IEEE TRANSACTIONS ON
EMERGING TOPICS
IN COMPUTING

Received 8 May 2013; revised 30 August 2013; accepted 21 October 2013. Date of publication 5 November 2013;
date of current version 7 May 2014.

Digital Object Identifier 10.1109/TETC.2013.2289385

Processor-Based Strong Physical Unclonable Functions With Aging-Based Response Tuning

JOONHO KONG (Member, IEEE) AND FARINAZ KOUSHANFAR (Member, IEEE)

Department of Electrical and Computer Engineering, Rice University, Houston, TX 77005 USA

CORRESPONDING AUTHOR: J. KONG (joonho.kong@rice.edu)

This work was supported by the AFRL under Contract FA8750-12-2-0062. This work was supported in part by the NSF under Trust Hub R3C880 and the ARO under YIP Grant R17450.

ABSTRACT A strong physically unclonable function (PUF) is a circuit structure that extracts an exponential number of unique chip signatures from a bounded number of circuit components. The strong PUF unique signatures can enable a variety of low-overhead security and intellectual property protection protocols applicable to several computing platforms. This paper proposes a novel lightweight (low overhead) strong PUF based on the timings of a classic processor architecture. A small amount of circuitry is added to the processor for on-the-fly extraction of the unique timing signatures. To achieve desirable strong PUF properties, we develop an algorithm that leverages intentional post-silicon aging to tune the inter- and intra-chip signatures variation. Our evaluation results show that the new PUF meets the desirable inter- and intra-chip strong PUF characteristics, whereas its overhead is much lower than the existing strong PUFs. For the processors implemented in 45 nm technology, the average inter-chip Hamming distance for 32-bit responses is increased by 16.1% after applying our post-silicon tuning method; the aging algorithm also decreases the average intra-chip Hamming distance by 98.1% (for 32-bit responses).

INDEX TERMS Physically unclonable function, multi-core processor, secure computing platform, post-silicon tuning, circuit aging, negative bias temperature instability.

I. INTRODUCTION

Achieving secure and trustworthy computing and communication is a grand challenge. Several known data/program security and trust methods leverage a root of trust in the processing units to achieve their goals. Microprocessors and other heterogeneous processing cores – which form the kernels of most modern computing and communication – have become increasingly mobile, limiting the amount of available energy and resources. Traditional security and trust methods based on classic cryptography are often computationally intensive and thus undesirable for low power portable platforms. Mobility and low power also favor smaller and simpler form factors that are unfortunately known to be more susceptible to attacks such as side-channels or invasive exploits. There is a search for low overhead and attack-resilient security methods that operate on low power computing platforms.

Physically unclonable function (PUF) is a promising circuit structure to address the pending security needs of several portable and resource-constrained computing platforms. Thanks to the unique and unclonable process variations (PVs) on each chip, PUFs can generate specific signatures for each manufactured IC. Technically, PVs mainly affect threshold voltage (V_{th}) or effective gate length (L_{eff}) of the devices in a chip [1], [2]. These unique device characteristics can be measured by the structural side-channel tests such as timing or current of specific test vectors. To ease integration into higher-level digital security primitives, it is desirable to transform the measured structural test results to digital values. The unclonability and inherent uniqueness properties of signatures makes PUF an attractive security primitive choice [3].

PUF signatures are typically extracted by a *challenge-response protocol*. In response to a *challenge (or input)*, the PUF generates a unique *response (or output)* that is dependent on the specific PV of the underlying chip. PUFs have been classified into two broad categories: Weak and Strong. Weak PUFs have a limited number of challenge-response pairs

(CRPs), which restricts their application scenarios to those requiring a few secret bits such as key generation. Strong PUFs generate an exponential number of CRPs from a limited number of circuit components. Strong PUFs enable a wider range of security and trust protocols by leveraging their huge space of CRPs.

Although the already proposed strong PUFs have shown promising results [4], their application is still limited due to their non-negligible overhead and instability. For example, AEGIS secure processor design [5] which realizes a trustworthy hardware platform, has a non-negligible hardware overhead of the added logic including the arbiter PUF for supporting secure execution. Apart from the PUF logic itself, a large portion of hardware overhead often comes from error correction logic. Since PUFs should be able to produce stable outputs under various environmental conditions (e.g., voltage and temperature fluctuations), error correction logic overhead is inevitable, yet desired to be reduced. Moreover, natural PUFs may have undesirable statistical distributions in terms of inter-chip variations, which significantly restricts their practical applicability. The statistical distribution becomes even worse when spatial correlations between the device characteristics due to process variation (in particular, systematic variations) are prevalent across the chips.

In this paper, we introduce an alternative strong PUF architecture, based on a conventional multi-core processor. Our PUF design is a realization of a low-overhead and stable strong PUF. By leveraging the built-in structures (adders in ALUs) in typical multi-core microprocessors instead of building additional delay logic (e.g., a series of switches and a series of inverter chains in arbiter PUFs and ring oscillator (RO) PUFs [6], respectively), our design realizes a low-overhead and secure strong PUF which can be employed to many security applications. A proof-of-concept implementation is demonstrated on a two-core architecture. To further improve security, reliability, and stability of the PUFs as well as make up for possible drawbacks of the two-core PUF design, we also propose a systematic post-silicon tuning method for our PUF. Our new algorithm leverages an intentional aging method based on one of the most significant circuit aging mechanisms: negative bias temperature instability (NBTI) [7]. Our proposed post-silicon aging algorithm does not incur any performance overhead in most of the chips by careful consideration of selecting the gates that will be intentionally aged. Also, our algorithm greatly improves statistical properties of our PUF design in terms of both inter-chip and intra-chip variations.

Our main contributions include:

- We propose a low overhead strong PUF design, two-core PUF, which leverages built-in components in general processor architectures.
- Our new PUF design shows good statistical results, comparable to the previously proposed strong PUF designs. The hardware overhead of the new PUF is lower than the previously proposed ones.
- We propose a systematic method to further enhance statistical properties of our multi-core PUF in terms of both inter-chip and intra-chip variations by leveraging intentional aging, which complements the possible drawbacks of our PUF design.
- Our simulation results on a two-core architecture prove that our intentional aging algorithms successfully improve the statistical property of the two-core PUF with negligible performance overhead in most cases.

The rest of this paper is organized as follows. Section II outlines background information for process variation, delay model, and circuit aging mechanism/model. Section III explains our two-core PUF design while Section IV introduces our systematic tuning method by leveraging intentional aging to tune the statistical properties of the introduced PUF. Evaluation results for the two-core realization and intentional aging algorithms are discussed in Section V. Section VI provides a brief review of the recent literatures regarding PUFs and intentional post-silicon aging methods. Lastly, we conclude in Section VII.

II. BACKGROUND AND PRELIMINARIES

In this section, we provide general background information and preliminaries for process variation, delay, and aging mechanism. The background and preliminaries are to make the paper self-contained and accessible to a broader audience who may not be familiar with process variation, delay model, and aging.

A. PROCESS VARIATION

Process variation (PV) generates inherent randomness in silicon structures. PV mainly affects threshold voltage (V_{th}) and effective gate length (L_{eff}) of devices, resulting in various side-effects (e.g., delay and power consumption) across chip instances.

PV can be classified into two broad categories: random and systematic variation. Random variation is caused by random dopant fluctuations or random defects in devices. Random variation does not have any spatial correlation between the devices. Unlike random variation, systematic variation incurs spatially correlated device fluctuations. It means that the devices which are close together have a higher probability to have similar device characteristics than those located far away. In contemporary process technologies, both random and systematic variation coexist in manufactured chips.

Figure 1 shows sample V_{th} distribution maps generated by a quad-tree PV model [1]. V_{th} distribution is shown to be fairly random in a single chip as well as across the chips, while similar colors tend to agglomerate together (i.e., V_{th} distributions are spatially correlated).

B. DELAY MODEL

To figure out the V_{th}-dependent gate delay, we use the delay model described in [8]. The gate-level delay model can be

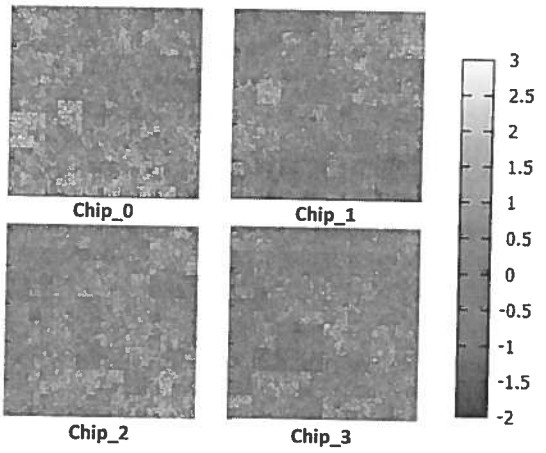

FIGURE 1. Four process variation map examples generated by quad-tree process variation model [1]. The number in the right side of the figures means Z value of Gaussian distribution.

represented as follows:

$$Delay \propto (\frac{L_{eff}}{\phi_t})^2 \times \frac{V_{dd}}{(ln(e^{\frac{(1+\sigma)V_{dd}-V_{th}}{2n\phi_t}}+1))^2} \quad (1)$$

where ϕ_t and σ are thermal voltage and subthreshold slope, respectively. There are several other key factors that affect gate-level delay: supply voltage (V_{dd}), threshold voltage (V_{th}), and effective gate length (L_{eff}). Due to process variations, these factors fluctuate, which in turn results in delay differences across the gates in chips. Furthermore, circuit aging (it will be covered in detail in Section II-C) also affects gate delay since circuit aging increases V_{th} of the gate.

C. AGING MODEL

Circuit aging is a phenomenon in which performance of the circuits is degraded by the circuit usage. This may eventually result in a malfunction of the circuit under intensive utilizations or extreme environmental conditions (e.g., extremely high temperature). Compared to fresh chips (i.e., not aged), aged chips have relatively lower performance due to V_{th} shift by hot carrier injection (HCI) and negative bias temperature instability (NBTI). V_{th} of devices is continuously increased as those devices are switched or have a high duty cycle, resulting in higher delay and lower power consumption.

In deep submicron process technologies, NBTI is known to be the most threatening aging mechanism [7]. Thus, in this paper, we consider NBTI as our main aging mechanism. The V_{th} shift (ΔV_{th}) by NBTI is commonly modeled as follows:

$$\Delta V_{th} = A \times e^{(BV_g)} \times e^{\frac{-E_\alpha}{kT}} \times t^{0.25} \quad (2)$$

where V_g and E_α are gate voltage and activation energy respectively. A and B are technology dependent constants. As shown in Equation 2, the V_{th} shift heavily depends on temperature (T) and stress time (t). By applying this aging model, one can derive an appropriate stress time (t) under a certain temperature (T) to intentionally increase a certain amount of V_{th}.

Stress time t is strongly dependent on the signal probability (SP) [9] that represents a fraction of time when a gate output stays logic high (1) during the circuit operation. Depending on SP of a gate, V_{th} of the gate will be increased (stress period) or decreased (recovery period). Hence, to make the gate intentionally aged, one should carefully determine SP of the gate so that it stays in the stress period much more than in the recovery period.

III. TWO-CORE PUF

A. DESIGN PHILOSOPHY AND DESIGN DECISIONS

1) BASE PLATFORM - MULTI-CORE MICROPROCESSOR

Since our design is fundamentally based on the delay comparison mechanism of arbiter PUFs, we need symmetric (homogeneous) structures to generate diverse path delays affected by process variations. The symmetric multi-core microprocessor is one of the best design candidates since most commodity microprocessors (or microcontrollers) have multiple homogeneous cores.

Typical strong PUF designs have separate delay circuits to generate PUF responses, which incur additional area and power overhead. In contrast, our PUF design utilizes built-in components in typical multi-core microprocessors, which minimizes additional hardware and communication overhead. Compared to the AEGIS design [5] which employs separate switches to implement an arbiter PUF, our design is implementable with a much smaller logic overhead.

2) PATH DELAY SOURCE—ALUs

Our design chooses ALUs as path delay sources. The main reason is that ALUs can accept an exponential number of operands, which can also be used as challenge inputs. Moreover, they can generate challenge-dependent responses when using *add* instructions by stimulating the complex carry-chains in adder structures. *Add* instructions can have an exponential number of different operands (2^{64} with 32-bit operands) and our PUF can also generate an exponential number of diverse responses depending on the challenge inputs as well as disorders in silicon structures. It means our ALU-based PUF design can be classified as a strong PUF.

The other reason for choosing ALUs as path delay sources is that ALUs are combinational logics in microprocessors and they have delay paths which are comprised of a long series of gates. It makes adversaries difficult to perform a model building attack. This is because the adversaries should perform multiple stages of gate-level delay table lookups and additions to obtain the accurate path delays through their PUF model. Determination of carry propagation behaviors also introduce a lot of control dependencies, which means it is difficult for adversaries to exploit the massively parallel computations in order to acquire a PUF response time comparable to that from the real PUF hardware. In this case, one can give a timing constraint (time-bound) during the PUF challenge in order to

distinguish the real PUF and the modeled PUF. Time-bounded authentication by PUF has been introduced earlier [10].

Our PUF design can be applied to any adder structures, though in this paper we build our PUF based on ripple-carry adders (RCAs) as a proof-of-concept. In fact, PUFs are broadly used in small embedded systems (e.g., sensor nodes or RFIDs) [11], [12] or FPGAs [13], [14], [15] in which RCAs are more beneficial for energy-efficiency than high-performance adders such as carry-lookahead adders (CLAs). Note that the first design consideration of those embedded systems is typically energy-efficiency, not performance.

B. OVERALL DESIGN

1) PUF DESIGN

Delay-based PUFs [6] exploit delay differences between multiple paths which have inherently different delays across chips due to process variations. One may deploy arbiters (or counters/comparators in case of ring-oscillator PUFs) to capture the delay difference between two delay lines and convert it into a digitized value. In this paper, we propose an alternative strong PUF design which utilizes already built-in components in a processor architecture as our delay lines instead of building separate delay lines (e.g., a series of the switches in arbiter PUFs or a series of the inverters in ring oscillator PUFs).

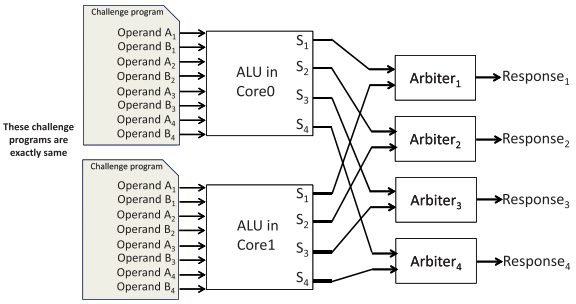

FIGURE 2. The basic structure of our two-core PUF (bit width = 4-bit).

Although our new strong PUF can be built based on any multi-core processor architecture, in the remainder of the paper we focus on a two-core proof-of-concept design. Generalization to more cores is straightforward. Figure 2 shows a high-level design of our two-core PUF. For simplicity, we provide a simple 4-bit two-core PUF design in this figure. Our PUF utilizes arithmetic logic units (ALUs) in the multi-core microprocessors/controllers as symmetric delay lines. In order to give a challenge input to the PUF, the identical challenge program runs in both cores. As shown in Figure 2, two 4-bit operands (operand A and B) are fed into each ALU and a 4-bit output ($S_1 \sim S_4$) can be obtained from each ALU. For delay comparison, the n-th output lines (S_n) from each ALU are connected to the n-th arbiter ($Arbiter_n$). The challenge program should start at the same cycle in both cores to guarantee correct PUF operations. Note that the arbiters in the circuit layout should be very carefully placed for correct operations of the two-core PUF. In addition, the wire lengths from two ALUs to the arbiter should be symmetric not to generate biased PUF outputs.

In our proof-of-concept example, *bitwidth* of our base microprocessor is 32-bit. Hence, each core has a 32-bit ALU. S_n from Core0 and Core1 are connected to the $Arbiter_n$, where n is 1-32. Thus, we need 32 arbiters for delay comparison. Note that our design can be easily extended to 64-bit microprocessors by simply adding 32 more arbiters and connecting the corresponding ALU output ports to those arbiters.

2) SECURITY ENHANCEMENT BY XOR OBFUSCATION

Typical security applications desire a high inter-response variations (i.e., high unpredictability). A low inter-response variation may make the PUF vulnerable to the modeling attack [16] because only a small set of CRPs may enable an accurate modeling of a specific PUF by adversaries. For better inter-response variations of our PUF design, one can deploy an additional XOR obfuscation step between two different response bits as described in [17].

By paying a little more hardware cost, one can perform an XOR operation between i-th bit and $(i + \frac{bitwidth}{2})$-th bit from a response, as shown in Figure 3. PUF operations should be performed twice with different challenges in order to generate a *bitwidth*-bit response, which also incurs timing overhead. Considering the trade-off among the hardware cost, performance, and security, one can employ the additional XOR obfuscation step only for the case where a high level of security is required.

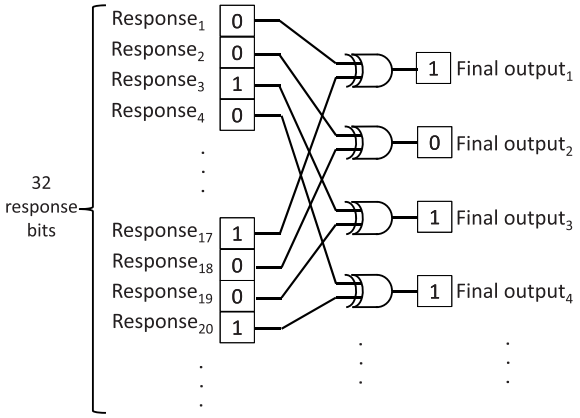

FIGURE 3. Additional logic for XOR obfuscation.

As shown in Figure 4, the inter-response variation is greatly improved by adding the XOR obfuscation step. Comparing between the case with and without XOR obfuscation, an average inter-response Hamming distance is increased from 5.06 bits to 10.64 bits and from 11.81 bits to 20.53 bits when using 32-bit and 64-bit two-core PUF, respectively.

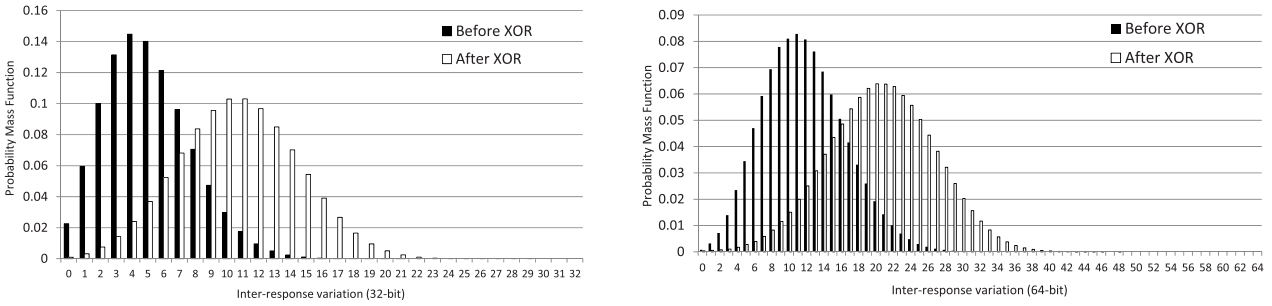

FIGURE 4. Inter-response Hamming distance variations when 10,000 random different inputs are fed into the two-core PUF. The x-axis and y-axis corresponds to the Hamming distances and probability mass function.

C. DETAILED DESIGN AND ARCHITECTURAL MODIFICATIONS

Delay characteristics in our PUF depend on the carry propagation behavior in the conventional ripple-carry adder (which is included in ALUs). As shown in Figure 5, two operands (A_i and B_i) are fed into the full adders. Between the full adders, there are carry bits (C_i), which depend on the operands (A_i and B_i) and previous carry bit (C_{i-1}). Depending on the carry bit, delay characteristics of the full adder rely on those of either the preceding full adders or only the current full adder. These carry propagation behaviors generate an exponential number of the signal propagation behaviors in the adder, which eventually enables a generation of challenge-dependent PUF outputs. The summation result bits (S_i) from the ALU (in each core) are connected to the arbiters. S_i is also connected to the ALU output storage which is already implemented in general processor architectures, though it is not shown in Figure 5. The signals from two separate ALUs race to the arbiter, which in turn generates a digitized output depending on which delay line is faster. The arbiter output is stored to a temporary register ('PUF Response$_i$' in Figure 5).

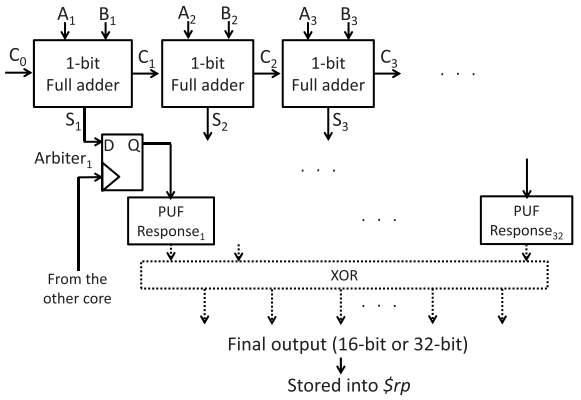

FIGURE 5. A more detailed structure of our two-core PUF. For simplicity, only one arbiter and one temporary register (flip-flop) are shown in the figure. The XOR obfuscation logic is drawn in a dashed-line since it is an optional logic.

As we explained in Section III-B.2, the response bits may be XOR-ed together (i-th bit $\oplus$ ($i + \frac{bitwidth}{2}$)-th bit) and the XOR-ed results are finally stored into one half of the final output register (*$rp*: a special purpose register to store the output from the two-core PUF). The other half of the output register is filled by performing the PUF operation once again with different challenge inputs. After the results are stored to the PUF output register, the challenge program can access this register for later usages.

D. CHALLENGE PROCEDURE

In order to give challenge inputs to our two-core PUF, we utilize a software-level challenge program. Figure 6 shows an example program for a PUF query based on MIPS assembly codes. One-time PUF query is performed as follows. Before starting the PUF operation, the operands (A and B) are loaded into the registers (Line 1-2 in Figure 6). The actual PUF operation is performed by four consecutive addition operations (Line 3-6 in Figure 6). Among these four *add* instructions, the instructions in Line 3 and 5 in Figure 6 are used to initialize the ALU output ports to '0' and '1', respectively. In addition, these instructions also initialize the signals in the carry propagation chains (from C_1 to C_{32}) to '0'. The *add* instructions in Line 4 and 6 in Figure 6 are to perform an actual PUF operation by stimulating the internal gates in the ALUs. The instructions in Line 3-4 and Line 5-6 are dedicated to capture 0 $\rightarrow$ 1 and 1 $\rightarrow$ 0 transitions in the arbiter, respectively. In this work, we use dual-trigger latches (arbiters) to capture both up (0 $\rightarrow$ 1) and down-transitions (1 $\rightarrow$ 0). Note that the operating system can block the other program execution during the PUF operation to prevent the unintended resource (ALU) sharing which may incur cycle-level discrepancy between the two cores.

An example assembly code for one-time PUF query			
1	*addi*	*$1, $0, A* (or from register)	# load operand A to register r1
2	*addi*	*$2, $0, B* (or from register)	# load operand B to register r2
3	*add*	*$5, $0, $0*	# initialization for delay measurement
4	*add*	*$3, $1, $2*	# the first add operation - $r3=$r1+$r2
5	*addi*	*$5, $0, 0xffffffff*	# initialization for delay measurement
6	*add*	*$3, $1, $2*	# the second add operation - $r3=$r1+$r2

FIGURE 6. An example challenge program (instruction sequence) for one-time PUF query (bitwidth = 32-bit).

E. PRACTICALITY ISSUES

Since our design utilizes an in-built structure (adder) instead of the specialized circuit for PUF, some imple-

mentation issues may arise. In this subsection, we address several practicality issues of the two-core PUF design.

1) INTERMEDIATE SIGNAL FLUCTUATIONS IN THE OUTPUT PORT

In the general circuit structures, there could be some ripples (fluctuations of the signal before capturing the true signal) in the output port. If the multiple input ports are connected to one output port, these fluctuations may occur because signal propagation delays from those input ports connected to the output port are likely to be diverse. Therefore, if the path delay sources for a delay-based PUF are generated from the general circuit structures, it could be problematic due to the ambiguity of when to capture the transition signal in the arbiters (i.e., selecting the signal to capture).

However, in the case of a ripple-carry adder that constitutes the path delay sources in the two-core PUF, the signal in the output ports fluctuates **at most twice**. In most cases, the first and second output signal fluctuations result from the operands fluctuations (i.e., when A_i and B_i are fed into the full adder) and carry propagations (i.e., a signal transition in C_{i-1}), respectively. Once the carry signal (C_{i-1}) is converted from 0 to 1, it does not make a transition into 0 again within one *add* operation, which restricts the maximum number of possible transitions in the output port of the adder to 2.

There can be 6 different cases of signal fluctuations captured by the arbiter in our PUF: $0 \rightarrow 1 \rightarrow 0$, $1 \rightarrow 0 \rightarrow 1$, $0 \rightarrow 1$, $1 \rightarrow 0$, $0 \rightarrow 0$ (not fluctuating from 0), and $1 \rightarrow 1$ (not fluctuating from 1). Among them, only the cases of $0 \rightarrow 1$ and $1 \rightarrow 0$ generate valid outputs in the arbiters. In the other cases, the values generated in the arbiter are ignored. The following subsection describes sorting of valid and invalid output bits.

2) SORTING OF THE VALID AND INVALID OUTPUT BITS

In order to make use of only valid output bits, one may need additional MUXes between the arbiter and temporary register to generate desired PUF outputs.

As shown in Figure 7, one can deploy a MUX between the arbiter and temporary register. By referring to the control signal, the MUX selects the value either from the arbiter or from the temporary register. Control signals can be generated by referring to the summation result bit (S_i). In the first phase of the PUF query, which corresponds to Line 3-4 in Figure 6, S_i is directly fed into the control port of the MUX. If S_i is '1', the MUX selects the value from the arbiter whose output is generated by capturing $0 \rightarrow 1$ transitions. Otherwise, the arbiter output is ignored by selecting the temporary register value in the MUX. In contrast, the negation of S_i is fed into the control signal of the MUX in the second phase (Line 5-6 in Figure 6) to identify the valid arbiter output generated by capturing $1 \rightarrow 0$ transitions.

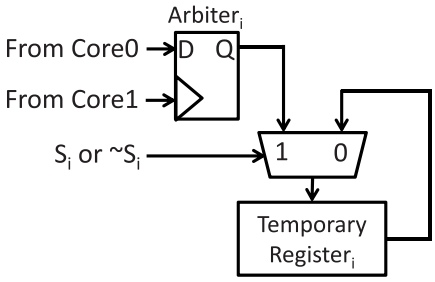

FIGURE 7. Selection of the valid PUF outputs by using a MUX.

3) RUNTIME TEMPERATURE DIFFERENCE BETWEEN TWO CORES

Since our two-core PUF design is based on the structures in different processor cores, there may be a temperature difference between two cores which may incur delay differences (i.e., delay behavior may be biased). Since thermal behaviors of the two cores are likely to be diverse depending on characteristics of the program previously executed before the PUF operation, it may make our PUF responses different from the expected responses.

To deal with different thermal behaviors of two ALUs, we can employ thermal sensors to detect the temperature difference between the ALUs. Typical microprocessors already have the thermal sensors in their expected localized hotspots [18], which means one does not need to deploy additional thermal sensors only for the two-core PUF. To guarantee the PUF operation correctness, operating systems (OSes) can read the temperature from the thermal sensors before the PUF operation begins. If there is a temperature difference between two ALUs, the OS cools the hotter ALU down by enforcing the sleep mode. Though it may incur performance overhead due to the sleep period in one core, the performance loss is insignificant in the authentication process (i.e., it is not performed in general program runtime, but only in authentication program runtime).

For a design-level solution, one can utilize two ALUs from one core in the case of superscalar processors. The ALUs in one core are likely to have similar thermal behaviors due to their close physical distance. Otherwise, one can also add redundant ripple-carry adders in the microprocessor, which will yield a little more hardware overhead, though our PUF has only a small implementation overhead (a detailed analysis on the hardware overhead will be described in the following subsection).

F. IMPLEMENTATION OVERHEAD

Typical microprocessors or microcontrollers already have several cores or ALUs to support multi-programmed/multi-threaded workloads or higher instruction-level parallelism (ILP). Our PUF design realizes a strong PUF with much lower hardware overhead by leveraging built-in components. Assuming one builds a two-core PUF based upon already built-in ALUs in a 32-bit processor, an additional hardware

cost is only arbiters, MUXes, and temporary storage for 32-bit data. If one needs an additional XOR obfuscation stage, only additional XOR gates are to be added. Even if one builds a two-core PUF without an underlying processor architecture, our PUF design only needs 96 2-to-1 MUXes (or 288 NANDs), 128 XOR gates, 32 arbiters, and 32 flip-flops including the logic shown in Figure 7. Compared to the conventional arbiter PUF (32-input/32-output) which needs 2048 2-to-1 MUXes and 32 arbiters, our PUF design incurs far less hardware cost. As a result, our PUF design yields much lower area/power overhead compared to the conventional strong PUF designs.

IV. POST-SILICON TUNING OF TWO-CORE PUF VIA INTENTIONAL AGING

A. HIGH-LEVEL DESCRIPTION OF OUR POST-SILICON TUNING

1) RATIONALE

Though our two-core PUF provides fairly good statistical distributions in general cases (see Section V-B), the manufactured PUFs may not show sufficiently good statistical properties in practice. In this case, one may have to discard manufactured PUF chips due to the low quality statistical properties, which results in yield losses of the chips.

The two possible problematic conditions for our manufactured PUFs include:

- Low inter-chip variations: This case may often happen (particularly for two-core PUFs) because the two ALUs in each core are not close together, which may result in systematic bias between two ALUs across the chip instances [19]. It can also be incurred by suboptimal layouts (i.e., asymmetric placement of the arbiters and different wire lengths between each ALU to the arbiters) as well as inherent process variation. In this case, regardless of the process variation in chips, arbiter outputs would be biased to either '0' or '1', which in turn results in losing the uniqueness of the PUF instances (i.e., reduced randomness among the PUF outputs from different chip instances).
- High intra-chip variations: Delay-based PUFs are often susceptible to various environmental conditions. PUFs should be able to produce stable outputs even under extreme environmental situations. Fluctuating environmental conditions include voltage/temperature variation and arbiter metastability. Since circuit delay is heavily dependent on the voltage/temperature variations, the PUF output might also be diverse under voltage/temperature variations. Arbiter metastability is another source of PUF output instabilities. In general arbiter-based delay PUFs, if the delay difference between two delay lines connected to one arbiter is less than the setup and hold time of the arbiter, the PUF output is not stable and may fluctuate depending on environmental conditions.

Manufactured PUF instances should avoid those two conditions that definitely degrade quality of the PUFs. In this paper, we introduce a systematic intentional aging method to make the statistical quality of the two-core PUF much better in terms of both inter- and intra-chip variations. Our aging method complements the possible drawbacks of our PUF design.

2) STRATEGY

Since the aging process is a one-way process and may degrade circuit's performance, a careful intentional aging strategy is desired. In particular, our PUF design leverages in-built structures as our path delay sources (i.e., not deploying additional dedicated circuits). In this case, aging may in turn degrade the entire circuit performance. In pipelined processors, though the execution stage where a processor performs ALU operations [20] does not typically lie in the critical path of processors [21], performance of a few chips which have their critical path in the execution stage may be adversely affected by increased ALU delay after the intentional aging process.

Our aging strategy is to apply intentional aging only to the gates which do not lie in the critical path of the adder. Figure 8 shows the structure of a full adder which is a substructure of our two-core PUF. The critical path of the entire adder is a carry propagation chain (NAND2 and NAND3 gates), which implies XOR1, XOR2, and NAND1 gates except those in the last full adder (FA) do not affect the critical path delay. In summary, the XOR1, XOR2, and NAND1 are safe to apply the intentional aging while the NAND2 and NAND3 gates might be very sensitive to the circuit's entire performance. Thus, to minimize side-effects from the intentional aging, we selectively apply intentional aging only to XOR1, XOR2, and NAND1 gates in the full adder.

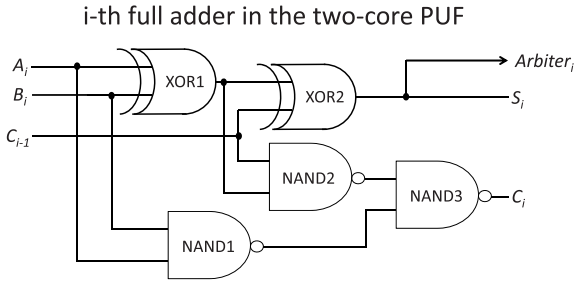

FIGURE 8. ***i*-th full adder structure in the two-core PUF.**

A careful selection of the full adders that must be intentionally aged is also important. In our PUF design, the n-th bit response is closely related to delay characteristics of the n-th full adder. Our strategy is to utilize statistical metrics to determine which full adders have bad statistical properties. First, we figure out which PUF output bits show a relatively bad statistical quality by investigating its output behaviors (i.e., the probability of occurring '0' and '1' in each PUF output bit). And then, we choose the full adders which correspond to those PUF output bit. Our main targets for intentional

aging algorithms are XOR1, XOR2, and NAND1 gates in the selected full adders.

3) FIGURING OUT THE INPUT VECTORS FOR AGING

Figure 9 shows how to generate input vectors for intentional aging. One input vector is an operand A = 0x*ffffffff* (unsigned) and B = 0x00000000 (unsigned) with an initial carry bit $C_0 = 1$ (see Figure 5), assuming 32-bit two-core PUF is used. The first input vector (two operands) is stationary regardless of which full adder (FA) is aged. For the other input vector, operand A is an operand of all '0' (A = 0x00000000). The operand B of the second input vector has different bit sequences depending on which full adders are intentionally aged. The *i*-th bit of the operand B is '1' if the *i*-th FA must be aged and the rest of bits are all '0'. For example, let us suppose that the first and third FA should be aged. In this case, one can make the operand B of the second input vector as 0x00000005 (i.e., within the 32-bit operand, only the first and third bit are '1' and all other bits are '0'). Note that the initial carry bit of the second input vector should be '0' ($C_0 = 0$). For our aging process, the first and second input vectors are fed into the two-core PUF alternately.

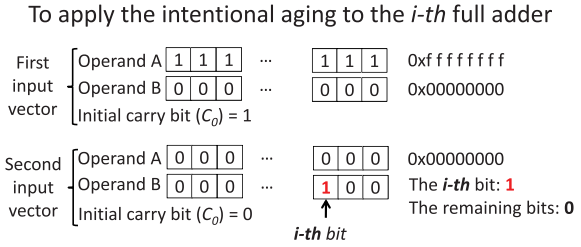

FIGURE 9. Input vector generation for our intentional aging process.

Our aging input vector generation leverages the stress and recovery mechanism of CMOS NBTI [22]. When a gate has an output of '1', the current passes through the PMOS devices, which means the gate is in the stress period of NBTI. Otherwise, the gate is in the recovery period. Thus, to age a gate, one should enforce the gate in the stress period more than the recovery period.

As we explained in Section IV-A.2, our goal is to age only XOR gates and NAND1 gate in the full adder. The first input vector enforces all of the full adders to reside in the state 5 in Table 1. In this cycle, XOR1, NAND1, and NAND3 gates are in the stress period (i.e., gate output = 1) while XOR2 and NAND2 gates are in the recovery period (i.e., gate output = 0). In the next cycle, by using the second input vector, the full adders which must be aged are enforced to be in the state 2 in Table 1 while the other full adders are in the state 0.

Table 2 shows the ratio between the stress and recovery period when our first and second aging input vector are alternately fed into the two-core PUF. As a result, only XOR1 and NAND1 gate are aged while the other gates are minimally

TABLE 1. A truth table of full adders.

State	A_i	B_i	C_{i-1}	XOR1	XOR2	NAND1	NAND2	NAND3
0	0	0	0	0	0	1	1	0
1	0	0	1	0	1	1	1	0
2	0	1	0	1	1	1	1	0
3	0	1	1	1	0	1	0	1
4	1	0	0	1	1	1	1	0
5	1	0	1	1	0	1	0	1
6	1	1	0	0	0	0	1	1
7	1	1	1	0	1	0	1	1

TABLE 2. Stress/recovery period ratio and duty cycle of each gate in a full adder (FA).

	FA in which aging is applied			FA in which aging is not applied		
	Stress period	Recovery period	Duty cycle	Stress period	Recovery period	Duty cycle
XOR1	100%	0%	1	50%	50%	0.5
XOR2	50%	50%	0.5	0%	100%	0
NAND1	100%	0%	1	100%	0%	1
NAND2	50%	50%	0.5	50%	50%	0.5
NAND3	50%	50%	0.5	50%	50%	0.5

affected because the stress and recovery period occur alternately. Though it seems that NAND1 gates are aged far more than the other gates, NAND1 gates hardly affect output delays since they are neither directly connected to the paths to the arbiters nor placed on the critical path of the adder.

NAND2 and NAND3 gates may also be a little aged together due to the partial recovery mechanism of NBTI. In the case of stress period = 50% and recovery period = 50% (i.e., duty cycle = 0.5), the gate is aged 2∼3 times less than the gate with a stress period of 100% [7]. However, assuming that one increases V_{th} by 0.1V via our aging process, the entire impact on the critical paths of the adder is only 7.53% in the worst case. Note that increasing V_{th} by 0.1V is fairly sufficient to obtain a good statistical property of our PUF (detailed results will be shown in Section V-C). It means that our aging process hardly affects the entire circuit performance because most processors have their critical path in the cache access (MEM stage) or register file access pipeline stage (RF/ID stage) [21]. Note that there is no additional hardware overhead required only for our intentional aging which can be performed with the specialized input vectors or programs.

During our aging process, the wires as well as logic gates would be aged. However, the NBTI aging mechanism mostly affects PMOS devices [7], which means the aging in wires is negligible compared to the aging in the logic gates.

For efficient intentional aging in the post-silicon stage, designers or manufacturers can perform the intentional aging process with appropriately high temperature environment. We also note here that the high temperature environment used in our intentional aging process should not incur any break down in devices, but only accelerate aging process.

4) SAMPLE SPACES TO MEASURE THE STATISTICAL PROPERTIES

There are two types of the sample spaces which are used in our aging algorithms: inter-chip sample space and intra-chip sample space.

- *inter-chip sample space*: it is composed of the PUF responses from different chip instances when the same

challenge input is fed. Note that one can feed a large number of challenges to generate a representative and sufficiently large number of *inter-chip sample spaces*. In this paper, we give 1,000,000 random challenge inputs to the PUF instances (i.e., we generate 1,000,000 *inter-chip sample spaces*).

- *intra-chip sample space*: the bit samples are the PUF responses from the same chip with the same challenge under different environmental conditions. The sample space is composed of 11 PUF responses and each response is extracted under different environmental conditions. There are three factors considered to generate different environmental conditions: voltage variation, temperature variation, and arbiter metastability. For generating 11 different environmental parameters, the voltage and temperature are randomly selected within the range of 1.0V-1.2V and 253K-393K, respectively. Arbiter metastability is also considered as a source of unstable PUF responses. For high representativeness of the samples, 1,000,000 different *intra-chip sample spaces* are generated with 1,000,000 different challenge inputs for each chip sample.

B. AGING ALGORITHM TO INCREASE INTER-CHIP VARIATIONS

To increase inter-chip variations, one should make the probability occurring '0' and '1' as close as possible for each response bit across PUF instances to minimize a bias in the responses. For *i*-th PUF response bit, *i*-th full adder (FA) mainly contributes to the delay to *i*-th arbiter. Thus, in the case that one tries to change the *i*-th bit response, one can selectively apply the aging process to the *i*-th full adder. Depending on the occurring frequency of 0 and 1 for each response bit in the *inter-chip sample spaces*, one can determine which core's *i*-th full adder must be aged.

Algorithm 1 Algorithm for Intentional Aging to N-bit Two-Core PUF to Increase Inter-Chip Variations.

```
Input: Statistical distribution of the PUF results
       in the inter-chip sample space
1   for i ← 1 to k // Main loop
2       Update PUF results and statistical distribution;
3       for j ← 1 to N
4           P_j ← prob. of occurrence of '1'
                  in j-th bit within the responses
                  across different chip instances;
                  (i.e., in the inter-chip sample spaces);
5           if(P_j ≥ 0.6)
6               Apply aging (V_th increment of 0.01V)
                to j-th FA in Core0;
7           if(P_j ≤ 0.4)
8               Apply aging (V_th increment of 0.01V)
                to j-th FA in Core1;
9       endfor
10  endfor
```

Algorithm 1 shows a detailed algorithm to make inter-chip variations of the two-core PUFs higher. For the input of this algorithm, the statistical distribution of the PUF responses in the *inter-chip sample spaces* is required. To determine which core's full adders must be aged, our algorithm investigates the P_j which represents probability of occurring '1' in *j*-th bit in the *inter-chip sample spaces* (Line 4 in Algorithm 1). In this paper, when updating the PUF statistical distributions with the generated *inter-chip sample spaces*, we use 1,000,000 random challenge programs **which are newly selected for each iteration**. Thus, the bias towards a certain set of challenges (i.e., 1,000,000 challenges used for updating the statistical distributions) is removed. If P_j is greater than 0.6, it means that the *j*-th FA in the core0 tends to be faster than that in the core1. Thus, our algorithm apply the aging process to the *j*-th FAs in Core0. Our algorithm increases V_{th} of the XOR1 gate in the FA by 0.01V for each iteration of the intentional aging process. The input vectors for our aging process are alternately fed into the PUF until the V_{th} of the XOR1 gate is increased by 0.01V. By using Equation 2, one can obtain an appropriate stress time to increase V_{th} by 0.01V under a certain temperature. The input vector sequence for selectively applying the aging process to a specific FA is already explained in Section IV-A.3. On the other hand, if P_j is less than 0.4, our algorithm applies the aging process to *j*-th FAs in Core1 to make Core1's FAs slower than before. This process is iterated k times and the number of iterations (k) can be determined by considering the degree of the delay bias. Note that this algorithm is applied globally to the chips.

C. AGING ALGORITHM TO REDUCE INTRA-CHIP VARIATION

To reduce intra-chip variation, one should make the delay difference between two delay lines (which are connected to the arbiters) larger so that the PUF can be stable under a certain degree of the environmental variabilities. By doing so, we try to make the probability of occurrence of unstable PUF responses as low as possible.

Algorithm 2 Algorithm for Intentional Aging to N-bit Two-Core PUF to Reduce Intra-Chip Variation.

```
Input: Statistical distribution of the PUF results
       in the intra-chip sample space
1   for i ← 1 to k // Main loop
2       Update PUF results and statistical distribution;
3       for j ← 1 to N
4           P_j ← prob. of occurrence of '1' in j-th bit
                  from the responses in the sample space
                  across various environmental parameters;
                  (i.e., in the intra-chip sample spaces);
5           if(P_j ≥ 0.5)
6               Apply aging (V_th increment of 0.01V)
                to j-th FA in Core1;
7           else
8               Apply aging (V_th increment of 0.01V)
                to j-th FA in Core0;
9       endfor
10  endfor
```

Algorithm 2 describes our algorithm to reduce intra-chip variations. With a given or updated statistical distribution

of the PUF responses in *intra-chip sample spaces* (as in Algorithm 1, one million random challenges which are newly chosen for each iteration are used), our algorithm applies the aging process to make the PUF responses more stable. For $j = 1$ to 32, if P_j is greater than 0.5 (i.e., Core0's j-th FA tends to be faster than Core1's j-th FA), then our algorithm ages the j-th FA in Core1 to make the delay difference between the FAs in Core0 and Core1 larger than before. Otherwise, our algorithm ages j-th FAs in Core0. Similar to Algorithm 1, V_{th} of the XOR1 gate in the FA is increased by 0.01V for each iteration and the main loop is repeated by k times. Unlike Algorithm 1, Algorithm 2 is individually applied to each chip.

Applying the intentional aging by using our algorithms may incur a huge post-processing cost. In case that the huge post-processing overhead is expected, one can selectively apply the Algorithm 2 to the PUF chips of which stability does not meet a quality standard, which can be determined by PUF designer or manufacturer considering the field usage of the PUFs. There can also be an efficient trade-off between the post-processing cost and PUF quality standard (or yield of the manufactured PUF), though analyzing the detailed trade-off between them is out of scope of this paper.

D. RELIABILITY AND SECURITY DISCUSSIONS ON AGING

For our PUF design, there are several security, reliability, and maintainability issues on aging: malicious usage of our aging algorithm and natural aging effect. In this subsection, we address those issues and introduce possible countermeasures.

1) MALICIOUS USAGE OF OUR ALGORITHM

An adversary may try to make the PUF responses of a certain chip as he or she desires by using our algorithms' capability of changing the PUF responses. For example, the adversary can try to use our aging algorithm in the opposite way, which may make the statistical property of our PUF worsened. One possible way to prevent this attack is to deploy aging sensors [23], [24], which can detect how much the circuit is aged by measuring the frequency of ring oscillators or delay elements. If the aging sensor detects a certain degree of the aging within a short time period, the OS can enforce the PUF to reside in a sleep mode so that the ALU can be cooled down and stop executing the malicious code. As another solution, already employed thermal sensors can also detect the execution of the malicious code for malicious aging. This is because the malicious code which tries to age the PUF by an adversary should intensively access the ALU, which makes it significantly hot. It triggers dynamic thermal management (DTM) to prevent thermal emergency in a microprocessor [18] [25]. The DTM also cools down the ALU by engaging the coercive sleep mode in the microprocessor. Since the NBTI aging heavily depends on the circuit temperature, cooled ALUs are affected little from the malicious aging by the adversary.

2) NATURAL AGING AND RECOVERY DUE TO THE NBTI

The natural aging may affect the PUF responses. However, there are two important reasons which support the claim that our PUF design is safe against the natural aging effect. i) Our PUF is based on the delay comparison between two symmetric delay lines by using the arbiter. In real usage cases, both paths are generally aged together, so that the PUF responses are not likely to be affected by natural aging effects. ii) NBTI mechanism has a recovery period (i.e., increased V_{th} due to the aging is recovered to a certain extent) when PMOS devices are not used. It means that the PUF structure may not be aged too much under the assumption that the most of the gates have a duty cycle of 0.5. For a safeguard mechanism, one can also deploy the aging sensors as introduced in the previous subsection, in order to detect the natural aging as well as malicious aging.

In addition to the natural aging, increased V_{th} due to our intentional aging may be recovered by the NBTI recovery. In this case, the improved statistical property of the PUFs may be worsened again. However, the recovered V_{th} by the NBTI recovery cannot reach the original V_{th} due to the partial recovery of the NBTI [7]. In case that a high degree of natural NBTI recovery is expected, one can make that the partial recovery from NBTI hardly affects the PUF responses (e.g., further conducting the aging process considering the expected partial recovery). Note that quantifying the impact of NBTI recovery and investigating a detailed mechanism for the aging process considering the NBTI recovery are out of the scope of this paper. We leave them as our future work.

V. EVALUATION

A. EVALUATION SETUP

Our evaluation results are based on an accurate gate-level delay simulation framework. We gave a threshold voltage (V_{th}) variation to each chip instance. Though, in the case of V_{th}, random variation is known to be more remarkable than systematic variation [1], V_{th} distributions are also spatially correlated [2], [21], [26]. Hence, we used a quad-tree process variation model [1] to precisely model both the random and systematic variation of the V_{th}. Assuming V_{th} distributions in a chip follow the normal distribution $N(\mu, \sigma^2)$, we considered three different process variation severities: $3\times\sigma/\mu = V_{th}\times 20\%$, $V_{th}\times 30\%$, and $V_{th}\times 40\%$. We generated 1,000 different chip instances for our Monte Carlo simulations for each process variation severity. Our ALU (adder) model is based on the Xilinx fast ripple-carry adder model [27]. The placement information is used to map V_{th} parameters (generated by [1]) to each gate in a chip. We obtained a nominal gate delay from HSPICE circuit simulations with 45nm process technology. By using Equation 1, we figured out delay of each gate according to V_{th} of that gate. Note that we exclude the simulation results when the additional XOR obfuscation step is deployed since it is not essential but optional in our PUF design.

TABLE 3. Average inter-chip Hamming distance results.

	Mean		
$3\times\sigma/\mu$	$V_{th}\times 20\%$	$V_{th}\times 30\%$	$V_{th}\times 40\%$
32-bit output	12.38 (38.69%)	12.21 (38.16%)	12.48 (39.00%)
64-bit output	22.67 (35.42%)	21.92 (34.25%)	22.92 (35.81%)
	Standard deviation		
$3\times\sigma/\mu$	$V_{th}\times 20\%$	$V_{th}\times 30\%$	$V_{th}\times 40\%$
32-bit output	2.76	2.75	2.76
64-bit output	3.83	3.80	3.84

TABLE 4. Average intra-chip Hamming distance variation results under 1,000,000 different challenge inputs.

	Mean		
$3\times\sigma/\mu$	$V_{th}\times 20\%$	$V_{th}\times 30\%$	$V_{th}\times 40\%$
32-bit output	3.59 (11.22%)	2.77 (8.66%)	1.36 (4.25%)
64-bit output	6.76 (10.56%)	5.06 (7.91%)	3.31 (5.17%)
	Standard deviation		
$3\times\sigma/\mu$	$V_{th}\times 20\%$	$V_{th}\times 30\%$	$V_{th}\times 40\%$
32-bit output	1.79	1.59	1.14
64-bit output	2.46	2.16	1.77

B. STATISTICAL RESULTS FOR TWO-CORE PUFs

1) INTER-CHIP VARIATIONS

In this subsection, we present inter-chip variation results of our two-core PUF for both 32-bit and 64-bit ALUs. To quantify the inter-chip variations, we measure inter-chip Hamming distances between different PUF instances when we feed the same challenge program to both cores. We show the average inter-chip Hamming distance results from 1,000,000 different challenge inputs (challenge programs).

Table 3 shows the inter-chip Hamming distances (mean and standard deviation) across three difference process variation severities. As Table 3 suggests, uniqueness of our two-core PUF is comparable to the existing strong PUF designs [4]. On average, the inter-chip Hamming distance is 12.35 bits (38.61% - ideally 50%) within the 32-bit responses. Regardless of process variation severities ($3\times\sigma/\mu = V_{th}\times$ 20%, 30%, and 40%), the inter-chip variations are shown to be around 38%. It means that our two-core PUF can be a low-overhead alternative for conventional strong PUFs.

We also provide the results for 64-bit two-core PUF design since many commodity microprocessors are using the 64-bit datapath. Across three process variation severity cases, the average inter-chip Hamming distance is 22.51 bits (35.16%).

2) INTRA-CHIP VARIATIONS

In this subsection, we present intra-chip variation results under various environmental circumstances. We consider three cases that can affect the intra-chip variations: voltage variation, temperature variation, and arbiter metastability.

Table 4 shows intra-chip variation results. To estimate the intra-chip variations, we first give the same challenge to the same chip by 11 times and collect the PUF responses. Each of 11 PUF operations is performed under random environmental conditions (the voltage and temperature are randomly selected within the range of 1.0V-1.2V and 253K-393K, respectively). Arbiter metastability also generates some noise to PUF responses. Intra-chip Hamming distance results are collected under 1,000,000 different challenge inputs and also averaged out to obtain the final results.

As shown in Table 4, the average intra-chip Hamming distances are 2.58 bits and 5.04 bits (8.05% and 7.88% - ideally 0%), in the case of 32-bit ALUs and 64-bit ALUs, respectively. Since our PUF design is based on the delay comparison mechanism of the arbiter, most of the intra-chip variations are due to the arbiter metastability. One thing worth noting is the intra-chip Hamming distances under more severe process variation scenarios tend to be lower. This is because the delay differences between two symmetric delay paths tend to be higher under severe process variations, which makes our PUF more robust under a certain level of the environmental variations.

Though the intra-chip Hamming distances under less severe process variation scenarios may seem to be non-negligible, it can be alleviated by our post-silicon intentional aging algorithms. The post-silicon tuning results for intra-chip variation reduction will be presented in Section V-C.2.

C. STATISTICAL RESULTS FOR POST-SILICON TUNING

1) FOR INTER-CHIP VARIATION IMPROVEMENT

To figure out the effectiveness of our proposed intentional aging method, we performed a Monte Carlo simulation with 1,000 different chip instances. The process variation severity is $3\times\sigma/\mu = V_{th}\times 30\%$. In this subsection, we provide two different practical cases to apply our intentional aging method to increase inter-chip variations. The first is a normal case where the delay of two cores are not significantly biased. The other is an extreme case in which the delay of two cores is significantly biased [19]. For the extreme case, after generating the chip instances using the quad-tree process variation model, we gave an additional 5% delay bias effect between two cores in each chip instance so that the ALU in one core tends to be faster than that in the other core.

TABLE 5. Average inter-chip Hamming distance results before and after our intentional aging process.

	Normal case	Extreme case
Before aging	12.47 (38.96%)	5.79 (18.10%)
After aging	14.47 (45.23%)	13.58 (42.45%)

Table 5 shows average inter-chip Hamming distance (HD) results before and after our intentional aging process. Before applying our intentional aging process, the baseline (before aging) inter-chip HD is 12.47 (12.47/32 = 38.96% – 50% is an ideal case) and 5.79 (5.79/32 = 18.10%) for the normal and extreme case, respectively. It is quite natural that the inter-chip HD for the extreme case is lower than that for the normal case because it has a higher possibility not to have unique responses but to have biased responses across the different chip instances. For the normal case, after

applying Algorithm 1 with $k = 20$, the inter-chip HD becomes 14.47 (14.47/32 = 45.23%), which means uniqueness of the PUF responses across different chips is significantly improved. For the extreme case, the average inter-chip HD becomes up to 13.58 (13.58/32 = 42.45%). It implies that our intentional aging method makes the chip instances practically usable even in the case that a significant delay bias exists between two cores due to systematic process variations.

To see how one can determine the parameter k in practice, we also show a trend of the inter-chip HD as we increase k in Figure 10. After 7 iterations (i.e., V_{th} increase by at most 0.07V via our intentional aging), the inter-chip HD is almost saturated. In other words, only with 7 iterations one can gain the maximum obtainable uniqueness for a certain set of the chip samples.

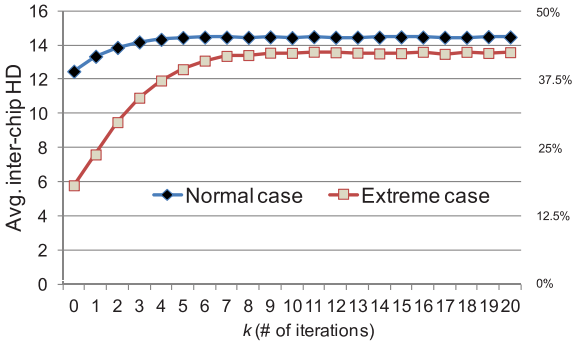

FIGURE 10. Average inter-chip Hamming distance results with regard to the number of iterations (k) in Algorithm 1.

2) FOR INTRA-CHIP VARIATION REDUCTION

Figure 11 shows effectiveness of Algorithm 2. Before applying Algorithm 2, the average intra-chip Hamming distance (HD) is 3.77 (3.77/32 = 11.78% – 0% is an ideal case), which implies there exist Hamming distances of 3-4 bits upon the repetitive measurements with the same challenge. However, after applying Algorithm 2 to each chip with $k = 20$, the intra-chip HD is reduced to 0.07 (0.07/32 = 0.26%), which implies a significant the intra-chip HD reduction. With only 3 iterations of our algorithm (i.e., V_{th} increase by at most 0.03V via our intentional aging), one can get the the intra-chip HD values below 1.0, which implies that there is an average of at most only one-bit Hamming distances within the 32-bit responses upon the repetitive measurements. In this case, one can deploy a light-weight error correction method (e.g., single error correction double error detection) instead of the high overhead error correction methods such as BCH coding [5], [6]. If the PUF designers want to make the PUF responses in terms of intra-chip variations more robust, it is possible to apply a higher k in Algorithm 2.

Figure 12 shows the average intra-chip HD per bit results within the 32-bit responses before and after applying Algorithm 2 with $k = 20$. It is measured by slicing the 32-bit

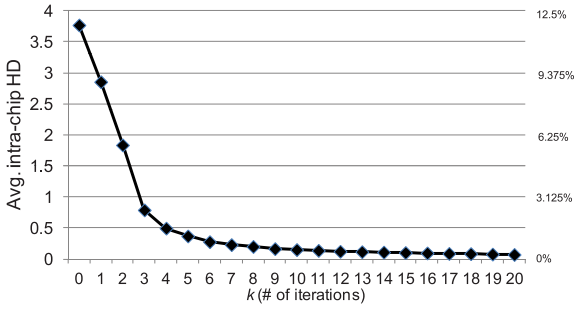

FIGURE 11. Average intra-chip Hamming distance results with regard to the number of iterations (k) in Algorithm 2.

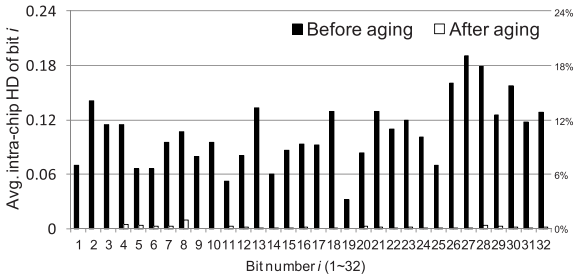

FIGURE 12. Average intra-chip Hamming distance of bit i (i = 1–32) results within 32-bit responses before and after applying Algorithm 2 with $k = 20$.

response into each bit and measuring the HD for each bit i of the PUF responses in the *intra-chip sample spaces*. Though the average intra-chip HD per bit depends on the delay characteristics of each chip, after applying Algorithm 2, the intra-chip HD of bit i becomes under 0.01 for all is (1-32), which means the bit responses become very stable across different environmental conditions. On average, our algorithm reduces the average intra-chip HD of bit i from 0.11 to 0.002 after 20 iterations of the intentional aging process, which means the average intra-chip HD per bit is reduced by 98%.

VI. RELATED WORK

A. PHYSICALLY UNCLONABLE FUNCTIONS (PUFs)

A plausible method for unique and unclonable identification of devices and objects is based on the inherent and hard to forge randomness or disorder of their underlying physical fabrics. To overcome the exposure associated with storage of digital keys, a novel class of secret embedding, storage, and extraction widely known as PUFs has emerged. The secret generation and storage mechanisms in PUFs are based on the inherent disorder present in the silicon [3]. Memory-based PUFs, which are a type of weak PUFs [4], [28], [29], are typically used for secure key storages. Arbiter PUFs [6], that are known to belong to the strong PUF family, are composed of a series of switches (MUXes), which change delay paths according to the input challenge bits. For better statistical properties and to make the structure resilient to modeling attacks, different PUF outputs can also be XOR-ed [30].

Ring-oscillator (RO) PUFs [6] are composed of a long chain of inverters. Glitch PUFs [31] exploit a glitch propagation variability along the delay paths. In [5] and [32], the PUF structures combined with microprocessor architecture are proposed. Apart from PUF design studies, there exists work in literature on detailed analysis [33] [34], formal models [35], and modeling attacks on PUFs [16].

In this work, we proposed a new strong PUF design, which is fundamentally based on delay comparison between two symmetric paths by using arbiters. Our PUF design is instruction-controlled, and leverages built-in components, i.e., arithmetic logic unit (ALU) in a classic processor architecture for path delay sources instead of deploying separate delay sources as presented in [6].

B. LEVERAGING AGING TO PUFs AND CIRCUITS

Circuit aging is a common mechanism by which performance of the circuits is degraded as they are used. Though a large body of work for aging resilience in circuit structures has been studied, in this paper, we focus on the case where one leverages the intentional aging of the PUFs for tuning the statistical properties of the PUF responses. Reference [36] provided the first set of formal properties for the statistical distribution of the PUF responses in terms of the inter-chip and intra-chip variation.

A hardware aging-based software metering technique [37] precisely tracks the software usage by feeding the test vectors to the specific circuit. Device-aging based PUF design [38] leverages aging mechanism to shape the PUF responses. It can also be used for a graduation of the PUF responses which is robust to PUF modeling attacks or for better statistical properties of the PUF by changing the PUF responses. Public PUFs (PPUFs) [39], [40], [41] leverage the aging to shape the PUF responses. The main purpose of applying aging to PPUFs is to make the responses of the PUFs, which are shared among the trusted parties, identical for low-power consumption and fast authentication. Leveraging intentional aging for generating stable outputs in SRAM (static random access memory) PUFs was also proposed [42]. Negative bias temperature instability (NBTI) aging mechanism enables a more stable output generation from SRAM PUFs.

To the best of our knowledge, our work is the first to introduce systematic aging of a strong PUF (two-core PUF) to get a better statistical distribution of PUF responses (i.e., signatures) both in terms of inter-chip and intra-chip variations.

VII. CONCLUSION

In this paper, we proposed a two-core strong PUF architecture. Our design is low overhead and robust to systematic variations because of its inherently symmetric construction. To improve the statistical distribution of the PUF outputs, we devised a novel intentional aging algorithm which makes the PUF instances much more secure and stable in terms of both inter- and intra-chip variations. Our evaluation results suggest that our proposed algorithms greatly improve the quality of the PUF challenge-response statistical properties. By applying the algorithm to increase inter-chip variations, one can obtain the PUF responses which have higher uniqueness across different chip instances. Also, the algorithm to reduce intra-chip variations make our PUF much more robust to the environmental fluctuation, which also enables a deployment of low overhead error correction schemes for robustness and stability of our PUFs.

ACKNOWLEDGMENT

The contractor acknowledges government support in the publication of this paper. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of AFRL.

REFERENCES

[1] B. Cline, K. Chopra, D. Blaauw, and Y. Cao, "Analysis and modeling of CD variation for statistical static timing," in *Proc. IEEE/ACM Int. Conf. Comput., Aided Design*, Nov. 2006, pp. 60–66.

[2] S. R. Sarangi, B. Greskamp, R. Teodorescu, J. Nakano, A. Tiwari, and J. Torrellas, "VARIUS: A model of process variation and resulting timing errors for microarchitects," *IEEE Trans. Semicond. Manuf.*, vol. 21, no. 1, pp. 3–13, Feb. 2008.

[3] U. Ruhrmair, S. Devadas, and F. Koushanfar, *Security Based on Physical Unclonability and Disorder*. New York, NY, USA: Springer-Verlag, 2011.

[4] R. Maes and I. Verbauwhede, *Physically Unclonable Functions: A Study on the State of the Art and Future Research Directions*. New York, NY, USA: Springer-Verlag, 2010.

[5] G. E. Suh, C. W. O'Donnell, and S. Devadas, "AEGIS: A single-chip secure processor," *Inf. Security Tech. Rep.*, vol. 10, no. 2, pp. 63–73, 2005.

[6] G. E. Suh and S. Devadas, "Physical unclonable functions for device authentication and secret key generation," in *Proc. 44th ACM/IEEE DAC*, Jun. 2007, pp. 9–14.

[7] R. Vattikonda, W. Wang, and Y. Cao, "Modeling and minimization of PMOS NBTI effect for robust nanometer design," in *Proc. 43rd Annu. Design Autom. Conf.*, 2006, pp. 1047–1052.

[8] D. Markovic, C. Wang, L. Alarcon, T.-T. Liu, and J. Rabaey, "Ultralow-power design in near-threshold region," *Proc. IEEE*, vol. 98, no. 2, pp. 237–252, Feb. 2010.

[9] F. Najm, "A survey of power estimation techniques in vlsi circuits," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 2, no. 4, pp. 446–455, Dec. 1994.

[10] M. Majzoobi and F. Koushanfar, "Time-bounded authentication of FPGAs," *IEEE Trans. Inf. Forensics Security*, vol. 6, no. 3, pp. 1123–1135, Sep. 2011.

[11] S. Schulz, A.-R. Sadeghi, and C. Wachsmann, "Short paper: Lightweight remote attestation using physical functions," in *Proc. 4th ACM Conf. Wireless Netw. Security*, 2011, pp. 109–114.

[12] S. Devadas, G. E. Suh, S. Paral, R. Sowell, and T. Ziola, "Design and implementation of PUF-based 'unclonable' RFID ICs for anti-counterfeiting and security applications," in *Proc. IEEE Int. Conf. RFID*, Apr. 2008, pp. 58–64.

[13] L. N. Chakrapani, K. K. Muntimadugu, A. Lingamneni, J. George, and K. V. Palem, "Highly energy and performance efficient embedded computing through approximately correct arithmetic: A mathematical foundation and preliminary experimental validation," in *Proc. Int. Conf. Compil., Architect. Synth. Embedded Syst.*, Oct. 2008, pp. 187–196.

[14] Z. M. Kedem, V. J. Mooney, K. K. Muntimadugu, and K. V. Palem, "An approach to energy-error tradeoffs in approximate ripple carry adders," in *Proc. 17th IEEE/ACM ISLPED*, Jan. 2011, pp. 211–216.

[15] D. G. Bailey, *Design for Embedded Image Processing on FPGAs*. New York, NY, USA: Wiley, 2011.

[16] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber, "Modeling attacks on physical unclonable functions," in *Proc. 17th ACM Conf. Comput. Commun. Security*, 2010, pp. 237–249.

[17] M. Majzoobi, M. Rostami, F. Koushanfar, D. Wallach, and S. Devadas, "Slender PUF protocol: A lightweight, robust, and secure authentication by substring matching," in *Proc. IEEE Symp. SPW*, Jun. 2012, pp. 33–44.

[18] J. Kong, S. W. Chung, and K. Skadron, "Recent thermal management techniques for microprocessors," *ACM Comput. Surv.*, vol. 44, no. 3, pp. 1–42, 2012.
[19] E. Humenay, D. Tarjan, and K. Skadron, "Impact of process variations on multicore performance symmetry," in *Proc. Conf. Design, Autom. Test Eur.*, 2007, pp. 1653–1658.
[20] D. A. Patterson and J. L. Hennessy, *Computer Organization and Design—The Hardware/Software Interface* (ser. Comput. Architecture and Design), 4th ed. San Mateo, CA, USA: Morgan Kaufmann, 2012.
[21] J. Kong, Y. Pan, S. Ozdemir, A. Mohan, G. Memik, and S. W. Chung, "Fine-grain voltage tuned cache architecture for yield management under process variations," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 20, no. 8, pp. 1532–1536, Aug. 2012.
[22] S. V. Kumar, C. H. Kim, and S. S. Sapatnekar, "NBTI-aware synthesis of digital circuits," in *Proc. 44th ACM/IEEE DAC*, Jun. 2007, pp. 370–375.
[23] M. Chen, V. Reddy, S. Krishnan, V. Srinivasan, and Y. Cao, "Asymmetric aging and workload sensitive bias temperature instability sensors," *IEEE Design Test Comput.*, vol. 29, no. 5, pp. 18–26, Oct. 2012.
[24] M. Valdes-Pena, J. F. Freijedo, M. M. Rodriguez, J. Rodriguez-Andina, J. Semiao, I. Teixeira, J. Teixeira, and F. Vargas, "Design and validation of configurable online aging sensors in nanometer-scale FPGAs," *IEEE Trans. Nanotechnol.*, vol. 12, no. 4, pp. 508–517, Jul. 2013.
[25] J. Kong, J. K. John, E.-Y. Chung, S. W. Chung, and J. S. Hu, "On the thermal attack in instruction caches," *IEEE Trans. Dependable Secure Comput.*, vol. 7, no. 2, pp. 217–223, Apr./Jun. 2010.
[26] J. Kong and S. W. Chung, "Exploiting narrow-width values for process variation-tolerant 3-D microprocessors," in *Proc. 49th Annu. DAC*, Jun. 2012, pp. 1197–1206.
[27] P. Zicari and S. Perri, "A fast carry chain adder for Virtex-5 FPGAs," in *Proc. 15th IEEE Medit. Electrotech. Conf.*, Apr. 2010, pp. 304–308.
[28] D. Holcomb, W. Burleson, and K. Fu, "Power-up SRAM state as an identifying fingerprint and source of true random numbers," *IEEE Trans. Comput.*, vol. 58, no. 9, pp. 1198–1210, Sep. 2009.
[29] S. S. Kumar, J. Guajardo, R. Maes, G. J. Schrijen, and P. Tuyls, "The butterfly PUF: Protecting IP on every FPGA," in *Proc. IEEE Int. Workshop Hardw., Oriented Security Trust*, Dec. 2008, pp. 67–70.
[30] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Lightweight secure PUFs," in *Proc. IEEE/ACM ICCAD*, Nov. 2008, pp. 670–673.
[31] D. Suzuki and K. Shimizu, "The glitch PUF: A new delay-PUF architecture exploiting glitch shapes," in *Proc. 12th Int. Conf. Cryptograph. Hardw. Embedded Syst.*, 2010, pp. 366–382.
[32] A. Maiti and P. Schaumont, "A novel microprocessor-intrinsic physical unclonable function," in *Proc. 22nd Int. Conf. FPL*, Aug. 2012, pp. 380–387.
[33] S. Katzenbeisser, Ü. Kocabas, V. Rozic, A.-R. Sadeghi, I. Verbauwhede, and C. Wachsmann, "PUFs: Myth, fact or busted? A security evaluation of physically unclonable functions (PUFs) cast in silicon," in *Proc. 14th Int. Conf. Cryptograph. Hardw. Embedded Syst.*, Sep. 2012, pp. 283–301.
[34] M.-D. M. Yu, R. Sowell, A. Singh, D. M'Raïhi, and S. Devadas, "Performance metrics and empirical results of a PUF cryptographic key generation ASIC," in *Proc. IEEE Int. Workshop Hardw., Oriented Security Trust*, Sep. 2012, pp. 108–115.
[35] F. Armknecht, R. Maes, A.-R. Sadeghi, F.-X. Standaert, and C. Wachsmann, "A formalization of the security features of physical functions," in *Proc. IEEE Symp. Security Privacy*, Aug. 2011, pp. 397–412.
[36] M. Majzoobi, F. Koushanfar, and M. Potkonjak, "Techniques for design and implementation of secure reconfigurable PUFs," *ACM Trans. Reconfigurable Technol. Syst.*, vol. 2, no. 1, pp. 1–33, 2009.
[37] F. Dabiri and M. Potkonjak, "Hardware aging-based software metering," in *Proc. Conf. Design, Autom. Test Eur.*, 2009, pp. 460–465.
[38] S. Meguerdichian and M. Potkonjak, "Device aging-based physically unclonable functions," in *Proc. 48th DAC*, 2011, pp. 288–289.
[39] M. Potkonjak, S. Meguerdichian, A. Nahapetian, and S. Wei, "Differential public physically unclonable functions: Architecture and applications," in *Proc. 48th DAC*, Jun. 2011, pp. 242–247.
[40] S. Meguerdichian and M. Potkonjak, "Matched public PUF: Ultra low energy security platform," in *Proc. 17th IEEE/ACM Int. Symp. Low-Power Electron. Design*, Aug. 2011, pp. 45–50.
[41] S. Meguerdichian and M. Potkonjak, "Using standardized quantization for multi-party PPUF matching: Foundations and applications," in *Proc. IEEE ICCAD*, Nov. 2012, pp. 577–584.
[42] M. Bhargava, C. Cakir, and K. Mai, "Reliability enhancement of bi-stable PUFs in 65 nm bulk CMOS," in *Proc. IEEE Int. Symp. HOST*, May 2012, pp. 25–30.

JOONHO KONG (S'07–M'11) received the B.S. degree in computer science from Korea University, Seoul, Korea, in 2007, and the M.S. and Ph.D. degrees in computer science and engineering from Korea University in 2009 and 2011, respectively. He is currently a post-doctoral researcher with the Department of Electrical and Computer Engineering, Rice University. His research interests include computer architecture design, temperature-aware microprocessor design, reliable microprocessor cache design, and hardware security.

FARINAZ KOUSHANFAR (S'99–M'06) received the Ph.D. degree in electrical engineering and computer science and the M.A. degree in statistics from the University of California Berkeley, in 2005, and the M.S. degree in electrical engineering from the University of California Los Angeles. She is currently an Associate Professor with the Department of Electrical and Computer Engineering, Rice University, Houston, TX, USA, where she directs the Texas Instruments DSP Leadership University Program. Her research interests include adaptive and low power embedded systems design, hardware security, and design intellectual property protection.

She is a recipient of the Presidential Early Career Award for Scientists and Engineers, the ACM SIGDA Outstanding New Faculty Award, the National Academy of Science Kavli Foundation Fellowship, the Army Research Office Young Investigator Program Award, the Office of Naval Research Young Investigator Program Award, the Defense Advanced Project Research Agency Young Faculty Award, the National Science Foundation CAREER Award, the MIT Technology Review TR-35, the Intel Open Collaborative Research Fellowship, and the Best Paper Award at Mobicom.