PUF-Based Authentication
Invited Paper

Wenjie Che - Univ. of New Mexico (wjche@unm.edu)
Fareena Saqib - Florida Institute of Technology (fsaqib@fit.edu)
Jim Plusquellic - Univ. of New Mexico (jimp@ece.unm.edu)

Abstract -- *In the context of hardware systems, authentication refers to the process of confirming the identity and authenticity of chip, board and system components such as RFID tags, smart cards and remote sensors. The ability of physical unclonable functions (PUF) to provide bitstrings unique to each component can be leveraged as an authentication mechanism to detect tamper, impersonation and substitution of such components. However, authentication requires a strong PUF, i.e., one capable of producing a large, unique set of bits per device, and, unlike secret key generation for encryption, has additional challenges that relate to machine learning attacks, protocol attacks and constraints on device resources. In this paper, we describe the requirements for PUF-based authentication, and present a PUF primitive and protocol designed for authentication in resource constrained devices. Our experimental results are derived from a 28 nm Xilinx FPGA.*[1]

1. Introduction

Authentication is traditionally characterized as a process that verifies "something you know", e.g., a password, "something you have", e.g., hardware one-time-password tokens, and "something you are", e.g., your fingerprints. Multi-factor authentication requires two or more of these components from different categories. PUF-based authentication provides individual devices with a set of passwords (bitstring responses to challenges) that uniquely identify it (a fingerprint), so in this sense, it can be characterized as a multi-factor authentication mechanism. PUFs derive their fingerprint from random variations that occur in the manufacturing process of a chip or board. For example, a delay-based PUF measures and digitizes variations that occur in paths and/or gates within the chip or along wires in a printed circuit board (PCB) [1]. Although we use variations in path delays as the entropy source in this paper, there are many other sources of variations that can be leveraged, as is evident from the published literature on PUFs.

PUFs have been proposed for other types of applications including encryption, for detecting malicious alterations of design components and for activating vendor specific features on chips. Each of these applications has a unique set of requirements regarding the security properties of the PUF. For example, PUFs that produce secret keys for encryption are not subject to model building attacks (as is true for PUF-based authentication) which attempt to 'machine learn' individual path delays for a chip as a means of predicting the complete response space of the PUF. This is true for encryption because the responses to challenges are typically not 'readable' from an interface on the chip. In general, the more access a given application provides to the PUF externally, the more resilience it needs to have to adversarial attack mechanisms.

Authentication as an application for PUFs clearly falls in the category of extended access. The term 'hardware token' or *prover* is typically used to identify a fielded device that embeds the PUF, such as a smart card, and the term 'secure server' is used in reference to the verifier.

Applications such as authentication require a **strong PUF**, i.e., a PUF that can produce a very large number of challenge-response-pairs or CRPs. Challenges and responses are the digital inputs and corresponding outputs of the PUF. In order for authentication to work, it must be necessary and impractical for an adversary to apply all possible challenges to the PUF on a chip as a means of obtaining all of its responses. Making this infeasible makes it impossible for an adversary to build a 'clone' of the chip that replicates the CRP behavior. However, the requirement of a very large CRP space is, in general, challenging to meet for PUFs. It requires a large source of entropy, which can become expensive area-wise when the PUF is implemented using dedicated and specialized components.

Authentication is typically characterized as having two phases: enrollment and regeneration. Enrollment occurs immediately after manufacture and involves the verifier generating a random set of challenges which are applied to the token to generate a corresponding set of responses. The set of CRPs are stored on the verifier for each chip individually in a secure environment. The stored CRPs can then be used to carry out authentication in the field with the token. The verifier only needs to store a limited set of CRPs in the secure database because the very large CRP space of the strong PUF combined with the secrecy of the chosen CRPs makes it difficult or impossible for an adversary to know how to respond using a clone of the token.

Bear in mind, authentication can also be implemented by having the PUF generate a secret key for encrypting communication between the prover and verifier. The enrollment process involves the PUF generating a shared key that is stored on the server through a one-time interface, i.e., an interface that can be disabled, along with helper data. The helper data is later transmitted to the token as needed for authentication in the field to enable precise regeneration of the key. The token in this scenario needs to incorporate an encryption algorithm, which adds to the required resources. Although this method requires only a weak PUF that is capable of producing only a small number of bits (a plus), the encryption operation carried out by the token is subject to side-channel attacks that attempt to learn the key (a minus). Once learned, the security mechanism is defeated. Therefore, strong PUFs that have a very large CRP space provide an advantage by making it infeasible for an adversary to extract all the secrets embedded in each token.

Most authentication proposals also limit the amount (or eliminate completely) the need for helper data and instead allow for fuzzy matching to occur between server stored responses and those generated in the field by the token. In other words, a *small* number of differences are tolerated in the response bitstrings. Although fuzzy matching reduces the storage requirements for the verifier by eliminating the helper data, it also increases the possibility of aliasing and impersonation, i.e., the likelihood that two devices produce the same responses (within the noise margin).

[1] This work supported by NSF grant 1118025.

In this paper, we propose a hardware-embedded delay PUF called HELP as a strong PUF for authentication. HELP leverages entropy present in functional units already present in the chip, and therefore, it does not require the insertion of dedicated components. Moreover, the overhead associated with integrating HELP into functional unit is very small relative to the size of the functional unit. HELP is unique in that it leverages delay variations in structures that are not identical and implicitly provides tamper protection of the functional unit(s). This paper contributes beyond previously published work in [2][3] in the following ways:

- We implement HELP on a Xilinx 28 nm 7020 Zynq chip embedded on AVNET's Zedboard [4] using both glitchy and glitch-free functional units as the source of entropy and analyze the statistical quality of the bitstrings.
- We isolate and analyze entropy introduced from multiple sources and discuss the trade-offs and impact on security.
- We propose an authentication protocol using HELP.

2. Related Work

An excellent survey and critical review has been recently published that covers the state-of-the-art with regard to PUF authentication for resource constrained devices [5]. The criteria used to review the existing methods assume a low-cost resource constrained token and resource-rich server, and the use of a strong PUF. The authors indicate that protocols which require NVM are less attractive because of the increased cost of manufacturing of NVM components in CMOS technologies and because of recently disclosed vulnerabilities of NVMs to probing attacks. The PUF protocols proposed in [6-22] are evaluated against the following characteristics [5]:

- Resilience to measurement and temperature/voltage (TV) noise sources.
- Resilience to machine learning via use of cryptographic hash functions and XOR functions as needed.
- Are techniques needed to expand the response space (PRNG) of the strong PUF?
- Ease of instantiation of the PUF authentication mechanism.
- Resistance to protocol attacks, i.e., token and/or server impersonation and denial of service attacks.

The authors conclude that the main problems with the protocols are rooted in the PUF itself and that research should focus on developing a truly strong PUF with solid cryptographic properties.

3. Overview

3.1 Goals and Objectives

One of the goals of this work is to isolate and characterize the main sources of delay variations (the entropy source) on the chip, namely, 1) within-die delay variations that occur within individual FPGA LUT primitives, 2) global delay variations that occur across all LUTs on the chip and 3) delay variations introduced by static and dynamic logic hazards. All of these sources of variations change the delay characteristics of paths uniquely on each chip.

A key objective is to determine the magnitude of these variations with respect to measurement and temperature/voltage (TV) noise sources. We refer to this noise as "TV noise" since TV dominates even when repeated sampling and TV compensation techniques are applied. TV noise works to impede access to the entropy provided by delay variations, and reduces the amount of usable entropy. Delay variations introduced by within-die process variations are relatively small even when measured through a single LUT. On the other hand, global variations and variations introduced by hazards are well above the TV noise margin, making them attractive as a source of entropy. However, there is a downside to leveraging these larger sources of entropy as discussed below.

We integrate HELP into a GF(4) subcomponent and a full-blown GF(256) version of the Advanced Encryption Standard (AES) SBOX functional unit [23]. The GF(4) version can be implemented using a logic depth of 1, which allows individual LUT delays to be analyzed. We implement the GF(256) in two ways referred to as: *Standard*: without any type of special logic style or constraints and *WDDL*: without glitches using wave-differential dynamic logic [24]. The Standard implementation includes all three sources of entropy. Inter-chip hamming distance (HD), Inter-chip HD and the results of NIST statistical tests are reported to understand the trade-off of the two logic styles on bitstring generation and reproduction [25][26].

A *modulus* technique is used in combination with a helper data string as a mechanism to maximize the strength of the cryptographic properties of the PUF in the proposed authentication protocol. Glitch-free logic implementations of the functional unit, such as WDDL, provide a distinct advantage in resource-constrained authentication applications by reducing bit flips while improving access to the limited, but most important source of entropy, namely that provided by within-die variations.

3.2 Attack Scenarios and Assumptions

Traditional "resource-constrained" applications such as RFID and smart cards utilize memory, small microcontrollers and/or ASICs for implementing functions. The attack models and assumptions that we describe in the context of FPGAs can be extended to these implementations as noted below. Although HELP is proposed as an FPGA authentication mechanism in this paper, the concept and techniques presented are also applicable to ASIC implementations [2].

Secure computing using FPGAs requires encryption of the programming bitstream. Modern FPGAs integrate encryption/ decryption modules, and NVM-based key storage mechanisms, to support this requirement. Beyond protecting Intellectual Property, encryption also prevents tampering with the design. Although our technique can detect tamper within functional unit(s), we assume an attacker is not able to defeat the bitstream encryption mechanism. No security mechanism, PUF or otherwise, is secure if this requirement is not met.

We consider two attack scenarios. First, the adversary can gain (temporary) possession of the token and attempt to read out all responses or enough of them to "machine learn" the entropy source. Once known, a clone can be 'programmed'. In general, strong PUFs can significantly impede, or make impossible, the success of this type of attack. For PUF architectures in which machine learning is effective, the proposed protocols typically incorporate obfuscation mechanisms to prevent direct control of the PUF and observation of its responses. The second attack mechanism is similar except that the adversary carries out a 'man-in-the-middle' attack, i.e., he or she listens to exchanges between the token and the server.

Other types of attack scenarios can be avoided. For example, some protocols require one-time interfaces to be present during enrollment but such interfaces can be 'undone' using focused ion beam techniques. Still other protocols require the use of small NVMs, which add cost and weaken security because 'read-out' mechanisms are becoming increasingly effective. Therefore, avoiding one-time interfaces and NVM is a plus.

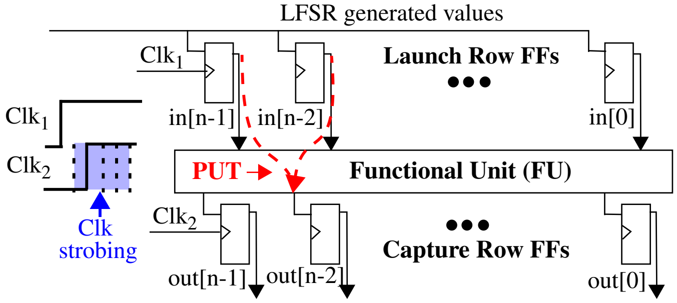

Fig. 1. Configuration of the functional unit (FU).

4. Experiment Setup

4.1 HELP Overview

HELP measures path delays using a simplified version of a embedded test structure called REBEL [2]. The simplified version eliminates the delay chain component and instead samples the path delays at the capture FF directly. Fig. 1 shows the test setup with the 'functional unit' or FU representing the entropy source. The inputs and outputs of the FU are connected to a set of Launch Row and Capture Row flip-flops (FFs), resp.

The delay of a path is determined using the fine phase adjust feature of a Xilinx embedded MMCM (mixed mode clock manager). A series of launch-capture clocking events are applied to the functional unit using two clocks, Clk_1 and Clk_2, as shown on the left side of Fig. 1. The phase shift between Clk_1 and Clk_2 is adjusted dynamically across the sequence of launch-capture tests. The digitally selected value of the fine phase shift between the two clocks is referred to as the launch-capture interval (**LCI**). The smallest LCI interval that allows the propagating edge along a path to be captured in the capture FF is used as the digitized timing value for the path. The MMCM on the Zynq FPGA clocked at 25 MHz provides a resolution of 18 ps. Digital values between 150 (smallest LCI with value of approx. 18 ps * 150 = 2.7 ns) and 2,200 (largest LCI with value approx. 39.6 ns) are used as the path delay value. The repeated testing of the FU at different LCIs is referred to as clock strobing. The LCI used to represent the delay of a path is referred to a PUFNum or **PN**. The signed difference of two randomly selected PNs is referred to as a **PNDiff**.

4.2 TV Compensation

The majority of the delay variations introduced by changes in temperature and voltage is removed by applying a TV compensation process. TV compensation is carried out by computing the mean (offset) and range (multiplier) from a set of PNDiffs for each chip and for each TV corner separately. The offset and multiplier computed during enrollment are used with the offset and multiplier computed at each TV corner to compensate the PNDiffs generated at the TV corners using Eq. 1.

$$zval_i = \frac{(PNDiff_{TVx} - \mu_{TVx})}{rng_{TVx}} \qquad \textbf{Eq. 1.}$$

$$PNDiffs_{TVComp} = zval_i rng_{TVEnroll} + \mu_{TVEnroll}$$

Here, $zval_i$ represents a standardized PNDiff after subtracting the mean and dividing by the range computed using a set of PNDiffs produced at the TV corner, *TVx*, for a specific chip. The individual $zval_i$ are then transformed using the mean and range computed earlier for the same chip during enrollment, i.e., at *TVEnroll*. We refer to the PNDiffs generated during enrollment as the **reference**. This linear transformation is very effective at eliminating the shifting and scaling that occurs to path delays at different TV corners (note: using the PNDiffs directly without this type of compensation does not compensate for scaling).

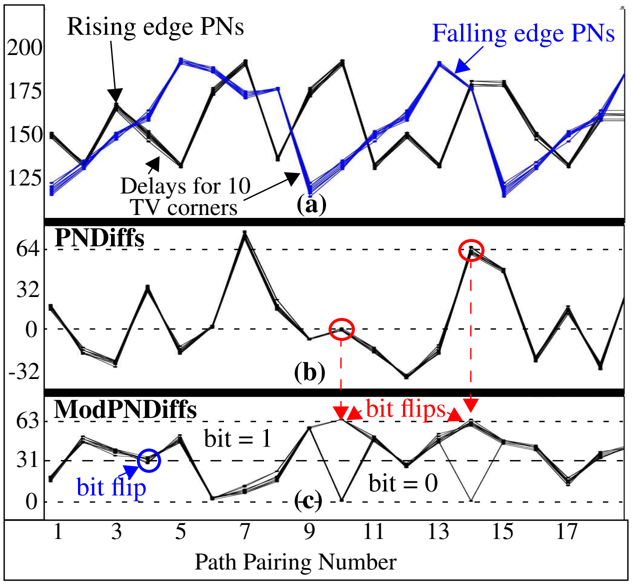

Fig. 2. Example rising and falling path PNs (top), random pairings of rising and fall PN differences (middle), PN differences modulo 64 (bottom).

4.3 Bit Generation Algorithm

The bit generation uses the signed difference in two path delays (PNDiff) as a means of both hardening the algorithm against model building and increasing the diversity in the PUF responses. A **ModPNDiff** is defined by computing a signed difference between two arbitrary selected PNs, and then applying a *modulus*. The modulus is necessary because the paths in the FU vary in length, for example, in our experiments, short paths consist of 1 LUT while the longest paths consist of 13 LUTs, which is captured in the PNDiffs. The modulus removes the 'path length' bias while fully preserving the smaller within-die delay variations.

For example, the top of Fig. 2(a) shows two sets of waveforms labeled 'Rising edge PNs' (black) and 'Falling edge PNs' (blue). The points in the waveforms represents the delay values (PNs) measured from a set of paths in chip C_1 in the AES SBOX GF(4) experiment. Each group of waveforms with similar shape and color represent the PNs measured at each of the 10 TV corners after a TV compensation method is applied (a process identical to the TV compensation applied to the PNDiffs described above). The vertical spread in the 10 points represent uncompensated TV noise. The waveforms shown in (b) represent the PNDiffs computed from randomized pairings of rising and falling edge PNs in (a). Although only chip C_1 data is shown, the shape of the difference waveforms is similar for other chips because of the path length bias. The **ModPNDiffs** shown in (c) are the result of applying a modulus of 64 to the PNDiffs in (b). The modulus effectively 'wraps' all differences into the range of 0 to 63 and reduces and/or eliminates the bias. The bit generation algorithm assigns ModPNDiffs in the range from 0 to 31 as '0' while those in the range of 32 to 63 are assigned '1'.

The red circles on points 10 and 14 show bit flips. Bit flips occur when some, but not all, of the 10 points in each group cross over one of the boundaries given by 0 or 63. An additional bit flip is shown by the blue circle for point 4, where the points cross over the boundary between ‘0’ and ‘1’. The close grouping of the 10 points makes it is possible to apply a predictive screening process that avoids most/all of these bit flips as we show below. Moreover, the modulus parameter can be used to remove bias as described but it is also useful for increasing the input-output space of the HELP PUF, which is also discussed in the following sections.

4.4 Functional Unit Synthesis Flow

The AES SBOX is used as the functional unit in our experiments because its interconnection implementation structure is random and complex. Although only the SBOX is used in this work, the technique can be extended to the full implementation of AES and other types of functional units (see [2] and [3]). As indicated earlier, we implement the SBOX using a special glitch-free logic style called WDDL [24] as a means of distinguishing between the underlying sources of entropy, and as a means of improving the reliability of HELP. WDDL eliminates functional and logic hazards by imposing stimulus constraints and restricting the implementation to use only AND and OR gates. WDDL is proposed as a mechanism to harden a design unit such as AES against side-channel attacks, and therefore, also attempts to eliminate information in the power curves. This latter feature is not required to improve the reliability of HELP and therefore, we are also looking into simpler glitch-free-only strategies that have less area overhead [27]. The benefit of WDDL is that it is simple to implement and provides a nice test bed for evaluation of glitch-free logic implementation.

Fig. 3 illustrates the design flow followed to implement the WDDL version of the AES SBOX. A behavioral VHDL description of the SBOX along with a standard cell library are used as input to the CADENCE RC synthesis tool. The standard cell library only includes 2-input to 6-input AND and OR gates to match the LUT capabilities on the FPGA, and a NOT gate. No timing constraints were used in the synthesis and therefore, RC optimized for area.

A structural netlist consisting of only AND, OR and NOT gates represents the output of the synthesis. This file along with a set of synthesis and implementation constraints are processed by a perl script to produce a WDDL version of the netlist. One example transformation is shown in the figure where a AND gate followed by an NOT gate is converted to a complementary pair of AND/OR gates, with the outputs swapped for connections downstream as a means of emulating (and eliminating) the NOT gate.

The WDDL version therefore is constructed by creating a complementary OR gate (with complementary inputs) for all existing AND gates, and vise versa. The 8 primary inputs of the SBOX are also replicated and are driven with complementary values during evaluation. The operation of WDDL consists of two phases: a pre-charge phase in which all primary inputs (including the complementary inputs) are driven with ‘0’. This forces ‘0’s on the inputs and output of all gates throughout the circuit. The evaluate phase applies the true and complementary values to the 8 true and complementary primary inputs, resp., and causes a set of rising transitions to propagate through the circuit. For the SBOX implementation, half of the true outputs and half of the complementary outputs transition on average during evaluate. Therefore, for each of the 256 possible input transitions, i.e., from 0000000->xxxxxxxx, 8 PNs are obtained to produce a total of 2048 PNs. Another 2048 are obtained for the precharge phase, i.e., from xxxxxxxx->00000000, so a total of 4096 PNs are produced, from which a set of 2048 PNDiffs can be uniquely constructed.

From Fig. 3, the WDDL version of SBOX is combined with the HELP engine (described using behavioral-level VHDL) in a project that is processed by the Xilinx Vivado synthesis and implementation tool. The constraints added by the perl script prevent the FPGA synthesis and implementation tools from optimizing the WDDL structural netlist. The programming bitstream generated by Vivado is then used to program the Xilinx 7020 Zynq chip on a Zedboard [4], which is placed in a temperature chamber.

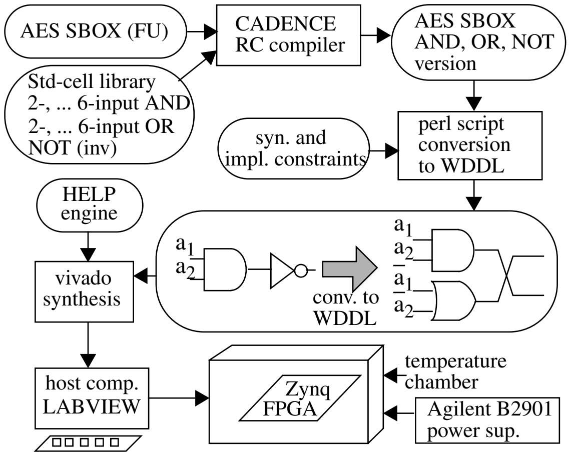

Fig. 3. Process Flow

We also synthesized AES SBOX GF(4) and GF(256) versions using a *standard* synthesis flow to serve as a comparison to the WDDL implementation. The flow for the *standard* versions simply uses VHDL descriptions of the GF(4) and GF(256) as input to the Xilinx Vivado synthesis tool without any constraints. We instantiate two copies of the GF(256) in the standard version, with the inputs to the 2nd copy complemented, to model the complementary network within the WDDL version as a means of making the two implementations as similar as possible. A similar strategy is used for the GF(4) except four copies are instantiated (each copy has only 4 inputs/outputs). The input transition sequence used for the WDDL version are also used here. Note that there are significant differences in the resource usage by the two GF(256) versions, however. For example, the standard version uses 80 LUTs in a 2-level logic structure while the WDDL version uses 756 LUTs in a multi-level logic style of up to 13 levels. The GF(4) has only 16 LUTs in 1 level of logic and therefore allows a single LUT delay to be measured.

5. Experimental Results

We ran our experiments on 30 copies of the Zedboard [4]. Commercial grade 7020 Zynq chips are incorporated on the Zedboard, which restricts the temperature range between 0^{o}C and 85^{o}C and the operating voltage between 0.95 V and 1.05 V (5% around the nominal 1.00 V). The Agilent precision power supply and ESPEC temperature chamber are controlled using a LABVIEW program running on a host computer. The Zedboards were tested at 25^{o}C, 1.00 V, which we use as enrollment data, and 9 regeneration corners, which includes all combinations of three temperatures, 0^{o}C, 25^{o}C and 85^{o}C and three voltages, +/- 5% and nominal. The MMCM on the FPGA is configured with a 25 MHz clock frequency.

5.1 AES SBOX GF(4) Analysis

The goal of the GF(4) analysis is to determine the magnitude of within-die variations in the shortest constructible path on an FPGA, i.e., paths with 1 launch FF, 1 LUT and 1 capture FF. Fig. 4

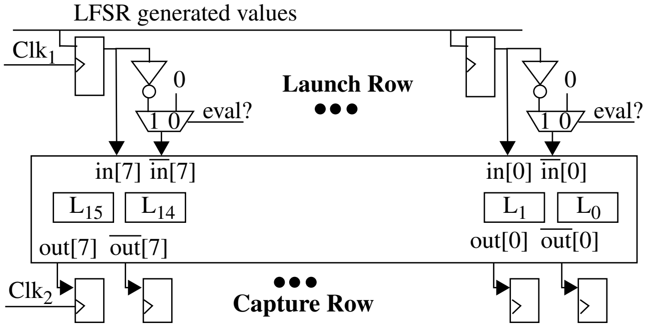

Fig. 4. Configuration of the AES SBOX FG(4) [23].

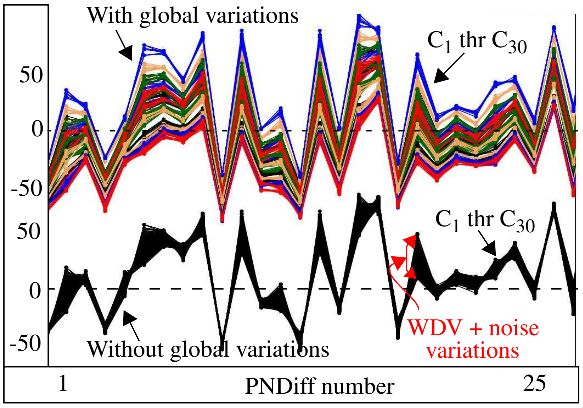

Fig. 5. TV compensated PNDiffs with (top) and without (bottom) global variations for 30 chips.

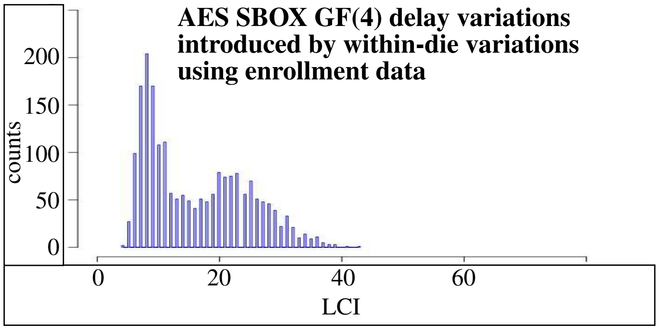

Fig. 6. Histogram of enrollment delay variations using TV compensation of PNDiffs with no global variations.

shows the configuration synthesized by Vivado. Two copies of the logic expressions for GF(4) given in [23], and two copies implementing their inverse, synthesized to a set of 16 4-input LUTs labeled L_{15} down to L_0. The inputs, e.g., in[7]/$\overline{\text{in}}$[7] fan-out to the LUTs of the true and inverse copies, resp. and the outputs, e.g., out[7]/$\overline{\text{out}}$[7], wire to a row of capture FFs. Given all inputs are applied simultaneously, there is no glitching that occurs on the outputs even though the potential exists given the diverse truth tables implemented with the LUTs.

A 25 point sample of the 2048 PNDiffs measured from the 30 chips at the 10 TV corners is shown in Fig. 5. The PNDiffs are computed by selecting a unique random pair (chosen by an LFSR) of PNs, one from the rising paths and one from the falling paths (see Fig. 2(a)). The groups of waveforms of the same color shown along the top have been TV compensated as described in Section 4.2, i.e., using the enrollment values for each chip as the 'reference'. The vertical offsets between the waveform groups are caused by global (chip-wide) variations, i.e., variations in the overall performance characteristics of the chips. Although global variations can be leveraged as a source of entropy, similar to within-die variations, there are drawbacks to depending on it.

To illustrate this problem, the black waveforms shown along the bottom of Fig. 5 are again from the 30 chips but are TV compensated using a special process in which the enrollment data from chip C_1 is used as the reference for all chips. This effectively eliminates the global variations and leaves only measurement noise, uncompensated TV noise and within-die variations (WDV) (see label in figure). In a large population of chips, it is highly likely that sets of chips will have the same level of global variations, so this graph illustrates this case, where only within-die variations can be leveraged as a source of entropy.

The magnitude of the noise sources is reflected in the width of the band of same colored waveforms shown along the top of Fig. 5. Measurement noise (with 16 sample averaging) is approx. 1 PN on average (approx. 18 ps), so the majority of the variation is introduced by uncompensated TV noise. The mean value of variation, computed as the mean of the 3σ values of the 10 TV compensated PNDiffs, that remains in the waveforms is on average approx. +/- 2.5 LCIs or 45 ps above or below the enrollment value, and the worst case value is less than +/- 8 LCIs or 145 ps. This number is important since it represents the amount of entropy that is lost, i.e., within-die variations less than this LCI value are more difficult to leverage. Within-die variations are reflected in the change in shape of the waveform groups for each chip. The magnitude of the variations introduced by within-die variations is, on average, approx. 4x larger (20 LCIs) than the average variation introduced by TV noise (5 LCIs), i.e., 360 ps vs 90 ps, resp.

A quantitative analysis of the entropy provided by within-die variations is shown in Fig. 6 using the 2048 PNDiffs from the 30 chips. The range across the 30 chips for each of the 2048 PNDiffs is computed using the TV compensated waveforms shown along the bottom of Fig. 5, i.e., those without global variations. Only the enrollment PNDiffs are considered here, so the histogram plots the distribution of the 2048 ranges without TV noise. Given that measurement noise is very low, the shape of the histogram is predominated determined by within-die variations. As indicated above, the average value is close to 20 but the ranges vary from 10 to more than 40.

Fig. 7 provides a second quantitative analysis using the hamming distances (HD) of bitstrings computed using the proposed bitstring generation algorithm and the ModPNDiffs with and without global variations. The analysis is carried out over a set of *PN modulus* (**PNMod**) values plotted along the x-axis. Inter-chip HD is computed by counting the number of bits that are different in the 2048-bit bitstrings produced by two chips during enrollment and then dividing by the number of bits. The values plotted are the average Inter-chip HDs across all possible pairings of the bitstrings (30*29/2=435 pairings). Intra-chip HD is computed in a similar fashion except the pairings are defined using the bitstrings produced at the 10 TV corners for each chip (10*9/2=45 pairings). The value plotted is again the average computed across the 30 individual chip values. Worst-case Intra-chip HD is simply the maximum value produced by one of the individual chips.

The curves for worst case and average case Intra-chip HD in Fig. 7 reflect the noise levels, while the difference between the Inter-chip and Intra-chip HD curves reflect the range of usable entropy. The results *with* global variation included are shown in

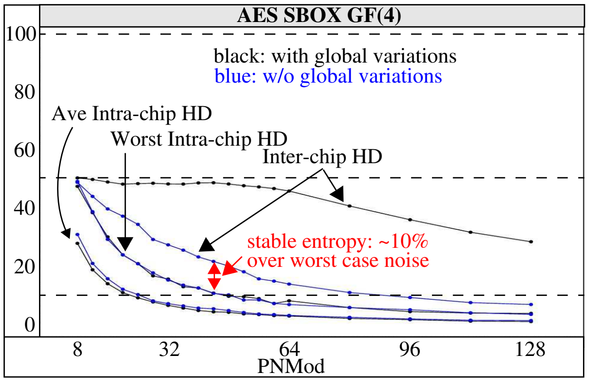

Fig. 7. Inter-chip HD and worst case and average case Intra-chip HD as a function of PN modulus.

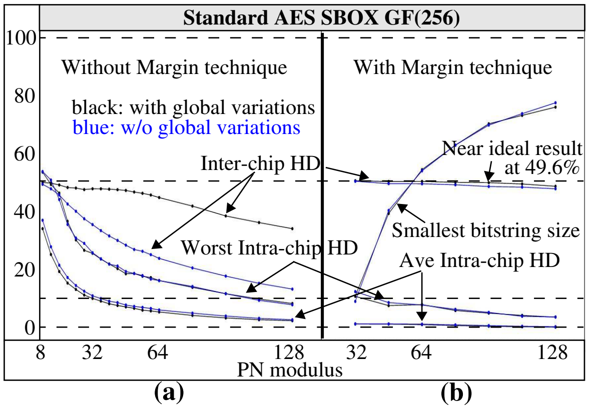

Fig. 8. Hamming distance (HD) results without (a) and with (b) the Margin technique for the Standard design.

black while the results *without* global variations are shown in blue.

The bit flips created by uncompensated TV noise remains relatively constant independent of whether global variations are present or not, as shown by the superposition of the black and blue Intra-chip HD curves. The difference between the Inter-chip HD without global variations and the worst-case Intra-chip curves varies between 0% on the left to approx. 15% at the widest point around PNMod = 28. The worst-case Intra-chip HD at PNMod of 48 is approx. 10% while the Inter-chip HD is approx. 20%. This suggests that the average Inter-chip HD of a large chip population will be smaller than its ideal value of 50% without some type of entropy amplification process. The Inter-chip HD with global variations shows that the ideal value of 50% is nearly achieved for PNMods up to approx. 64. Unfortunately, as just mentioned, this is not likely to hold true as the number of chips used in the HD calculation increases well beyond the 30 available in our experiments. Therefore, in these experiments and on this 28 nm FPGA, either entropy amplification methods or other sources of entropy need to be leveraged to produce good quality bitstrings.

5.2 AES SBOX GF(256), Standard vs. WDDL

The test setup for the Standard GF(256) and WDDL versions of the AES SBOX is similar to that shown in Fig. 4. As indicated above, the structure of the Standard version is un-constrained and therefore, is subject to static and dynamic hazards occurring internally and on some outputs, which act to increase the occurrence of bit flips.

Fig. 8(a) presents the statistical HD results in the same fashion as discussed in relation to Fig. 7. The results are very similar to the GF(4) version except for the approx. doubling of the worst- and average-case Intra-chip HD over the GF(4) version. The increase in bit flips is directly attributable to presence of glitching. Note that glitching can increase both Intra-chip and Inter-chip HD. For paths whose delays are affected by glitches consistently across all TV corners, the effect is beneficial because the path delay typically changes by 10 to 100 LCIs, and therefore represents a significant source of within-die variations. For those paths where the glitch is present at some TV corners and disappears at others, the effect is detrimental, resulting in bit flips. The worst-case Intra-chip HD and Inter-chip HD curves illustrate that both types occur because the distance between the curves (and their shape) is similar to the corresponding curves shown in Fig. 7. Although Inter-chip HD increases, this benefit is partially offset by the increase in worst-case bit-flips. Average-case Intra-chip HD, on the other hand, only increases slightly. Although we cannot present the results in detail here, it turns out that a small subset of our chips have many more occurrences of the detrimental form of glitching than the remaining chips. It was also possible to identify these glitchy chips by the difference in their rising and falling delays as shown in Fig. 2(a), using data from the WDDL version of the AES SBOX. The falling PNs (blue waveforms) are offset downwards from the rising PNs (black waveforms) in the extra glitchy chips, i.e., the falling delays are noticeable smaller than the rising delays. The extra glitchy chip Intra-chip HDs are 3 times larger than the less glitchy chips.

5.3 Margin Technique

Fig. 8(b) shows the results after applying a **Margin technique**. The method identifies PNDiffs during enrollment that have the highest probability of introducing bit flips. The PN modulus technique illustrated in Fig. 2 shows several examples of bit flips that occur at data points 4, 10 and 14. All of these data points are close to the lines that represent the boundaries between ‘0’ and ‘1’, i.e, 0, 31 and 63. The Margin technique classifies an enrollment PNDiff as ‘invalid’ if it falls within a small region (a margin) around these boundaries. The margin is set ideally to the worst case TV noise level for best results, but can be tuned according to the level of tolerance the server has to bit flips. A helper data bitstring is constructed during enrollment that records the valid status of each PNDiff data point. The helper data is stored on the server along with the margin, PNMod, challenge and response bitstrings. During regeneration, the server sends the margin, PNMod, challenge and helper data to the token, which uses the helper data to discard the ‘weak’ bits in the response.

The Margin technique significantly improves both the Intra-chip and Inter-chip HD results, as shown on the Fig. 8(b). We used a Margin of 7 as the threshold to identify ‘weak’ bits in the response. Inter-chip HD improves because the PNDiffs corresponding to the generation of the ‘strong’ bits in different chips can now vary. This is true because within-die variations cause PNDiffs for some chips to fall within the margins, while on others, those same PNDiffs are outside the margins. Another important characteristic is the lower sensitivity of the results to whether global variations are present or not, which we indicated earlier is a highly desirable feature.

The size of the smallest bitstring generated by one of the 30 chips is also plotted in Fig. 8(b) to illustrate the overhead associated with the helper data. By selecting a PNMod that is >= 64, the helper data bitstring is no larger than twice the size of the response bitstring in the worst case. It is also possible to use the complement of the helper data to generate a second response bitstring when the

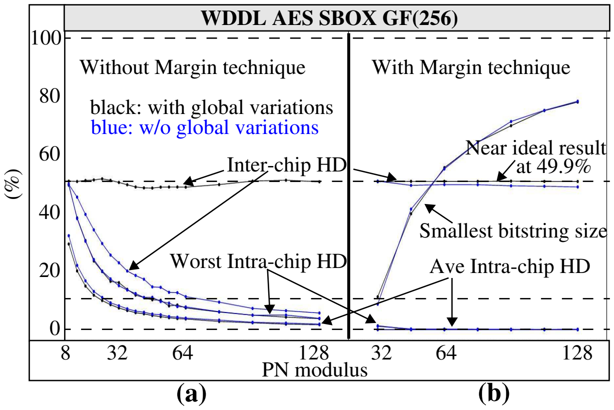

Fig. 9. Hamming distance (HD) results without (a) and with (b) the Margin technique for the WDDL design.

sum of the regions delineated by the margins is equal to the sum of the 'valid' regions defined for '0' and '1'. For example, a PNMod of 64 as shown in Fig. 2 requires the margins to be set to 8, yielding valid regions of size 16. The second response bitstring uses the same set of PNDiffs but first adds an offset equal to 1/4 of the PNMod (16 in the example) before applying the modulus operation, which effectively shifts the distribution and converts all of the previous 'weak' bits into 'strong' bits (and vise versa), thereby making the helper data to response data ratio 1.

The results using the WDDL version are shown in Fig. 9. The longer paths present in the WDDL version are responsible for the improvement in the Inter-chip HD to nearly ideal as shown on the left side in Fig. 9(a). We confirmed this in a separate set of experiments (not shown) in which the path lengths in the Standard version are doubled. Therefore, longer paths improve Inter-chip HD but only in the case where global variations are preserved, i.e., the Inter-chip HD curve without global variations shows a very different result. The results using the Margin technique shown in Fig. 9(b), on the other hand, are nearly ideal with or without global variations. The Intra-chip HD curves also illustrate that the majority of the bit flips that remain in the corresponding results from Fig. 8(b) are attributable to the glitches produced in the Standard version, i.e., margining is not effective for glitches because the change in delay is larger than the worst case TV noise used as the margin. This is evident by the near 0 values for the worse case and average Intra-chip HD for the WDDL version.

5.4 NIST Statistical Test Results

The enrollment bitstrings generated in each of these 8 experiments were used as input to the NIST statistical test suite [26]. The small size of the bitstrings (largest is 2,048 bits), allowed up to 10 of the 15 NIST tests to be applied. The test is classified as passed if at least 28 of the 30 chip bitstrings pass the test. The NIST results are similar for the four sets of results in Fig 8(a), where all tests are passed for PNMod values less than 64. The PNMod of 64 represents a cut-off where some tests are failed but by only 2-3 chip in the worst case. The fail rates increase for PNMods larger than 64, with only a few passing some of the tests at the largest PNMod values. In contrast, the NIST results for the WDDL experiments shown in Fig. 9 are good throughout the entire PN modulus range, with only a few instances of fails, and by only 3 chips in the worst case. These results suggest that glitchy implementations of the FU produce bitstrings of good statistical quality but impose restrictions on the PNMod values, while glitch-free FUs are able to produce high quality bitstrings under a wider range of modulus values.

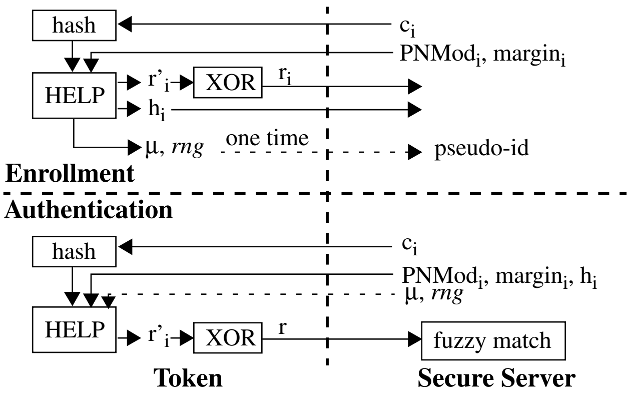

Fig. 10. Proposed authentication protocol.

5.5 ATPG Analysis of Entropy

We used CADENCE Encounter Test (ET) to analyze the number of paths in the WDDL version of the AES SBOX. The underlying entropy source consists of both individual LUT gate delays and the interconnect routing delays, which are combined in unique ways and measured as path delays by HELP. Therefore, the number of paths reflects the amount of entropy present in the functional unit. This analysis will help support our claim that HELP is a strong PUF, with both a large input and output space, when used with functional units in which the number of paths is exponentially related to the number of its inputs.

A WDDL implementation contains two networks of interconnected logic gates (true and complemented) that 'cross-over' at points where inverters occur in the original network. The RC synthesized AND-OR-NOT version of the AES SBOX (see Fig. 3) produced 26 NOT gates in a network of 570 total gates, so the number of cross-overs is fairly limited. With 16 inputs, the expected number of paths would be 2^16 or 65,536. ET reports 15,511 structural paths, which reflects the small interconnection structure between the two networks. As expected, automatic test pattern generation (ATPG) reports that 98.6% of all paths are hazard-free robust testable, which indicates that the paths are independent. Using the set of 512 WDDL vectors (Section 4.4), 37.8% of these paths are tested, which indicates that the remaining paths can only be tested by violating the complementary input patterns required with WDDL. However, testing the WDDL implementation using illegal patterns is possible and recommended when operating the functional unit in PUF mode.

6. Authentication Protocol

The proposed authentication protocol is shown in Fig. 10. During enrollment, the server generates random challenges, c_i, $PNMod_i$ and $margin_i$ which are used by the token as a seed to an LFSR (or a pair of LFSRs to enable arbitrary two vector sequences to be applied). The PUF produces response r_i and helper data h_i, which are stored on the server with the challenge information. In cases where global variations are utilized, a µ and *rng* are also computed for the chip and stored on the server (note these values can also be used as a pseudo-id for the chip). The challenge is optionally passed through a cryptographic hash function to increase the difficulty of model building attacks which attempt to systematically apply a set of seeds designed to carry out path delay tests in a deterministic manner. The hash makes it difficult to determine how to choose c_i such that the output of the hash is controlled to specific seed values. The XOR obfuscation function of the response is optionally added for a similar purpose (note that only one of the input and output obfuscation methods is needed). As indicated in [5], XOR networks amplify bit flip behavior in *r* and therefore, are applicable only when Intra-chip HDs are very low. Authentication is carried out in a similar fashion except for the direction of transmission of the helper data, h_i, µ and *rng*. Note that µ and *rng* are not needed if the PNDiffs are TV compensated to a universal standard (which also eliminates entropy from global variations).

As indicated, the margin and PNMod parameters are also beneficial because they expand the CRP space. However, allowing these parameter to be set without constraints can be used by an adversary to assist with model building. Our experiments suggest that a hard coded margin or allowing only a small range of values, e.g., between 5 and 8, accomplishes the goal of improving the statistics while maintaining a limited information leakage channel. The same is true of the PNMod parameter, where only a limited set of values should be allowed, e.g., restricting to powers of 2 also significantly simplifies the implementation of the modulus operation while providing a 'limited' expansion of the CRP space.

7. Summary and Conclusion

In this paper, we investigated the strengths and weaknesses of using a delay-based strong PUF for authentication. Glitch-free functional units were used as the entropy source and shown to enhance the quality of the generated bitstrings. Within-die variations by itself is not large enough to produce unique bitstrings across a large population of chips. A margining technique is shown to significantly improve the statistical quality of the bitstrings while adding moderately to the storage overhead in the secure database.

The following areas will be investigated in future work. We will investigate the use of ATPG generated input vectors as challenges, which can target additional sources of entropy represented by 'random pattern resistant' paths, that are not likely tested using an LFSR scheme. We will also investigate enrollment schemes which store PNDiffs directly through a one-time interface. These 8-bit values can then be used to generate a set (>> 8) of bitstrings by changing the modulus and margin parameters, thereby improving the storage efficiency on the server. Alternative, lower overhead, glitch-free logic implementation styles will be investigated as an alternative to WDDL. Low power techniques that only reduce the occurrence of glitches will also be investigated.

Although not reported on in this paper, we have also evaluated a voltage-based enrollment (VBE) scheme, which uses the bitstrings generated at a fixed set of supply voltages, in particular, those at the extremes of the specification range, and then records, as weak bits in the helper data, those bits that flip in the regenerated bitstrings. VBE works well to reduce the Intra-chip HD for normally synthesized functional units, i.e., those with glitches. We also found significant diversity is created by the synthesis tool in path delays and the corresponding bitstrings when inconsequential changes are made to the HDL, which again can be used to expand the input/output space of HELP. Last, we are investigating the applicability of techniques described here to board-level authentication as described in [1].

8. References

[1] F. Zhang, A. Henessy, and S. Bhunia, "Robust Counterfeit PCB Detection Exploiting Intrinsic Trace Impedance Variations", *VLSI Test Symposium*, April 2015.

[2] F. Saqib, M. Areno, J. Aarestad and J. Plusquellic, "An ASIC Implementation of a Hardware-Embedded Physical Unclonable Function", *IET Computers & Digital Techniques*, Vol. 8, Issue 6, Nov. 2014, pp. 288-299.

[3] J. Aarestad, J. Plusquellic, D. Acharyya, "Error-Tolerant Bit Generation Techniques for Use with a Hardware-Embedded Path Delay PUF", *HOST*, 2013, pp. 151-158.

[4] http://zedboard.org/product/zedboard

[5] J. Delvaux, D. Gu, R. Peeters and I. Verbauwhede, "A Survey on Lightweight Entity Authentication with Strong PUFs", *Cryptology ePrint Archive: Report 2014/977*.

[6] R. S. Pappu. Physical One-Way Functions. *PhD thesis*, MIT, 2001.

[7] B. Gassend, D. E. Clarke, M. van Dijk, and S. Devadas, "Silicon Physical Random Functions", *Conference on Computer and Communications Security*, 2002, pp. 148-160.

[8] L. Bolotny and G. Robins, "Physically Unclonable Function-based Security and Privacy in RFID Systems", *PerCom*, 2007, pp. 211-220.

[9] E. Ozturk, G. Hammouri, and B. Sunar, "Towards Robust Low Cost Authentication for Pervasive Devices", *PerCom*, 2008, pp. 170-178.

[10] G. Hammouri, E. Ozturk, and B. Sunar, "A Tamper-Proof and Lightweight Authentication Scheme, *Pervasive and Mobile Computing*, 2008, 807-818.

[11] L. Kulseng, Z. Yu, Y. Wei, and Y. Guan, "Lightweight Mutual Authentication and Ownership Transfer for RFID Systems", *INFOCOM*, 2010, pp. 251-255.

[12] A.-R. Sadeghi, I. Visconti, and C. Wachsmann, "Enhancing RFID Security and Privacy by Physically Unclonable Functions", *Information Security and Cryptography*, 2010, pp. 281-305.

[13] S. Katzenbeisser, Unal Kocabas, V. Van Der Leest, A. Sadeghi, G. J. Schrijen, H. Schroder, and C. Wachsmann, "Recyclable PUFs: Logically Recongurable PUFs", *CHES*, 2011, pp. 374-389.

[14] A. Van Herrewege, S. Katzenbeisser, R. Maes, R. Peeters, A.-R. Sadeghi, I. Verbauwhede, and C. Wachsmann, "Reverse Fuzzy Extractors: Enabling Lightweight Mutual Authentication for PUF-enabled RFIDs", *Vol. 7397 of Lecture Notes in Computer Science*, 2012, pp. 374-389.

[15] U. Kocabas, A. Peter, S. Katzenbeisser, and A. Sadeghi, "Converse PUF-Based Authentication" *TRUST,* 2012, pp. 142-158.

[16] Y. S. Lee, T. Y. Kim, and H. J. Lee, "Mutual Authentication Protocol for Enhanced RFID Security and Anticounterfeiting", *WAINA*, 2012, pp. 558-563.

[17] Y. Jin, W. Xin, H. Sun, and Z. Chen, "PUF-Based RFID Authentication Protocol against Secret Key Leakage", *Vol. 7235 of Lecture Notes in Computer Science*, 2012, pp. 318-329.

[18] M. Majzoobi, M. Rostami, F. Koushanfar, D. S. Wallach, and S. Devadas, "Slender PUF Protocol: A Lightweight, Robust, and Secure Authentication by Substring Matching", *Symposium on Security and Privacy Workshop*, 2012, pp. 33-44.

[19] Y. Xu and Z. He, "Design of a Security Protocol for Low-Cost RFID", *WiCOM*, 2012, pp. 1-3.

[20] Y. S. Lee, H. J. Lee, and E. Alasaarela, "Mutual Authentication in Wireless Body Sensor Networks Based on Physical Unclonable Function", *IWCMC*, 2013, pp. 1314-1318.

[21] M.-D. M. Yu, D. M'Rahi, I. Verbauwhede, and S. Devadas, "A Noise Bifurcation Architecture for Linear Additive Physical Functions, *HOST*, pp. 124-129.

[22] S. T. C. Konigsmark, L. K. Hwang, D. Chen, and M. D. F. Wong, "System-of-PUFs: Multilevel Security for Embedded Systems", *CODES*, pp. 27:1-27:10, 2014.

[23] S. Nikova, V. Rijmen and M. Schlaffer, "Using Normal Bases for Compact Hardware Implementations of the AES S-Box", *Security and Cryptography for Networks*, *Lect. Notes in C.S.,* Volume 5229, 2008, pp 236-245.

[24] K. Tiri and I. Verbauwhede, "A Logic Level Design Methodology for a Secure DPA Resistant ASIC or FPGA Implementation", *DATE*, 2004, pp. 246-251.

[25] http://en.wikipedia.org/wiki/Hamming_distance

[26] NIST: Computer Security Division, Statistical Tests, http://csrc.nist.gov/groups/ST/toolkit/rng/stats_tests.html

[27] S. M. Nowick and C. W. O'Donnell, "On the Existence of Hazard-Free Multi-Level Logic", ASYNC, 2003.