Know Time to Die – Integrity Checking for Zero Trust Chiplet-based Systems Using Between-Die Delay PUFs

Aleksa Deric and Daniel Holcomb

University of Massachusetts, Amherst MA, USA
aderic@umass.edu,dholcomb@umass.edu

Abstract. Industry trends are moving toward increasing use of chiplets as a replacement for monolithic fabrication in many modern chips. Each chiplet is a separately-produced silicon die, and a system-on-chip (SoC) is created by packaging the chiplets together on a silicon interposer or bridge. Chiplets enable IP reuse, heterogeneous integration, and better ability to leverage cost-appropriate process nodes. Yet, creating systems from separately produced components also brings security risks to consider, such as the possibility of die swapping, or susceptibility to interposer probing or tampering. In a zero-trust security posture, a chiplet should not blindly assume it is operating in a friendly environment.

In this paper we propose a delay-based PUF for chiplets to verify system integrity. Our technique allows a single chiplet to initiate a protocol with its neighbors to measure unique variations in the propagation delays of incoming signals as part of an integrity check. We prototype our design on Xilinx Ultrascale+ FPGAs, which are constructed as multi-die systems on a silicon interposer, and which also emulate the general features of other industrial chiplet interfaces. We perform experiments on, and compare data from, dozens of Ultrascale+ FPGAs by making use of Amazon's Elastic Compute Cloud (EC2) F1 instances as a testing platform. The PUF cells are shown to reject clock and temperature variation as common mode, and each cell produces approximately 5 ps of unique delay variation. For a design with 144 PUF cells, we measure the mean within-class and between-class distances to be 68.3 ps and 847.7 ps, respectively. The smallest between-class distance of 686.0 ps exceeds the largest within-class distance of 124.0 ps by more than 5× under nominal conditions, and the PUF is shown to be resilient to environmental changes. Our findings indicate the PUF can be used for authentication, and is potentially sensitive enough to detect picosecond-scale timing changes due to tampering.

Keywords: Chiplet · PUF · FPGA · Advanced Packaging · Zero Trust Architecture

1 Introduction

Chip manufacturers have traditionally relied on Moore's Law to create increasingly complex integrated systems on each single silicon die. However, more recently with increasing design complexity, smaller node sizes and a shift to Systems-on-Chip (SoCs) architectures, large monolithic designs are becoming impractical as the limits of technology are tested and yields decrease. An increasingly popular solution to these challenges is partitioning a large design into multiple smaller components known as chiplets. Chiplets are smaller dies that are separately fabricated with standardized interfaces, which are then integrated into a larger system by assembly on a passive silicon interposer or bridge that connects the chiplets to each other.

Chiplets can be viewed as a middle ground between monolithic single die systems-on-chip and PCB-based systems, inheriting salient features of each. Like connections between functional units on monolithic SoCs, the connections between chiplets are dense

and unterminated; the connections provide high bandwidth, low-latency, and sub-pJ energy-per-bit. On the other hand, like connections between chips in PCB-based systems, chiplets allow for heterogeneity of process nodes and vendor choices, and can be reused in different combinations across product lines.

Chiplet-based designs are not without downsides. Compared to monolithic chips the logistics of manufacturing, testing and assembling multiple smaller dies means increased packaging cost, and the die-to-die connections incur latency and power penalties. Moreover, due to the latency penalty, it can be challenging to split large elements such as coherent memories across dies without performance loss. Last but not least, splitting a design across multiple chiplets opens new attack vectors and potentially exposes the system to a greater risk of tampering. Likewise, chiplets also have some downsides when viewed as an alternative to PCB-based system, in that they cannot easily be serviced or changed after assembly, and that their increased compute density generates more heat within a small area that must be dissipated. Although there will continue to be some applications better suited to a monolithic die or PCB-based system, industry is trending toward increasing use of chiplets in the coming years.

The term "zero trust" describes a security paradigm in which "there is no implicit trust granted to assets or user accounts based solely on their physical or network location" [RBMC20]. For reasons described more fully in Section 2.3, hardware disaggregation creates a scenario in which a zero trust posture between the chiplets of a system may be warranted. A chiplet should not blindly assume that signals it receives and transmits are being communicated exclusively to honest actors that share a common goal of a secure system. Yet even if adopting a zero trust posture, it is not obvious what actions can be taken in support of it. To establish a measure of trust in the integrity of other chiplets within a system, we propose a scheme in which a chiplet can measure the delays of signals arriving from its neighbors, and check that information against its expectations to decide whether to trust the neighbor or not. The specific contributions of our work are as follows:

- We present a novel design that can extract robust delay signatures from inter-die connections through an interposer in chiplet-based systems, with common-mode rejection of undesirable clock and environmental variations.
- We prototype and validate the design on Xilinx VU9P FPGAs locally and across a population of chips on Amazon's EC2 F1 instances to show ability to extract sufficient entropy as a PUF, while being able to measure picosecond-scale delay changes that might arise from tampering.
- We perform extensive analysis across a variety of design manipulations to identify the specific sources of entropy in the system.

2 Background and Related Work

2.1 Chiplets and Advanced Packaging

A survey of chiplets is found in [LHY+20]. Details given below are for several popular chiplet interfaces currently used by large players in the industry. The design that we propose in this paper is demonstrated on Xilinx chiplets, but it uses generic features of source-synchronous clocking that are shared by all the currently leading chiplet connection technologies. In source-synchronous clocking, the transmitting device forwards its transmit clock along with the data wires to the receiving device. The receiver uses a controllable delay line followed by H-tree routing to de-skew the received clock and ensure that the arriving data signals are sampled near the center of each bit for reliable communication.

Intel AIB and EMIB Intel offers the advanced interface bus (AIB) as the interface for chiplet connections, and Embedded Multi-Die Interconnect Bridge (EMIB) as the

packaging technology to realize the AIB connections between adjacent edges of two chiplets. An array of microbumps along the edge of each chiplet connect it to the EMIB; the microbumps are arrayed at $55\mu m$ pitch [LMS+20], with plans to scale to $35\mu m$ pitch, providing greater density than the typical $100\mu m$ pitch in a flip-chip package. The density of connections through EMIB in Intel's Stratix 10 FPGA is currently 256 wires per mm of shoreline, and it can scale up to 1024 per mm of shoreline. The EMIB contains multiple layers of metal connections, typically 4 layers with all wires shielded [LMS+20], to carry the signals across the shoreline between neighboring chiplets. Clock forwarding and delay tuning on the receiver side is used for communication. Each wire carries 1 Gbps when used in Single Data Rate (SDR) configuration [Keh17] [MQV+19]. Currently Intel uses EMIB/AIB to integrate an FPGA main die in a leading CMOS process, with various I/O dies from different process nodes; only the largest Stratix 10 device currently splits the FPGA logic itself across multiple chiplets.

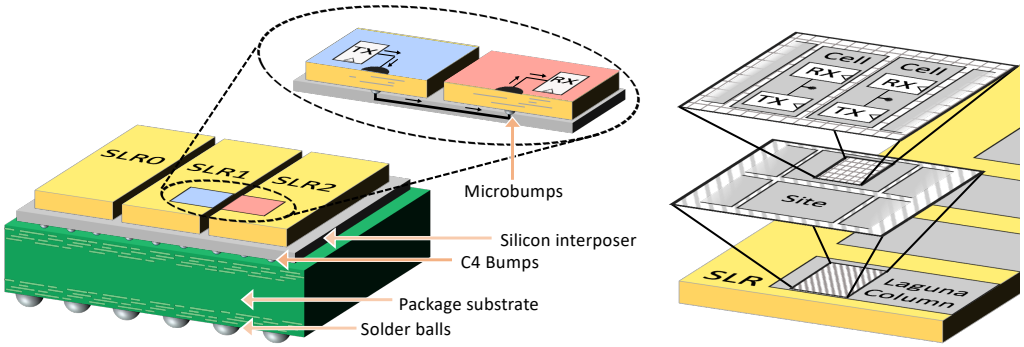
TSMC LIPINCON interface and CoWoS packaging TSMC similarly provides their own technology for connections between chiplets. Their interface is denoted as LIPINCON (standing for low-voltage in-package interconnect) and the packaging technology is denoted as CoWoS (chip on wafer on substrate). Unlike Intel's EMIB which is a bridge underneath only the chiplet edges, CoWoS is a full silicon interposer layer underneath the chiplets. In 2019 TSMC and ARM presented a 7nm proof-of-concept ARM based octo-core system-in-package processor. The processor is made up of two identical chiplets that run at 4.0 GHz and communicate using LIPINCON, achieving nearly 320 GB/s of aggregate bandwidth [LHT+19].

Xilinx Ultrascale+ Driven by the ever increasing demand for higher capacity FPGAs Xilinx developed a multi-die technology (see Fig. 1a) termed Stacked Silicon Interconnect (SSI). SSI technology combines multiple FPGA dies into a single device using microbump connections to a shared silicon interposer [Sab12]. The FPGA chiplet dies are referred to as Super Logic Regions (SLRs), and the passive silicon interposer is fabricated in a 65 nm process. Four layers of metalization realize tens of thousands of low latency connections called Super Long Lines (SLLs) that connect adjacent edges of neighboring SLRs; state-of-the-art Virtex UltraScale+ devices have as many as 17,280 SLLs. Through-silicon vias (TSV) through the interposer connect down to the package substrate. Like similar interfaces, Xilinx SSI technology also allows for the integration of heterogeneous components such as high speed IO transceivers. Because FPGAs provide users with control over clocking, and their reconfigurable logic enables transmitting arbitrary known patterns on demand across chiplet boundaries, we make use of Xilinx multi-die FPGAs as a prototyping platform in this work. Analogous capabilities exist in AIB and LIPINCON compliant chiplets, but they are not exposed to user control which prevents them from being similarly used for prototyping without producing a cost prohibitive multi-chiplet system.

2.2 Related Work

Chiplet and System-in-Package technologies are gaining traction and making their way from size-constrained mobile systems into mainstream computation over the past few years. The variety of manufacturer-specific interfaces listed in the prior subsection are one consequence of that evolution.

Standardization of interface protocols to allow for easy plug-and-play compatibility between chiplets is required to realize the full potential of the technology [MKN+21]. A set of standardized models including power, IO, behavioral and test are needed in order to facilitate this transition. Notably, the same work that advocates for standardization [MKN+21] also cites an emerging need for operational security of chiplets and the resulting packaged devices; traceability and authentication are listed among top priorities and, although no concrete solution is proposed, physically unclonable functions (PUFs) are mentioned as a



(a) Model and view of SLL circuit (b) Xilinx Hierarchy

Figure 1: Xilinx multi-chiplet system. (a) shows the physical representation of the chiplet-based system, with an expanded view of the path of a single SLL through the interposer, the delay of which is used as an entropy source. (b) shows the hierarchy of the Xilinx architectural primitives for communicating between SLRs that are used in this work.

possible remedy. Other works focus specifically on the security challenges and opportunities of chiplet designs with respect to hardware Trojans and supply chain [SBK20], and on side channel attacks, hardware Trojans, secure manufacturing and IP piracy [GLS+16] Yet, these works do not propose solutions for authentication.

To support security of communication between chiplets, several works [CMK+21, NAP+20] propose that an active interposer manufactured by a trusted foundry can be used as a secure-by-construction root of trust (RoT). The active interposer contains logic to continuously monitor and police traffic between chiplets to ensure authenticity and system integrity. Although promising for some applications, the active interposer brings yield challenges as it must be large enough to interface to all chiplets. Another 2021 paper [XMYW21] proposes that circuits connected to FPGA outputs may be able to detect impedance changes caused by the presence of a physical probe on the output wire, and the authors note that a similar technique could be considered in the future for authentication, but no experiments in this direction are performed.

Commercially-available systems that address multi-die system security largely rely on cryptographic protocols to create secure channels between dies. One example is CEVA Fortrix [CEV22], which features secure message passing, authentication, attestation, and firmware download while protecting against firmware copying/tampering, chiplet counterfeiting and chiplet disabling/modifying. The use of already-trusted cryptographic protocols is attractive in that approach, but the substantial area cost of the computations performed likely limits it to the subset of applications that can justify paying a high cost for security. Nonetheless, the emergence of these competing solutions supports the high interest in finding ways to establish system level security from chiplets, and our work represents a novel and low-cost technology to support this objective based on common features of chiplet interfaces.

2.3 Security in Chiplets

The modularity and interoperability that enable heterogeneous integration with chiplets also has implications for how they can be attacked, and how they must be protected. The chiplet attack surface is unlike that of a traditional single-die SoC in which there is a clear distinction between on-chip and off-chip threats, with a well-defined perimeter separating the two. Oversimplifying a bit, one can consider the world beyond the I/Os of a single-die SoC as untrusted and beyond the control of the vendor, and within the die is the domain

of what can be secured; in this framing physical protection techniques such as PUFs or enclosures are defending against attacks that try to breach the secure domain. On-die circuitry is not assumed secure by default due to a potential malicious foundry or 3rd Party IP, but in this framing an SoC producer can have some confidence that a functional block will have a consistent neighbor across all instances and across its lifetime. If the SoC is secured once, and the perimeter is not breached, it likely will remain so. Chiplets, on the other hand, must operate outside of the castle walls. A chiplet will be re-used across product lines, and may interface to chiplets created by entities that it does not know or trust. As such, it is not feasible to assume honest collaboration toward system security, or that other chiplets will have security-enhancing features such as PUFs.

The critical question, which we try to address in this work, is how can a trusted chiplet support integrity attestation of a system in which it is one component among many without special privilege, and without assuming that other chiplets are honest. Relevant threats in this setting include the possibility that a neighboring chiplet has been changed across time, or that its communications through an interposer are being probed, spoofed, subjected to man-in-the-middle attack, and so on. This is a daunting scenario, and without physical protections it may not be possible to mitigate all threats that are within scope. Yet we argue in this paper that a novel realization of a delay-based PUF can be an important primitive toward verifying system integrity, and that it can be accomplished using capabilities that are already widely available in chiplets.

2.4 Physical Unclonable Functions and Device-Tied Entropy

Physical unclonable functions (PUFs) are circuits that produce outputs using entropy from the manufacturing process variations of each instance. PUF outputs are persistent over time, but can be influenced by noise. The source entropy of a PUF generally grows proportionally to its area, but there are different approaches for generating values from the available entropy, often termed as *strong* PUFs or *weak* PUFs. Strong PUFs, which have a large space of input challenges, are exemplified by the well-known Arbiter PUF that maps inputs to outputs according to delay variations of ASIC [GCVD02] or FPGA [MKD10] cells. Because each input-output pairing leaks information about the source entropy, strong PUFs are subject to modeling attacks, in which an adversary uses a set of known input-output examples to train a model that can eventually predict the PUF output for any input. Techniques like the lockdown PUF [YHD+16] aim to get around this problem by limiting the number of input-output pairings available to the adversary.

Weak PUFs lack a large input space and can therefore be viewed as device-tied constants, subject to noise. Following early chip-ID circuits [LDT00, SHO07], weak PUFs gained prominence in the work of Guajardo et al. [GKST07] that uses entropy from the unique power-up states of uninitialized SRAM on FPGAs, and is commercially available through Intrinsic ID. SRAM values are initialized at power-up on modern Xilinx and Intel FPGAs, although some workarounds [SGB+10, WG14] attempt to circumvent initialization with limited success on specific devices. The initial state of flip-flops [MTV08] are highly biased, and Butterfly PUFs based on cross-coupled latches [KGM+08] can have diminished uniqueness due to asymmetry in the FPGA routing resources between the latches [MMS10]. Distinct from the earlier FPGA implementations of delay-based arbiter and ring oscillator PUFs [SD07], a delay-based weak PUF based on fixed-function FPGA carry chains has also been explored [And10]. Decay-based DRAM PUFs are also viable and were used by Tian et al [TXG+20] to study and map the entire Amazon Web Services (AWS) FPGA infrastructure.

The output of a PUF can be kept secret if it is used as a key in a cryptographic protocol, which requires error correction to be performed with helper data to compensate for noise. Protocols such as fuzzy extractors rely on standard error correcting codes which compromise some entropy. Index-based syndrome (IBS) coding [YD10] is an error

correction scheme for PUFs with real-valued outputs, in which some of the source entropy is traded away to obtain highly reliable outputs. Information can be leaked in IBS if the PUF outputs do not satisfy the assumption of being independent and identically distributed [BWG15]. Other schemes include the equiprobable quantization methods explored for the secure physical enclosures of Immler et al. [ION+18] that make best use of source entropy.

3 Methodology

The solution that we propose for integrity checking calls for one chiplet to measure the propagation delays of signals arriving through the interposer from a neighboring chiplet. We prototype and evaluate our design using a population of Xilinx Virtex Ultrascale+ (VU9P) FPGAs, which use multiple dies connected through an interposer. To stay consistent we use Xilinx terminology when describing the design. The chiplets are referred to as Super Logic Regions (SLRs), and the wires through the interposer as Super Long Lines (SLLs).

3.1 Use Case and Requirements

A PUF implementation such as ours has several security and hardware requirements that must be satisfied. First, for the PUF itself, the physical structure used to derive responses should be difficult to clone but easy to measure. Tampering with the structure should permanently alter its response in subsequent measurements. Inter-die interconnect satisfies these properties. Thus the basis for security of our PUF is that it derives its entropy from process variation of the interposer and chiplets. Depending on the application, persistent memory and hardware primitives for signing, encryption, decryption and verification of measured delays must be available. We envision our system being used similarly to other PUFs with an initial enrollment phase in which the device generates and securely stores the first set of measurements. In the field, measurements would be retaken and compared to the enrolled data to generate a key or a security flag. Depending on the application, the measurements can be checked at boot time to authenticate neighboring dies or at regular intervals to check for probes. We make the assumption that the receiving chiplet is trusted, but that circuitry beyond its die is not. This assumption is justified by inter-die wires being physically larger and more exposed than on-die wires, and by the existence of other techniques that can already address the problem of deriving trust in a single die using PUFs.

As described in Sections 1 and 2 chiplets present a unique security challenge in that they expose new attack surfaces that differ from those in monolithic devices. Our work aims to provide a method through which a die can authenticate its neighbor and verify system integrity. More concretely, we consider unaddressed threats such as tampering and probing attacks against chiplet systems on passive interposers. One representative tampering threat is die-swapping of a transmitting chiplet. To defeat our system an adversary trying to swap a chiplet must *1)* measure the flop-to-flop delays along the transmitter-interposer-receiver paths, and *2)* tune the replacement chip to have the same picosecond-level delays. Critically, the attacker must perform these tasks without in situ delay measurements which are known only to the receiver chiplet. Similarly, physical probing represents a real threat to mitigate within a die, or between dies on a PCB [XMYW21] with circuitry that detects an impedance change on output wires. We do not experimentally validate ability to detect probing, but estimates show it is likely that delay changes induced by probing interposer wires would be detectable. Specification sheets of on inter-chiplet connections generally claim energy numbers around 0.85 pJ/bit with 0.9V supply [Keh17]. Assuming negligible short circuit current during switching this equates to 1pF capacitance on the microbumps and the SLL itself. In comparison, a sophisticated active picoprobe has 0.04pF

(40fF) capacitance [GGB]. An additional 0.04pF capacitance from probing on a 1pF line would cause approximately a 4% delay increase, which exceeds the typical between-class distances of our PUF and should therefore be as detectable as die swapping. There are other important practical considerations with probing as well, such as the shielding around the interposer wires, the density of the wires, and that probing may require different detection criteria that are sensitive to changes in a small number of wires instead of using an aggregate metric of similarity that considers all wires. While the tasks faced by a die-swapping or probing attacker appears daunting, we are cautious to note that it is hard to confidently quantify the capabilities of well-equipped adversaries with state-of-the-art equipment.

3.2 Measuring Propagation Delay with Dynamic Phase Shifting

Our design for measuring propagation delay relies on clock phase adjustment on the receiving chiplet. Clock phase adjustment is common for source-synchronous clocking in chiplets to de-skew a forwarded clock on the receiving die so that the clock will reach the sampling flip-flops a time coinciding with the center of each bit. Our method, however, uses the phase adjustment differently, to measure the propagation delay of arriving signals.

Unlike the conservative notion of delay in timing analysis, we define delay to be exactly the time required for each signal to propagate from the TX flip-flop on one chiplet to the RX flip-flop on another. More precisely, we define delay to be the time difference (skew) between the clocks of the TX and RX flip-flops that cause a transmitted rising transition to be sampled by the receiver as 0 and 1 with equal probability, or in other words as having a failure probability (p_{fail}) of 0.5. When the clock skew is smaller than the wire delay the rising transition is more likely to be incorrectly sampled as a 0 ($p_{fail} > 0.5$), and when skew is larger than the wire delay it is less likely to be sampled as a 0 ($p_{fail} < 0.5$). The phase of the receive clock can only be adjusted in discrete steps, so there will not be one particular step that causes p_{fail} to be exactly 0.5. Instead we measure p_{fail} across range of phase steps and then interpolate to infer the phase that would have caused $p_{fail} = 0.5$. The clock skew, in picoseconds, at this inferred phase shift is considered to be the measured delay of the SLL. The delay of each SLL is measured independently on each chip using this method.

Note that the setup we use for clock sweeping to measure delay is similar to the principle of Time-to-Digital Converters (TDCs) based on tapped delay lines. The samples we take at different clock phases are analogous to the time-shifted samples collected from different delay taps in the TDC. It is not possible to have multiple taps on each SLL in the Xilinx FPGA because each SLL has a single fixed endpoint at the input of the RX flip-flop on the receiving die, which is the reason we use phase sweeping instead of a tapped-delay TDC.

3.3 Design

Following Xilinx's hierarchy (see Fig. 1b), our design is organized in columns, of which the general schematic is given in Fig. 2. Six instances of the design are placed on the FPGA, each with eight Laguna sites on the transmitting and receiving SLRs. Each Laguna site contains six Laguna cells, which are TX-RX flip-flop pairs dedicated to specific SLLs. As depicted in Fig. 1a, the TX flip-flop in one Laguna cell drives the signal through the microbump connections of its own die, across the interposer wire, and through the microbump of the neighbor die where it reaches the corresponding RX flip-flop. Our design includes 48 SLLs per column, and 288 SLLs in total. The 48 SLLs per column represent a small fraction of the 1,440 SLLs available. Having a relatively low utilization reduces congestion and simplifies placement, but in principle much higher utilization is possible to produce a richer delay fingerprint.

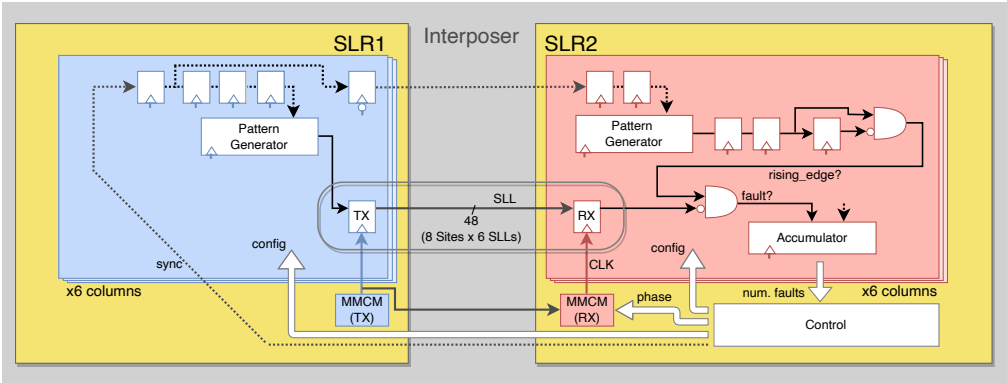


Figure 2: Schematic showing single column of the design. The 48 SLLs are organized in eight Laguna sites, each with six SLLs. The wide control signals are asynchronous.

As shown in Fig. 2, both SLRs contain pattern generators, and the receive side also includes additional fault detection logic. The transmit and receive pattern generators are identical 16-bit LFSRs. Their purpose is to generate a pseudorandom sequence containing rising transitions to transmit across the interposer while checking for timing faults. On the transmitting side the pattern generator is producing the value in each cycle that is actually sent across the interposer. On the receiving side the pattern generator is producing a fault-free copy of the pattern sent by the transmitter, to determine which samples should be checked for rising transition timing faults. During a clock sweep, when the receiver determines that a rising edge was sent, a fault count is incremented if a 0-value was sampled, which indicates a timing fault.

Because the two SLRs are configured as different clock domains, with variable clock phase over a certain range during clock sweeping, some care is required to ensure proper synchronization between the two pattern generators. Pipelining is used on the synchronization signal to relax placement and ensure timing constraints can be met within each SLR. Additionally, a negative edge triggered flip-flop is used to send the synchronization signal across the clock domain boundary on the negative edge of the transmit clock, which makes it insensitive to the range of clock phase differences used during the sweep.

3.4 Xilinx and AWS Implementation

We implement the design on Virtex UltraScale+ VU9P FPGAs using Xilinx Vivado IDE. A floorplan of the implemented design is given in Fig. 3a. The VU9P FPGA contains 3 vertically arranged SLRs with 17,280 SLLs going across each boundary. We implemented the design locally for testing on Xilinx's VCU118 development board, which features part number xcvu9p-flga2104-2L-e. We use AWS EC2 to scale up the experiment for the sake of measuring uniqueness on a larger population. AWS EC2 F1 instances feature the same VU9P FPGA (part number xcvu9p-flgb2104-2-i) found in the VCU118 kit. F1 instances contain additional shell logic to handle communication. The shell logic is confined to the right six columns of the FPGA, and is visible in Fig. 3a where it is shown in orange and blue on the right half of the chip.

To avoid Amazon's logic, the six leftmost Laguna columns are used in our design, which are denoted in Vivado as x4, x12, x21, x30, x40, and x49. Within each of those columns, we use every sixth Laguna site; in the default configuration, the sites which are transmitting across the SLLs are on SLR1 and are denoted as y342, y336, y330, y324, y318, y312, y306, and y300. By default, one pattern generator and one accumulator are instantiated per site, and the delays of the six SLLs in the site are measured sequentially.

Only one SLL per column is active at a time and using the mentioned resources.

Xilinx's Mixed-Mode Clock Manager (MMCM) macro is used for clock generation as it allows for phase shifting of clocks at runtime. The phase of the receive clock is shifted, relative to the forwarded clock from the transmitter, in increments of 14.286 picoseconds.

3.5 Porting Effort

Currently Xilinx is the only vendor that allows unrestricted access to all of its interface and clock primitives and as such is our experimental platform of choice. We however foresee that our design could be ported with minimal effort to other manufacturers' interfaces. Both Intel's AIB [Keh17] and Open Compute Project's (OCP) Bunch of Wires (BoW) [ACF+20] interface incorporate controllable delay lines on the receive side in order to deskew the forwarded clock. Given these capabilities our method could be adapted to them by modifying its control logic to interface with the existing delay primitives of each manufacturer.

4 Analysis of SLL Delays

Our proposed PUF, which is covered in Sec. 5, is a differential design that uses entropy from delay variations in SLLs. To be used as part of a PUF, the SLL delay variation across chip instances should exceed the delay variation across trials on a given instance. Before getting to the PUF, we first analyze in this section the delays of the SLLs themselves. The analysis performed in this section is intentionally anecdotal; its purpose is to give readers an intuitive understanding of the characteristics of SLL delays, as background for the proper analysis that follows in Sec. 5.

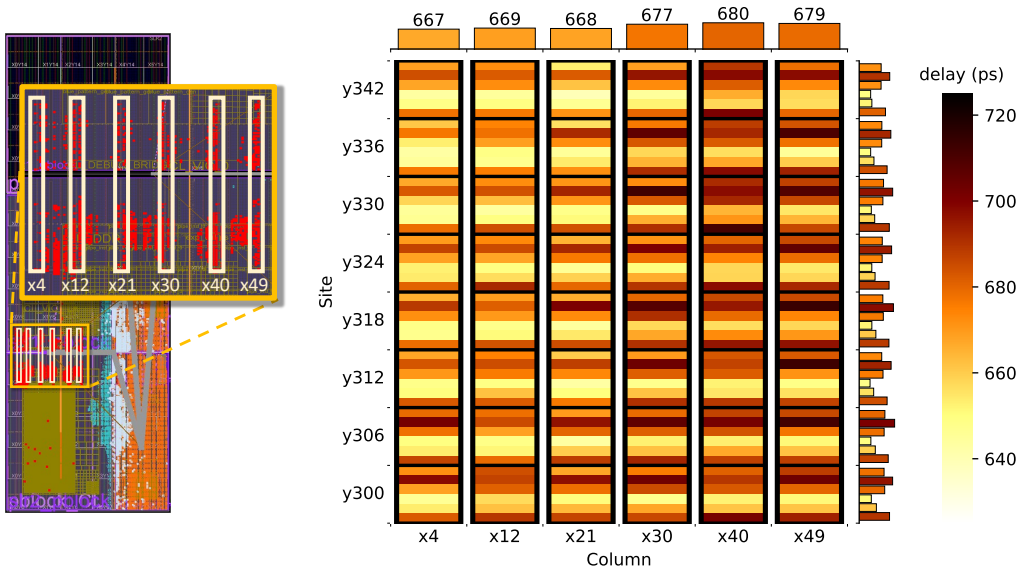
The heatmap in Fig. 3b shows the measured delay values of the 288 SLLs, from a single characterization trial on a single chip. The mean and standard deviation across the 288 SLLs are 673.4 ps and 17.4 ps respectively. The bar graph across the top of Fig. 3b shows the average delay of the 48 SLLs within each column. The bar graph along the right side shows the average delay for each of the 48 SLLs, averaged across all six columns.

4.1 Delay Differences Across Trials and Instances

Fig. 4a shows the mean delay of the 288 SLLs across 20 instances, which can be viewed as a simple representation of the typical delay for each SLL. Fig. 4b shows the delay difference between the trial in Fig. 3b and the typical delays from Fig. 4a; across the 288 SLLs the differences have a mean and standard deviation of 14.9 ps and 6.6 ps respectively. It is apparent from the figure that the measured SLL delays in the trial are generally higher than typical, but that the difference is not uniform, which implies some amount of random variation. Fig. 4c is the difference between the example trial of Fig. 3b and the average delays across 100 trials of that same chip instance; the mean and standard deviation are 0.51 ps and 0.52 ps respectively, which gives an indication that the impact of noise is relatively low.

4.2 Distinguishing Useful Variation from Bias

In the typical delays of Fig. 4a, one can see patterns that suggest skew or bias associated with the column, and with the cell within each site. The consistently positive delay differences in Fig. 4b also indicate that there is a bias associated with the overall speed of a given chip. One can reasonably expect the biases to be additive, meaning that the expected delay of an SLL can be predicted as the sum of its biases. To understand the significance of these factors, we fit the experimental data to a model [PVG+11] that



(a) Floorplan

(b) Measured delays of the 288 SLLs in one trial

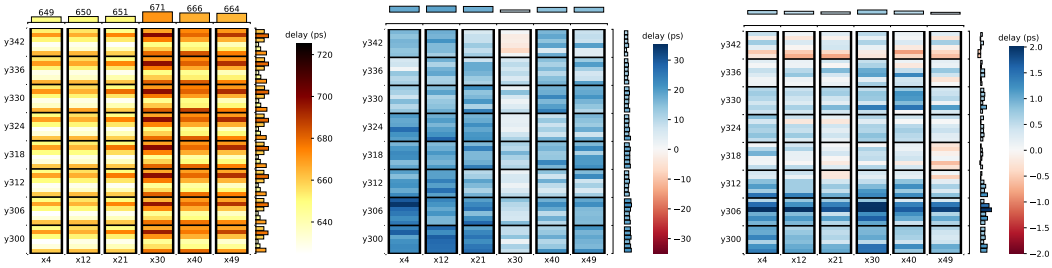
Figure 3: Floorplan shows the six circuit columns spanning SLR1 to SLR2 (a). Heatmap in (b) shows the measured delay of all 288 SLLs from a single trial, organized by column and site. The bars above show the average delay within each column, and at right show average delay of the 48 SLLs across all columns.

describes delay of each SLL as additive with respect to its chip, column, site, and cell. The model is of the form $Xw = y$, where w represents the unknown bias associated with each feature. The model is fit to a dataset in which y is a column vector with 5,760 SLL delays, corresponding to the 288 SLLs on each of 20 FPGA instances; the delay used for each SLL is the mean across 100 trials to reduce the impact of noise. Matrix X is 5760×40, where the 40 features are $(x_0, \ldots, x_{19})$ to denote the bias of the FPGA chip instance on which the SLL delay is measured, $(x_{20}, \ldots, x_{25})$ to denote the bias associated with the column of the SLL, $(x_{26}, \ldots, x_{33})$ to denote the bias associated with the site of the SLL within the column, and $(x_{34}, \ldots, x_{39})$ to denote the bias associated with the cell of the SLL within its site. Each of the 5,760 SLL delay measurements is associated to one chip, one column, one site, and one cell, and therefore every row of X has four 1-values and 36 0-values.

Ridge regression finds weights w that minimize Eq. 1, which are plotted in Fig. 5a. The weights associated with the column and cell indexes of the SLLs match well to the observed data across the top histogram and right histogram of Fig. 4a. The delay does not appear to vary much across sites, and a large part of the overall SLL delay is attributed to the bias of the FPGA instance as a whole. Importantly, there is a significant component of delay that is not explainable by the model, which is consistent with being random variation. Fig. 5b shows a histogram of the 5,760 residual delays, which are the differences between the measured SLL delay and the SLL delay predicted by the model. The standard deviation of the residual delay is 7.309 ps. Fig. 6 shows the same 5,760 residual delays, but now plotted according to the SLL position on each chip instead of being summarized in a histogram.

$$\|Xw - y\|_2^2 + \alpha\|w\|_2^2 \tag{1}$$

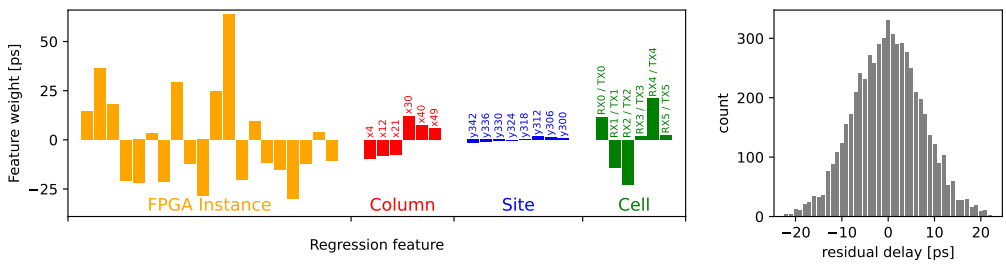
Note that the model does not explicitly account for bias that might be caused by



(a) all-chip mean (b) trial vs all-chip mean (c) trial vs own-chip mean

Figure 4: Measured SLL delays collected from EC2 F1 instances. (a) shows mean delays of each SLL across 20 chips. (b) and (c) show how much the delays from Fig. 3b deviate from the values in (a), and from the mean delays of the same chip that produced Fig. 3b. Note that (b) and (c) have different scales.

differences in routing through the interposer or differences in routing on the SLR to reach the appropriate location of the arrayed microbump that connects the SLR to the interposer. The good fit of the model, and the lack of any consistent patterns in Fig. 6 imply that there is not a significant bias aside from what is captured in the model. This indicates either that the unaccounted-for routing is uniform across all connections, or that it has an asymmetry that mirrors how the SLLs are organized into columns or cells. In the latter case, part of the column bias or cell bias discovered by the model is actually accounting for the asymmetry.



(a) coefficients of fitted model (b) residual delays

Figure 5: Plot shows regression model coefficients and residual delays. The residual delays show that the model cannot explain all of the variation between the SLL delays. The standard deviation of the residuals is 7.309 ps.

5 Using as a PUF

The intended use case for our PUF is a protocol by which a trusted chiplet can obtain a physical signature to check the integrity of its system and the authenticity of its neighboring chiplet. The trusted chiplet (shaded red in Fig. 2) initiates the protocol by prompting the untrusted chiplet (blue in Fig. 2) to send back a known pattern through the interposer. The pattern itself is non-secret and known to both chiplets. While the pattern is being transmitted, the receiver performs clock sweeping as described in Section 3.2 to measure the PUF and extract a delay fingerprint that is caused by the physical variation of the two chiplets and the shared interposer.

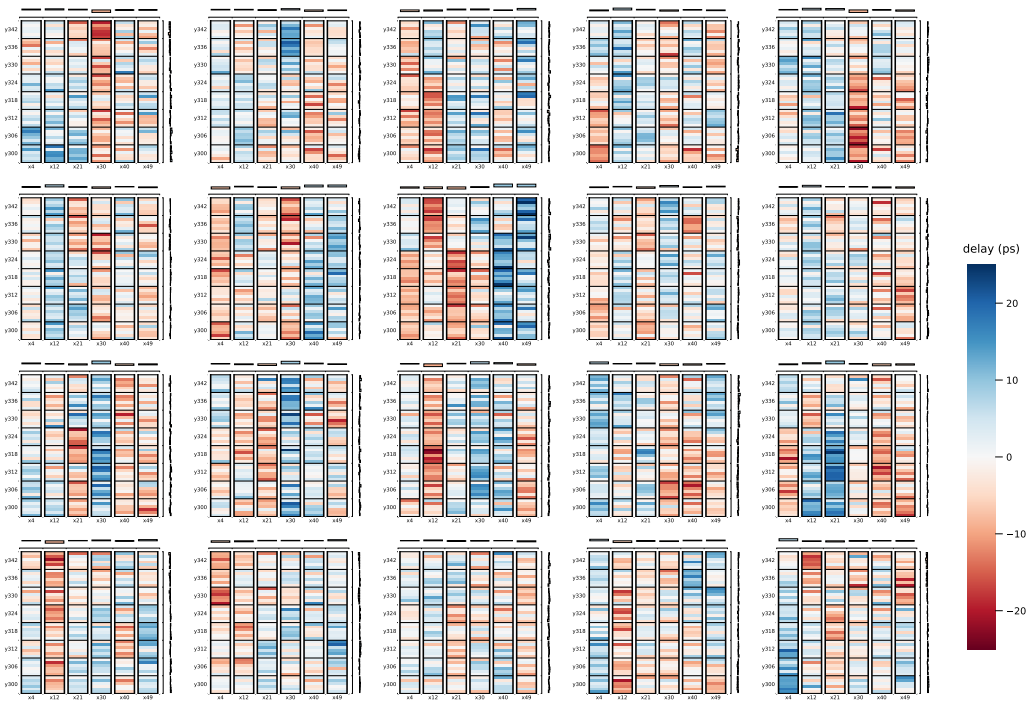


Figure 6: Across 20 chip instances, the difference between measured delays on a chip and the delay from the regression model; in other words, the residual of the model. The histogram in Fig. 5b summarizes the same delays.

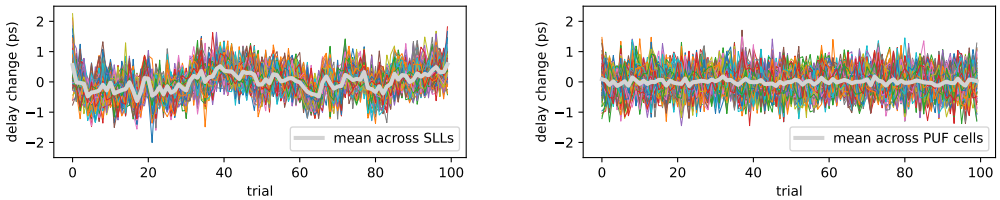
5.1 PUF-Based Delay Fingerprints

The circuit that generates the fingerprint is a type of weak PUF. As is typical of weak PUFs, it comprises repeated instantiations of a unit cell that each operate independently of the others. Our PUF cell is not based on measuring delay of a single SLL; instead it is a differential delay measurement.

5.1.1 Motivating Principle of Differential PUF Cell

Sec. 4 analyzed the delays of individual SLLs to give the technical background for our PUF design; we henceforth refer to those measurements as single-ended delay measurements. We now explain why single-ended measurements are unsuitable as the basis for a PUF, and we then present our differential PUF cell. The major limitations of single-ended delay are as follows:

1. In single-ended delay measurement, delay variation in the clock network aliases to delay variation in the SLL itself. Aliasing can therefore cause the uniqueness of the SLL delay to be overestimated. In fact, aliasing is observable in Fig. 6; in that figure the difference between the measured delays and the delays predicted by regression show spatial patterns coinciding with clock distribution. The six SLLs in each site tend to be faster or slower as a unit, which would occur if the delay variation is actually coming from the clock node, because the six SLLs in the site all come from TX flip-flops connected to the same clock tree node, and the corresponding six RX flip-flops similarly all connect to the same clock-tree node. Additionally, columns



(a) Single-ended delay (b) PUF output (differential delay)

Figure 7: Delay drift across 100 trials on a single chip instance, using both single-ended delays and the PUF outputs. The differential nature of the PUF mitigates drift caused by factors that are common-mode to its two SLLs.

tend to be faster or slower as a unit, which might be explained by delay variation at a point further up the clock hierarchy.

2. Single ended delay is based on comparing an SLL to a clock, which adds unwanted noise because the SLL and the clock are dissimilar. Specifically, the clock path includes the phase compensation circuit and it is carried on a clock tree with the goal of balancing skew at leaf nodes, unlike the SLL which is a direct and dedicated path straight between two flip-flops with the goal of minimizing delay. Because of their dissimilarity, they will respond differently to environmental fluctuations.

5.1.2 PUF Cell

Our PUF unit cell is given in Fig. 8. The cell comprises two SLLs and the TX and RX flip-flops of the Laguna cells on both ends of each SLL. We pair the six SLLs of each Laguna site into three PUF cells using the arbitrary pairings (0,1), (2,3), and (4,5). Because each Laguna site has a single input for TX clock and for RX clock, the two SLLs of each PUF cell within the site will be commonly impacted by noise and variation associated with clock distribution (shown in black in Fig 8), which reduces the impact of clock on the PUF response. To obtain the response of each PUF cell, the single-ended delay of each SLL is measured by clock sweeping as before, and the output of the PUF cell is simply the difference between the two measured delays.

Fig. 7 shows the stability of the single-ended delays (Fig. 7a) and the stability of PUF outputs (Fig. 7b) across 100 trials. In each case, the plotted values are the difference between a measurement (SLL delay or PUF output) in one trial and the average of that same measurement across all 100 trials. The single-ended delays (Fig. 7a) tend to increase and decrease together, presumably due to environmental fluctuations. The PUF outputs (Fig. 7b) are stable and do not drift significantly because common changes to the delay of the two SLLs in the cell will cancel out, which illustrates the benefit of using the differential measurement.

5.2 Uniqueness and Reliability

The distribution of between-class and within-class distances of the PUF outputs are shown in Fig. 9. Each distribution is obtained by making 10,000 random comparisons using a dataset of 100 trials from 20 chips. Value $D_{t,s}$ represents the output (differential delay) in trial t of PUF cell at location s. When comparing outputs from individual PUF cells (Fig. 9a), the mean distance between trials from the same chip (within-class distance) is 0.472 ps, and between trials from different chips (between-class distance) is 5.922 ps. When comparing trials of entire chips, using the total distance between the 144 PUF cells on each chip as the metric, the mean within-class distance is 68.331 ps, and the mean between-class

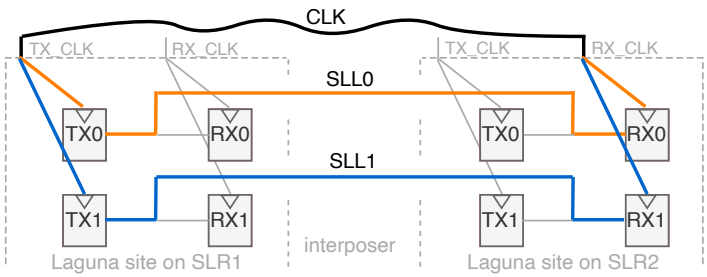
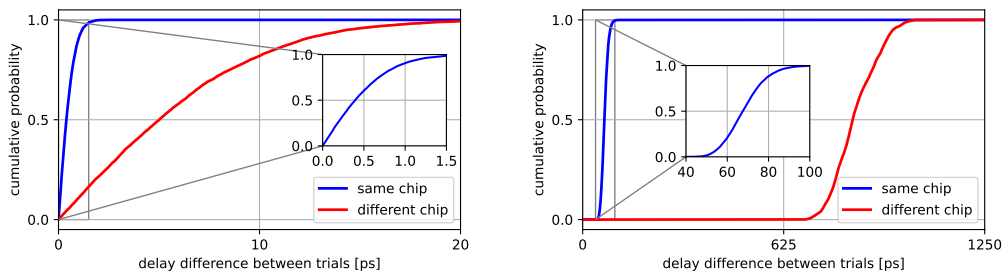


Figure 8: Schematic of a PUF unit cell, and its differential delay paths using two SLLs.



(a) PUF cell difference $\left|D_{t,s} - D_{t',s}\right|$ (b) instance difference $\sum_{s=1}^{144} \left|D_{t,s} - D_{t',s}\right|$

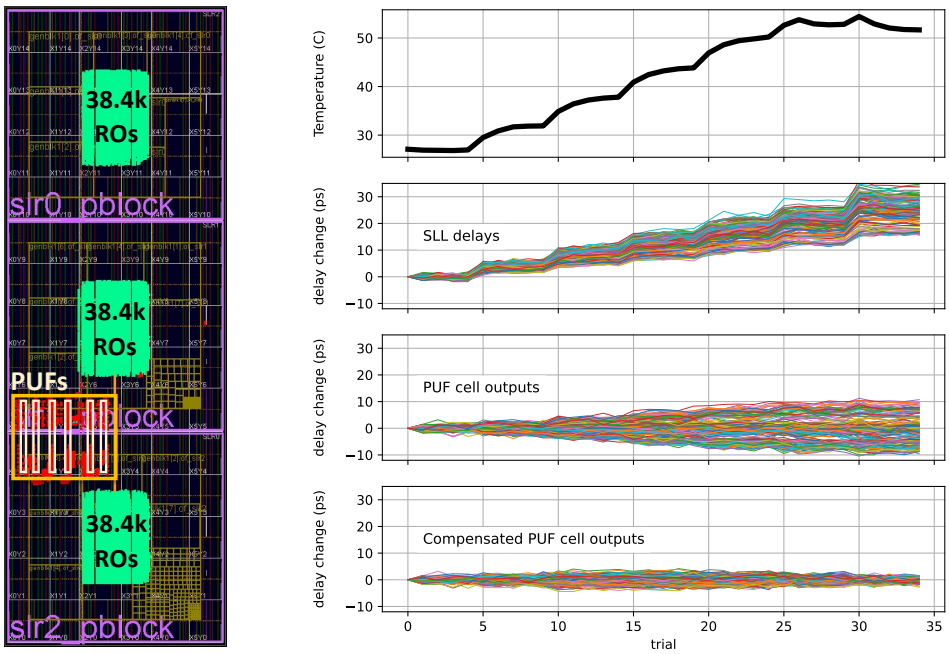
Figure 9: Cumulative distribution of within-class and between-class distance for individual PUF cells (a) and for entire chip instances when all 144 PUF cells are used (b).

distance is 847.713 ps. Across the 10,000 randomly chosen comparisons of each type, the maximum within-class distance was 123.992 ps and the minimum between-class distance was 686.071 ps; the separation between these values is consistent with the PUF being a reliable and unique fingerprint.

5.3 Impact of Temperature and Voltage Disturbance

To study reliability in an extreme scenario, we add 38,400 power wasting ring oscillators (ROs) to each of the three SLRs (see Fig. 10a). While multiple trials of PUF data are being collected over 30 minutes, increasing numbers of ROs are activated, which is known to disturb supply voltage and cause heating. The die temperature during the experiment (1st subplot of Fig. 10b) is logged using the provided on-die temperature sensor; the temperature stops increasing when the fans turn on at approximately 54°C. We repeat the analysis from Fig. 7 to examine how the SLL delays and PUF outputs are changing over this time. The SLL delays (2nd subplot of Fig. 10b) increase with the measured die temperature, jumping higher each time more ROs are activated. The SLL delays increase by an average of 24.610 ps (3.70%) from the first trial to the last over a temperature difference of 24.57°C. While the average sensitivity of the 288 SLLs is therefore 1.002 ps/°C, the sensitivities are not uniform across SLLs, but instead are distributed between 0.645 ps/°C and 1.384 ps/°C. Due to the non-uniform sensitivity of SLL delay to temperature, the PUF outputs drift over the course of the experiment (3rd subplot of Fig. 10b), with the direction of drift depending on which of the two SLLs is more sensitive to temperature.

If the PUF outputs are used without compensation, within-class comparisons made across a large temperature disparity approach the between-class distances under nominal conditions (Fig. 11a). Fortunately, the drift of each SLL is notably linear with respect to temperature, which allows for a simple compensation to make the PUF robust across a



(a) Floorplan with ROs

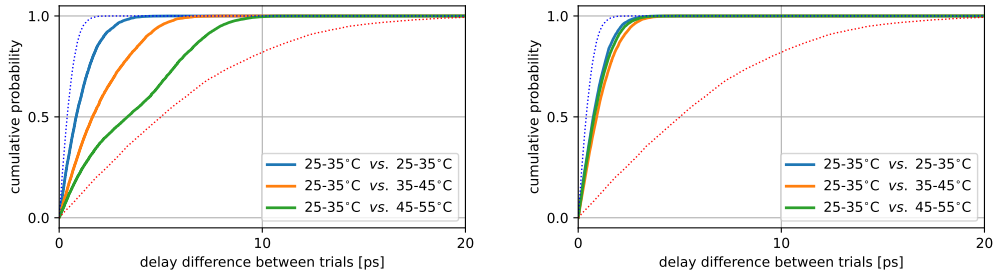
(b) Drift across temperature

Figure 10: Performance across temperature. (a) shows the PUF design instantiated together with 38,400 ROs on each SLR used to create heating. (b) shows how the SLL delays and the PUF outputs change across the trials as the chip heats up.

wide temperature range. It is obvious that one way to compensate is to collect temperature data during PUF measurement, but this assumes the availability of a sensor in a location able to make representative measurements. Instead we compensate the delay of individual SLLs based on the amount of aggregate SLL delay change between two measurements. We find for each SLL a compensation factor, which is a unitless quantity that denotes how much its own delay changes relative to a certain amount of average delay change across all SLLs. An SLL with typical sensitivity will have a compensation factor of 1.0, meaning that it is expected to get 1 ps slower when the average delay across all SLLs increases by 1 ps. More temperature sensitive SLLs will be above 1.0, and less sensitive SLLs will be below 1.0. When two sets of measurements are taken at different temperatures, and they therefore have different mean SLL delays, the 2nd is compensated to the mean SLL delay of the 1st by adjusting the SLL delays according to their compensation factors, which allows for a robust comparison between the two measurements. Note that temperature is not used in the compensation. The compensated PUF outputs are shown in the 4th subplot of Fig. 10b, and the distribution of within-class distances are shown in Fig 11b, indicating that with simple compensation, comparisons across the largest temperature disparities are only slightly degraded.

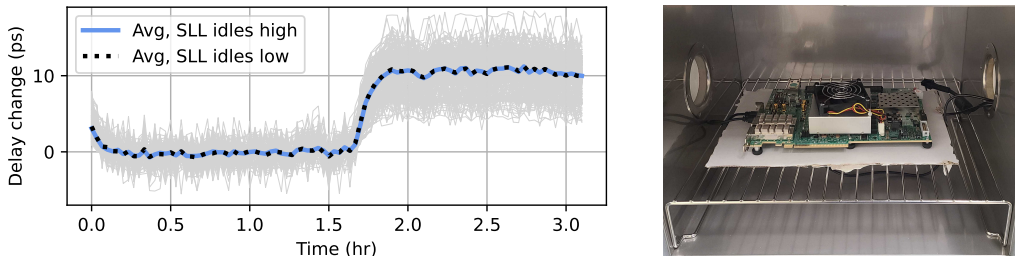
5.4 Aging

Aging is a phenomenon that can affect the long-term reliability of delay PUFs or SRAM PUFs. We test the impact of aging on our design by randomly dividing 288 SLLs into two groups, and aging the two groups in opposite directions. In one group, the SLLs are pulled low when idle between measurements, and in the other the SLLs are pulled high



(a) PUF cell difference (b) PUF cell difference, compensated

Figure 11: Cumulative distribution of within-class distances when compensation is, or is not, used. Dashed lines show for comparison the results from Fig. 9a, which are collected under default conditions without disturbance from oscillators.



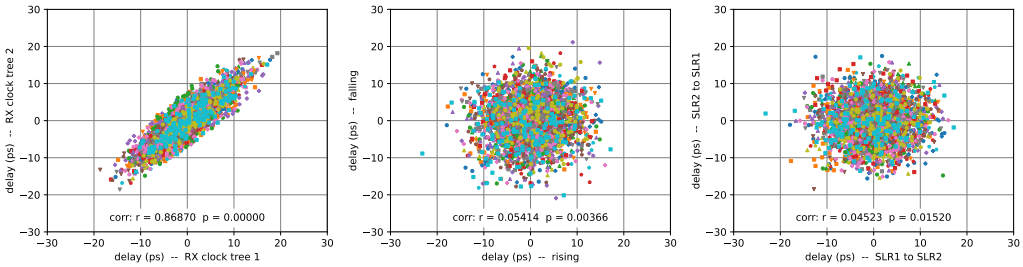
(a) SLL delay fluctuations with aging between measurements (b) VCU118 in thermal chamber

Figure 12: Average delay drift measurements for two groups of SLLs, which are aged using opposite states. The temperature is kept at 25°C for the first 1.5 hours, and then raised to 50°C. The delays fluctuate across time and temperature as expected, but there are no differences between the delay changes of the two groups due to their aging condition.

between measurements. We then run our design and collect samples at 2 minute intervals. To accelerate aging, the experiment is performed in a TestEquity Model 115A [the] Temperature Chamber (Fig. 12b) and the chamber temperature is raised from 25°C to 50°C halfway through the experiment. If NBTI or PBTI had a significant effect one would expect the two groups to diverge with time. The results of our experiment (Fig. 12) show no such evidence of aging, which implies that aging is having little or no impact.

5.5 Quantifying Entropy Source

We conduct a set of targeted experiments to understand better the performance of the PUF cell and identify the source of its entropy. In each of these experiments, we measure the correlation between values taken under scenarios in which there is a specific difference applied to the design. We use a population of 20 EC2 F1 instances in each experiment. The outputs of the PUF cells do not have a mean value of 0, because the 2 SLLs in the PUF cell will generally have different typical delays, as was shown in Sec. 4 when discussing SLL delays. To ensure that the correlation is not falsely inflated by the bias, in this subsection we first de-bias the PUF outputs by subtracting from the PUF outputs of each chip, the mean output of the same PUF across the other 19 chips.



(a) changing clock routing (b) rising vs falling transition (c) swapping TX and RX

Figure 13: Scatterplot shows correlations between the PUF outputs for (a) the case where clock routing is changed; (b) where a rising transition is replaced by a falling one; and (c) where the direction of the propagating signal is inverted.

5.5.1 Sensitivity to Changes in Clock Skew

The first experiment is to evaluate our hypothesis that the differential measurement of the PUF causes clock variations to be common-mode and thus having minimal impact on the PUF outputs. We mimic a massive change in clock skew by creating an additional design variant in which the MMCM of the receiver has been moved across the SLR boundary, which is observed to greatly change the clock path and its skew according to the timing report. We then deploy the two variants sequentially to the same 20 EC2 F1 instances, and compare the results of the 144 PUF cells across the variants as through they come from the same design. The scatterplot in Fig. 13a evaluates the correlation between the de-biased versions of the same PUF cells on both chips. The strong positive correlation of 0.86870 indicates the clock has a negligible impact on the PUF outputs, which implies that the variability observed in the PUFs comes from the SLLs themselves, as intended.

5.5.2 Rising vs Falling Transitions

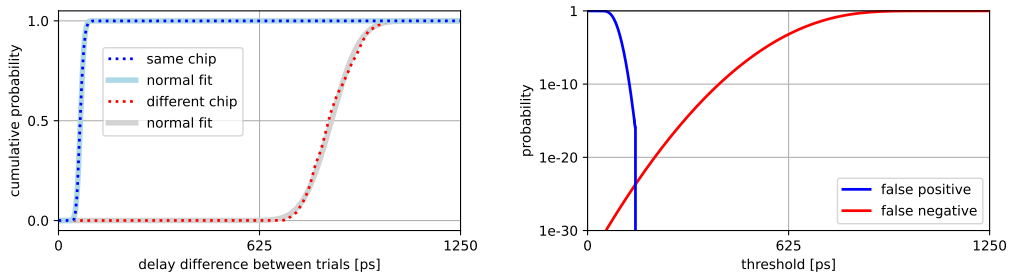
For the scatterplot in Fig. 13b, the design is configured so that it can be set to measure delay based on rising transitions as usual, or can be inverted to measure delay based on timing faults from falling transitions. The results show a weakly positive correlation, with a p-value of 0.0037 indicating that the finding is significant. The weakness of the correlation can be explained by the rising and falling transitions being logical complements of each other; the pull-up transistor that drives the rising transition is distinct from the pull-down transistor that drives the falling transition, so any entropy associated with the driver will not be common across the two scenarios. Similarly, the receiver flip-flop will have asymmetry in how it samples each transition.

5.5.3 Driving Same SLL in Opposite Directions

Each interposer wire can be driven in either direction depending on whether the TX or RX flop of each Laguna cell is activated (see Fig. 8). An interesting case is to evaluate whether there is correlation across opposite directions. The scatterplot shown in Fig. 13c again finds a weakly positive correlation, slightly weaker than between the rising/falling comparison for a single direction. As in the previous case, the transistors that drive the wire are again distinct for the two cases, but now they also differ in position which may contribute environmental differences that further reduce similarity.

5.6 Modeling of False Positive and Negatives

One of our security objectives is to authenticate the neighboring chiplet, and for that application it is important to prevent false negatives, in which a chiplet is replaced by another but accepted as authentic, and false positives, in which an authentic chiplet is labeled inauthentic. Toward that goal, our within-class and between-class distances (Fig. 9b) and the large margin between them are promising. We observe that within-class and between-class distance distributions are approximately normally distributed as illustrated in Fig. 14a, and use the fitted normal distributions to estimate false positive rates and false negative rates of a larger population. The mean and standard deviation are 68.1 ps and 9.8 ps respectively for within-class comparisons, and 848.5 ps and 68.6 ps for between-class comparisons. The achievable tradeoffs of false positives and false negatives are shown in Fig. 14b. A threshold of 150.31 ps causes false positive and false negative probabilities to be equal, with both probabilities being 2.259E-24 at that threshold. Although the probability of misclassification is already low, it can be further reduced by using a larger number of PUF cells.



(a) normally fitted distributions (b) classification probabilities

Figure 14: Fitted distribution lines for within-class and between-class distributions from Fig. 9b and type I and type II error probabilities across classification thresholds.

6 Performance

6.1 Resource Utilization and Power

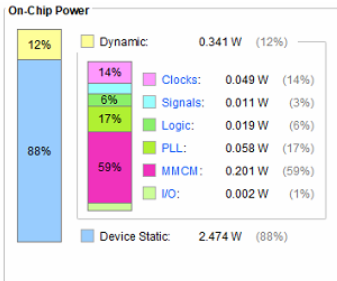
Fig. 15a offers a detailed summary of resource utilization. Our design is lightweight and utilizes only 3,136 LUTs and 7,933 flip-flops, which is 0.27% and 0.34% of all the available resources respectively. It utilizes two of the 30 MMCM, which is the highest utilization share among any resource type. Similarly, only 288 SLLs that span across two SLRs are utilized for the PUF, with additional 48 wires reserved for asynchronous control signals. This accounts for less than 0.2% of the total number of SLLs available at each crossing, although it may be desirable to increase the number of SLLs used to offer stronger security. Due to its small footprint the design is also efficient, consuming only 0.815 W, nearly 90% of which is static power dissipation. The power report summary generated by Vivado is displayed in Fig. 15b.

6.2 Speed

Currently, our PUF design is configured to have a single pattern generator and a single accumulator per Laguna site, meaning only one SLL in each site can be scanned at a time. Once scanned, the values from the accumulator for each SLL can be read out sequentially for processing. In typical fashion one can significantly reduce the time needed to measure

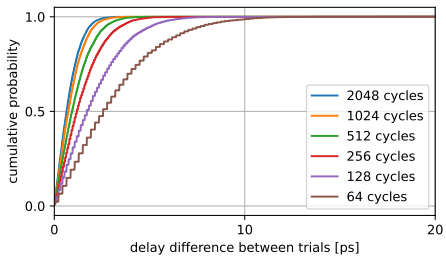
Resource	Utilization	Available	Utilization (%)
LUT	3,136	1,182,240	0.27
FF	7,933	2,364,480	0.34
IO	4	832	0.48
BUFG	8	1,800	0.44
MMCM	2	30	6.67
PLL	1	60	1.67

(a) Utilization summary

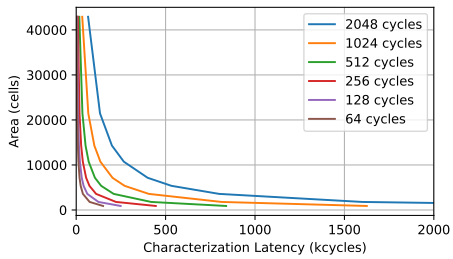


(b) Power estimate summary

Figure 15: Resource utilization and power estimate summaries



(a) Within-class PUF cell distance



(b) Latency vs area tradeoff

Figure 16: Design space exploration

all the SLLs by trading away area to increase parallelism, or by reducing the number of edges transmitted. Fig. 16a and Fig. 16b help visualize the area and time tradeoffs. More specifically, in Fig. 16a one sees that, experimentally, reducing the number of samples collected for each SLL will loosen its delay distribution, which in turn means that the error rate will increase and detection sensitivity will suffer; the discreteness in Fig. 16b arises because there are a relatively few delay numbers that can be measured when there are a small number of possible faults at each clock phase. Fig. 16b on the other hand appears more attractive, as increasing area can dramatically decrease latency by unlocking parallelism, without hurting the PUF sensitivity as was the case with shortening the applied pattern length for each SLL.

Finally, one can further cut down runtime by doing a focused phase sweep, meaning instead of sweeping every possible phase one can only sweep a small number around the expected delay value of each SLL. Because dynamic phase shifting is deterministic and consumes 12 cycles to increment or decrement the phase of the controlled clock, focused phase sweeping also scales linearly.

7 Conclusions

To get around reticle limits and cope with the need for larger and larger chips many manufacturers are switching to the chiplet model. With new possibilities and challenges of building devices by stacking multiple dies also comes new security concerns and opportunities. PUFs have long been used for authentication of chips, however with multiple chiplets it is not sufficient to verify integrity of an individual die. We present a novel method of chiplet neighbor authentication and tamper sensing by building a PUF out of the interposer wires that connect two dies. We implemented our PUF on Xilinx VU9P

FPGAs and demonstrated its effectiveness by scaling up our experiments using AWS EC2 F1 instances. With minimal overhead and all the necessary implementation resources already available our PUF design can potentially be adapted to inter-chiplet buses from other manufacturers in future work.

Acknowledgements

This work has been supported in part by a grant from the National Science Foundation (NSF), and by the MITRE Corporation. Portions of this technical data were produced for the U. S. Government under Contract No. FA8702-19-C-0001 and W56KGU-18-D-0004, and is subject to the Rights in Technical Data-Noncommercial Items Clause DFARS 252.227-7013 (FEB 2014). ©2022 Aleksa Deric, Daniel Holcomb, The MITRE Corporation. All rights reserved. Approved for Public Release; Distribution Unlimited. 22-1081.

References

[ACF+20] Shahab Ardalan, Halil Cirit, Ramin Farjad, Mark Kuemerle, Ken Poulton, Suresh Subramanian, and Bapiraju Vinnakota. Bunch of wires: An open die-to-die interface. In *2020 IEEE Symposium on High-Performance Interconnects (HOTI)*, pages 9–16, 2020.

[And10] Jason H. Anderson. A PUF design for secure FPGA-based embedded systems. In *Proceedings of the 2010 Asia and South Pacific Design Automation Conference*, pages 1–6, 2010.

[BWG15] Georg T. Becker, Alexander Wild, and Tim Güneysu. Security analysis of index-based syndrome coding for PUF-based key generation. In *2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 20–25, 2015.

[CEV22] CEVA. Fortrix: Self-contained IP platform for Root-of-Trust and cybersecurity in chiplets and SoCs, 1 2022. Product note.

[CMK+21] Gino Chacon, Tapojyoti Mandal, Johann Knechtel, Ozgur Sinanoglu, Paul Gratz, and Vassos Soteriou. Coherence attacks and countermeasures in interposer-based systems, 2021.

[GCVD02] B Gassend, D Clarke, and M Van Dijk. Silicon physical random functions. In *Proceedings of the ACM Conference on Computer and Communications Security*, pages 148–160, 2002.

[GGB] GGB INDUSTRIES INC. Picoprobe Models 28 and 29. Product note.

[GKST07] J Guajardo, S Kumar, G Schrijen, and P Tuyls. FPGA intrinsic PUFs and their use for IP protection. *Cryptographic Hardware and Embedded Systems*, pages 63–80, 2007.

[GLS+16] Peng Gu, Shuangchen Li, Dylan Stow, Russell Barnes, Liu Liu, Yuan Xie, and Eren Kursun. Leveraging 3D technologies for hardware security: Opportunities and challenges. In *2016 International Great Lakes Symposium on VLSI (GLSVLSI)*, pages 347–352, 2016.

[ION+18] Vincent Immler, Johannes Obermaier, Kuan Kuan Ng, Fei Xiang Ke, JinYu Lee, Yak Peng Lim, Wei Koon Oh, Keng Hoong Wee, and Georg Sigl. Secure physical enclosures from covers with tamper-resistance. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019(1):51–96, Nov. 2018.

[Keh17] David Kehlet. Accelerating innovation through a standard chiplet interface: The advanced interface bus (AIB). *Intel White Paper*, 2017.

[KGM+08] S S Kumar, J Guajardo, R Maes, G J Schrijen, and P Tuyls. The butterfly PUF protecting IP on every FPGA. *IEEE International Workshop on Hardware-Oriented Security and Trust*, pages 67–70, 2008.

[LDT00] Keith Lofstrom, W Robert Daasch, and Donald Taylor. IC identification circuit using device mismatch. In *2000 IEEE International Solid-State Circuits Conference. Digest of Technical Papers (Cat. No. 00CH37056)*, pages 372–373. IEEE, 2000.

[LHT+19] Mu-Shan Lin, Tze-Chiang Huang, Chien-Chun Tsai, King-Ho Tam, Cheng-Hsiang Hsieh, Tom Chen, Wen-Hung Huang, Jack Hu, Yu-Chi Chen, Sandeep Kumar Goel, Chin-Ming Fu, Stefan Rusu, Chao-Chieh Li, Sheng-Yao Yang, Mei Wong, Shu-Chun Yang, and Frank Lee. A 7nm 4GHz Arm-core-based CoWoS chiplet design for high performance computing. In *2019 Symposium on VLSI Circuits*, pages C28–C29, 2019.

[LHY+20] Tao Li, Jie Hou, Jinli Yan, Rulin Liu, Hui Yang, and Zhigang Sun. Chiplet heterogeneous integration technology—status and challenges. *Electronics*, 9(4), 2020.

[LMS+20] Hyung-Jin Lee, Ravi Mahajan, Farhana Sheikh, Ramune Nagisetty, and Manish Deo. Multi-die integration using advanced packaging technologies. In *2020 IEEE Custom Integrated Circuits Conference (CICC)*, pages 1–7. IEEE, 2020.

[MKD10] M. Majzoobi, F. Koushanfar, and S. Devadas. FPGA PUF using programmable delay lines. In *International Workshop on Information Forensics and Security*, pages 1–6, Dec 2010.

[MKN+21] Anthony Mastroianni, Benjamin Kerr, Jawad Nasrullah, Kevin Cameron, Hockshan James Wong, David Ratchkov, and Joseph Reynick. Proposed standardization of heterogenous integrated chiplet models. In *2021 IEEE International 3D Systems Integration Conference (3DIC)*, pages 1–8, 2021.

[MMS10] Sergey Morozov, Abhranil Maiti, and Patrick Schaumont. An analysis of delay based PUF implementations on FPGA. In *Reconfigurable Computing: Architectures, Tools and Applications*, pages 382–387. Springer, 2010.

[MQV+19] Ravi Mahajan, Zhiguo Qian, Ram S. Viswanath, Sriram Srinivasan, Kemal Aygün, Wei-Lun Jen, Sujit Sharan, and Ashish Dhall. Embedded multidie interconnect bridge—a localized, high-density multichip packaging interconnect. *IEEE Transactions on Components, Packaging and Manufacturing Technology*, 9(10):1952–1962, 2019.

[MTV08] Roel Maes, Pim Tuyls, and Ingrid Verbauwhede. Intrinsic PUFs from flip-flops on reconfigurable devices. In *3rd Benelux Workshop on Information and System Security*, volume 17, 2008.

[NAP+20] Mohammed Nabeel, Mohammed Ashraf, Satwik Patnaik, Vassos Soteriou, Ozgur Sinanoglu, and Johann Knechtel. 2.5d root of trust: Secure system-level integration of untrusted chiplets. *IEEE Transactions on Computers*, 69(11):1611–1625, 2020.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[RBMC20] Scott W Rose, Oliver Borchert, Stuart Mitchell, and Sean Connelly. Zero trust architecture. *NIST Special Publication*, 800(207), 2020.

[Sab12] Kirk Saban. Xilinx stacked silicon interconnect technology delivers breakthrough FPGA capacity, bandwidth, and power efficiency. *White Paper: Virtex-7 FPGAs*, 2012.

[SBK20] Mohammed Shayan, Kanad Basu, and Ramesh Karri. Security assessment of interposer-based chiplet integration, 2020.

[SD07] G Edward Suh and Srinivas Devadas. Physical unclonable functions for device authentication and secret key generation. In *2007 44th ACM/IEEE Design Automation Conference*, pages 9–14. IEEE, 2007.

[SGB+10] O. Sander, B. Glas, L. Braun, K.D. Müller-Glaser, and J. Becker. Intrinsic identification of Xilinx Virtex-5 FPGA devices using uninitialized parts of configuration memory space. In *International Conference on Reconfigurable Computing and FPGAs*, pages 13–18, Dec 2010.

[SHO07] Y. Su, J. Holleman, and B. Otis. A 1.6pj/bit 96% stable chip-id generating circuit using process variations. In *2007 IEEE International Solid-State Circuits Conference. Digest of Technical Papers*, pages 406–611, 2007.

[the] TestEquity Chambers Model 115A Temperature Chamber. http://www.testequity.com/115A-specs.

[TXG+20] Shanquan Tian, Wenjie Xiong, Ilias Giechaskiel, Kasper Rasmussen, and Jakub Szefer. Fingerprinting cloud FPGA infrastructures. In *The 2020 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, pages 58–64. ACM, 2020.

[WG14] A. Wild and T. Guneysu. Enabling SRAM-PUFs on Xilinx FPGAs. In *International Conference on Field Programmable Logic and Applications*, pages 1–4, Sept 2014.

[XMYW21] Zhenyu Xu, Thomas Mauldin, Qing Yang, and Tao Wei. Runtime detection of probing/tampering on interconnecting buses. In *2021 IEEE 29th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, pages 247–251, 2021.

[YD10] Meng-Day Yu and Srinivas Devadas. Secure and robust error correction for physical unclonable functions. *IEEE Design & Test of Computers*, 27(1):48–65, 2010.

[YHD+16] Meng-Day Yu, Matthias Hiller, Jeroen Delvaux, Richard Sowell, Srinivas Devadas, and Ingrid Verbauwhede. A lockdown technique to prevent machine learning on PUFs for lightweight authentication. *IEEE Transactions on Multi-Scale Computing Systems*, 2(3):146–159, 2016.