Practical Public PUF Enabled by Solving Max-Flow Problem on Chip

Meng Li[1], Jin Miao[2], Kai Zhong[3], David Z. Pan[1]
[1]Electrical and Computer Engineering Department, University of Texas at Austin, Austin, Tx USA 78712
[2]Cadence Design Systems, Inc., San Jose, CA USA 95134
[3]Institue of Computational Engineering and Science, University of Texas at Austin, Austin, Tx USA 78712
meng_li@utexas.edu, jmiao@cadence.com, zhongkai@ices.utexas.edu, dpan@ece.utexas.edu

ABSTRACT

The execution-simulation gap (ESG) is a fundamental property of public physical unclonable function (PPUF), which exploits the time gap between direct IC execution and computer simulation. ESG needs to consider both advanced computing scheme, including parallel and approximate computing scheme, and IC physical realization. In this paper, we propose a novel PPUF design, whose execution is equivalent to solving the hard-to-parallel and hard-to-approximate max-flow problem in a complete graph on chip. Thus, max-flow problem can be used as the simulation model to bound the ESG rigorously. To enable an efficient physical realization, we propose a crossbar structure and adopt source degeneration technique to map the graph topology on chip. The difference on asymptotic scaling between execution delay and simulation time is examined in the experimental results. The measurability of output difference is also verified to prove the physical practicality.

1. INTRODUCTION

Many electronic systems require solutions for security, unique identification and authentication. Physical unclonable function (PUF) has been proposed as a promising solution [1–3]. A PUF is a pseudo-random function that exploits the randomness inherent in the scaled CMOS technologies to generate unique output response given certain input challenge. A Public PUF (PPUF) is a PUF that is created so that its simulation model is publicly available but large discrepancies exist between the execution delay and simulation time [4–6]. A PPUF relies on the time gap between execution and simulation to derive its security, which is promising because no secret information needs to be kept, and the enrollment phase before using a PUF (during which large amount of responses need to be characterized and stored) is also eliminated [5]. Therefore, PPUFs are able to underlie multiple public-key protocols and have potentially much more applications compared with traditional PUFs [6].

For a PPUF to be an effective security primitive, execution-simulation gap (ESG) acts as a fundamental property and needs to be justified in terms of theoretical soundness and physical practicality. Theoretical soundness requires the ESG to be bounded rigorously, especially considering the advanced parallel and approximate computing scheme. Physical practicality further requires that the ESG can be realized effectively considering the existing fabrication technique and the generated output must be measurable.

Although a number of PPUF designs have been proposed in the literature over the years [5–9], most of them do not justify the proposed ESG in the two aspects above. The first PPUF is proposed in [6] and relies on exclusive-or networks to convert the delay variation into small voltage glitches. Because the amount of glitches ideally increases exponentially relative to circuit depth, the authors claim that keeping record of all the glitches requires exponential computation. Although the idea is innovative, the PPUF is hard to realize because the generate glitches usually have very small pulse width and are very likely to be attenuated during propagation to output due to the electrical property of the logic gates [10]. Therefore, actual time gap is much smaller compared with the ideal expectation. Another security primitive, termed as SIMulation Possible, but Laborious (SIMPL) system, leverages the time gap between the real optical interference and solving the differential equations underlying the optical system [5]. However, its security against attacks relies on the nonlinearity of optical medium, which is still an open problem. In [8], a nano-PPUF design based on memristors is proposed. The authors justify the ESG by the complexity of matrix multiplication operation used in SPICE simulation. However, the authors ignore that matrix multiplication can be effectively paralleled to reduce the simulation time significantly.

While previous designs are more conceptual, we introduce a practical PPUF whose execution is equivalent to solving the max-flow problem in a complete graph. The equivalence enables us to use the max-flow problem [11] as the simulation model and bound the ESG rigorously. To enable an efficient physical realization of our design, we propose a crossbar structure to map the graph topology to silicon. The PPUF basic building block is designed with MOS transistors working in saturation region and enhanced by source degeneration (SD) technique [12] to instantiate flow constraints on chip. ESG is examined in experimental results by verifying the difference between asymptotic scaling of execution delay and simulation time. We summarize our contributions as follows:

- A new PPUF design is proposed with rigorous ESG achieved by solving max-flow problem in a complete graph on chip.
- A crossbar structure is proposed and SD technique is adopted to map the graph topology and flow constraints on chip and enable an efficient physical realization.

The rest of the paper is as follows. Section 2 describes preliminaries on max-flow problem and discusses the algorithms that aim to solve it. Section 3 introduces our PPUF topology and basic building blocks, which maps the max-flow problem on chip. ESG is also analyzed in Section 3. Section 4 describes the physical realization of the PPUF and also discusses the PPUF challenge-response pairs (CRPs). We evaluate the performance of the PPUF in Section 5 and conclude the paper in Section 6.

DAC '16, June 05-09, 2016, Austin, TX, USA

DOI: http://dx.doi.org/10.1145/2897937.2898067

2. MAX-FLOW PROBLEM IN DIRECTED GRAPH: PRELIMINARIES

Let $G = (V, E)$ represent a directed graph with $|V| = n$ vertices and $|E| = m$ directed edges. If $\forall v_i, v_j \in V, \exists (v_i, v_j) \in E$, G is called complete with $m = n(n-1)$. In the directed graph G, we distinguish a set of source vertices $S \subset V$ and sink vertices $T \subset V$ and assign a non-negative capacity $c(v_i, v_j)$ to each edge $(v_i, v_j) \in E$. An instance of the max-flow problem consists of the directed graph G and the set of capacities.

Given an instance of a max-flow problem, a function, $f : E \to R^+$, is called a flow function if it satisfies the following conservation and capacity constraints:

$$\sum_{(v_i,v_j)\in E} f(v_i, v_j) = \sum_{(v_j,v_k)\in E} f(v_j, v_k) \qquad \forall v_j \in V - S - T$$

$$0 \le f(v_i, v_j) \le c(v_i, v_j) \qquad \forall (v_i, v_j) \in E$$

The value of a flow f is defined as the net flow from a source node. The max-flow problem is to find a maximum-value flow function on a given instance of a flow problem.

Max-flow problem has been shown to be computationally demanding and difficult to parallel [11]. Traditional methods include augmenting path algorithm [13], push-relabel algorithm [14], blocking flow method [13] and so on. All these methods have at least $O(n^3)$ complexity for complete graph. Recent efforts on calculating the exact solution for max-flow problem can be classified into parallel and approximate methods. The best known parallel method shown in [15] leverages blocking flow algorithm and achieve a parallel runtime of $O(n^3 \log(n)/p)$, where $p \le n$ is the number of processors. Therefore, the best achieved complexity of parallel algorithms is lower bounded by $O(n^2 log(n))$. The best known approximate algorithm targeting at max-flow problem is proposed in [16]. To get an ϵ-approximate solution, the complexity of the proposed algorithm is $O(m^{1+o(1)}\epsilon^{-2})$. In our case, for the complete graph, the complexity is $O(n^{2+o(1)}\epsilon^{-2})$. Therefore, considering the parallel and approximate algorithms, the complexity of solving max-flow problem is still lower bounded by $O(n^2)$ with respect to number of nodes in the graph.

While solving the max-flow problem is computational intensive, it is much easier to check the optimality of a flow f. Define the residual capacity $r_f(v_i, v_j)$ of an edge (v_i, v_j) to be $c(v_i, v_j) - f(v_i, v_j)$. The residual graph $G_f = (V, E_f)$ for a flow f is the directed graph whose vertex set is V and edge set E_f is the set of edges with positive residual capacity. f is optimal iff $\forall t \in T, s \in S$, t is not reachable from s in the residual graph. To check optimality, we just need to create the residual graph and do a breadth first search from source to sink, which is highly parallelizable and can be finished with $O(n^2/p)$ complexity for a complete graph [17].

3. PPUF TOPOLOGY AND ESG ANALYSIS

In this section, we introduce our PPUF design and rigorously prove the ESG. Our main intuition is to build a PPUF circuit which is equivalent to solving the max-flow problem but requires asymptotically less time compared with best known algorithms. The main difficulty comes from mapping the constraints and objective functions on chip. As we will show, our PPUF topology together with the basic building block guarantees the equivalence and thus, enables a rigorous ESG.

3.1 PPUF Topology and Basic Building Block

The proposed PPUF topology is shown in Figure 1. The PPUF consists of a pair of nominally identical networks that are different only because of process variation. The circuit nodes correspond to the vertices in the graph while each building block as shown in Figure 2 (d) instantiates one directed edge. Inputs to the PPUF are used to select the source nodes and sink nodes, and control the current capacity of each edge. The selected source and sink nodes are connected to $V(s)$ and ground, respectively. Output is generated by comparing the current flowing into the source node.

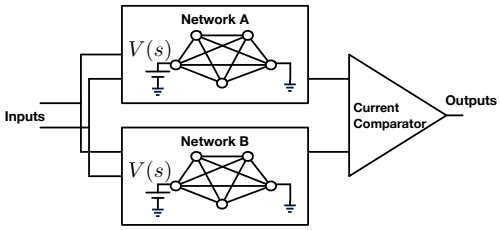

Figure 1: Topology of the proposed PPUF design.

To explain our design methodology for the basic building block, we list our requirements below and describe the proposed circuit block step by step to ensure all the requirements are satisfied.

Requirement 1 *The maximum current of the basic building blocks must be controllable.*

This is because the basic block is used to instantiate capacity constraint on each edge. To satisfy the requirement, we use the MOS transistors working in saturation region and set the gate to source bias (V_{gs}) to control the saturation current as shown in Figure 2 (a). The diodes are used to ensure the direction of the current. However, due to channel length modulation and other short channel effects (SCE), the saturation current still changes as drain to source bias given fixed V_{gs}, which is undesired.

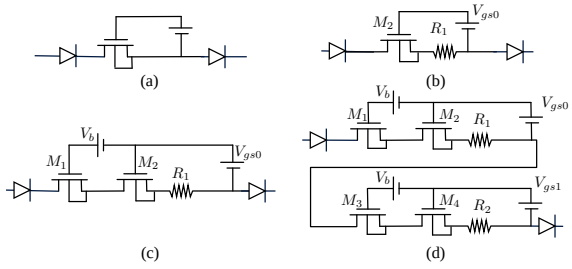

Figure 2: Evolution of basic building block design to satisfy all the requirements.

To reduce the change of saturation current, we adopt the SD technique from analog design. SD technique can help stabilize the current and mitigate the impact of SCE by creating negative feedback with resistors or MOS transistors. In Figure 2 (b), R_1 acts as the degeneration circuit for M_2. After M_2 enters the saturation region, the change of current caused by the increase of drain to source bias can be compensated by the increased voltage drop on R_1. The degeneration circuit can be nested to further suppress the change of saturation current. As in Figure 2 (c), two levels of source degeneration are nested: R_1 works as the degeneration circuit for M_2 while M_2 and R_1 together work as the degeneration circuit for M_1. The additional voltage source (V_b) is used to ensure both M_1 and M_2 are working in saturation region. Figure 3 (a) shows the I-V relation of the three circuits in Figure 2 (a)-(c). As we can see, with SD technique, the impact of SCEs is mitigated. Better control over saturation current can be gained with more levels of SD technique, which can also lead to large design overhead. To decide the sufficiency of the SD technique, we propose the following requirement.

Requirement 2 *The impact of process variation on the saturation current needs to be much larger compared to the inaccuracy induced by SCEs.*

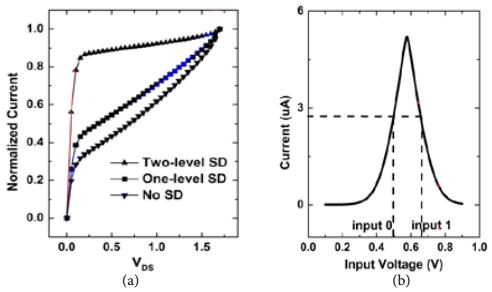

Figure 3: I-V relation of proposed circuit unit: a) comparison on saturation current change for different building block designs; b) relation between saturation current and control voltage V_{gs0}.

This requirement ensures that the inaccuracy will not lead to false response calculated from simulation compared to PPUF execution. Experimental results from SPICE based Monte Carlo simulation indicate that with two-level SD technique, the amplitude of saturation current variation of the basic block is around 130X larger than the current change induced by SCE, which indicates the sufficiency of the two-level SD technique.

Requirement 3 *The boundry between PPUF 0-response and 1-response needs to be nonlinear.*

The requirement aims to ensure good resilience to model-building attack. Model-building attack aims to model the challenge-response behavior of PPUF with machine learning techniques [18]. To ensure the resilience, a nonlinear boundary between PPUF 0-output and 1-output is required. To accomplish this, we replicate the basic building block and connect them in serial as in Figure 2 (d). We also limit the sum of the V_{gs0} and V_{gs1} to be a constant (V_c) and choose the control voltage for input 0 and 1 such that their nominal saturation current are the same as shown in Figure 3 (b). Because the current of the basic building block is limited by different MOS transistors for input 0 and 1, given the current information for input 0, the current information for input 1 remains unknown and vise versa. Meanwhile, because the current of all the edges connected to the node sum up to 0 for each internal node, the current flowing through one edge is not only determined by the voltage of the edge, but also impacted by all the other edges connected to the node. Therefore, all the inputs are closely correlated to achieve a nonlinear boundry between 0-output and 1-output. The requirement is also verified in the experimental results with both parametric and non-parametric model-building techniques.

Besides satisfying all the requirements above, another intriguing property of the building block is its incremental passivity [19]. A memoryless component is incrementally passive if its current increases monotonically as the increase of voltage. The proposed building block satisfies this condition. As we will show, the incremental passivity of basic block helps ensure that the steady state current of the PPUF circuit is optimal solution to the corresponding max-flow problem.

3.2 Lower Bound of PPUF Simulation

In this section, we will prove the equivalence between execution of PPUF and calculation of max-flow in a complete graph to derive the lower bound of the simulation time.

First let us consider the capacity constraints for each edge in the graph. As we have discussed in Section 3.1, the diodes on the two sides of the basic block limit the direction of the current such that it is always non-negative. Meanwhile, once the control voltage is given for the building block, its current is limited by the saturation current I_{sat}. Therefore, for each basic block, we have

$$0 \leq I \leq I_{sat}$$

Flow conservation constraint is realized naturally. Based on Kirchhoff's current law, for each internal node v_i, we have

$$\sum_{(v_i,v_j)\in E} I(v_i,v_j) = \sum_{(v_j,v_k)\in E} I(v_j,v_k)$$

The objective function of max-flow problem corresponds to the current flowing into the source node in the circuit. Based on Kirchhoff's current laws for the source node, we have

$$I(s) = \sum_{(s,v)\in E} I(s,v)$$

where $I(s)$ is the current flowing into the source node.

Because the PPUF is only composed of basic building blocks that are incrementally passive, the circuit is also incrementally passive [19]. Such incremental passivity guarantees:

- As the increase of $V(s)$, the current flowing into the PPUF circuit increases monotonically.
- For any voltage input and initial condition, the circuit will converge to a unique steady-state solution.

Therefore, increasing $V(s)$ will always try to maximize the current flowing into the circuit under the edge capacity constraints and conservation constraints.

Now, we are able to show the mapping between constraints and objective function of max-flow problem with the PPUF circuit. Therefore, we can conclude that the PPUF execution is equivalent to solving a max-flow problem in a directed graph. Specifically, since each node in the PPUF is designed to be connected with all the other nodes, the direct graph is complete. The equivalence enables us to use the max-flow problem as the simulation model. More importantly, because the max-flow problem is hard to parallel and approximate, we are able to rigorously derive the lower bound for the simulation time: with the best known algorithm, the simulation time scales at least $O(n^2)$ as the increase of PPUF node number.

Though hard to simulate, it is much easier to verify the optimality of a solution as described in Section 2. The verifier can leverage the asymmetry between verification and calculation of max-flow problem by asking for the residual edges from PPUF holder or attacker. Based on the information, the verifier can build the residual graph and decide whether the sink is reachable from the source to determine the optimality. As described in Section 2, the verification process can be finished efficiently in $O(n^2/p)$ time.

3.3 Upper Bound of PPUF Execution

Rigorous analysis of ESG requires an accurate upper bound of the PPUF execution time. For the proposed design, the execution delay is the time required for the current from the source node to become stable, which can be upper bounded by the time required for the voltage of all circuit nodes to become stable. In this section, we aim to derive the upper bound on the execution time by considering the charging delay of each node. To be noticed here, different from traditional RC tree structure, driving and loading networks of a vertex in the complete graph are not explicit. We modify the method proposed in [20] to create the delay relation for all the circuit nodes, based on which rigorous upper bound on charging delay can be derived.

Consider a vertex $v_i \in V$, and let $R(v_i, v_j)$ denote the resistance of the edge (v_i, v_j) as shown in Figure 4 (a). [20] proves that we can decompose the capacitance of v_i, denoted as $C(v_i)$, into $n-1$ parts and redistribute each part to all the edges pointing to v_i,

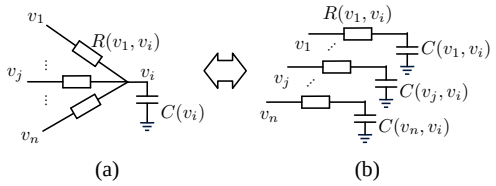

Figure 4: Capacitance decomposition and redistribution for delay estimation.

denoted as $C(v_j, v_i)$, as shown in Figure 4 (b), such that

$$T(v_i) = T(v_j) + R(v_j, v_i)C(v_j, v_i) \qquad \forall v_j \in V - \{v_i\}$$
$$\sum_{(v_j,v_i)\in E} C(v_j, v_i) = C(v_i)$$

Here, $T(v_i)$ denotes the delay from source node to v_i. $C(v_j, v_i)$ can be either positive or negative depending on the relation between $T(v_i)$ and $T(v_j)$, while $R(v_j, v_i)$ is always positive.

Consider the node with largest delay in the PPUF circuit, denoted as u. Then, we have $T(u) \geq T(v), \forall v \in V - \{u\}$. For the redistributed capacitance

$$0 \leq C(v, u) \leq C(u) \qquad \forall v \in V - \{u\}$$

Since in the complete graph, u is connected with source s directly, we have

$$T(u) = R(s, u)C(s, u) + T(s) = R(s, u)C(s, u) \leq R(s, u)C(u)$$

Here $R(s, u)$ is the resistance of the edge connecting s and u, which remains unchanged as the increase of node number. $C(u)$ is the capacitance of node u, which increases linearly because the number of edges incident on u increases linearly. Therefore, the delay for the PPUF scales at most $O(n)$ relative to circuit node number.

The analysis above shows rigorous ESG considering the parallel and approximate computing scheme. The proposed ESG can be further amplified by deploying the feedback loop technique proposed in [5]. Instead of calculating the response for one challenge, the verifier can present the PPUF with a challenge C_1 and force the PPUF holder or attacker to determine a sequence of challenge-response pairs $(C_1, R_1), \ldots, (C_k, R_k)$ with R_k being the final response. The later challenge C_i is determined by earlier response R_{i-1}, where $2 \leq i \leq k$. In this way, the lower bound of the simulation time becomes $O(kn^2)$ and the upper bound of execution delay becomes $O(kn)$ and thus, ESG can be amplified by k times.

4. PPUF PHYSICAL REALIZATION

Although the ESG is proved rigorously in Section 3, realizing a complete graph and the basic building block on chip is non-trivial. In this section, we describe our strategies towards a practical and efficient on-chip realization of the proposed PPUF design.

4.1 Complete Crossbar Structure

The completeness of the graph requires each circuit node to be connected with all the other nodes. To realize the complete connection, we propose a $n \times n$ crossbar structure as shown in Figure 5 (b). In the crossbar structure, the number of horizontal and vertical bars are the same as the node number. The ith horizontal bar and ith vertical bar are connected directly through a wire. These two bars together represent one node in the graph. Then, at the intersection of ith vertical bar and jth horizontal bar $(i \neq j)$, there is a basic building block. The direction of the building blocks are always pointing from the vertical bars to the horizontal bars. In this way, each bar is connected with all the other bars through the basic building blocks, which realizes the complete connection

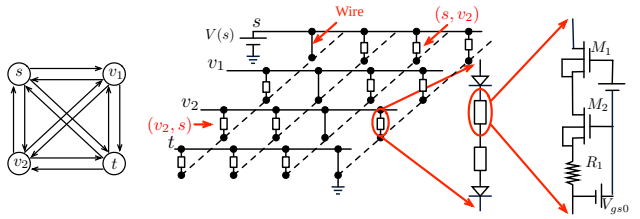

Figure 5: Crossbar structure to map the complete graph on chip: (a) example graph; (b) crossbar structure that instantiates graph in (a).

of vertices in the graph. Figure 5 (b) shows an example of the crossbar structure for the graph in Figure 5 (a).

To ensure enough ESG, the circuit size of the PPUF can be large. Therefore, systematic variation across die must be taken into consideration. To mitigate the impact of systematic variation, we choose to place the transistors in the same positions from the two different networks side by side. In this way, transistors in the same positions can be assumed to have the same systematic variation. Combined with the differential structure of the proposed PPUF, the impact of systematic variation can be suppressed.

4.2 Grid Partition for Control Signal

As we have mentioned, the capacity of each edge is controlled by input signal. Though using one input signal for each basic block can provide very large challenge-response space, the number of individual control signals increases quadratically relative to the node number, which leads to high cost for large design. To reduce the number of voltage sources, we partition the crossbar structure into $l \times l$ grids, where $l \leq n$, and use one input signal to control all the building blocks within the grid. It shall be noted that since we use relative voltage source in our basic block design, we cannot connect different voltage sources with the control signal directly. Instead, we can leverage the input signal to control the charging and discharging of multiple capacitors, which work as the voltage bias for the basic blocks within one grid. In this way, the requirement on using individual independent voltage source is eliminated.

Now, we analyze the CRPs of the PPUF. For a PPUF to be a strong PUF, one important requirement is the large challenge-response space. In the PPUF circuit, we identify two types of control inputs, denoted as type-A inputs and type-B inputs. Type-A inputs are used to choose the source and sink nodes of the network. The chosen source and sink nodes are connected to $V(s)$ and ground, respectively while all the other nodes are left floating. Therefore, the size of type-A input space is $n(n-1)$. Type-B inputs are used to control the maximum current for the circuit units. Since we partition each network into $l \times l$ grids, with one input controlling all the edges within the grid, the size of the Type-B challenge space should be 2^{l^2}. However, we argue that not all the CRPs can be used. This is because to ensure good unpredictability, when a single input bit is flipped, the ideal probability for a output bit to flip is 0.5. We try to achieve the same effect by posing requirement on the minimum hamming distance (HD) between different challenges. To be more specific, we select a subset from the whole challenge space such that the minimum HD for any two challenges in the subset is at least d. We demonstrate the impact of d in the experimental results. To decide the number of challenges that satisfy this requirement, it is equivalent to constructing binary codes of length l^2 and minimum HD d. As proved by [21], the size of the Type-B challenge space is larger than $2^{l^2}/(\sum_{i=0}^{d-1} \binom{l^2}{d})$. Then the total number of CRPs (N_{CRP})

satisfies

$$N_{CRP} \geq n(n-1) \times \frac{2^{l^2}}{\sum_{i=0}^{d-1} \binom{l^2}{d}}$$

Consider a PPUF with $n = 200$ circuit nodes. Assume $l = 15$ and $d = 2l$, then $N_{CRP} \geq 6.53 \times 10^{+35}$. Large challenge-response space makes it impossible for an adversary to enumerate all the CRPs exhaustively.

5. EXPERIMENTAL RESULTS

In this section, we examine the security properties of the proposed PPUF design. The experiments fall into the following categories: accuracy of the simulation model, asymptotic scaling of ESG, PPUF output measurability and power consumption, statistical evaluation of PUF metrics and model-building attack resilience.

The current output and execution delay of the PPUF circuit is acquired using SPICE simulation with 32 nm predictive technology model [22]. 32 nm technology node is chosen because we want to have good control over SCEs while ensure enough impact of process variation. We assume the threshold voltage variation follows normal distribution with a standard deviation of 35 mV, a value consistent with ITRS [23]. Concerning the voltage settings, $V(s)$ is set to be $2V$ because of the voltage drop on the diodes. $V_b = 0.1V, V_c = 1.2V$. If input is 1, V_{gs0}, as shown in Figure 2 (d), is set to be $0.5V$ while if input is 1, V_{gs0} is set to be $0.67V$. The simulation model is implemented in C++. Because the best known sequential and parallel algorithms are more conceptual with no packages available, we instead choose the widely used push-relabel and augmenting-path algorithms from boost library [24]. Meanwhile, because the statistical evaluation for PPUFs with large number of nodes is too time-consuming, we demonstrate most of the tests on relatively small PPUFs and use extrapolation to estimate the performance for large PPUF. The simulation is run on an Intel Xeon 2.93 GHz workstation with 74G memory.

We first demonstrate the accuracy of using max-flow problem as simulation model. We compare the max-flow results from execution and simulation for PPUFs with different number of nodes. We define the inaccuracy as $|I_{max,exe} - I_{max,sim}|/I_{max,exe}$. For each PPUF, we run 100 simulations and show the inaccuracy in Figure 6. As we can see, the average inaccuracy is less than 1%. Compared with the inaccuracy, the average variation of the maximum current flow is around 9.27% for a 100-node PPUF. The comparison ensures that we can get accurate response from the simulation model.

Next, we demonstrate the ESG by comparing the PPUF execution and simulation time. To be noticed here, though it is possible to reduce the simulation time by running on better machine or using more efficient algorithm, we argue that the lower bound of the simulation time still exists and justified ESG can be guaranteed as we have proved. The scaling of execution delay and simulation time is shown in Figure 7 (a). Then, ESG can be calculated as the difference between the execution delay and simulation time. We show the ESG with/without feedback loop technique in Figure 7 (b). For feedback loop technique, we set the loop number to be the same as the node number in PPUF. As we can see, to achieve 1s ESG, which is shown to be a reasonable requirement in [4], 900 nodes are needed for our PPUF design while with feedback loop technique, the required number of nodes reduces to 190.

Another aspect that we investigate is the measurability of PPUF output. This serves as a measure of the PPUF practicality. We measure the average current from the two crossbar structures and their difference because they impose requirements on the input range and resolution of the comparator. We use extrapolation to infer these two parameters for large design based on Figure 8. For a 900-node PPUF, the average current is $33.6\mu A$, while the current difference is $2.89\mu A$. These requirements are easy to accomplish by designs shown in existing papers [25,26], which proves the practicality of the proposed design. We also estimate the power consumption for the 900-node PPUF. The power of the two crossbar structure is around $134.4\mu W$. As for the current comparator, we use the data from [25], which is $153\mu W$. Based on Figure 7 (a), the execution delay for a 900-node PPUF is estimated to be $1.0\mu s$. Therefore, for one evaluation, the total power consumption is around $287.4pJ$.

Metrics	Ideal	40-node PPUF		100-node PPUF	
		Mean	Stdv	Mean	Stdv
Inter-class HD	0.5	0.5009	0.1371	0.4977	0.1075
Intra-class HD	0	0.0673	0.1104	0.0853	0.1321
Uniformity	0.5	0.4946	0.208	0.4672	0.158
Randomness	0.5	0.4946	0.0277	0.4672	0.0361

Table 1: Statistical evaluation on 40-node and 100-node PPUF.

We further examine the PPUF performance over several commonly used metrics that quantify the quality of the PPUF design: inter-class HD, intra-class HD, randomness and uniformity [27]. In our experiments, intra-class HD accounts for supply voltage variation of 10% and temperature variation ranging from $-20\,^{\circ}$C to $80\,^{\circ}$C. We evaluate these metrics for a 40-node and a 100-node PPUF. As we can see in Table 1, the average performance of both PPUFs are close to ideal value.

We also evaluate the relation between output flip probability and minimum HD (d) of PPUF challenge: changing d inputs, we check the probability for the output bit to flip. Here we run experiments on 100 40-node PPUF circuits with gird size $l = 8$. For each PPUF and each minimum HD d, we random sample 1000 input vectors. The change of output flip probability relative to d is shown in Figure 9. As we can see, when $d = 16$, the average output flip probability is approaches 0.5.

To evaluate the model-building attack resilience, we leverage both parametric and non-parametric machine learning algorithms, including Support Vector Machines (SVMs) [28] and K Nearest Neighbor (KNN) [29]. We employ a nonlinear radial bias function (RBF) kernel for SVM algorithm while for KNN algorithm, we run a series of empirical KNN tests with $K = 1, 3, \ldots, 21$. The final prediction inaccuracy is the minimum of SVM and KNN tests. The prediction error for 40-node and 100-node PPUFs is shown in Figure 10. Compared to the arbiter PUF with the same input length, our PPUF achieves more than an order of magnitude higher prediction error than arbiter PUF, which indicates much better model-building attack resilience.

6. CONCLUSION

In this paper, we propose a PPUF with practical ESG in terms of theoretical soundness and physical practicality. The execution of PPUF is proved to be equivalent to calculating max-flow in a complete graph, which enables us to use the max-flow problem as the simulation model to rigorously bound the simulation time. The execution time is also bounded for the proposed design. Therefore, rigorous ESG can be shown based on the difference on the asymptotic scaling. To enable an efficient realization of PPUF, we propose a crossbar structure and adopt the SD technique to build the PPUF basic building blocks to map the complete graph on chip. Our PPUF exhibits good performance as shown in the experimental results.

7. REFERENCES

[1] B. Gassend, D. Clarke, M. Van Dijk, and S. Devadas, "Silicon physical random functions," in *CCS*, 2002.

[2] J. W. Lee, D. Lim, B. Gassend, G. E. Suh, M. Van Dijk, and S. Devadas, "A technique to build a secret key in integrated circuits

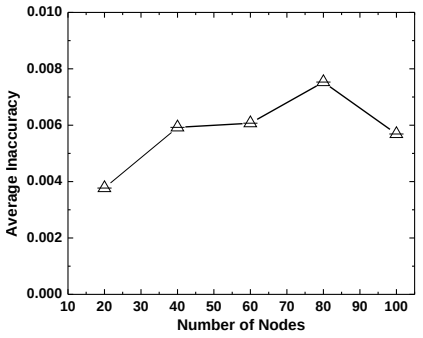

Figure 6: Inaccuracy of simulation model compared with PPUF execution.

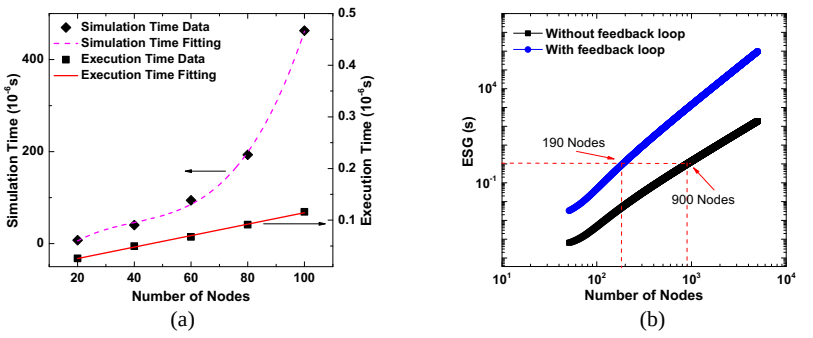

Figure 7: Comparison between execution and simulation time: (a) scaling of execution and simulation time and polynomial fitting; (b) scaling of ESG with/without feedback loop technique.

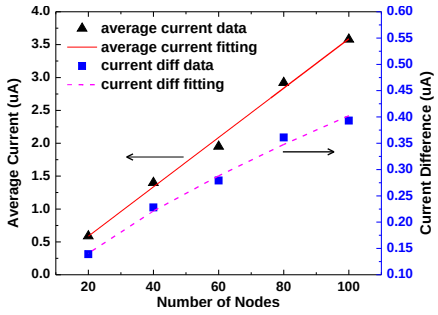

Figure 8: Scaling of output current average and difference.

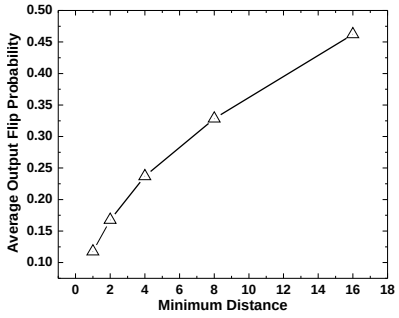

Figure 9: Output bit flip probability with respect to minimum distance of input challenges.

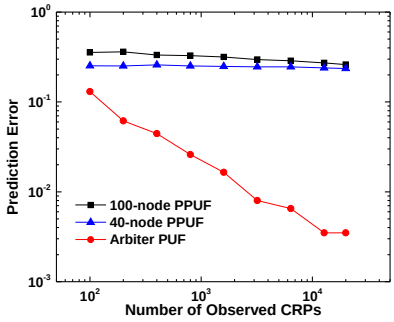

Figure 10: Comparison on prediction error for a 40-node and 100-node PPUF with arbiter PUF.

for identification and authentication applications," in *VLSI Circuits*, 2004.

[3] M. Gao, K. Lai, and G. Qu, "A highly flexible ring oscillator puf," in *DAC*, pp. 1–6, 2014.

[4] M. Potkonjak and V. Goudar, "Public physical unclonable functions," *Proceedings of the IEEE*, vol. 102, no. 8, pp. 1142–1156, 2014.

[5] U. Rührmair, "Simpl systems: On a public key variant of physical unclonable functions.," *IACR Cryptology ePrint Archive*, vol. 2009, p. 255, 2009.

[6] N. Beckmann and M. Potkonjak, "Hardware-based public-key cryptography with public physically unclonable functions," in *Information Hiding*, pp. 206–220, 2009.

[7] M. Potkonjak, S. Meguerdichian, A. Nahapetian, and S. Wei, "Differential public physically unclonable functions: architecture and applications," in *DAC*, pp. 242–247, 2011.

[8] J. Rajendran, G. S. Rose, R. Karri, and M. Potkonjak, "Nano-ppuf: A memristor-based security primitive," in *ISVLSI*, pp. 84–87, 2012.

[9] M. Majzoobi and F. Koushanfar, "Time-bounded authentication of fpgas," *Information Forensics and Security, IEEE Transactions on*, vol. 6, no. 3, pp. 1123–1135, 2011.

[10] T. Karnik and P. Hazucha, "Characterization of soft errors caused by single event upsets in cmos processes," *Dependable and Secure Computing, IEEE Transactions on*, vol. 1, no. 2, pp. 128–143, 2004.

[11] L. M. Goldschlager, R. A. Shaw, and J. Staples, "The maximum flow problem is log space complete for p," *Theoretical Computer Science*, vol. 21, no. 1, pp. 105–111, 1982.

[12] I. Mehr and D. R. Welland, "A cmos continuous-time g m-c filter for prml read channel applications at 150 mb/s and beyond," *Solid-State Circuits, IEEE Journal of*, vol. 32, no. 4, pp. 499–513, 1997.

[13] E. Dinits, "Algorithm of solution to problem of maximum flow in network with power estimates," *Doklady Akademii Nauk SSSR*, vol. 194, no. 4, p. 754, 1970.

[14] A. V. Goldberg and R. E. Tarjan, "A new approach to the maximum-flow problem," *Journal of the ACM (JACM)*, vol. 35, no. 4, pp. 921–940, 1988.

[15] Y. Shiloach and U. Vishkin, "An o (n 2 log n) parallel max-flow algorithm," *Journal of Algorithms*, vol. 3, no. 2, pp. 128–146, 1982.

[16] J. A. Kelner, Y. T. Lee, L. Orecchia, and A. Sidford, "An almost-linear-time algorithm for approximate max flow in undirected graphs, and its multicommodity generalizations," in *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 217–226, SIAM, 2014.

[17] A. Yoo, E. Chow, K. Henderson, W. McLendon, B. Hendrickson, and Ü. Çatalyürek, "A scalable distributed parallel breadth-first search algorithm on bluegene/l," in *SC*, pp. 25–25, 2005.

[18] U. Rührmair, F. Sehnke, J. Sölter, G. Dror, S. Devadas, and J. Schmidhuber, "Modeling attacks on physical unclonable functions," in *CCS*, 2010.

[19] C. Mead and M. Ismail, *Analog VLSI implementation of neural systems*, vol. 80. Springer Science & Business Media, 2012.

[20] T.-M. Lin, C. Mead, *et al.*, "Signal delay in general rc networks," *TCAD*, vol. 3, no. 4, pp. 331–349, 1984.

[21] M. Plotkin, "Binary codes with specified minimum distance," *Information Theory, IRE Transactions on*, vol. 6, no. 4, pp. 445–450, 1960.

[22] W. Zhao and Y. Cao, "New generation of predictive technology model for sub-45 nm early design exploration," *TED*, vol. 53, no. 11, pp. 2816–2823, 2006.

[23] "International technology roadmap for semiconductors." http://public.itrs.net.

[24] B. Schäling, *The boost C++ libraries*. Boris Schäling, 2011.

[25] Y. Sun, Y. Swang, and F. Lai, "Low power high speed switched current comparator," in *MIXDES*, pp. 305–308, 2007.

[26] N. K. Chasta, "A very high speed, high resolution current comparator design," *International Journal of electric, electronics science and engineering*, vol. 7, no. 11, 2013.

[27] A. Maiti, V. Gunreddy, and P. Schaumont, "A systematic method to evaluate and compare the performance of physical unclonable functions," in *Embedded systems design with FPGAs*, pp. 245–267, 2013.

[28] J. A. Suykens and J. Vandewalle, "Least squares support vector machine classifiers," *Neural processing letters*, vol. 9, no. 3, pp. 293–300, 1999.

[29] P. Cunningham and S. J. Delany, "k-nearest neighbour classifiers," *Multiple Classifier Systems*, pp. 1–17, 2007.