ClockPUF: Physical Unclonable Functions based on Clock Networks

Yida Yao, MyungBo Kim, Jianmin Li, Igor L. Markov, Farinaz Koushanfar‡
University of Michigan,2260 Hayward St., Ann Arbor, MI 48109
‡ Rice University, 6100 Mains St, MS. 380, Houston, TX 77005

ABSTRACT

Physical Unclonable Functions (PUFs) extract unique chip signatures from process variations. They are used in identification, authentication, integrity verification, and anti-counterfeiting tasks. We introduce new PUF techniques that extract bits from pairwise skews between sinks of a clock network. These techniques inherit the stability of clock network, but require a return network to deliver clock pulses to a certain region, where they are compared. Our algorithms select equidistant sinks and route the return network, then derive chip-specific random bits from available data with a moderate overhead. SPICE-based evaluation of clock-PUFs using a 45nm CMOS technology validates the operability, stability, uniqueness, randomness, and their low overhead.

1. INTRODUCTION

With the ever-increasing growth in microelectronic devices and applications, there is an equally pressing demand for ensuring product authenticity, security, and reliability of the electronic systems. The US government and semiconductor companies point out potential systems vulnerabilities resulting from the contract foundry model, hardware Intellectual Property (IP) and IC theft, as well as counterfeiting [1, 2].

During November 2011 US Congressional hearings on the DoD supply chain, the Semiconductor Industry Association (SIA) estimated US product-revenue loss due to counterfeiting at $7.5B/yr. In 2012, IHS iSuppli ranked counterfeited semiconductor types as *analog ICs* (25.2%), *microprocessors* (13.4%), *memory ICs* (13.1%), and *PLDs* (8.3%) [18] Standard identification methods, such as printed serial numbers, can be forged. Researches have explored using natural silicon variation to defeat copying and cloning. Unique, random, unclonable process variation of silicon ICs promises resiliency against side-channels and replication attacks. This approach does not require storing secret keys in hardware — a vulnerability in other chip-ID proposals.

Several forms of Physical Unclonable Functions (PUFs) for extraction of unique signatures for the silicon devices have been proposed [5, 6, 20, 7, 9]. When PUF is given an input (*challenge*) it produces an output (*response*) [5]. The response is unique to each device and depends on the specific process variations of each IC. The most dominant approaches for implementing PUFs either leverage the bistable circuit elements such as SRAM arrays [6, 9], or are based on variations in logic gate/wire delays or leakage currents [5, 20]. Signature extraction is often performed by an isolated module that could potentially be replaced or replayed without affecting the circuit's functionality or performance.

This paper introduces ClockPUF, a methodology for extracting the random and unique responses based on the pairwise differences between sink latencies in on-chip clock networks (*clock skews*). To quantify clock skew, our method creates a return network to route the clock pulses to a region where the pulses are compared. A major advantage of ClockPUF is that it is interwoven within the chip's functionality, making separation and tampering difficult. As clock networks are tuned to picosecond accuracy, even small glitches can hamper diagnosis and debugging of the chip. Clock networks are routinely overdesigned to be robust to failures, cross-coupling and environmental variations. They are *exceptionally stable* in practice. In particular, clock skew (from which we derive PUF bits) does not change if the delays of all clock paths change by the same amount should the chip be baked or frozen. The overhead of ClockPUF is low since a clock network occupies only a small fraction of the chip.

Our technical contributions for realization and evaluation of the ClockPUF methodology are as follows:

- The idea to use on-chip clock networks to build resilient PUFs, and an architecture for ClockPUFs.
- Using tunable delay buffers to enhance the entropy of ClockPUFs and enable a challenge-response protocol.
- An algorithm that selects clock sinks for pairwise comparisons such that their locations are equidistant from the location where multiplexors can be placed.
- A methodology for generating return paths for ClockPUF, while minimizing routing congestion.
- Extensive SPICE simulations using the 45nm infrastructure from the ISPD 2010 Clock-network Synthesis Contest (organized by Intel and IBM) to demonstrate the low overhead of ClockPUF and its effectiveness for IP protection. We evaluate the stability, uniqueness, randomness, and resilience to environmental and operational variations.

The remainder of the paper is organized as follows. Section 2 reviews the background and related literature. Section 3 describes architecture of ClockPUF and its blocks. Sections 4.1 and 4.2 discuss sink selection and generation of return paths, respectively. Process-variation characterization and tuning structures are covered in Section 5. Section 6 presents evaluation and analysis.

978-3-9815370-0-0/DATE13/

2. BACKGROUND

Unique chip identifiers based on process variations were first proposed in [12]. The extracted output IDs were fixed set and static, but suitable for cryptographic-key generation. Other static IDs based on bistable SRAM-like cells are [6, 9]. Challenge-response pairs (CRPs) and dynamic authentication can improve the strengths of PUFs based on process variations in nanometer-scale silicon [5]. The PUFs in [5] utilized delay differences between pairs of parallel timing paths with equal nominal delay. PUF bits were generated by a delay arbiter connected to these paths. To increase the number of CRPs, the paths were segmented and multiplexed by challenge bits. An ASIC implementation was demonstrated in 180nm CMOS [11].

Ring-oscillator PUFs exploit differences in the frequency of on-chip oscillators to generate CRPs [20], but PUFs in [5] allow for more CTPs. Postprocessing can increase the number of CRPs [15], but not the entropy available for PUF generation. PUF security and robustness are evaluated in [16] and lightweight techniques to improve PUF security in [17]. Interconnect variations in power-distribution networks was used for building a PUF by measuring the resistive differences of the wires [7, 8]. This was accomplished by passive resistive circuit components to measure voltage drop. Since interconnect variations are typically dwarfed by device variations, using both offers greater entropy for PUF generation.

As chip IDs based on silicon variability may be affected by environmental conditions, adverse effects can be mitigated by error-correction [4, 21, 13]. Further aspects of PUF design, performance, and security foundations applicable to our work can be found in [14, 19, 3].

3. OVERVIEW OF CLOCKPUF

We now introduce our proposed ClockPUF architecture. Using a clock network in a given IC, a ClockPUF compares the arrival times of certain clock signals and generates rather stable but unclonable bits. As illustrated in Figure 1, the major components of a ClockPUF are

- The *tap-outs* that branch out clock signals from their intended sinks without significantly affecting the performance of the clock network. Side capacitance is shielded by a small buffer.
- The return paths that bring branched-out clock signals together so they can be compared. The return paths are buffered and thus accumulate the impact of process variations from different regions of the chip.[1]
- A multiplexing network capable of selecting two clock signals for comparison. It can be distributed or implemented with pairs of adjacent stand-alone multiplexors or using distributed multiplexors.
- A pair of externally controlled tunable delay buffers with matched delays to maximize variational entropy,
- An arbiter unit (SR-latch) to determine which of two given clock signals transitions first.

The tap-outs are added to a few carefully selected sinks, so that return paths have matched lengths and propagation delays (Section 4.1). The arbiter compares pairs of clock

[1]Process variation impacts transistors more than wires.

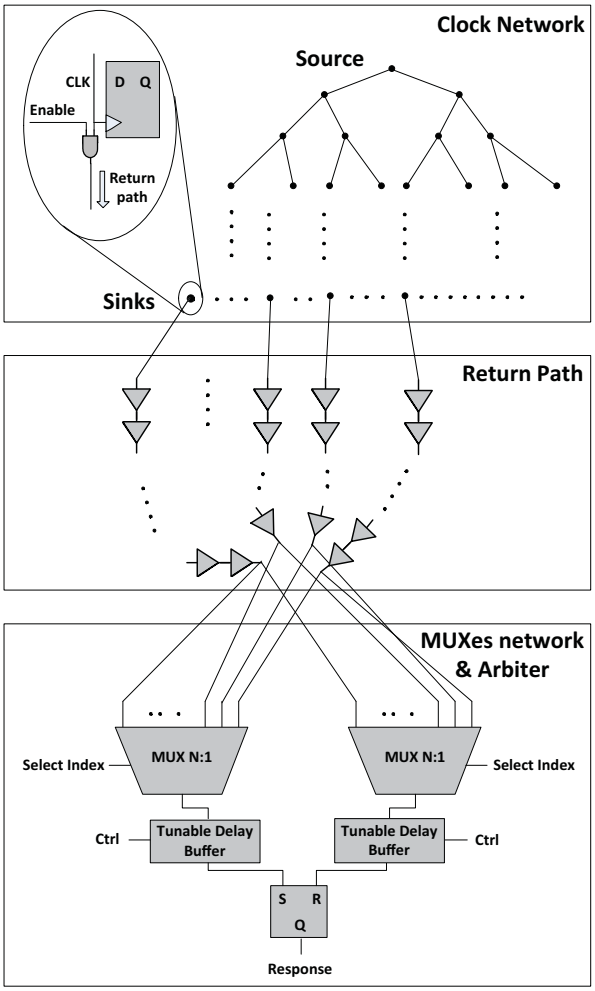

Figure 1: The architecture of ClockPUF.

transitions, producing one bit at a time, while *tunable delay buffers* can compensate for unintentional systematic delay biases encountered on the way. With systematic biases tuned out, the randomness of process variation on return paths is used to produce repeatable but unclonable PUF bits. Return paths are routed through as distant parts of the chip as possible to decrease the impact of spatial correlation. By covering certain regions of the chip, they also provide a limited protection from tampering, as cutting or detouring a return path will affect PUF bits.

3.1 Using an existing clock network

Key characteristics of clock networks useful for PUFs include stability to environmental variations and the fact that attempts at tampering with the clock network will quickly become obvious. Tampering is also obstructed by the picosecond-accurate tuning of modern clock networks [10, Chapter 7]. Despite our use of ASIC-style clock trees for illustration, our PUF proposal is also compatible with low-skew CPU-style clock meshes. As process variation is collected largely (but not only) on return paths (Figure 1), ClockPUF will work even when the skew between all pairs of sinks is zero. A comparable PUF *without a clock network* would require greater interconnect resources and much more careful tuning. Standalone PUFs are also easier subvert than PUFs

integrated with the clock network and spread out across the chip. ClockPUF can exploit any skew present in the clock network. Skew due to process variations increases the entropy of ClockPUF. Design-time skew is compensated for by tunable delay buffers *per sink*, enabling a challenge-response mechanism that thwarts numerous attacks.

3.2 The tap-outs and the return paths

When dealing with gated clocks and multiple clock domains, relevant clock domains must be active during PUF read-out. Working with one domain at a time is easiest, or one may draw different PUF bits from different domains.

After clock signals are tapped out from clock sinks, they must be brought toward the arbiter to facilitate a temporal comparison. ClockPUFs accomplish this using return paths and a multiplexing network. Intuitively, one can think of the return network as a "reverse clock tree". This analogy is significant, as classic algorithms for clock-tree design and optimization can be used to route the return network with smaller overhead. We also make an effort to reduce routing congestion in the return network, as explained in Section 4.2. In addition to minimizing the overhead, this also reduces the correlation between PUF bits and thus increases available entropy. An important aspect of our proposal is selecting the clock sinks to make the return network more efficient.

A minor technical issue in the design of the return network is to minimize the impact of tap-outs on the performance of the original clock network. As illustrated in Figure 1, this can be accomplished by placing a small AND gate near the clock sink to (*i*) enable/disable the return path, and (*ii*) shield away the capacitance of the tap-out. If the input-pin capacitance of this AND gate cannot be neglected, it can be added to the sink capacitance during the clock-network design (so that routine clock-network tuning compensates for it). Figure 2 illustrates the impact of adding tap-outs to a clock tree on the distribution of maximum skew. The data were obtained through 500 Monte-Carlo simulations on bench08 from the ISPD 2010 Clock-Network Synthesis contest. We shielded all tap-outs by buffers, whose input-pin capacitance was only 4-12% of input pin capacitance of a clock sink. Skew distribution in the original tree (without tap-outs) is compared to the skew distribution after adding tap-out buffers (increased loading) to the same tree. The third line shows skew distribution of a similar tree where skew-tuning was informed of larger sink capacitances due to tap-out buffers. As expected, the differences are fairly small.

Due to their length, the return paths include a number of uniformly-spaced buffers. These buffers ensure that the return paths bring the clock signals to the arbiter in one clock cycle (or in two clock cycles, if one clock cycle is too difficult to ensure). They also serve to increase the amount of delay variation observed on these paths. Therefore, we do not use clock buffers, which are larger but less susceptible to process variation. As return paths traverse the chip, they sample process variation more effectively than prior ring-oscillator PUFs that are local in nature.

3.3 The multiplexing network & the arbiter

After N return paths bring relevant clock signals close together, pairwise comparisons are performed by a pair of N-to-1 multiplexors (MUXes) and a delay arbiter connected through tunable delay buffers. Using $2\log_2 N$ *select bits*, one can connect any two return paths to the delay arbiter for comparison. Delay arbiters are implemented by SR-latches to provide better temporal resolution with smaller area than flip-flops. While only one bit is produced by checking which wire transitions faster, additional information can be extracted by adjusting delay buffers placed before the delay arbiter. Tunable buffers also compensate for systematic delay biases and support a challenge-response mechanism.

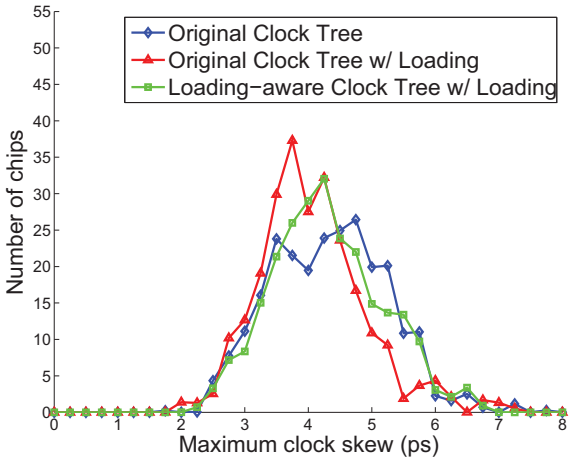

Figure 2: Clock skew distribution over 500 Monte-Carlo SPICE simulations on bench08 from the ISPD 2010 benchmark suite.

One design option is to instantiate two monolithic multiplexors and route the wires to their inputs. In some layouts, this may lead to excessive routing congestion. Alternatively, each N-to-1 MUX can be implemented by a number of smaller MUXes, geometrically distributed to avoid routing congestion and average out area impact. Tunable delay buffers can compensate for delay asymmetries.

To maximize the entropy of pairwise comparisons by adjusting tunable delay buffers, each chip must be characterized through post-manufacturing delay testing of the return paths and the multiplexor network. Such a characterization would result in challenge-vector setting for delay buffer and expected responses, for each pair of clock sinks. This tuning technique will be discussed in Section 5.

4. DESIGN AUTOMATION FOR CLOCKPUF

We introduce design-automation techniques that implement ClockPUF architecture and minimize its overhead. These techniques were used in our empirical evaluation.

4.1 Selecting sinks for ClockPUF

Given the locations of N clock sinks, we select n sinks for use in ClockPUF such that their locations are approximately equidistant (in the Manhattan metric) from some *center-point*. Thus, the delays from sinks to MUXes along the return paths can be matched (Figure 3).

The inner part of our algorithm selects the sinks for a given center-point, whose location is optimized in the outer part. For a given center, we discard sinks that are too close and

sort the remaining sinks by Manhattan distance. We then sweep the sinks by ascending distance and maintain a range of n sinks, keeping track of the max-min spread in this range. Several ranges with small spread are saved and evaluated based on the angular distribution of the sinks, and we select the one where the sink distribution is closest to uniform, as this decorrelates return paths.

The outer part of the algorithm can sweep through all possible center locations on a fine grid. But this is inefficient, as the desirable centers usually cluster around the arithmetic-average location of all sinks. Therefore, we start by evaluating this location and then generate a Gaussian sample centered at it. We evaluate each point from the sample, find the best center-point, and then repeat this search from the new center-point. The process, illustrated in Figure 5, stops after exhausting the maximal allowed number of steps (or when the improvement drops below a small threshold). In practice, the runtime is quite small, even when the best center-point is found far from the geometric center. The following pseudocode shows details.

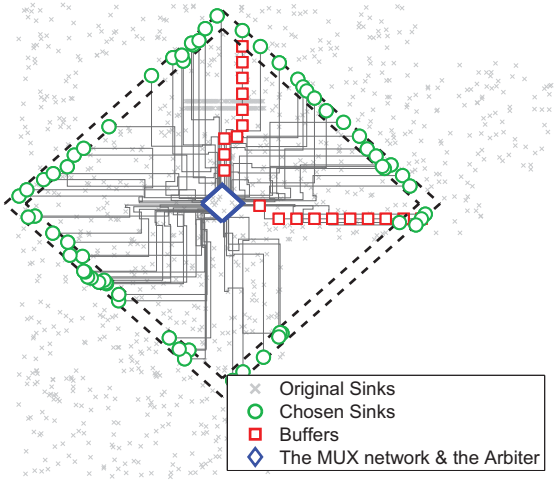

Figure 3: Chosen sinks approximately equidistant from the center with matched-delay return paths routed to decrease delay correlations. Buffers are shown only on two return paths, to avoid clutter.

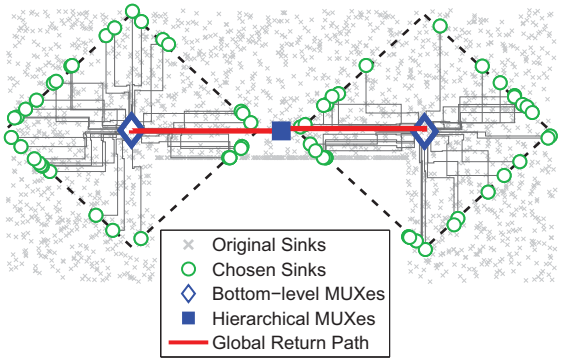

Figure 4: A configuration with two sink groups and routed return paths. Buffers are shown for only two paths to avoid clutter.

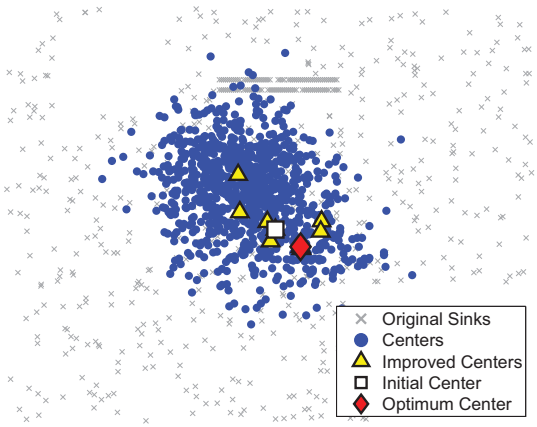

Figure 5: Gaussian search for a center-point.

Input: Sink set $\mathcal{S}$ with N sinks, their locations (x_i, y_i), number n of sinks to choose
Output: Chosen sink set S_c and a center-point C_c
1. $C(x_C,y_C)$ = GeomCenter($\mathcal{S}$);
2. S_C = SelectEquidistantSinks(C, n);
3. v_C = CalculateVariance(C, S_C);
4. $S_c = S_C$; $v_c = v_C$;
5. $C_c(x_c,y_c) = C(x_C,y_C)$;
6. **repeat** M **times** {
7. $C(x_{tmp},y_{tmp})$ = GaussianRandMove(C);
8. S_{tmp}= SelectEquidistantSinks(C, n);
9. v_{tmp} = CalculateVariance(C, S_{tmp});
10. **if** $(v_{tmp} < v_c)$ {
11. $S_c = S_{tmp}$; $v_c = v_{tmp}$;
12. $C_c(x_c,y_c) = C(x_{tmp},y_{tmp})$;
13. }
14.}

4.2 Routing return paths

Our baseline algorithm routes sinks toward the center. It first instantiates L-shape (single-bend) routes and optimizes their orientation to minimize overlap. With more than several sinks, L-shaped routing leads to excessive congestion along the x- and y-coordinates of the center. Therefore, we also admit shortest-path Z-shapes (two bends, rather than one). The initial routing with L-shapes identifies regions with most congestion and guides Z-shape routing. The resulting paths are buffered with equally-spaced buffers and evaluated with timing analysis. As the selected sinks are only *approximately equidistant* from the center, shortest-path delays can be slightly different. Thus, we estimate excess distance (slack) on each path and pass it to the router. Simplified pseudocode (w/o congestion mitigation) follows.

Input: Chosen-sink set $S_c[1...n]$ and its center $C_c(X_c,Y_c)$, number of buffers to add to each return path b
Output: Return paths $R[1...n]$, buffer locations $B[1...n]$
1. boundLine[1...4] = defineMuxKeepOutZone(C_c);
2. terminals[1...n] = getRoutTerminals(S_c, boundLine);
3. dist[1...n] = calcManhDistances(S_c, terminals);
4. maxPathLen = findMax(dist);
5. **for** $(i = 1, i \leq n, i++)$ {
6. slack = maxPathLen - dist[i];
7. $R[i]$ = LZrouteWithSlack($S_c[i]$, terminals[i], slack);
8. $B[i]$ = InsertBuffers($R[i]$, b);
9.}

We first define a keep-out zone (KOZ) around the MUX and connect sinks to points on the boundary of KOZ. This creates flexibility in implementing the MUX as a network of smaller MUXes so as to avoid routing congestion.

5. PROCESS VARIATION AND TUNING

We simulated ClockPUF with clock networks from the ISPD 2010 Clock-network Synthesis Contest and an IBM 45nm library, using the contest protocol with 500 Monte-Carlo SPICE runs. The return paths and other components of ClockPUF were simulated using a fuller 45nm library from IBM and 10K Monte-Carlo SPICE runs. Simulating entire ClockPUFs was not practicable. The two sets of results were combined in a spreadsheet. The latter assumption holds because the buffers are far apart and experience much higher variation than interconnects.

A pair of tunable delay buffers at the inputs of the delay arbiter compensates for the intrinsic delay difference between paths (design-time and manufacturing biases). When nominal delays of two return paths are matched, their comparison generates a random bit. Small biases in such bits can be tolerated by combining multiple bits. The best settings of delay buffers are determined empirically based on a sufficiently large lot of chips. By setting delays to median values, one decreases systematic bias in delay differences and increase variational entropy available for PUF bit generation. To further increase entropy, several settings for delay buffers can be used. Using such settings as a challenge-response mechanism repeals attacks based on replaying or analyzing responses to a single setting. Finding good settings of delay buffers and PUF read-out does not require external tools when delay buffers can be programmed via multiplexed input pins.

6. EXPERIMENTAL VALIDATION

Using the infrastructure described in Section 5, we chose 64 sinks from each ISPD 2010 clock network and performed physical layout of ClockPUF, as explained in Section 4. Buffering was performed with minimum-sized inverters from the 45nm IBM cell library. The number of buffers was the same on all paths and minimized delay.

Inter-chip Hamming distances. To identify individual chips by their PUF response, the responses must sufficiently differ. Figure 6 illustrates the distribution of inter-chip Hamming distances over 500 Monte-Carlo SPICE simulations. Average inter-chip Hamming distance is distributed around 50%, thus we can conclude that ClockPUF has a good performance interms of response uniqueness. We also note that the standard deviation drops as the number of comparisons increases. Thus, more CRPs improve chip differentiation.

Reproducibility of response ensures a permanent chip ID regardless of environmental conditions. Small changes can be tolerated through error-correcting codes, as established in previous PUF literature. We studied the probability of bit-flips in PUF response for temperature variation (-20C to 120C) and voltage (+/- 10%) on 4 different clock networks from the ISPD 2010 contest with different chip size, number of sinks, and clock-skew distribution. Figure 6 shows that $< 10\%$ of pairwise comparisons (between 64 sinks) can flip in the worst case. This probability drops with the magnitude of environmental variations.

The impact of return paths on delay distribution. Return paths and their buffers not only deliver tapped clock signals to MUX network and arbiter, but also spread out the skew distribution, increasing the entropy available for PUF-bit generation (Section 3.2). To this end, Figure 6 compares the arrival-time distributions at 64 chosen sinks and at the ends of their return paths. In both cases, time is measured relative to the average of 64 datapoints. The return paths spread out the distribution of arrival times.

Overhead of ClockPUF is reported in Table 1 over four different clock networks from the ISPD 2010 contest. Area overhead, estimated based on the size of IBM 45nm standard cells, is $< 0.1\%$ of total chip area, thus negligible. Power overhead can be estimated relative to total power consumption of the original clock network. During idle operation, the return paths are turned off (gated) by AND gates with a global *enable* signal (Figure 1). Given that PUF read-out (with error-correction) requires several hundred cycles at most, dynamic-power overhead is limited by load capacitance of AND gates.[2]

Further extensions to minimize overhead are possible with hierarchical return networks that combine multiple sets of tap-out points equidistant from their centers (Figure 4) through hierarchical 2-to-1 MUXes. This can significantly reduce the total length of the return network, while preserving sufficient entropy for PUF read-out.

Comparison with RO-PUF in Table 2 evaluates ClockPUF and 3 RO-PUF configurations. We performed 500 Monte-Carlo SPICE simulations (with 1000-bit keys) using an IBM 45nm SOI technology. All PUF configurations show inter-Hamming distances $\sim$50% as desired.

Standard deviation of inter-Hamming distances drops as the number of inverters in RO-PUF increases. But the variance of ClockPUF exceeds that of RO-PUF because the intrinsic delay differences in the clock tree widen the delay distribution. Though ClockPUF negates this trend with return-path inverters, if delay of one specific clock-tree path is greater, then the inter-Hamming distance of that specific path from other paths is likely to be biased. This bias in delay distribution can be avoided using a delay buffer near the comparator. ClockPUFs offer 2x greater reproducibility than RO-PUFs thanks to a wider delay distribution.

[2]Power overhead of ClockPUF readout is estimated to be far below 20% relative its host clock domain. It can be reduced by optimizations, but we do not presently see sufficient need to develop them.

Table 1: Overhead of ClockPUF.

Bench No.	Area (mm^2)	No. of Sinks	Overhead (%)		
			Area	Power	
				Measure	Idle
01	64	1107	0.002	20.71	0.16
03	1.5	1200	0.075	25.02	0.57
06	1.7	981	0.066	22.64	0.70
08	3.7	1134	0.030	18.82	0.62

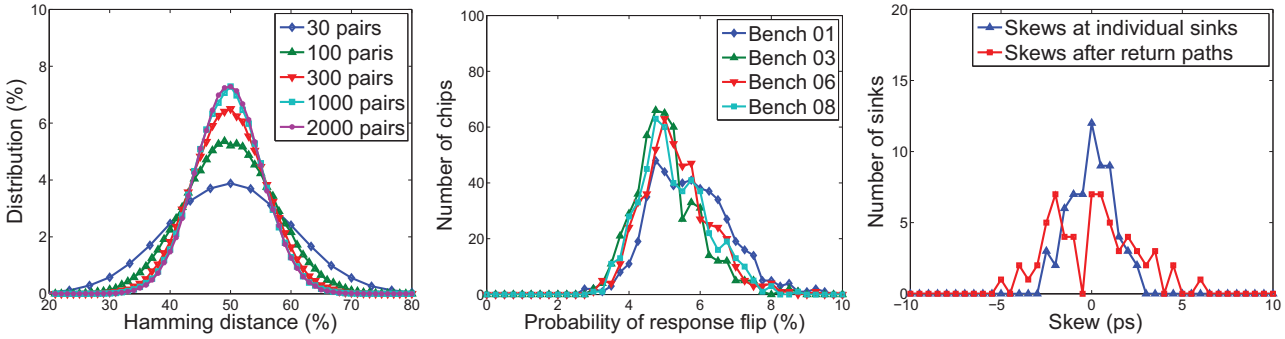

Figure 6: (left) Inter-chip Hamming distance distribution with multiple pairwise comparisons, (center) probability of bit-flips in PUF responses for four clock networks under environmental conditions ranging from (1.1V, -20C) to (0.9V, 120C), (right) Skew distributions at 64 chosen sinks and after return paths.

7. CONCLUSIONS AND FUTURE WORK

We have described a novel unclonable chip ID based on an existing clock network. A comprehensive empirical evaluation is performed using Monte-Carlo SPICE runs on clock networks from the ISPD 2010 contest and IBM 45nm library cells. Uniqueness and reproducibility of ClockPUF are established, and its overhead is shown to be negligible.

When viewing our ClockPUFs as weak PUFs [19], a valid comparison is with RO-PUFs. In addition to the empirical comparison we presented, ClockPUFs are more tamper-resistant because of integration within the clock network. Viewing ClockPUFs as strong PUFs, a valid comparison is with DelayPUFs, which have a larger overhead and are less tamper-resistant. Our empirical data show that ClockPUFs are competitive in their stability.

8. REFERENCES

[1] Defense Science Board (DSB) study on High Performance Microchip Supply, 2005.

[2] Defense Industrial Base Assessment: Counterfeit Electronics study by U.S. Dept. Of Commerce Bureau Of Industry & Security Office Of Tech. Evaluation, 2010.

[3] F. Armknecht, R. Maes, A.-R. Sadeghi, F.-X. Standaert, and C. Wachsmann. A formalization of security features of physical functions. In *IEEE Symp.Sec'ty & Privacy*, pages 397–412, 2011.

[4] C. Bösch, J. Guajardo, A. Sadeghi, J. Shokrollahi, and P. Tuyls. Efficient helper data key extractor on FPGAs. In *Crypto Hardware & Emb Sys (CHES)*, pages 181–197, 2008.

[5] B. Gassend, D. Clarke, M. van Dijk, and S. Devadas. Controlled physical random functions. In *ACSAC*, pages 149–160, 2002.

[6] J. Guajardo, S.S. Kumar, G. Schrijen, and P. Tuyls. FPGA intrinsic PUFs and their use for IP protection. In *Crypto Hardware & Emb Sys (CHES)*, pages 63–80, 2007.

Table 2: Our ClockPUF configuration compared to three configurations of RO-PUFs.

Configurations	Inter-Hamming dist.		Probability of
	mean	std. dev.	response flip
C 10inv	50.3%	7.3%	5.07%
R 4inv+nand	49.21%	5.62%	10.17%
R 10inv+nand	50.1%	5.28%	9.82%
R 30inv+nand	50.21%	4.93%	9.51%

[7] R. Helinski, D. Acharyya, and J. Plusquellic. A physical unclonable function defined using power distribution system equivalent resistance variations. In *DAC*, pages 676–681, 2009.

[8] R. Helinski, D. Acharyya, and J. Plusquellic. Quality metric evaluation of a physical unclonable function derived from an IC's power distribution system. In *DAC*, pages 240–243, 2010.

[9] D.E. Holcomb, W. Burleson, and K. Fu. Power-up SRAM state as an identifying fingerprint and source of true random numbers. *IEEE Trans. Comp.*, 58(9):1198–1210, September 2009.

[10] A. B. Kahng, J. Lienig, I. L. Markov, and J. Hu. *VLSI Physical Design - From Graph Partitioning to Timing Closure*. Springer, 2011.

[11] J.W. Lee, A. Lim, B. Gassend, G.E. Suh, M. van Dijk, and S. Devadas. A technique to build a secret key in integrated circuits for identification and authentication applications. In *Symp. VLSI*, pages 176–179, 2004.

[12] K. Lofstrom, W.R. Daasch, and D. Taylor. IC identification circuits using device mismatch. In *ISSCC*, pages 372–373, 2000.

[13] R. Maes, P. Tuyls, and I. Verbauwhede. Low-overhead implementation of a soft decision helper data algorithm for SRAM PUFs. In *Crypto Hardware & Emb Sys (CHES)*, pages 332–347, 2009.

[14] R. Maes and I. Verbauwhede. *Physically Unclonable Functions: a Study on the State of the Art and Future Research Directions*. Springer, 2010.

[15] A. Maiti and P. Schaumont. Improved ring oscillator PUF: An FPGA-friendly secure primitive. *J. Cryptology*, 24(2):375–397, 2011.

[16] M. Majzoobi, F. Koushanfar, and M. Potkonjak. Testing techniques for hardware security. In *ITC*, pages 1–10, 2008.

[17] M. Majzoobi, F. Koushanfar, and M. Potkonjak. Techniques for design and implementation of secure reconfigurable pufs. *ACM TRETS*, 2(1), 2009.

[18] D. McGrath. Ihs: Counterfeit parts represent $169b annual risk. 4/4/2012.

[19] U. Ruhrmair, S. Devadas, and F. Koushanfar. *Security Based on Phys. Unclonability and Disorder*. Springer, 2011.

[20] G. Suh and S. Devadas. Physical unclonable functions for device authentication & secret key generation. In *DAC*, pages 9–14, 2007.

[21] M.-D. Yu and S. Devadas. Secure and robust error correction for physical unclonable functions. *IEEE Desing & Test*, 27:48–65, 2010.