-no additional hardware; 3) have some robustness to environmental fluctuations; and 4) provide a usable throughtput. In this article, we propose a technique to generate a random number that meets the aforementioned requirements by exploiting latency variations of off-the-shelf nonvolatile FRAM chips. The key contributions of this article are as follows.



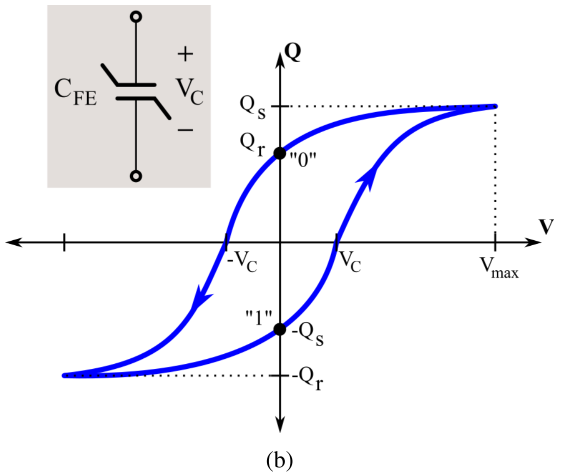


Fig. 1. (a) Simplified cell structure of 1T-1C FRAM. (b) Hysteresis characteristic of ferroelectric capacitor.

1) We violate the minimum chip enable (CE) active time during the read operation to create errors, which are used as a source of randomness. At the selected reduced timing parameter, the errors created from some of the memory cells are truly random. We propose an algorithm to select the most suitable cells that exhibit proper randomness in order to generate robust random numbers.
2) We demonstrate the robustness of the proposed TRNG in multiple off-the-shelf Cypress FRAM chips under a wide range of operating conditions.

The remainder of this article is organized as follows. Section II overviews the organization and operating principle of FRAM in brief. Section III expounds the proposed technique of true random numbers, including cell characterization and suitable bit selection algorithm. Section IV discusses the experimental setup, and the results for the procedures are delineated in Section III. It also includes an analysis of the quality of generated random numbers by evaluating the test results using the National Institute of Standards and Technology (NIST) test suite. Experimental results at various temperature and voltage conditions have been provided to substantiate the robustness of the proposed method. Finally, Section V concludes this article.

II. Fundamentals of FRAM

A. FRAM Architecture

The single-cell structure of a conventional single transistor–single capacitor (1T-1C) FRAM cell is similar to that of DRAM, except for a ferroelectric capacitor in place of a dielectric capacitor and a plate pine (PL) in place of ground connected at the positive end of the capacitor terminal. The simplified version of FRAM cell structure is shown in Fig. 1(a) [24]. The basic storage element of an FRAM is the ferroelectric capacitor. A ferroelectric capacitor is a capacitor that is based on ferroelectric material, such as lead zirconate titanate (PZT), instead of dielectric materials used in traditional capacitors. PZT has a perovskite crystal structure. The central atom of PZT has two equal and stable energy states. These states determine the polarization of the atom. If an electric field is applied in the proper direction, the atom will be polarized in one energy state and aligned according to the direction of the applied electric field. When the direction of the applied electric field is reversed, the atom will polarize from one energy state to another and, consequently, move in the opposite direction. Since both the energy states are stable, there remains a remnant polarization even when the applied electric field is removed. After that, if the applied electric field is increased in the opposite direction, the remnant polarization will diminish at a point, and then, the polarization will be reversed with the increment of an applied electric field. Similar behavior is observed for the opposite direction as well.

The polarization versus applied electric field forms a hysteresis loop [shown in Fig. 1(b)] [24]. The remnant charge, Q_r, the saturation charge, Q_s, and the coercive voltage, V_C are three significant parameters of a hysteresis loop. When the voltage across the capacitor is 0 V, the capacitor assumes one of the two stable states expressed by "0" or "1." The total charge stored on the capacitor is Q_r, for a "0," or $-Q_r$, for a "1." Both Q_r and $-Q_r$ are bound charges that cannot be released for sensing until a voltage pulse is applied to the ferroelectric capacitor. When voltage is applied to the terminal of the capacitor, its polarization may switch depending on the stored data. If it switches, the change in the total charge in the capacitor is $2Q_r + Q_s$, and if it does not, the change in charge will be $Q_s - Q_r$. Since

$$2Q_r + Q_s > Q_s - Q_r$$
$$\Rightarrow \Delta Q_{\text{switched}} > \Delta Q_{\text{unswitched}}.$$

As a result, the capacitance observed is different for switched and unswitched cases. Since by definition $C = dQ/dV$

$$C_{\text{swithced}} > C_{\text{unswitched}}. \quad (1)$$

B. FRAM Operation

FRAM stores data by exploiting the spontaneous polarization of the ferroelectric material of the ferroelectric capacitor. In order to write a "1" in a cell, the corresponding bitline (BL) is raised to V_{DD}, and wordline (WL) is raised to $V_{\text{DD}} + V_T$,

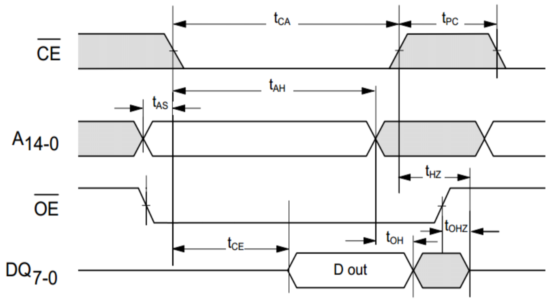


Fig. 2. Read cycle timing parameter for Cypress FM28V020 [42].

where V_T is the threshold voltage of the access transistor. Then, PL is pulsed. This allows a full $-V_{\mathrm{DD}}$ to appear across the ferroelectric capacitor for some time since the voltage convention is measured as shown in Fig. 1(a). In order to write “0” in the cell, BL is pulled down to the ground (GND) instead of raising to V_{DD}, and thus, the voltage applied across the capacitor is V_{DD} [24].

On the other hand, the read operation of a FRAM is destructive. Read access begins by precharging the BL to $0\ V$, followed by activating the WL and raising PL to V_{DD}. This establishes a capacitor divider consisting of C_{FE} and C_{BL} between the PL and the ground, where C_{FE} represents the capacitance of the ferroelectric capacitor, and C_{BL} represents the parasitic capacitance of the BL. Depending on the bit stored, the capacitance of ferroelectric capacitor can be approximated by C_0 and C_1. If the stored bit is 1, the polarization will switch when the PL is high. Therefore, C_1 will be greater than C_0 (1). The voltage developed on the BL (V_x) can be one of the following two values:

$$V_x = \begin{cases} V_0 = \dfrac{C_0}{C_0 + C_{\mathrm{BL}}} V_{\mathrm{DD}} \\ V_1 = \dfrac{C_1}{C_1 + C_{\mathrm{BL}}} V_{\mathrm{DD}}. \end{cases} \tag{2}$$

At this point, the sense amplifier is activated to drive the BL to full V_{DD} if the voltage developed on the BL is V_1 or to 0 V if the voltage on the BL is V_0. The WL is kept activated until the sensed voltage on the BL restores the original data back into the memory cell, and the BL is precharged back to 0 V.

The commercial FRAM manufacturer sets some timing parameters for the reliable operation of the memory chips. The timing diagram for the read operation of FM28V20 FRAM chip, manufactured by Cypress, is shown in Fig. 2.

Here,

t_{CE} = chip enable access time, i.e., the time to enable to chip operation after the CE pin is activated.

t_{CA} = chip enable active time, i.e., the time period for which the CE pin is kept activated.

There are many other timing parameters specified in the datasheet, but the two most important timing parameters are t_{CE} and t_{CA} from the perspective of this article.

III. Methodology

In the proposed methodology, we exploit the random ferroelectric switching at the reduced timing, manufacturer-recommended timing parameter, to generate random numbers. When the *chip enable active time*, t_{CA}, of FRAM (see Fig. 2) is reduced, it does not get enough time to develop stable voltages at all the internal transistors and nodes. Due to the process variation and this improper distribution of voltage within the chip, random variations are created in BL capacitance, ferroelectric capacitance, delay in access transistor, and other parameters as well. As a result, not all the cells can convey the stored information to the data bus at the same time. That is why the manufacturer specifies a set of timing parameters for the memory chip for the reliable read and write operations. Violating these manufacturer-recommended parameters renders faulty outputs during the read or write operation.

In general, the random process variations are originated from different steps involved in the fabrication process of modern memory chips, including lithography steps, etching, oxidation, ion implantation, chemical–mechanical polishing, and so on. Each of these process steps adds to the random variation by affecting the subsequent transistor parameters in differing manners [25]–[29] and is ultimately perturbed by microscopic thermal fluctuations from a heat bath [38]. These random variations can introduce a discrepancy in the delay of different access transistors and gates and have an impact on the BL capacitance as well. Furthermore, these discrepancy increases with the lowering of the applied voltage. Eisele *et al.* [30] showed that average delay variations of the low–high transitions and high–low transitions of a single inverter increase significantly with lower voltage. As a result, for the improper development of voltage throughout different nodes of the memory chip due to reduced timing parameters, the delay of each transistor and path will vary differently in an amplitude large enough to result in random data at the output.

Aside from the significant variation of access transistor parameters and BL capacitance, the improper distribution of voltage also affects the capacitance of the ferroelectric capacitors of each cell. The relationship between the capacitance of the ferroelectric capacitor and the applied voltage is nonlinear. Therefore, if the applied voltage of different capacitors varies from one another, the corresponding ferroelectric capacitance will not be the same even if the stored bits are equal. These variations in ferroelectric capacitance along with the variation in BL capacitance make the resultant RC delay different for different BLs, and hence, the data captured at sense amplifier will differ accordingly (2). Moreover, the power distribution to the cells connected with a PL is not uniform at the reduced latency. Consequently, if some of the cells store “1" and do not get sufficient power, not all the ferroelectric dipoles within the ferroelectric capacitors can be driven to polarize along the direction of the applied electric field. As a result, the net polarization may not be large enough to switch the capacitor state. Hence, an erroneous output may be detected by the sense amplifier. Since the distribution of power can be influenced by process variation and noise considerably with reduced timing, it can vary for a specific cell for multiple readings. As a consequence, the output of those cells for multiple readings will be unpredictable due to the aforementioned random processes.

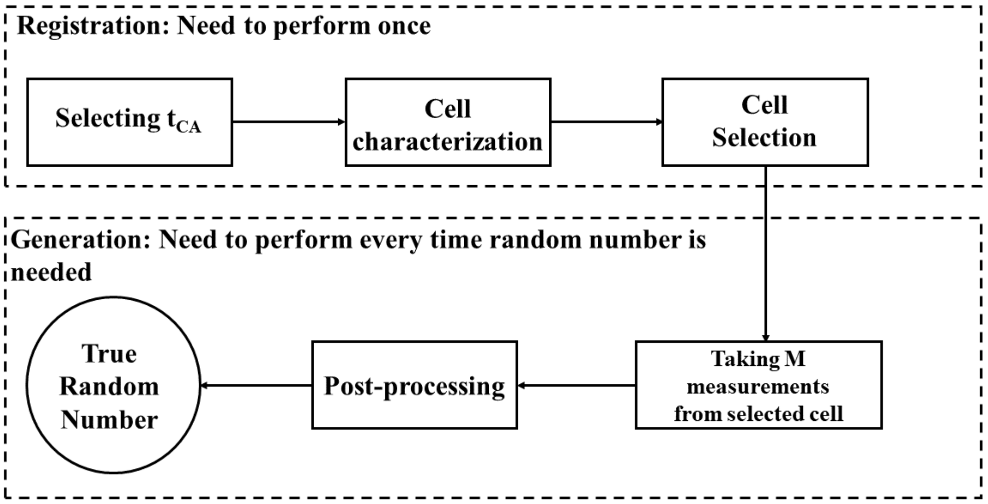

Registration: Need to perform once
Selecting t_{CA}
Cell characterization
Cell Selection
Generation: Need to perform every time random number is needed
True Random Number
Post-processing
Taking M measurements from selected cell

Fig. 3. True random number generation steps.

In order to generate true random numbers, the proposed methodology involves several steps. The flowchart in Fig. 3 shows the steps of generating a random number chronologically. In the registration phase, at first, we select reduced t_{CA} to introduce error during the read operation. Second, we characterize all the cells to understand the randomness behavior of FRAM cells at different t_{CA}s. Third, we use a cell selection algorithm to choose a FRAM cell that is used to generate robust random numbers. These three steps are required to be performed only once. For the future generation of random numbers, we collect data from multiple measurements from the selected cell and use a postprocessing scheme to generate high-quality random numbers. The registration and generation phases are distinguished in the dashed boxes of Fig. 3.

A. Selection of Timing Parameter

According to the timing diagram in Fig. 2, some of the cells may provide erroneous outputs if they are read at the reduced timing parameters. The number of these faulty cells varies within the cell activation time range $t = [0, t_{\mathrm{CE}})$. We change the t_{CA} and count the total number of erroneous output. The objective is to find a suitable t_{CA}, for which maximum erroneous output is achieved. In the next step, we devise an algorithm to select a suitable timing parameter for random number generation for any system through proper characterization. First, for all possible reduced timing parameters, the number of random cells is determined. The details about the selection of random cells are described in Section III-C. The timing parameter, for which the maximum number of random cells is observed, is selected to carry out the next steps to generate a random number.

B. Cell Characterization

Not all cells can be used to generate robust random numbers. Only entropic cells are of interest for this technique. To discover these random cells, at first, we characterize FRAM memory cells by writing a specific input data pattern to all memory cells. Then, we read whole memory chip N times at the reduced CE time t_{CA}. The value of N depends on the target length of the random output number. Larger N provides better results but slows down the characterization time. Based on the output of each cell from the sample readings, the memory cells can be classified into two categories.

1) Consistent Cells: These types of cells provide a stable output for all sample readings. These cells are fit for generating memory-based physically unclonable function [31]–[34] but not for random number generation because they manifest no randomness in their behavior at the reduced t_{CA}.

2) Inconsistent Cells: These kinds of cells provide different outputs for different readings. Some of these cells are found to be biased to a particular value either 0 or 1. Excluding those, the remaining inconsistent cells are appropriate for generating true random numbers.

C. Cell Selection for Random Number Generation

Suitable cells for TRNG should possess persistent high entropy and no bias over an increasing number of measurements (i.e., robust against aging). In order to select the random cells, first, we calculate the flip count, F_C for N measurements, where N is the number of measurements taken for the characterization process (Section III-B). F_c is a $1 \times N$ array containing the number of flips with respect to the input data for each of N measurements. The silicon results show that the F_C of suitable random cells follows a linear relation with the number of measurements. If the length of target random number is much larger, multiple measurements need to be taken from the random cell. In order to minimize the bias of inconsistent cells, we fit the F_C curve of the memory cells into a linear regression model from the N measurements. The cells for which the slope of the fitted model falls between a specified limit around 0.5 are selected to be the cells to generate a random number. This limit depends on the choice of N and the length of the target random number. The step-by-step procedure is shown in Algorithm 1. Each of these cells is capable of generating random numbers. If some of the random cells are spatially close, they may show spatial dependence in their output for a particular measurement due to sharing the

TABLE I
ANALYSIS OF RANDOMNESS BEFORE AND AFTER POSTPROCESSING

Operating Condition	**Normal**		**High Temperature**	
Test Condition	Before Post Processing	After Post Processing	Before Post Processing	After Post Processing
Number of NIST tests passed	15	15	8	15

Algorithm 1 Algorithm for Selecting Random Cells

```
procedure $select\_random\_cell(bitmap, input, UL, LL)$
1: /* $UL$ = Upper Limit of slope of fitted model to choose random cells */
2: /* $LL$ = Lower Limit of slope of fitted model to choose random cells */
3: /* $bitmap$ = $N \times num\_add \times num\_col$ matrix containing all the inconsistent cell data read at reduced $t_{CA}$ for all $N$ iterations */
4: /* $input$ = a $num\_add \times num\_col$ matrix containing input data stored to each memory cell */
5: $num\_add$ = total number of memory addresses containing inconsistent cells
6: $num\_col$ = number of inconsistent memory cells addressed by corresponding address
7: $rand\_add$ = [ ] /* Matrix to store the addresses of selected random cells */
8: $num\_randcell = 0$ /* Number of selected random cells */
9: for $i = 1$ to $num\_add$ do
10:    for $j = 1$ to $num\_col$ do
11:       $bitseq(i,j) = bitmap(:,i,j)$ /* bitstream from the output of the bit specified by (i,j) for N readings */
12:       $F_c(i,j,:) = xor(bitseq(i,j), input(i,j)))$
13:       $model = fit\_linear\_regrss\_model(F_c(i,j,:))$
14:       if $LL \leq model.slope \leq UL$ then
15:          $rand\_add.append([i,j])$
16:          $num\_randcell++$
17:       end if
18:    end for
19: end for
20: return $rand\_add$
end procedure
```

same BL or PL. However, for a single cell, spatial dependence will not have any effect on the temporal sequence of output. Therefore, the bitstream generated from a single random cell from multiple measurements shows no bias in the distribution of 0 and 1 s. The advantage of using a single cell is the low overhead of memory, which means that we will need to store the information of one memory cell only.

D. Postprocessing

Latency variation provides a truly entropic seed for postprocessing with secure hash algorithm (SHA)-256. This process conditions the random sequences much like other standard postprocessing schemes [40].

Adopting a cryptographic hash algorithm as a postprocessing technique to generate true random numbers is not uncommon. Talukder *et al.* [13] and Łoza and Matuszewski [45] proposed TRNG using secure hash algorithms. The advantage of using cryptographic hash algorithms, such as SHA-256, is that if the raw output gets biased under variations of operating conditions, one bit of flip can debias the entire output. Moreover, compared to the other debiasing technique such as the von Neumann algorithm or bit XORing, the SHA-256 algorithm usually requires fewer input bits to generate a random sequence with the same length entropy. In cryptography, the usage of hash algorithms is so common that modern hardware security modules, e.g., trusted platform module (TPM), include an embedded implementation of hash functions [47]. Moreover, most of the modern microprocessors have a separate instruction set to execute the SHA algorithm [46]. The hardware implementation of the SHA algorithm is much faster than software implementation [48], [49]. According to [49], SHA256 hardware acceleration on TM4C microcontroller running at 16-MHz frequency is experimentally found to be 4200% faster than the software implementation. As a consequence, devices that are concerned with security are likely to use any hardware security modules or secure cryptoprocessors [50], [51], and the application of hash algorithms is way faster than other common postprocessing techniques. In a system without an embedded cryptoprocessor or hardware security module, the software implementation will limit the throughput of the proposed scheme.

To generate a random number of target length N_T, at first, a selected random cell is to be chosen. After that, M sample readings at any chosen reduced t_{CA} are to be collected from the selected cell. The value of M depends on the target length and the postprocessing technique used. For SHA-256, $M = 512 \times \mathrm{ceil}(N_T/256)$. A bitstream, B_M, is created by concatenating the output of the random cell from these M sample readings. The B_M is chopped into $M/512$ chunks of length 512 bits. Next, cryptographic hash function SHA-256 is applied to each chunk of B_M, and the outputs for each chunk are concatenated and together to represent a true random number (see Algorithm 2). For example, if a 1024-bit random number is required, we will need to read 2048 times from a random cell at the reduced CE active time. These 2048 bits will be chopped into four 512-bit long sequences and will be fed to an SHA-256 algorithm to produce four sets of 256-bit random numbers. Concatenating these four numbers will give a total 1024-bit random number, as desired.

IV. Experimental Results and Analysis

We implemented the proposed methodology with a memory controller that changes the timing latency on the Alchitry Au

TABLE II
WORST CASE NIST TEST RESULT AT THE REDUCED t_{CA}

Model	FM18W08							
t_{CA}	15ns		25ns		30ns		35ns	
Result Type	p	S	p	S	p	S	p	S
Frequency	0.911413	10/10	0.534146	10/10	0.122325	10/10	0.534146	10/10
BlockFrequency	0.002043	10/10	0.350485	9/10	0.991468	10/10	0.739918	10/10
CumulativeSums	0.991468	10/10	0.739918	10/10	0.534146	10/10	0.350485	10/10
Runs	0.739918	10/10	0.739918	10/10	0.534146	10/10	0.739918	10/10
LongestRun	0.350485	10/10	0.739918	10/10	0.350485	10/10	0.911413	10/10
Rank	0.350485	10/10	0.122325	10/10	0.350485	10/10	0.035174	10/10
FFT	0.739918	10/10	0.911413	10/10	0.122325	10/10	0.739918	10/10
NonOverlappingTemplate	0.739918	9/10	0.534146	8/10	0.534146	9/10	0.739918	9/10
OverlappingTemplate	0.739918	9/10	0.534146	10/10	0.911413	10/10	0.350485	10/10
Universal	0.122325	9/10	0.122325	10/10	0.911413	10/10	0.911413	10/10
ApproximateEntropy	0.739918	10/10	0.739918	10/10	0.350485	10/10	0.350485	10/10
RandomExcursions	—-	8/8	—-	3/3	—-	7/7	—-	8/8
RandomExcursionsVariant	—-	8/8	—-	3/3	—-	6/7	—-	8/8
Serial	0.017912	9/10	0.739918	8/10	0.739918	10/10	0.534146	10/10
LinearComplexity	0.350485	10/10	0.739918	10/10	0.739918	10/10	0.739918	10/10

Algorithm 2 Algorithm for Generating Random Numbers

```
procedure $gen\_rand\_num(N_T, rand\_add)$
1:  /* $N_T$ = number of bits for target random number */
2:  /* $rand\_add$ = matrix containing the address of one selected random memory cell */
3:  $M = 512 \times ceil(N_T/256)$
4:  $bit\_string$ = [ ] /* Initializing bit stream from random memory cells */
5:  for $i = 1$ to $M$ do
6:      $x = read\_from\_rand\_add()$
7:      $bit\_string.append(x)$
8:  end for
9:  $rand\_string$ = [ ] /* matrix to store random number */
10: for $i = 1$ to $(M/512)$ do
11:     $B_M = bit\_string((i-1) \times 512 + 1 : i \times 512)$
12:     $post\_proc = sha256(B_M)$
13:     $rand\_string.append(post\_proc)$
14: end for
15: return $rand\_string$
end procedure
```

FPGA board [41]. The resolution of the timing parameter provided by the FPGA firmware was 5 ns. Due to the process variations, the controller path from different chips can create different delays. However, the slope of the fitted F_C line of the selected cell does not need to be precisely 0.5. Any cell that has a slope close to 0.5 generates a high-quality random number as well. Therefore, memory controllers have almost no effect on randomness. Data collected from different FPGAs show that the random number generated from FRAM chips is random. However, to achieve at least 15-ns resolution (selected $t_{\text{CA}} = 15$ ns, discussed in Section IV-A), the controller's minimum clock frequency should be 66.67 MHz. In order to analyze the applicability of the proposed method to different memory chips, we collected silicon data from two different FRAM memory models from Cypress, FM18W08, and FM28V020 and five from each type. We characterized the FRAM cells according to Section III-B with $N = 100$ readings. For the proposed random-cell selection algorithm, demonstrated in Algorithm 1, we used the upper and lower limit of the slope to be $0.5\% \pm 2\%$, respectively. The figure of merit for the generated random numbers is determined by the test result of the NIST statistical test suite (STS), one of the most commonly used and accepted test suits for randomness analysis [35]. It is a statistical package consisting of 15 tests. These tests are designed to inspect the quality of randomness of the binary sequence produced by hardware- or software-based cryptographic random number generators [35]. Tables II–IV show the results of the NIST test suits on random number generator with varied timing parameters, memory models, and different operating conditions, respectively. In the tables, the results are expressed in terms of p-value and S (score). The p-value represents the probability that a perfect random number generator would produce less random sequences than the sequence being tested [36]. A p-value of 1 (or 0) indicates that the sequence appears to be totally random (or nonrandom). When applied to a binary test sequence, all NIST tests result in one or more p-values depending on the number of subtests. Some tests, such as Runs, Random Excursions, and Random Excursions Variant, are not always applicable. These tests are applicable only if

TABLE III
WORST CASE NIST TEST RESULT FOR TWO DIFFERENT MEMORY MODELS

Model	FM18W08				FM28V020			
Sample Chip	Chip 1		Chip 2		Chip 1		Chip 2	
Result Type	p	S	p	S	p	S	p	S
Frequency	0.122325	10/10	0.213309	10/10	0.911413	10/10	0.739918	10/10
BlockFrequency	0.350485	10/10	0.122325	10/10	0.002043	10/10	0.739918	10/10
CumulativeSums	0.739918	10/10	0.534146	10/10	0.991468	10/10	0.534146	10/10
Runs	0.739918	10/10	0.739918	10/10	0.739918	10/10	0.350485	10/10
LongestRun	0.911413	10/10	0.739918	10/10	0.350485	10/10	0.911413	10/10
Rank	0.534146	10/10	0.213309	10/10	0.350485	10/10	0.213309	10/10
FFT	0.911413	10/10	0.350485	10/10	0.739918	10/10	0.122325	10/10
NonOverlappingTemplate	0.534146	8/10	0.739918	9/10	0.739918	9/10	0.739918	9/10
OverlappingTemplate	0.213309	9/10	0.534146	10/10	0.739918	9/10	0.122325	10/10
Universal	0.534146	10/10	0.122325	10/10	0.122325	9/10	0.739918	10/10
ApproximateEntropy	0.122325	10/10	0.122325	10/10	0.739918	10/10	0.350485	10/10
RandomExcursions	—-	7/7	—-	7/7	—-	8/8	—-	7/8
RandomExcursionsVariant	—-	7/7	—-	7/7	—-	8/8	—-	8/8
Serial	0.739918	10/10	0.534146	10/10	0.017912	9/10	0.035174	10/10
LinearComplexity	0.739918	10/10	0.350485	10/10	0.350485	10/10	0.739918	10/10

TABLE IV
WORST CASE NIST TEST RESULT FOR DIFFERENT OPERATING CONDITION

Model	FM18W08									
Operating Conditions	Normal Condition		High Voltage		Low Voltage		High Temperature		Low Temperature	
Result Type	p	S	p	S	p	S	p	S	p	S
Frequency	0.122325	10/10	0.911413	10/10	0.350485	9/10	0.534146	10/10	0.534146	10/10
BlockFrequency	0.350485	10/10	0.739918	9/10	0.911413	10/10	0.213309	10/10	0.534146	10/10
CumulativeSums	0.739918	10/10	0.739918	10/10	0.350485	9/10	0.911413	10/10	0.122325	10/10
Runs	0.739918	10/10	0.035174	10/10	0.534146	10/10	0.534146	10/10	0.350485	10/10
LongestRun	0.911413	10/10	0.534146	10/10	0.534146	10/10	0.213309	10/10	0.534146	10/10
Rank	0.534146	10/10	0.739918	10/10	0.350485	10/10	0.739918	10/10	0.534146	10/10
FFT	0.911413	10/10	0.122325	10/10	0.213309	10/10	0.350485	10/10	0.066882	10/10
NonOverlappingTemplate	0.534146	8/10	0.350485	9/10	0.739918	9/10	0.739918	8/10	0.534146	9/10
OverlappingTemplate	0.213309	9/10	0.739918	10/10	0.534146	9/10	0.534146	10/10	0.739918	10/10
Universal	0.534146	10/10	0.534146	10/10	0.122325	10/10	0.213309	9/10	0.350485	10/10
ApproximateEntropy	0.122325	10/10	0.066882	10/10	0.534146	10/10	0.739918	9/10	0.739918	10/10
RandomExcursions	-	7/7	-	8/8	-	7/7	-	7/7	-	8/8
RandomExcursionsVariant	-	7/7	-	8/8	-	7/7	-	7/7	-	8/8
Serial	0.739918	10/10	0.213309	10/10	0.066882	10/10	0.122325	10/10	0.739918	10/10
LinearComplexity	0.739918	10/10	0.066882	10/10	0.002043	10/10	0.739918	10/10	0.350485	10/10

the sequence meets certain criteria, such as the frequency test is passed and/or the number of cycles is greater than 500. If a test is not applicable, the resulting p-value is set to 0. If the p-value is 0 for all the subtests, the final result does not show any value in the p-value section. Score S, on the other hand, is the proportion of binary sequences that pass a test. The minimum pass rate for each statistical test with the exception of the random excursion (variant) test is approximately equal to 8 for a sample size of ten binary sequences. The binary sequences that pass all the NIST tests are considered to be random. The tables show the worst case NIST test results, i.e., for multiple scores of the same result type, the worst

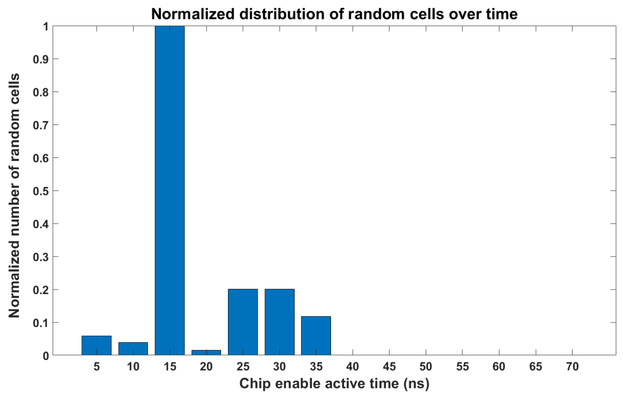


Fig. 4. Normalized distribution of the number of random cells with reduced t_{CA}.

one is presented. The experiment used ten bitstreams, each being larger than 10^6 for all chips and operating conditions. Table I shows the comparative analysis of the randomness of raw data and processed data. In normal operating conditions, the raw bitstream passes all the 15 NIST tests, but, in high temperature, the output of the selected cell tends to bias toward a particular value (either 0 or 1) and p-value decreases. That is why at high temperatures, it does not pass all the tests. With postprocessing, the output becomes robust against variation in operating conditions.

A. Selecting t_{CA}

The reduction of timing parameters depends on the driver clock frequency and system architecture. Random behavior is observed for a wide range of timing parameters. To compare the behavior of the random cells in different timing parameters, an analysis to determine the number of cells with random behavior was performed during the experiment. For the chips that we experimented on, the number of random cells falls within 3.81%–5.17% of all the memory cells. Since the number of random cells varies with different chips and random cell selection threshold, a normalized distribution of randomness, with respect to the maximum number of random cells, obtained for different t_{CA}s is shown in Fig. 4. According to Fig. 4, randomness is experienced in a comparatively large number of cells for $t_{\mathrm{CA}} = 15$ ns at a given 5-ns resolution for the experimental setup, which means that the randomness is best captured at a timing near 15 ns.

B. Chip Variability

A random number generation scheme has an increased prospect to be adopted for practical application if its applicability includes a varied range of memory models. To check the robustness of the proposed technique for chip variation, we analyzed the silicon data collected for two different models. The average number of selected random cells is observed to be different for these two memory chip models. This is because of the architectural variation, as well as the difference between interchip variations of the two models. However, the source of randomness, which is random process variation, applies to all the chips. Hence, random numbers can be generated with the proposed method for any memory chip. The random numbers generated from all of the experimented ten memory chips passed the NIST STS. The worst case test scores of three of each model are shown in Table III.

C. Temperature and Voltage Variations

Suitable random number generators must be robust against a wide range of operating conditions. In order to check the resilience of the proposed method under temperature and voltage variations, we analyzed silicon data collected at temperature ranging from 0 °C to 60 °C. We varied the operating V_{DD} by $\pm 5\%$ of nominal operating voltage. For high temperature and high voltage, the total number of random cells (see Section 1) is found to be comparatively less than the other operating conditions. The delay variation of the logic gates is reduced significantly at higher voltage, which results in the reduction of the number of random cells [30]. On the other hand, the mobility of charge carriers in Si decreases with increased temperature, depending on the doping concentration. When the charge carriers move slower, the propagation delay increases. At the same time, at a higher temperature, the threshold voltage decreases, which results in higher current and, therefore, a better delay. Hence, there is a competition between the two effects, and in general, the mobility effect is dominant [37]. Hence, the increased path delay may suppress the effect of random process variation and cause a decrease in the number of random bits. However, the proposed cell selection algorithm is robust, and therefore, the output is found to be robust for all the operating conditions. Table IV shows the NIST STS results at different operating conditions.

D. Throughput Analysis

The TRNG throughput of the proposed technique depends on the time required to read the memory cell at the reduced t_{CA} and the time required to execute SHA-256. Equation (3) represents the throughput of the proposed TRNG

$$T = \frac{N_T}{T_{\mathrm{mem}} + T_{\mathrm{SHA}}} \tag{3}$$

where

N_T	= length of target random number;
T_{mem}	= the time required to read a bitstream of length req_bits from memory, as in Algorithm 2;
T_{SHA}	= the time required to perform postprocessing on the bitstream read from memory, to generate a random number of length N_T.

In a secure device with embedded hardware security modules or secure cryptoprocessors, the implementation of SHA-256 is much faster than software implementation and other common postprocessing techniques. However, for a device without embedded hardware security module, software implementation is required, which limits the throughput of the proposed technique. The time required for implementing the SHA-256 algorithm is different for different processors. A statistical analysis of the performance of different cryptographic algorithms can be found at [39]. The time required

TABLE V
COMPARISON

Category	B. Ray et al. [5]	S. Chakraborty et al. [17]	Y. Wang et al. [18]	This Work
Memory technology	Flash	ReRAM and Flash	Flash	FRAM
Source of randomness	Read noise	Switching-time variability	RTN noise and thermal noise	Ferroelectric latency variability
Post-processing	Von- Neumann	XOR	Von- Neumann	SHA-256
Robustness against voltage fluctuation	–	–	–	Yes
Robustness against temperature variation	Yes	Yes	Yes	Yes
Number of Passed NIST tests	15	15	15	15
Throughput	1 Mbps	0.25 Kbp	11.48 Kbp	6.23 Mbps (at Intel Xeon Gold 6248)

for applying SHA-256 on a 512-bit-long input is, on average, 19.28 cycles/byte in the processor: Intel Xeon Gold 6248, CascadeLake architecture, 20×2500 MHz [39]. For a single processor, the required time would be 385.6 cycles, which is about 0.154 μs, neglecting the overhead for parallelization in multicores. In the experimental setup, in order to generate a random number of target length $N_T = 1024$, the average time required to take $2 \times 1024 = 2048$ measurements from a selected random cell is 174.08 μs for FM28V020 and 163.84 μs for FM18W08. Based on the performance of SHA-256 on the processor referred above, the average throughput of the proposed TRNG will be about 5.86 Mb/s on FM28V020 and 6.23 Mb/s, neglecting the communication overhead. This throughput is much higher than that of [5], [13], [17], and [18]. The works in [17] and [18] took the communication overhead into account while determining the throughput. Their works were implemented using FPGA with MicroBlaze softcore processor and ARM microprocessor, respectively. In recent years, several types of research have been performed on fast software hashing. Osvik [52] proposed a software implementation of the SHA-256 algorithm with a speed of 335 cycles/byte. Considering the minimum clock frequency, i.e., 66.67 MHz, the speed of SHA-256 will be 1.592 Mb/s. This will result in a minimum throughput of 1.285 Mb/s, which is still faster than [5], [13], [17], and [18]. The comparison of the results of the proposed techniques with other methods involving NVM is shown in Table V. The table demonstrates different NVM-based TRNGs and their corresponding source of randomness, NIST test results, and throughput, respectively.

E. Summary of Results

Overall, we draw the following main conclusions from the TRNG results.

1) The reduction of *chip enable active time* can be used to capture the entropy in FRAM chip.
2) The characterization and selection of cells are performed once in a lifetime during registration.
3) Only a single cell is used to generate random numbers. The location of this cell can be stored in an NVM memory. Therefore, the required storage is almost negligible.
4) The proposed TRNG is robust against a wide range of operating conditions and acceptably fast.

V. CONCLUSION

In this article, we proposed an efficient technique to generate true random numbers from commercially available nonvolatile FRAM chips by utilizing the variation of its internal latency. The NIST STS results tabulated in the result section validate that the proposed technique is truly random and robust against voltage and temperature variations. The throughput is also found to be relatively better than other TRNG techniques implemented in NVM chips.

REFERENCES

[1] M. T. Rahman, K. Xiao, D. Forte, X. Zhang, J. Shi, and M. Tehranipoor, "TI-TRNG: Technology independent true random number generator," in *Proc. 51st ACM/EDAC/IEEE Design Automat. Conf. (DAC)*, Jun. 2014, pp. 1–6.

[2] K. Marton, A. Suciu, and I. Ignat, "Randomness in digital cryptography: A survey," *Romanian J. Inf. Sci. Technol.*, vol. 13, no. 3, pp. 219–240, 2010.

[3] J. S. Kim, M. Patel, H. Hassan, L. Orosa, and O. Mutlu, "D-RaNGe: Using commodity DRAM devices to generate true random numbers with low latency and high throughput," in *Proc. IEEE Int. Symp. High Perform. Comput. Archit. (HPCA)*, Feb. 2019, pp. 582–595.

[4] J. Kelsey, B. Schneier, D. Wagner, and C. Hall, "Cryptanalytic attacks on pseudorandom number generators," in *Fast Software Encryption* (Lecture Notes in Computer Science), vol. 1372, S. Vaudenay, Ed. Berlin, Germany: Springer, 1998,

[5] B. Ray and A. Milenković, "True random number generation using read noise of flash memory cells," *IEEE Trans. Electron Devices*, vol. 65, no. 3, pp. 963–969, Mar. 2018.

[6] E. Kim, M. Lee, and J.-J. Kim, "8.2 8Mb/s 28Mb/mJ robust true-random-number generator in 65nm CMOS based on differential ring oscillator with feedback resistors," in *IEEE ISSCC Dig. Tech. Papers*, Feb. 2017, pp. 144–145.

[7] M. Drutarovsky and P. Galajda, "A robust chaos-based true random number generator embedded in reconfigurable switched-capacitor hardware," in *Proc. 17th Int. Conf. Radioelektronika*, Apr. 2007, pp. 1–6.

[8] E. Bejar, J. Saldana, E. Raygada, and C. Silva, "On the jitter-to-fast-clock-period ratio in oscillator-based true random number generators," in *Proc. 24th IEEE Int. Conf. Electron., Circuits Syst. (ICECS)*, Dec. 2017, pp. 243–246.

[9] X. Li, G. Zhang, and Y. Liao, "Chaos-based true random number generator using image," in *Proc. Int. Conf. Comput. Sci. Service Syst. (CSSS)*, Jun. 2011, pp. 2145–2147.

[10] F. Yu, L. Li, Q. Tang, S. Cai, Y. Song, and Q. Xu, "A survey on true random number generators based on chaos," *Discrete Dyn. Nature Soc.*, vol. 2019, Dec. 2019, Art. no. 2545123.

[11] M. T. Rahman, D. Forte, X. Wang, and M. Tehranipoor, "Enhancing noise sensitivity of embedded SRAMs for robust true random number generation in SoCs," in *Proc. IEEE Asian Hardw.-Oriented Secur. Trust (AsianHOST)*, Dec. 2016, pp. 1–6.

[12] I. Baturone, M. A. Prada-Delgado, and S. Eiroa, "Improved generation of identifiers, secret keys, and random numbers from SRAMs," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 12, pp. 2653–2668, Dec. 2015.

[13] B. M. S. Bahar Talukder, J. Kerns, B. Ray, T. Morris, and M. T. Rahman, "Exploiting DRAM latency variations for generating true random numbers," in *Proc. IEEE Int. Conf. Consum. Electron. (ICCE)*, Jan. 2019, pp. 1–6.

[14] C.-Y. Huang *et al.*, "A contact-resistive random-access-memory-based true random number generator," *IEEE Electron Device Lett.*, vol. 33, no. 8, pp. 1108–1110, Aug. 2012.

[15] H. Jiang *et al.*, "A novel true random number generator based on a stochastic diffusive memristor," *Nature Commun.*, vol. 8, no. 1, p. 882, Dec. 2017.

[16] C. Eckert, F. Tehranipoor, and J. A. Chandy, "DRNG: DRAM-based random number generation using its startup value behavior," in *Proc. IEEE 60th Int. Midwest Symp. Circuits Syst. (MWSCAS)*, Aug. 2017, pp. 1260–1263.

[17] S. Chakraborty, A. Garg, and M. Suri, "True random number generation from commodity NVM chips," *IEEE Trans. Electron Devices*, vol. 67, no. 3, pp. 888–894, Mar. 2020.

[18] Y. Wang, W.-K. Yu, S. Wu, G. Malysa, G. E. Suh, and E. C. Kan, "Flash memory for ubiquitous hardware security functions: True random number generation and device fingerprints," in *Proc. IEEE Symp. Secur. Privacy*, San Francisco, CA, USA, May 2012, pp. 33–47.

[19] S. Balatti *et al.*, "True random number generation by variability of resistive switching in oxide-based devices," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 5, no. 2, pp. 214–221, Jun. 2015.

[20] S. Sahay, M. Suri, A. Kumar, and V. Parmar, "Hybrid CMOS-OxRAM RNG circuits," in *Proc. IEEE 16th Int. Conf. Nanotechnol. (IEEE-NANO)*, Aug. 2016, pp. 393–396.

[21] H. Mulaosmanovic, T. Mikolajick, and S. Slesazeck, "Random number generation based on ferroelectric switching," *IEEE Electron Device Lett.*, vol. 39, no. 1, pp. 135–138, Jan. 2018.

[22] E. T. Peeters *et al.*, "Random number generation with ferroelectric random access memory," U.S. Patent 10 216 484 B2, Feb. 26, 2019.

[23] J. A. Rodriguez *et al.*, "Random number generation in ferroelectric random access memory (FRAM)," U.S. Patent 9 851 914 B2, Dec. 12, 2017.

[24] H. Ishiwara, M. Okyama, and Y. Arimoto, "Operation principle and circuit design issues," in *Ferroelectric Random Access Memories* (Topics in Applied Physics), vol. 93. Berlin, Germany: Springer-Verlag, 2004.

[25] S. K. Varanasi, "Study of effect of process variations on SRAM cell," M.S. thesis, Dept. Elect. Eng., Electron. Commun., Int. Inst. Inf. Technol., Hyderabad, India, Mar. 2020, pp. 16–20.

[26] A. Asenov, "Random dopant induced threshold voltage lowering and fluctuations in Sub-0.1m MOSFET's: A 3-D 'Atomistic' simulation study," *IEEE Trans. Electron Devices*, vol. 45, no. 12, pp. 2505–2513, Dec. 1998.

[27] J.-Y. Lee *et al.*, "Effect of line-edge roughness (LER) and line-width roughness (LWR) on sub-100 nm device performance," *Proc. SPIE*, vol. 5376, May 2004, Art. no. 534926, doi: 10.1117/12.534926.

[28] E. Gogolides, V. Constantoudis, G. P. Patsis, and A. Tserepi, "A review of line edge roughness and surface nanotexture resulting from patterning processes," *Microelectron. Eng.*, vol. 83, nos. 4–9, pp. 1067–1072, Apr./Sep. 2006.

[29] G. Palasantzas and J. T. M. De Hosson, "The effect of mound roughness on the electrical capacitance of a thin insulating film," *Solid State Commun.*, vol. 118, no. 4, pp. 203–206, Apr. 2001.

[30] M. Eisele, J. Berthold, D. Schmitt-Landsiedel, and R. Mahnkopf, "The impact of intra-die device parameter variations on path delays and on the design for yield of low voltage digital circuits," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 5, no. 4, pp. 360–368, Dec. 1997.

[31] B. M. S. Bahar Talukder, B. Ray, D. Forte, and M. T. Rahman, "PreLatPUF: Exploiting DRAM latency variations for generating robust device signatures," *IEEE Access*, vol. 7, pp. 81106–81120, 2019.

[32] M. T. Rahman, A. Hosey, Z. Guo, J. Carroll, D. Forte, and M. Tehranipoor, "Systematic correlation and cell neighborhood analysis of SRAM PUF for robust and unique key generation," *J. Hardw. Syst. Secur.*, vol. 1, no. 2, pp. 137–155, Jun. 2017.

[33] K. Xiao, M. T. Rahman, D. Forte, Y. Huang, M. Su, and M. Tehranipoor, "Bit selection algorithm suitable for high-volume production of SRAM-PUF," in *Proc. IEEE Int. Symp. Hardw.-Oriented Secur. Trust (HOST)*, May 2014, pp. 101–106.

[34] A. Mazady, M. T. Rahman, D. Forte, and M. Anwar, "Memristor PUF—A security primitive: Theory and experiment," *IEEE J. Emerg. Sel. Topics Circuits Syst.*, vol. 5, no. 2, pp. 222–229, Jun. 2015.

[35] L. E. Bassham, III, *et al.*, "Spp. 800-22 rev. 1a. A statistical test suite for random and pseudorandom number generators for cryptographic applications," Comput. Secur. Resource Center, NIST, Gaithersburg, MD, USA, Tech. Rep., 2010.

[36] M. Sys at al, "On the interpretation of results from the NIST statistical test suite," *Romanian J. Inf. Sci. Technol.*, vol. 18, no. 1, pp. 18–32, 2015.

[37] A. Johansson, "Investigation of typical 0.13 μm CMOS technology timing effects in a complex digital system on-chip," M.S. thesis, Dept. Electr. Eng., Linköping Univ., Linköping, Sweden. 2004. Accessed: Mar. 23, 2020. [Online]. Available: http://www.diva-portal.org/smash/

[38] C. A. Gobet and A. Knob, "Noise analysis of switched capacitor networks," *IEEE Trans. circuits Syst.*, vol. 30, no. 1, pp. 37–43, Jan. 1983.

[39] J. B. Daniel and T. Lange, Eds. *eBACS: ECRYPT Benchmarking of Cryptographic Systems*. Accessed: Mar. 23, 2020. [Online]. Available: https://bench.cr.yp.to

[40] S.-H. Kwok *et al.*, "A comparison of post-processing techniques for biased random number generators," in *Proc. IFIP Int. Workshop Inf. Secur. Theory Pract.* Berlin, Germany: Springer, 2011, pp. 175–190.

[41] *Alchitry FPGA Board*. Accessed: Mar. 21, 2020. [Online]. Available: https://alchitry.com/products/alchitry-au-fpga-development-board

[42] *Cypress FRAM FM28V020 Datasheet*. Accessed: Mar. 21, 2020. [Online]. Available: https://www.cypress.com/file/136591/download

[43] Texas Instruments. *MSP430 Ultra-Low-Power Microcontrollers*. Accessed: Mar. 21, 2020. [Online]. Available: https://www.ti.com/microcontrollers/msp430-ultra-low-power-mcus/overview.html

[44] Fujitsu Semiconductor Limited. *8-bit FRAM embedded Microcontrollers MB95R203A*. Accessed: Mar. 21, 2020. [Online]. Available: https://www.fujitsu.com/downloads/MICRO/fma/mcu/n712630.pdf

[45] S. Łoza and Ł. Matuszewski, "A true random number generator using ring oscillators and SHA-256 as post-processing," in *Proc. Int. Conf. Signals Electron. Syst. (ICSES)*, Sep. 2014, pp. 1–4.

[46] Intel Corporation. *Intel Advanced Encryption Standard (AES) New Instructions Set*. Accessed: Mar. 21, 2020. [Online]. Available: https://www.intel.com/content/dam/doc/white-paper/advanced-encryption-standard-new-instructions-set-paper.pdf

[47] D. Wooten, "Trusted platform module security," U.S. Patent 9 230 109 B2, Jan. 5, 2016.

[48] H. Mestiri, H. Mestiri, F. Kahri, B. Bouallegue, and M. Machhout, "Efficient FPGA hardware implementation of secure hash function SHA-2," *Int. J. Comput. Netw. Inf. Secur.*, vol. 7, no. 1, pp. 9–15, Dec. 2014.

[49] *Texas Instruments Forum*. Accessed: Jan. 9, 2014. [Online]. Available: https://e2e.ti.com/support/microcontrollers/other/f/908/t/313509?SHA256-Software-vs-Hardware-Comparison

[50] M. Larabel. (Aug. 2018). *Intel Open-Sources New TPM2 Software Stack*. photonix. [Online]. Available: https://www.phoronix.com/scan.php?page=news_item&px=Intel-New-Open-Source-TPM2

[51] T. Deneen, "The TPM2 software stack: Introducing a major open source release," Intel Developer Zone, Mountain View, VA, USA, Tech. Rep., Aug. 2018. Accessed: Jul. 11, 2020. [Online]. Available: https://software.intel.com/content/www/us/en/develop/blogs/tpm2-software-stack-open-source.html

[52] D. A. Osvik, "Fast embedded software hashing," École polytechnique fédérale de Lausanne, Lausanne, Switzerland, Tech. Rep. 2012/156, 2012. [Online]. Available: http://eprint.iacr.org/