On Improving Reliability of SRAM-Based Physically Unclonable Functions

Arunkumar Vijayakumar

Vinay C. Patil

Sandip Kundu

Journal of
Low Power Electronics and Applications

MDPI

Article

On Improving Reliability of SRAM-Based Physically Unclonable Functions

Arunkumar Vijayakumar, Vinay C. Patil * and Sandip Kundu

Department of Electrical and Computer Engineering, University of Massachusetts Amherst, Amherst, MA 01003, USA; avijayak@umass.edu (A.V.); kundu@umass.edu (S.K.)
* Correspondence: vcpatil@umass.com; Tel.: +1-413-404-7602

Academic Editors: Osnat Keren, Ilia Polian and Sanu Mathew
Received: 5 July 2016; Accepted: 5 January 2017; Published: 12 January 2017

Abstract: Physically unclonable functions (PUFs) have been touted for their inherent resistance to invasive attacks and low cost in providing a hardware root of trust for various security applications. SRAM PUFs in particular are popular in industry for key/ID generation. Due to intrinsic process variations, SRAM cells, ideally, tend to have the same start-up behavior. SRAM PUFs exploit this start-up behavior. Unfortunately, not all SRAM cells exhibit reliable start-up behavior due to noise susceptibility. Hence, design enhancements are needed for improving reliability. Some of the proposed enhancements in literature include fuzzy extraction, error-correcting codes and voting mechanisms. All enhancements involve a trade-off between area/power/performance overhead and PUF reliability. This paper presents a design enhancement technique for reliability that improves upon previous solutions. We present simulation results to quantify improvement in SRAM PUF reliability and efficiency. The proposed technique is shown to generate a 128-bit key in $\leq$0.2 µs at an area estimate of 4538 µm^2 with error rate as low as 10^{-6} for intrinsic error probability of 15%.

Keywords: physically unclonable function; SRAM PUF; reliability

1. Introduction

A computing system is built with multiple layers of abstraction from application to operating systems, to instruction set architecture and transistor level design. Generally speaking, a higher layer of abstraction such as application or OS is built atop a stack of abstracted layers beneath it which it must trust. The initial sources of this trust are known as *roots of trust*, which are typically hardware features. Such features must be small and secure by design to perform one or more security related functions such as verifying software, authenticating a device and protecting cryptographic keys. Silicon physically unclonable functions (PUFs) have been proposed as such roots of trust [1].

At a circuit level, a PUF harnesses device parametric variations to create a binary digital mapping that maps m-bit input challenge, c, to an n-bit response, r. The tuple (c, r) is called a challenge–response pair (CRP) of the PUF. PUFs with a limited challenge set, with low values of m, are called *Weak* PUFs [2], which are used for key generation. PUFs with exponential CRPs are called *Strong* PUFs [2] and primarily target authentication applications. Various circuits have been proposed to harness manufacturing process variations [3] to create Strong or Weak PUFs. SRAM based PUFs, which exploit start-up values in SRAM cells [4,5], and Arbiter PUFs [6], which exploit signal races, have received the most attention.

The salient properties that make a PUF attractive for security applications are its *tamper-proof* nature and *uniqueness*. It has been claimed that PUFs are tamper-proof because they exploit inherent disorder in the manufacturing process, and any attempt to break package or delayer metal interconnections will change that physical disorder [2]. *Uniqueness*, on the other hand, is harnessed by exploiting inherent variations of the semiconductor manufacturing process to create unique identifiers/keys. Thus, PUFs are unclonable by both the manufacturer and designer. These properties provide a major advantage over non-volatile memory based key storage, which has been shown to be vulnerable to physical attacks [7]. SRAM PUFs are the most popular Weak PUFs. An SRAM usually starts up in the same initial state upon power-up [4]. This state varies from chip to chip. This is the basis of SRAM PUF function [8].

Unfortunately, PUF circuit characteristics are affected by environmental variations, noise and aging. This impacts repeatability of the response, which is called *reliability*, in the context of PUFs. Reliability of PUFs is a key design concern in Weak PUFs, as the responses are typically used for cryptographic key generation (or identification) and need to be quite robust to prevent data corruption downstream.

The primary focus of this work is on improving the reliability of Weak PUFs. First, we propose a new voter based technique to reduce the error rate of the SRAM-based PUF responses by harnessing the statistical bias. Next, we present analytical results on error rate pertaining to the voter design. To enable a complete solution, we then propose a design for Design for Test (DFT)/testing based approach that capitalizes on the voter based characterization to improve overall reliability of the system. We have simulated example designs of the proposed method to measure error rates, area, performance, and yield of the proposed method, and compare it against prior approaches to demonstrate the benefits of this solution.

The paper is arranged as follows. In Section 2, we present the related background and literature. In Section 3, we present proposed voter method along with the related circuit design. In Section 4, we analyze our proposed method in greater detail using *random walk* models and also present DFT techniques to improve the overall reliability. In Section 5, we discuss the results and design examples and differentiate with related schemes. Finally, we conclude the paper in Section 6 with insight into future works.

2. Background

2.1. SRAM PUF and Reliability

SRAM cells that are constituent of embedded memories typically consist of cross-coupled inverters connected by access transistors. Figure 1 shows a typical 6-Transistor SRAM cell. Due to intrinsic process variations, an SRAM cell on start-up would, typically, consistently settle in either *logic-0* or *logic-1* values. The settlement state is determined by mismatch in process variations in the cell transistors. Settlement to consistent yet random states allow values from multiple cells to be collected for use as a key or identifier. An SRAM PUF is expected to produce this key each and every time during power-up operation. Unfortunately, noise during start up can impact the settlement state of the PUF resulting in unreliability. Specifically, cells with low mismatch due to process variations are more sensitive to noise than cells with greater mismatch. Cells with greater mismatch produce sufficient differential drive to overcome any impact of noise. Along with various noise sources, variations in ambient conditions and supply voltage and parametric changes due to aging of the transistors also impact reliability. As the properties of a PUF depend on process variations, long-term device degradation due to aging impacts PUF reliability. Reliability of a PUF is also influenced by ambient conditions, supply voltage and various sources of noise. Among the various noise sources, power supply noise, crosstalk, thermal noise, shot noise and random telegraphic noise play a role. In addition to environmental variations, aging of circuits due to negative-bias temperature instability (NBTI), hot-carrier injection (HCI) and time-dependent dielectric breakdown (TDDB) decreases life-time reliability of a PUF. Thus, a PUF is considered unreliable whenever it produces a response that is different from the enrolled ideal response.

As mentioned earlier, PUF circuits can be broadly classified as Weak and Strong PUFs. Weak PUFs such as SRAM PUFs typically have few CRPs, while Strong PUFs such as Arbiter PUFs have an exponentially large number of CRPs. A Strong PUF circuit may produce errors for certain CRPs.

Similarly, a Weak PUF may produce an identifier in which only a few bits of the identifier have a high error rate. Due to the difference in CRP count and usage model of Strong and Weak PUFs, the errors have to be handled differently. As Strong PUFs have exponential CRPs and are used for authentication applications, multiple challenges are applied and responses are obtained. Authentication using these error-prone responses can be carried-out successfully by setting a success threshold for responses. For example, a Strong PUF with 2% intrinsic error rate can be successfully authenticated if 97% responses are correct. In contrast, Weak PUF responses are typically used for cryptographic key generation (or identification) where the responses are expected to be extremely reliable. Hence, alternative solutions such as error-correcting codes are more critical for Weak PUFs. As the reliability requirements of Weak PUFs are more demanding, we focus on improving their reliability for the remainder of this paper.

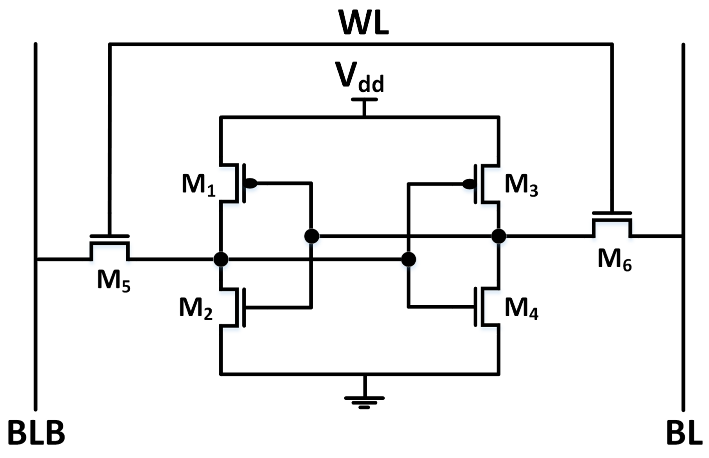


Figure 1. Traditional 6T SRAM Cell.

2.2. Related Literature

To derive a stable key from a string of bits where a few locations in the string are unreliable, a number of technology and algorithmic solutions have been proposed to improve the reliability of keys produced from SRAM PUFs. Since an ideal accuracy of 100% would require a large cost, typically, a low error rate such as an error rate of 10^{-6} is used as the design target [9,10]. The solutions are discussed below.

2.2.1. Error-Correcting Codes and Fuzzy Extractor

Error-correcting Codes (ECC) and Fuzzy Extractors have been proposed to correct errors in SRAM PUF [11–15]. Typically, these techniques create a helper data that is made public. The helper data is used to recover a key from the original data while minimizing the probability of error. This setup is shown in Figure 2. To derive a stable key of a certain length, one must start with a string that is longer in length, representing an overhead. In a fuzzy extractor, this overhead, along with processing overhead, increases rapidly with rising intrinsic bit error rate of SRAM PUFs. An SRAM PUF array with inherently low reliability requires more area and computation for error reduction using the above approaches. Hence, a reduction in error rate of the raw PUF cells would benefit in total area reduction. Apart from this overhead concern, such schemes should also ensure minimal information leakage from the helper data. These concerns have spurred an alternative line of research, which is described next.

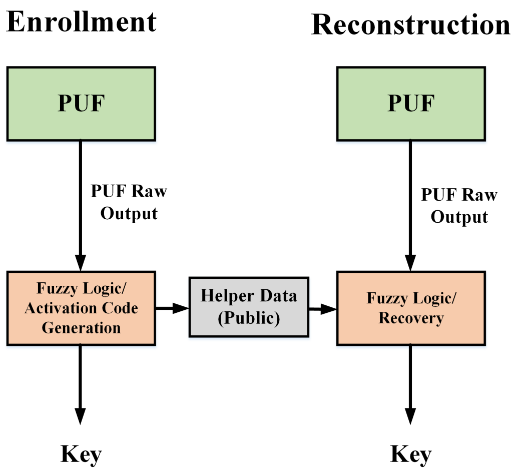


Figure 2. Enrollment procedure for SRAM PUFs.

2.2.2. Circuit and Manufacturing Technology Solutions

Several researchers have proposed technology and circuit solutions as an alternative to ECC. Mathew et al. have proposed a solution based on Temporal Majority Voting (TMV) and dark bits evaluation [10]. They also employ burn-in and aging effects to improve the PUF. Design changes to enable voting, along with uniqueness improvement through synchronous design, have also been proposed. However, the majority voting scheme proposed can only correct up to error rates of less than 8% stand-alone (with a 15-way voter), and additional techniques are mandatory to achieve practical application. Garg et al. have also proposed a method to improve the reliability of the PUF cells by aging the cells and increasing the mismatch [16]. Similarly, Bhargava et al. have proposed a technique to improve reliability through hot carrier injection (HCI) aging [17]. Maes et al. also investigate the effect on aging in SRAM PUFs and effectiveness of data-dependent strategies to improve reliability [18]. Jang and Ghosh [19] proposed an 8T SRAM PUF with a PMOS latch and a low-power 7T SRAM cell with an embedded Magnetic Tunnel Junction (MTJ) to enhance the reliability of the PUF in the presence of environmental fluctuations during multiple power-ups. The authors also focused on lowering the leakage power. Bucci and Luzzi [20] presented a differential circuit that captures the process mismatch and amplifies it to reduce the effect of noise. Once the difference has been sufficiently amplified, the differential outputs are latched to generate stable bits. Hofer et al. have proposed an alternative to error correction by pre-selecting the bit with greater mismatch to reduce error rate [21]. Böhm et al. implemented an SRAM PUF on a microcontroller and combined it with simple repetition codes to reduce the error rates. They also focused on checking the uniqueness of the PUFs and reliability over a wide range of temperatures. Cortez et al. have proposed a method by adapting voltage ramp-up time to ambient temperature to reduce the error rate of memory PUFs [22]. Though this technique improves the reliability, the auxiliary circuits needed for voltage ramp-up can be area intensive. In addition, shaping supply voltage is expensive for designs with large power delivery networks.

While many prior works have focused on circuit, design and manufacturing technology based solutions to improve reliability of SRAM PUFs, we seek a low-cost alternative for low-cost systems that avoids expensive technology solutions. We elaborate on this in the next sections.

3. Proposed Technique

In this section, we describe the preliminaries for the proposed method. Then, we present our approach and the design of associated circuits. We conclude this section with observations on the simulated reliability improvement.

3.1. Harnessing Statistical Bias for Improving Reliability

An SRAM PUF cell is expected to produce reliable value for key generation. If the relative drive strength of the cross-coupled inverters is low, then any noise present in the cell will determine the outcome of the PUF creating errors. When an SRAM cell exhibits errors during start-up, it is possible that if the cell is powered-up several times, it will exhibit a *statistical bias* (bias towards 0 or 1). That is, if we assume that the mean value of noise that gets coupled to the PUF cells is close to zero, multiple evaluation of the PUF's response would give the true statistical bias of the PUF response. Temporal Majority Voting (TMV) is one of the well-known techniques to extract such a bias in the presence of noise. If the statistical bias is strong, it may be detected using only a *limited* number of experiments. Contrarily, if the bias is weak, a much larger number of start-up experiments may be necessary to detect the true bias. A problem with a larger number of start-up experiments is that the associated circuits for bookkeeping will grow in size. Our proposed technique avoids this problem while extracting such a weak bias. Through simulation and analysis, we show that this technique is superior to traditional TMV.

3.2. Temporal Majority Voting

A simple way to reduce the error rate is using a Temporal Majority Voting scheme [10,23]. For example, a simple 4-bit counter based TMV counts from 0 to 15 and hence can be used as 15-way voting. If the resultant value after 15 evaluations of the cell's response is greater than 8, then the final value is classified as 1, or else it can be classified as 0. The concept of TMV has been discussed extensively in previous works by Mathew et al. and Xiao et al. [10,23]. The mathematical model of the TMV is a binomial counting process and, hence, the reduction in error rate can be calculated analytically. For example, a PUF cell whose error rate is $1 - p$, reduces to

$$P_e(N) = \sum_{m=k}^{N} \binom{N}{k} p^m (1-p)^{N-m}, \tag{1}$$

where $k = (N+1)/2$ (N is odd) when an N-way TMV is used [24]. The circuit implementation of the TMV typically consists of a n-bit counter, where $N = 2^n - 1$. The counter counts the number of ones; it is incremented by 1 if and only if the response from the SRAM cell is 1.

A major disadvantage of TMV is the high cost when the statistical bias of an SRAM PUF cell is weak. This will be explored further in Section 4.

3.3. New Voter Design

In this subsection, we present our voter design. Even though we demonstrate the proposed technique using SRAM-based PUFs, the technique is generic to use with other classes of Weak PUFs to improve their reliability. The proposed voter is based on an UP/DOWN counter (UDC). A simple counter starts at an initial value of count 0 and counts upwards. By contrast, the n-bit UP/DOWN counter used in this design starts at an initial value of $(2^n - 2)/2$. The counter value is incremented if the response from the current trial is 1, else (0) it is decreased. When the counter reaches a terminal value of 0 (or $2^n - 2$), the counter *saturates* and retains the terminal value.

The complete setup of the proposed voter scheme along with the PUFs is shown in Figure 3. The output of the PUF cell is used as input to the n-bit UP/DOWN counter. In the figure, we show a setup where an UP/DOWN counter is shared with four PUF cells, as an example. Starting at an initial counter value of $k = (2^n - 2)/2$, multiple trials are conducted until the n-bit UP/DOWN counter reaches terminal values of 0 or $2^n - 2$. Unlike TMV, where the number of trials is *fixed*, trials may continue *indefinitely* in an UP/DOWN counter until a terminal value is reached. When the counter reaches a terminal value of 0 (or $2^n - 2$), the value of the PUF cell is resolved as a *logic-0* (or *logic-1*). When a terminal value is reached, the trials are terminated, otherwise, in practice, the trials are continued for a pre-determined number of times. It is possible that no decision can be reached when the trials are terminated. The optimal values for n for varying error rates will be discussed in greater detail in Section 4.3. Multiple PUF cells can share a single UP/DOWN counter or each cell can be assigned an UP/DOWN counter, as done in previous designs [10]. The multiplexer is appropriately chosen (4:1 in Figure 3). As expected, sharing the counter across multiple cells would increase the run-time but reduce the area overhead.

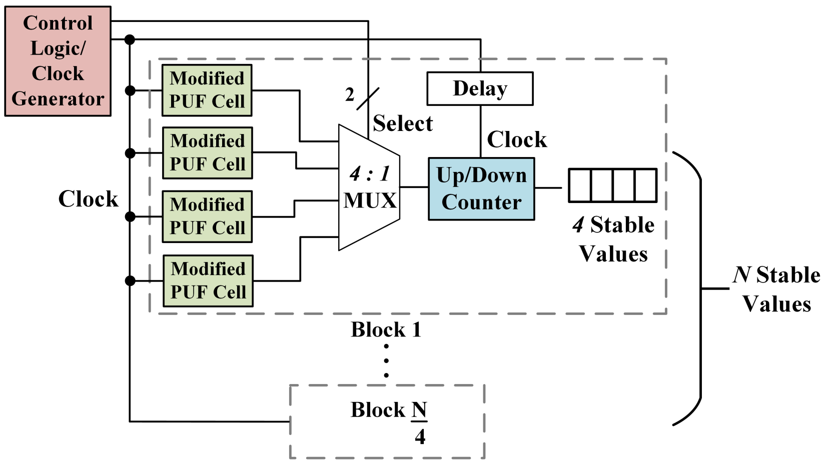


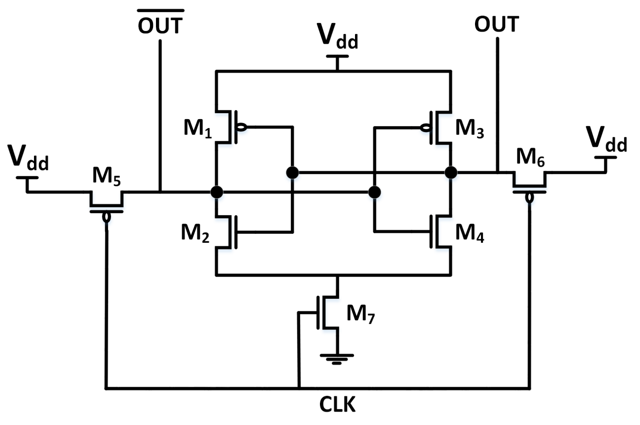
Figure 3. UP/DOWN counter (UDC) based voter scheme.

3.4. Circuit Design

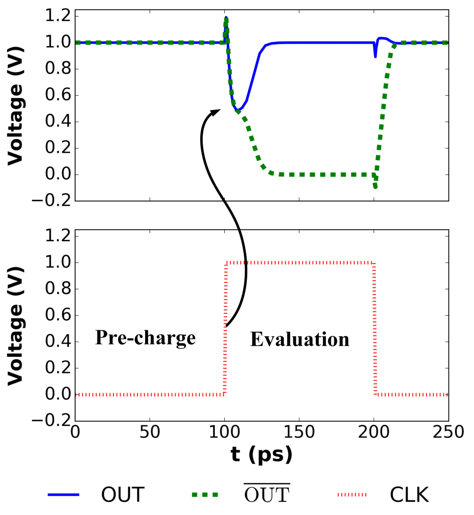
The voting process described above needs multiple evaluations of the PUF output, but multiple power-ups of the circuit are inefficient. Instead, the SRAM cell can be modified with minimal changes to implement the scheme. Instead of using start-up values during power-up, the circuit can be converted to a pre-charge/discharge circuit. This modified circuit is shown in Figure 4a. The clock signal first enables the paths from V_{dd} to OUT and $\overline{OUT}$ and pre-charges the nodes to the supply voltage. During evaluation, depending on process variation, the output will settle in *logic-1* or *logic-0* due to the mismatch in the strengths of the discharge paths. The timing diagrams are shown in Figure 4b. The cell output is registered by the counter, as shown in Figure 3, on the rising edge of the delayed clock signal. The clock delay is to allow the cell to settle to its stable value during evaluation. This design is similar to changes required for enabling TMV in related work [10], Sense-Amplifier PUFs [25] and meta-stability based true random number generators (TRNGs) [26]. The circuit changes above have minimal impact on the cell area. The UP/DOWN counter circuit can be implemented using flip-flops along with the required logic for initialization and saturation detection.

3.5. Error Rate from Simulation

We defer description of the details of simulation settings to Section 5. Here, we describe the methodology for obtaining error rate from simulation. We simulate a noisy PUF cell in SPICE using 45 nm technology [27] by randomly varying supply noise. The power supply noise distribution is varied to control the error rate of the SRAM cell. One million samples were collected from the cell. The outputs were fed into a 4-bit UP/DOWN counter, and the new, more reliable output bits were obtained. The new error rate was calculated for these bits to obtain the reliability improvement. As SPICE simulations are expensive, for an initial error rate of less than 0.24, we generated 10^8 bits with required initial error rate from a Python script and fed the bits to the UP/ DOWN counter. As shown in Figure 5, the error rate of the proposed technique obtained through simulation is less than 10^{-6} for an initial error rate of ≤ 0.16. We reserve the reliability improvement over traditional TMV for the next section. As the order of magnitude of the error rate obtained through simulation is in the range of 10^{-6}, a large number of simulations may be required to get an accurate estimate. In order to estimate the error rate with higher accuracy, analysis of the proposed technique is presented in the next section.



(**a**)



(**b**)

Figure 4. Modified Cell schematic and timing diagram. (**a**) modified SRAM cell for multiple evaluations; and (**b**) timing diagrams for bit evaluation.

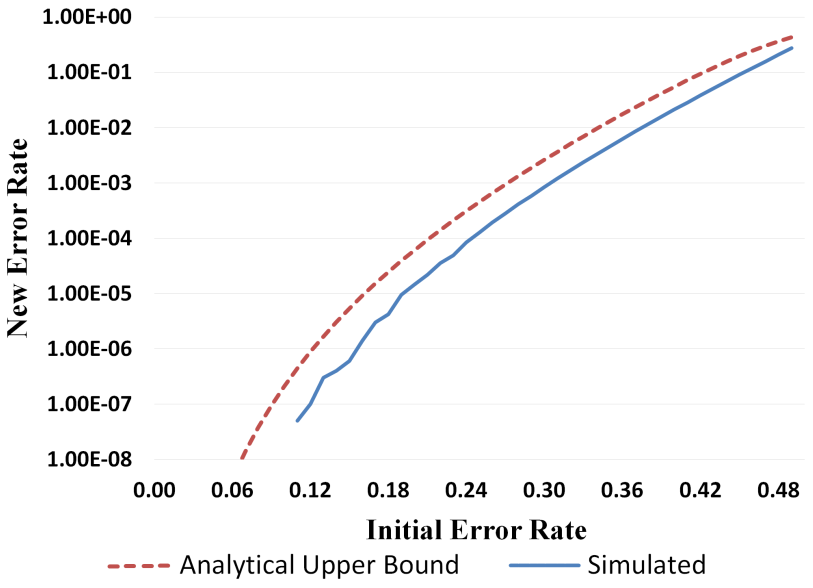


Figure 5. Error rate results from simulation.

4. Analysis of the Proposed Counter Based Design

In the following subsections, we describe the basics of the random walk model and use it to analytically evaluate the UP/DOWN counter scheme. In addition, we compare our approach to Temporal Majority Voting (TMV) and present a DFT scheme to improve overall reliability.

4.1. Operation of the Proposed Voter as Random Walk

Let $(U_1, U_2, \ldots)$ be a sequence of independent random variables from the set $\{1, -1\}$. Let the probability of value 1 from a trial be p, where $p \in [0, 1]$. Then, the probability of value -1 is $1 - p$. If X_N represents the sum of such sequence after N-trials, then

$$X_N = \sum_{i=1}^{N} U_i\,; \quad \text{where } N \text{ is the number of trials.} \tag{2}$$

The path traced by X_N is called a simple random walk [28]. This is an elementary one-dimensional random walk on an integer number line. The properties of random walk and associated problems are well studied and are related to the Markov process.

The UP/DOWN counter based scheme proposed above can be modeled and analyzed using random walk based models. For the purpose of analysis, without any loss in generality, let us assume that, in the absence of any noise, the PUF cell settles at *logic-1* due to process variation. In the presence of noise, there is a possibility that the cell will settle at *logic-0*, which is opposite to the inherent process variation of the cell.

In Table 1, we describe the notations used in our analysis of the UP/DOWN counter. In this analysis, we assume the noise experienced by the cell has zero mean; hence, for a statistical bias towards *logic-1*, p must be >0.5. If the cell has a strong bias towards *logic-1*, then $p \gg 0.5$; otherwise, for a weak bias, p has a value slightly above 0.5.

Table 1. Definition of *symbols*.

Symbols	Definitions
p	Probability of *logic-1* from a PUF cell
$q = 1 - p$	Probability of *logic-0* from a PUF cell
n	Length of the UP/DOWN counter
T	Total number of trials
$k = \frac{(2^n-2)}{2}$	Initialization state—it is also the number of steps from initialization to end states
P_s	Probability of *logic-1* from UP/DOWN counter (probability of success)
P_e	Probability of *logic-0* from UP/DOWN counter (probability of error)

As mentioned earlier, the n-bit UP/DOWN counter is initialized at $k = (2^n - 2)/2$ and multiple trials are conducted. The state transition diagram to better illustrate this process is shown in Figure 6. If the PUF cell creates *logic-1*, it is symbolized by $U_i = +1$ else, $U_i = -1$ in Label (2). However, due to the absorbing saturation (decision) barrier at the end as shown in Figure 6, Label (2) cannot be used directly to model UP/DOWN counter. Nevertheless the UP/DOWN counter is related to the well-studied *Gambler's ruin* problem [28] and, hence, the metrics of interest can be determined. For the proposed voter scheme, we are concerned with three probabilities: (i) probability of reaching the correct state, P_s (probability of success); (ii) probability for reaching the wrong state, P_e (probability of error); and (iii) probability of unresolved output $1 - (P_e + P_s)$. If we consider our working example, as the cell is biased towards *logic-1*, the probability of reaching the end state $(2^n - 2)$ is the probability of success P_s. Similarly, the probability of the trials leading to the end state 0 is the probability of error P_e. The last probability case may arise when the counter has not reached a saturating state after a given number of trials. As the circuit is designed to resolve to a decision, we are concerned with P_e and P_s with a limited number of trials.

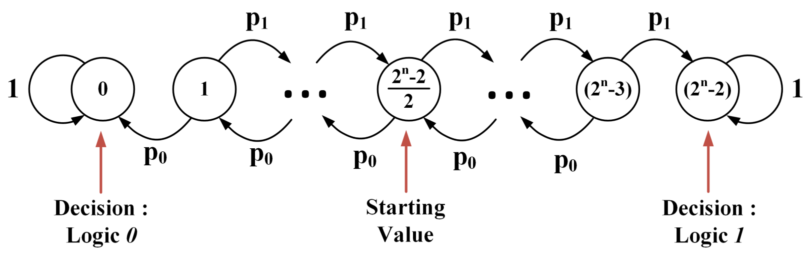


Figure 6. Markov Chain model for the voter scheme.

4.2. Error Rate

A random walk with absorbing barrier is akin to the random walk in Gambler's ruin problem as mentioned earlier. Instead of deriving probability of error in T trials, the probability of errors occurring in an infinite number of trials can be derived [28] and is given by

$$P_e = 1 - \frac{1-\left(\frac{q}{p}\right)^k}{1-\left(\frac{q}{p}\right)^{2k}}. \tag{3}$$

This value serves as an upper bound on the worst case error rate, resulting with the use of the UP/DOWN counter. The above expression is directly related to the ruin probability in Gambler's ruin problem [28] and the derivation is excluded for brevity. As Label (3) signifies the probability of error under infinite trials, the probability of error *is lower* under fixed number of trials. Nevertheless, the expression gives meaningful bounds for the design, signifying that the resultant error rate is always less than the expression derived in Label (3). This is shown in Figure 5, where the error rate using our scheme using simulation and analytical results are plotted. Thus, the equation gives insights into the benefits of using the proposed voting scheme.

Similar to probability of error, the probability of reaching correct state (probability of success) under infinite trials can be derived as

$$P_s = 1 - P_e. \tag{4}$$

From Labels (3) and (4), we can infer that, under infinite trials, the UP/DOWN counter saturates at either one of the saturating ends when $p \neq 0.5$. In reality, due to the limited number of trials, the counter value can be struck in a value between the saturating values. As this primarily occurs in cells with high error rate, they can be neglected for key generation if the design has redundant cells. A testing/DFT scheme using this method is discussed in Section 4.4.

4.3. UP/DOWN Counter vs. TMV

In Figure 7, the comparisons of error rates after using the TMV and UP/DOWN counter scheme are plotted against the initial error rate of the cell. Analytical results were used for both schemes. The expression for error rate of TMV can be found in related publications [10]. For comparison, the error rate reduction by using 4−, 5− and 6-bit TMV counters and UP/DOWN counters are plotted. For the target error rate of the order of 10^{-6}, the new voter design offers a significant advantage over a traditional TMV. The 4-bit UP/DOWN counter is capable of handling 2× the initial error rate compared to a 4-bit, 15-way TMV to get a final error rate $\leq 10^{-6}$. Similarly, 5-bit and 6-bit UP/DOWN counters offer 2× and 1.8× improvement. We also note that, for a 5-bit UP/DOWN counter, the counter still outperforms a 6-bit TMV counter. Hence, we expect an n-bit UP/DOWN counter to do better than an $n+1$-bit TMV counter for $n \geq 5$. Another point to mention is that the improvement in reliability when using an UP/DOWN counter is a conservative estimate, as Label (3) is an upper bound, and, in reality, better gains are expected. The UP/DOWN counter has an area penalty of ∼10% to ∼15% compared to a similar simple counter used by TMV in 45 nm technology Nangate open cell library [29]. This area increase is acceptable for the significant improvement offered in error rate reduction.

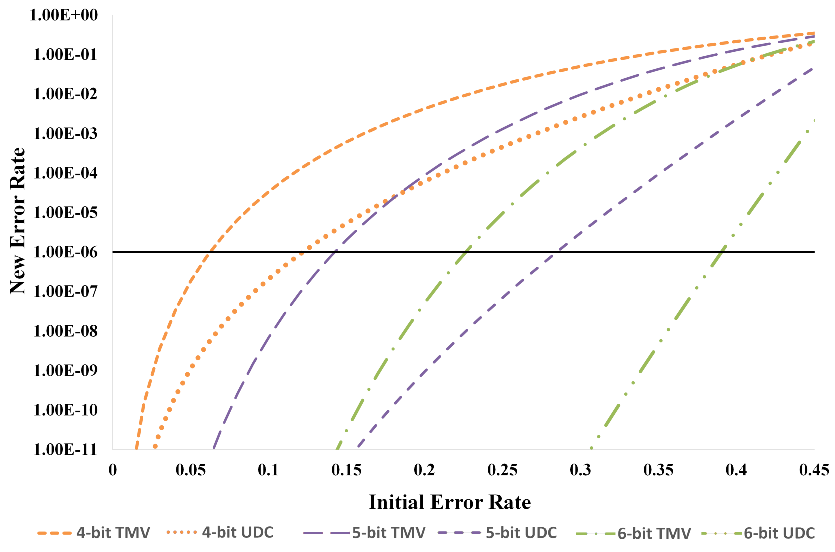


Figure 7. Error rate reduction comparison using TMV and UDC.

4.4. DFT Based on Trials to Settlement

Label (4) quantifies the probability of success when infinite trials are applied to each cell. In reality, the UP/DOWN counter may settle in a value between the end states due to the limited number of trials. This is related to the inherent error rate of each cell. The expected number of trials needed by the UP/DOWN counter to reach a decision state [28] can be derived as

$$d_k = \frac{k}{(p-q)}\left[2\left(\frac{1-\left(\frac{q}{p}\right)^k}{1-\left(\frac{q}{p}\right)^{2k}}\right)-1\right], \quad (5)$$

where $k = (2^n - 2)/2$. This expected value for a 4-bit UP/DOWN counter is plotted in Figure 8 for different initial error rates. As illustrated, the average number of trials for reaching the saturation increases with the error rate. The exact distributions for probability of reaching the correct value under given number of trials can be derived using probability generating functions. Such explicit expressions for number of trials and probability of not reaching any state can be used to improve the design. The explicit expression permits the designers to estimate how much provision is to be made for the masked bits in the design.

As the average number of trials to reach the end state is related to inherent error rate of the cell, this information can be used during the trials to filter out cells with high error rates. For example, consider the test setup shown in Figure 9. The PUF array has redundant cells so that, during the trials, the cells with high error rates can be discarded for operation. Hence, the aim of the test is to generate *mask information* that indicates which of the cells in the PUF array should be considered for real-time operation. One simple way to achieve this is to set an empirical or analytical threshold based on Label (5) (or using tighter bound expressions) for the number of trials to apply to the PUF array. If a PUF cell does not reach the end state within the target number of trials, the cell is marked as invalid. As this mask does not reveal any information about the PUFs' values, it can be made public. During real-time operation, this mask value along with raw PUF response is combined to filter out responses of cells with high error rates. The resultant identifier is used as a key. This technique may also be combined with ECC or other post-processing techniques to reduce the probability of error further. This technique can also be used to determine whether a PUF chip is reliable enough for operation. For example, if the mask implies that the number of cells that are reliable in an array is less than the length of the desired key length, the chip can be marked as unreliable and hence, rejected during testing. Bhargava et al. have proposed similar mask generation, but by adjusting supply voltage [30].

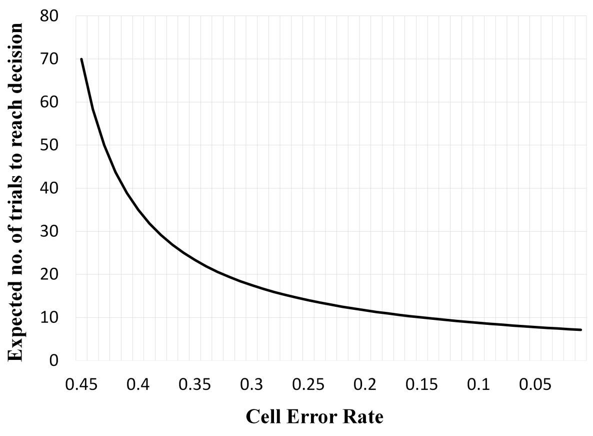


Figure 8. Expected number of trials to reach saturation (decision) in a 4-bit UDC.

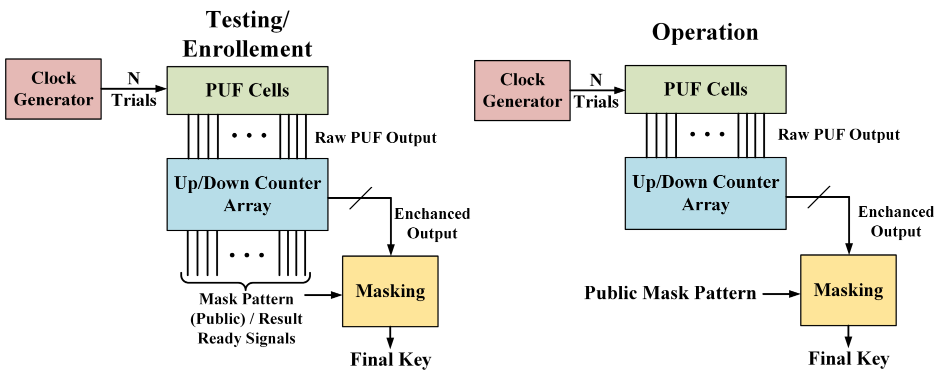


Figure 9. Testing/DFT method for identifying high error rate cells.

Hence, even though the UP/DOWN scheme has subtle changes in comparison to the TMV scheme, they offer significant improvement in error rate and also provide leverage to identify high error rate cells.

5. Results and Case Studies

In this section, we first perform a case study to illustrate how yield and error rates can be improved using redundancy with the proposed voter based design. Later, we make comparisons in terms of area and performance with other related works.

5.1. Case Study: Redundancy to Improve Yield and Reduce Error Rate

In this subsection, we demonstrate how combining the redundancy of the PUF cells along with the proposed testing technique can be used to improve reliability and yield of a chip using SRAM-based PUF. We also estimate how many extra redundant cells are required for guaranteeing high yield.

Let us assume that, under a particular manufacturing process and SRAM design, $X\%$ of cells are unreliable beyond correction by UP/DOWN voter scheme. We name such cells *bad cells*. Hence, we can assume that each cell has a probability of x ($0 \leq x = \frac{X}{100} \leq 1$) of turning up as a bad cell after manufacturing. We also assume that the reliability for different cells is independent. We are interested in calculating, for a particular value of x, how many redundant cells are required to create a stable

128 bit-key with high probability. We aim at reducing the probability of not finding 128 reliable bits cells to $\leq 10^{-6}$. This can be calculated using a binomial counting process as given below:

$$P_{yield\_loss} = \sum_{m=128}^{N} \binom{N}{128} x^m (1-x)^{n-m}, \tag{6}$$

where P_{yield_loss} is defined as the probability of not finding 128 stable cells in a design with N cells. For example, if a particular manufacturing process has $x = 0.05$, with just 151 cells, we can guarantee that 128 stable bits can be found with high probability $(1 - 10^{-6})$. This ensures that expected yield loss is around 1 in a million chips for the above example.

5.2. Design Area/Performance Comparisons

In this subsection, we present a design example along with estimates on other metrics. The 45 nm technology Nangate open cell library [29] is used for all area calculations. We use 4-bit UP/DOWN for the estimation, which can correct initial error rate of 15% to less than an order of 10^{-6}.

First, we obtain the statistics on the number of trials required for decision using the UP/DOWN counter in the presence of noise (which is related to the analysis presented in Label (5)). Ten thousand SRAM cells were instantiated with PVT variations modeled as Gaussian threshold voltage, V_{th}, variations with 3σ value of 150 mV. The voltage supplies connected to the source of M_5 and M_6, of Figure 4a were each used as noise sources. This is done to simulate the differential thermal noise that introduces errors across various evaluations of an SRAM cell [31]. One hundred such outputs were generated with varying supply noise, and the outputs were fed to a 4-bit UP/DOWN counter. The histogram for the number of outputs that each SRAM cell needed to reach saturation in the presence of supply noise is shown in Figure 10. The results indicate that the majority of the cells (99.7% or $\pm 3\sigma$) take ≤ 50 counts, for a 4-bit UP/DOWN counter, to reach a decision. Hence, if a 4-bit UP/DOWN counter is used, 50 counts are adequate to cover the majority of the cells. This empirical number for trials is used in key generation time calculations below.

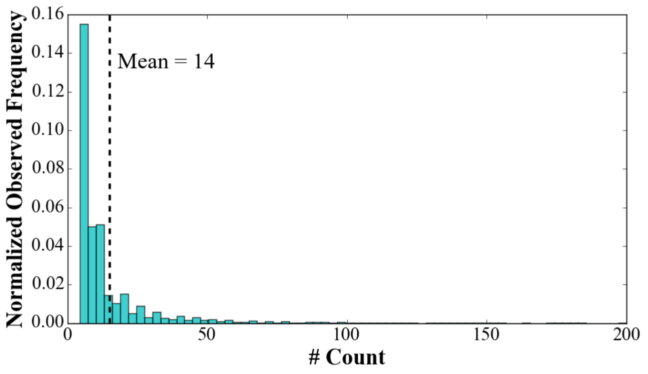


Figure 10. Histogram of the number of times that a SRAM cell was read before saturation.

To make a comparison with a fuzzy logic based implementation, we chose the fuzzy extractors proposed by Bosch et al. [32] for generating a stable key from SRAM PUFs, given 15% error probability. The authors make use of concatenated codes. Based on the results presented in the work, we estimated the areas for Golay, G_{23}, and Reed Muller, RM, decoder implementations, where the code parameters were set to yield a final error rate of $\leq 10^{-6}$. The initial number of SRAM cells needed were 3105 and 5040 cells for Golay and Reed Muller implementations, respectively, to generate 171 stable bits. For this comparison, we do not consider the hashing function area from the work. Considering that 5% of the total cells are beyond correction by our scheme, our implementation requires a minimum of 180 PUF cells to get 171 stable bits. We consider 192 cells ($\in 16\mathbb{Z}$) for ease of implementing our multiplexing solutions.

The total area of the PUF system comprises an area of the PUF cells, the area for UP/DOWN counter, selection multiplexers and the clock generator/controller logic. For calculating the PUF cell area, we approximated a 50% overhead for our design over the regular 6T SRAM cell area used for previous ECC solutions [32]. In addition, the different multiplexing options are built using the 2:1 Muxes from the Nangate library [29], and their areas are obtained accordingly. Table 2 tabulates all the results and shows that our implementation offers 1.7×–2.9× range of improvement in area. The penalty for fuzzy logic implementations comes from the large number of SRAM PUFs needed to generate reliable keys. However, for high reliability systems, our voting scheme can be combined with fuzzy extraction and ECC based schemes to create efficient hybrid techniques.

Table 2. Area estimates using Nangate Cell Library [29].

Implementation		Area (μm^2)
repetition[9,1,9]; Golay[23,12,7] [32]		7648
repetition[9,1,9]; Reed Muller[16,5,8] [32]		7945
UP/DOWN counter scheme	4 : 1 Mux	4538
	8 : 1 Mux	3266
	16 : 1 Mux	2630

Bhargava et al. [30] have shown significant timing improvements compared to fuzzy logic-based works with data obtained from test chips. Hence, we compare our work against theirs for key generation time. The authors load mask data, termed *reliability map*, and generate the key by obtaining 171 bits. Similar to their work, we consider loading mask information in 4-bits/cycle sequence. Hence, in our case, for 192 bits, we need 48 cycles. We take 50 cycles as the maximum number of times that each PUF cell is read. The results are tabulated in Table 3. When clocked at 2 GHz, this corresponds to $\leq$0.2 µs key generation time. Higher order multiplexing options benefit from lower area, but there is a trade-off in terms of time taken to generate the final key (e.g., 848 cycles for 16:1 Mux option). These results indicate the efficiency of the proposed solution.

Table 3. Generation time results.

Implementation		# Bits generated	# Cycles
Bhargava et al. [30]		171	286
Our Approach	4:1	192	248
	8:1		448
	16:1		848

6. Conclusions

SRAM based PUFs are popular for key/ID generation. Such keys may be unreliable due to noise. Since the reliability of keys is of paramount importance, various ECC and fuzzy extraction based techniques have been proposed previously. Unfortunately, they are also expensive. We present an alternative design based on a new voter scheme which reduces the error rate significantly. Our integrative solution, encompassing analysis, design and test techniques, ensures low error rate, high yield and simplicity of design. The proposed technique can be viewed as both alternative to ECC schemes in low cost systems and complementary to ECC in high reliability design requirements.

The future extension of this work includes analysis of the impact of aging on reliability improvement of the proposed scheme. We also plan to fabricate this design to conduct silicon experiments in order to validate our results experimentally.

Acknowledgments: This work is supported in part by grants from the National Science Foundation (NSF) (Grant No. 1421352) and Intel Corporation, Hillsboro, OR, USA.

Author Contributions: The primary idea for this work was developed by Arunkumar Vijayakumar and Sandip Kundu. Arunkumar Vijayakumar was responsible for the mathematical analysis showcased in this work. The simulation setup, generation of results and comparison with previous works were performed, jointly, by Arunkumar Vijayakumar and Vinay C. Patil. The manuscript was written by Arunkumar Vijayakumar and Vinay C. Patil, with technical and formatting aspects handled by Vinay C. Patil. Sandip Kundu supervised the entire research and provided valuable guidance to help with making crucial observations. He also directed the writing of this paper and its technical proofing.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Gassend, B.; Clarke, D.; van Dijk, M.; Devadas, S. Silicon physical random functions. In Proceedings of the 9th ACM Conference on Computer and Communications Security, Washington, DC, USA, 18–22 November 2002; pp. 148–160.
2. Rührmair, U.; Sölter, J.; Sehnke, F. On the foundations of physical unclonable functions. *IACR Cryptol. ePrint Arch.* **2009**, *2009*, 277.
3. Maes, R.; Verbauwhede, I. Physically unclonable functions: A study on the state of the art and future research directions. In *Towards Hardware-Intrinsic Security*; Sadeghi, A.R., Naccache, D., Eds.; Information Security and Cryptography; Springer: Berlin/Heidelberg, Germany, 2010; pp. 3–37.
4. Guajardo, J.; Kumar, S.S.; Schrijen, G.J.; Tuyls, P. FPGA intrinsic PUFs and their use for IP protection. In Proceedings of the 9th International Workshop on Cryptographic Hardware and Embedded Systems (CHES'07), Vienna, Austria, 10–13 September 2007; Springer: Berlin/Heidelberg, Germany, 2007; pp. 63–80.
5. Holcomb, D.; Burleson, W.; Fu, K. Power-up SRAM state as an identifying fingerprint and source of true random numbers. *IEEE Trans. Comput.* **2009**, *58*, 1198–1210.
6. Lee, J.; Lim, D.; Gassend, B.; Suh, G.; van Dijk, M.; Devadas, S. A technique to build a secret key in integrated circuits for identification and authentication applications. In Proceedings of the 2004 Symposium on IVLSI Circuits. Digest of Technical Papers, Honolulu, HI, USA, 17–19 June 2004; pp. 176–179.
7. Lim, D. Extracting Secret Keys from Integrated Circuits. Master's Thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2004.
8. Maiti, A.; Casarona, J.; McHale, L.; Schaumont, P. A large scale characterization of RO-PUF. In Proceedings of the 2010 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), Anaheim, CA, USA, 13–14 June 2010; pp. 94–99.
9. Virtual Laboratories in Probability and Statistics. Available online: http://www.math.uah.edu/stat/ (accessed on 1 January 2015).
10. Mathew, S.; Satpathy, S.; Anders, M.; Kaul, H.; Hsu, S.; Agarwal, A.; Chen, G.; Parker, R.; Krishnamurthy, R.; De, V. 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 2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC), San Francisco, CA, USA, 9–13 February 2014; pp. 278–279.
11. Dodis, Y.; Ostrovsky, R.; Reyzin, L.; Smith, A. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.* **2008**, *38*, 97–139.
12. Maes, R.; Tuyls, P.; Verbauwhede, I. A soft decision helper data algorithm for SRAM PUFs. In Proceedings of the IEEE International Symposium on Information Theory (ISIT 2009), Seoul, Korea, 28 June–3 July 2009; pp. 2101–2105.
13. Maes, R.; Tuyls, P.; Verbauwhede, I. Low-overhead implementation of a soft decision helper data algorithm for SRAM PUFs. In *Cryptographic Hardware and Embedded Systems—CHES 2009*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 332–347.
14. Delvaux, J.; Gu, D.; Schellekens, D.; Verbauwhede, I. Helper data algorithms for puf-based key generation: Overview and analysis. *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* **2015**, *34*, 889.

15. Maes, R.; van Herrewege, A.; Verbauwhede, I. Pufky: A fully functional puf-based cryptographic key generator. In *Cryptographic Hardware and Embedded Systems—CHES 2012*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 302–319.
16. Garg, A.; Kim, T. Design of SRAM PUF with improved uniformity and reliability utilizing device aging effect. In Proceedings of the 2014 IEEE International Symposium on Circuits and Systems (ISCAS), Melbourne, Australia, 1–5 June 2014; pp. 1941–1944.
17. Bhargava, M.; Mai, K. A high reliability PUF using hot carrier injection based response reinforcement. In *Cryptographic Hardware and Embedded Systems—CHES 2013*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 90–106.
18. Maes, R.; van der Leest, V. Countering the effects of silicon aging on SRAM PUFs. In Proceedings of the 2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), Arlington, VA, USA, 6–7 May 2014; pp. 148–153.
19. Jang, J.W.; Ghosh, S. Design and analysis of novel SRAM PUFs with embedded latch for robustness. In Proceedings of the Sixteenth International Symposium on Quality Electronic Design, Santa Clara, CA, USA, 2–4 March 2015; pp. 298–302.
20. Bucci, M.; Luzzi, R. Identification Circuit and Method for Generating an Identification Bit Using Physical Unclonable Functions. U.S. Patent 8,583,710, 12 November 2013.
21. Hofer, M.; Boehm, C. An alternative to error correction for SRAM-like PUFs. In Proceedings of the 12th International Conference on Cryptographic Hardware and Embedded Systems (CHES'10), Santa Barbara, CA, USA, 17–20 August 2010; Springer: Berlin/Heidelberg, Germany, 2010; pp. 335–350.
22. Cortez, M.; Hamdioui, S.; van der Leest, V.; Maes, R.; Schrijen, G.J. Adapting voltage ramp-up time for temperature noise reduction on memory-based PUFs. In Proceedings of the 2013 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), Austin, TX, USA, 2–3 June 2013; pp. 35–40.
23. Xiao, K.; Rahman, M.; Forte, D.; Huang, Y.; Su, M.; Tehranipoor, M. Bit selection algorithm suitable for high-volume production of SRAM-PUF. In Proceedings of the 2014 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), Arlington, VA, USA, 6–7 May 2014; pp. 101–106.
24. Lam, S.W.L. Theory and application of majority vote: From Condorcet Jury Theorem to pattern recognition. In Proceedings of the 2nd International Conference on Mathematics Education into the 21st Century: Mathematics for Living, Amman, Jordan, 18–23 November 2000.
25. Bhargava, M.; Cakir, C.; MAI, K. Attack resistant sense amplifier based PUFs (SA-PUF) with deterministic and controllable reliability of PUF responses. In Proceedings of the 2010 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST), Anaheim, CA, USA, 13–14 June 2010; pp. 106–111.
26. Suresh, V.; Burleson, W. Robust metastability-based TRNG design in nanometer CMOS with sub-vdd pre-charge and hybrid self-calibration. In Proceedings of the 2012 13th International Symposium on Quality Electronic Design (ISQED), Santa Clara, CA, USA, 19–21 March 2012; pp. 298–305.
27. NCSU FreePDK 45 nm. Available online: http://www.eda.ncsu.edu/wiki/FreePDK45:Contents (accessed on 10 January 2009).
28. Ibe, O.C. *Elements of Random Walk and Diffusion Processes*; John Wiley & Sons: Hoboken, NJ, USA, 2013.
29. Nangate Open Cell Library. Available online: http://www.si2.org/openeda.si2.org/projects/nangatelib (accessed on 2 January 2010).
30. Bhargava, M.; Mai, K. An efficient reliable PUF-based cryptographic key generator in 65 nm CMOS. In Proceedings of the Conference on Design, Automation & Test in Europe Conference and Exhibition, Dresden, Germany, 24–28 March 2014; p. 70.
31. Holcomb, D.E.; Fu, K. Bitline PUF: Building native challenge-response PUF capability into any SRAM. In *Cryptographic Hardware and Embedded Systems*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 510–526.
32. Bosch, C.; Guajardo, J.; Sadeghi, A.R.; Shokrollahi, J.; Tuyls, P. Efficient helper data key extractor on FPGAs. In Proceedings of the 10th International Workshop on Cryptographic Hardware and Embedded Systems (CHES'08), Washington, DC, USA, 10–13 August 2008; Springer: Berlin/Heidelberg, Germany, 2008; pp. 181–197.