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

Aleksa Deric, Daniel E. Holcomb
University of Massachusetts Amherst

CHES'22

University of Massachusetts Amherst

MITRE

Introduction

Industry trends toward chiplets as a replacement for monolithic fabrication

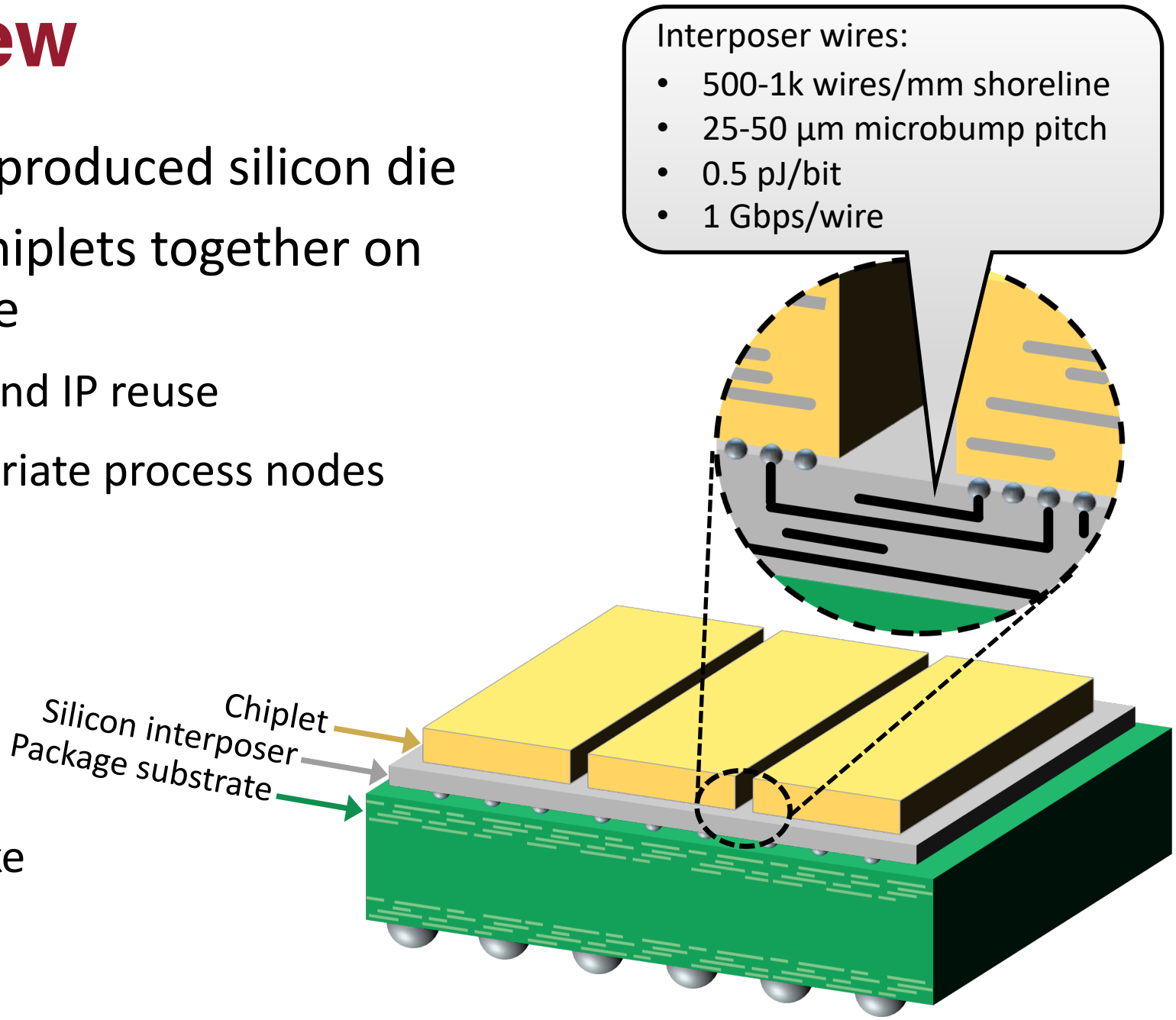
The modularity of chiplets brings new and interesting security threats

This work: inter-die delay PUF as a security primitive for chiplets

Chiplets - Overview

- Each chiplet is a separately-produced silicon die
- SoC created by packaging chiplets together on a silicon interposer or bridge
  - Heterogeneous integration and IP reuse
  - Able to leverage cost-appropriate process nodes
  - Increased yield

- Recent examples
  - AMD Ryzen
  - Intel Meteor Lake, Arrow Lake
  - Xilinx Virtex Ultrascale+



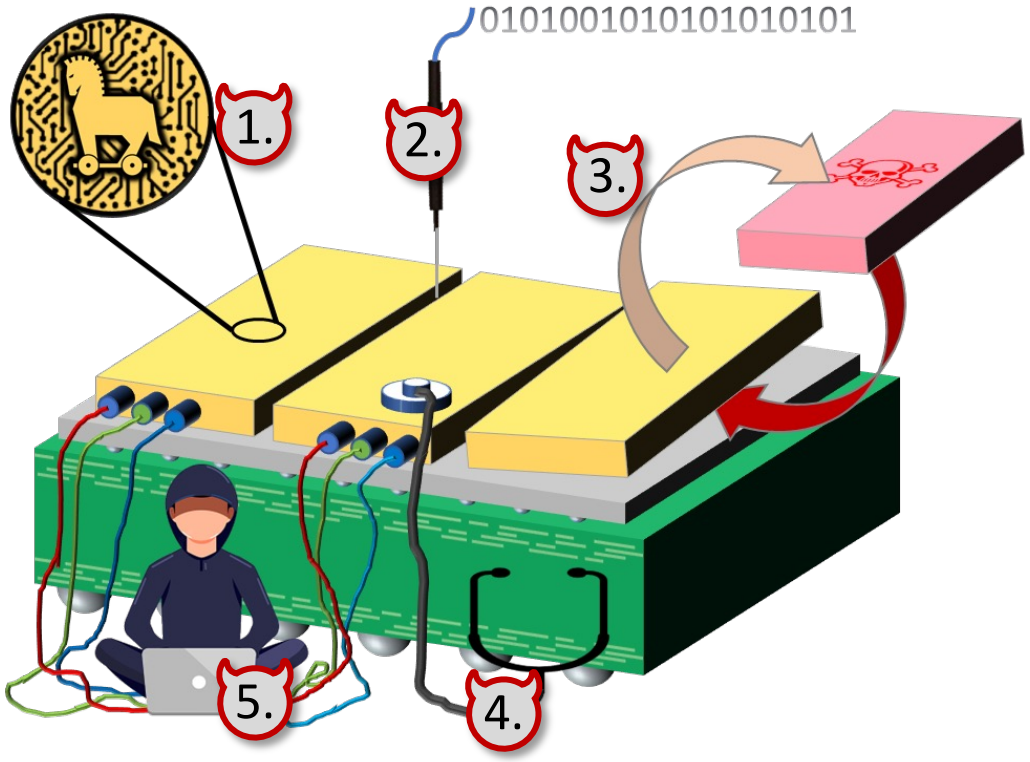
Chiplets – Motivation & Related Work

- Different threats possible with chiplets vs monolithic fabrication
  - Which are critical and how to defend?

- Zero trust: chiplets cannot blindly assume they are operating in a friendly environment
  - Root of trust needed
  - Using cryptography and PUFs [1]
  - Trusted security-enforcing interposer with active traffic policing [2]
  - Secure-by-construction interposer Networks-on-Chip with message checking [3]

- This work: inter-chiplet delay fingerprints through interposer for physical security

1. Trojans in co-packaged chiplets
2. Probing exposed interposer wires
3. Die-swapping
4. Side-channels from within package
5. Man-in-the-middle



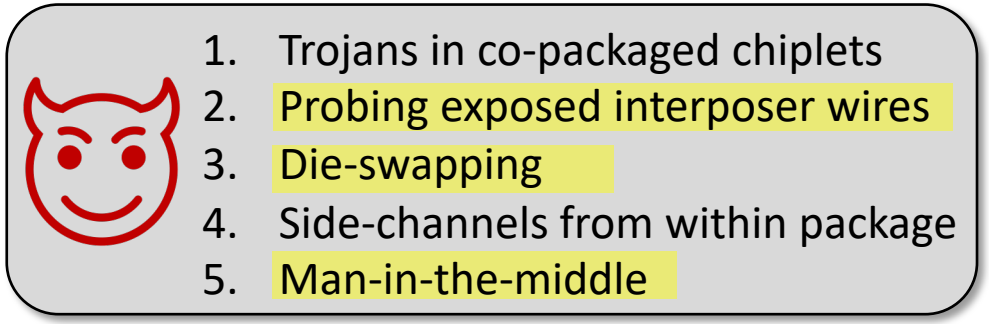
[1] CEVA. Fortrix: Self-contained IP platform for Root-of-Trust and cybersecurity in chiplets and SoCs. Product note. 2022
[2] Nabeel et al. 2.5d root of trust: Secure system-level integration of untrusted chiplets. IEEE T-Comp, 2020
[3] Chacon et al. Coherence attacks and countermeasures in interposer-based systems, 2021

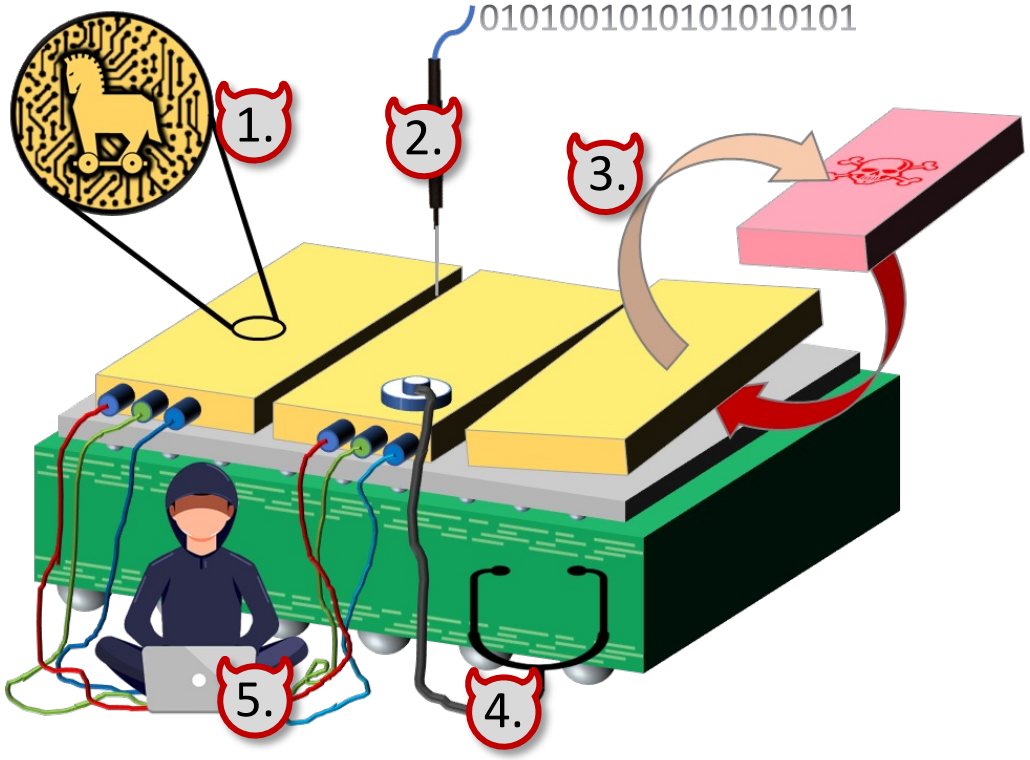
Chiplets – Motivation & Related Work

- Different threats possible with chiplets vs monolithic fabrication
  - Which are critical and how to defend?

- Zero trust: chiplets cannot blindly assume they are operating in a friendly environment
  - Root of trust needed
  - Using cryptography and PUFs [1]
  - Trusted security-enforcing interposer with active traffic policing [2]
  - Secure-by-construction interposer Networks-on-Chip with message checking [3]

- This work: inter-chiplet delay fingerprints through interposer for physical security





[1] CEVA. Fortrix: Self-contained IP platform for Root-of-Trust and cybersecurity in chiplets and SoCs. Product note. 2022
[2] Nabeel et al. 2.5d root of trust: Secure system-level integration of untrusted chiplets. IEEE T-Comp, 2020
[3] Chacon et al. Coherence attacks and countermeasures in interposer-based systems, 2021

Overview

Measuring Delay

Design & Implementation

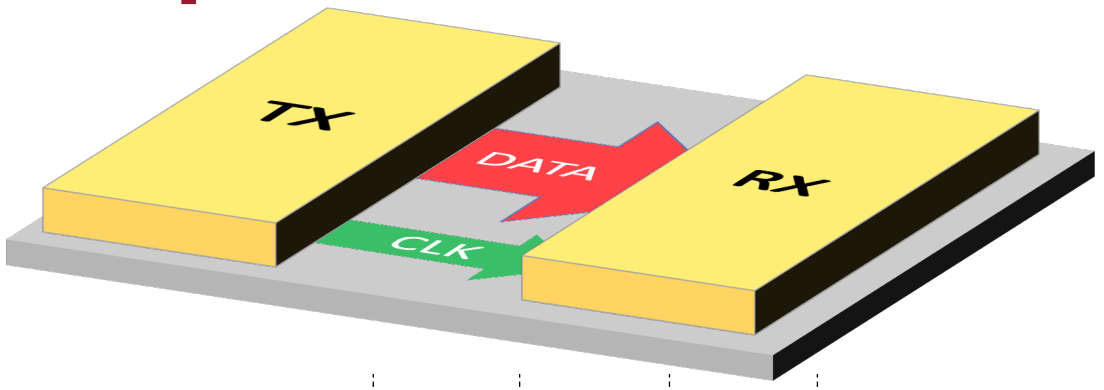
Statistics

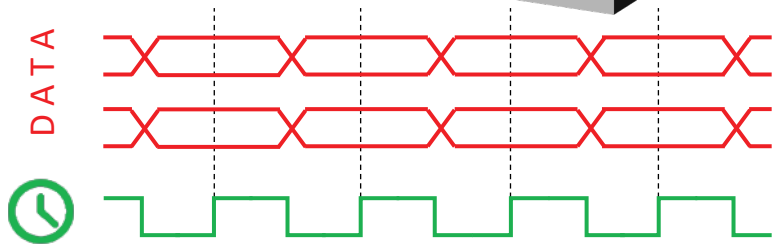
Characterization Experiments

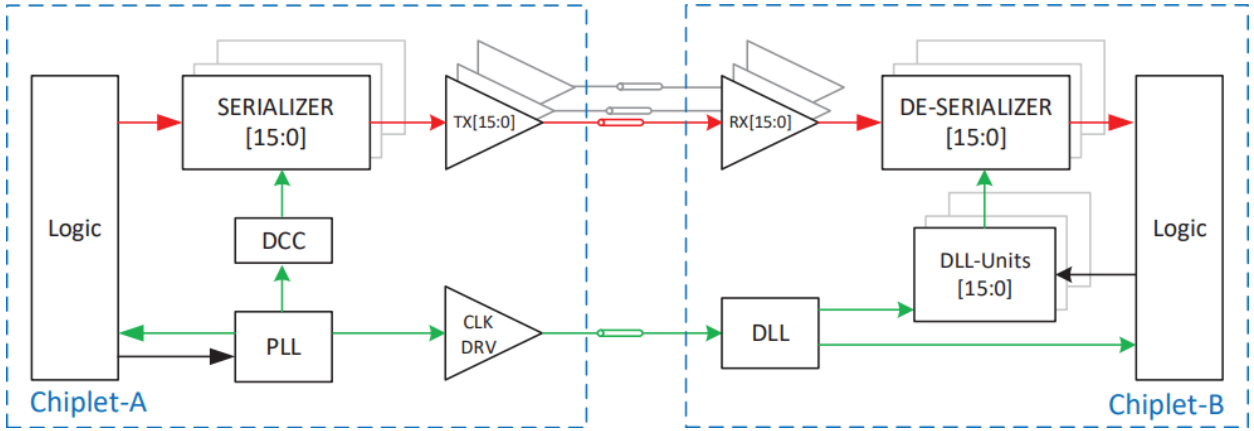
Summary

Communication Between Chiplets

- Typically source synchronous clocking
  - Data and clock forwarded from TX
  - Wires crossing through interposer
  - Registered I/O
  - Tunable delay on RX deskews sampling clock
- Emerging standards
  - Intel AIB [1]
  - TSMC LIPINCON [2]
  - UCIe [3]
  - Bunch-of-Wires [4]





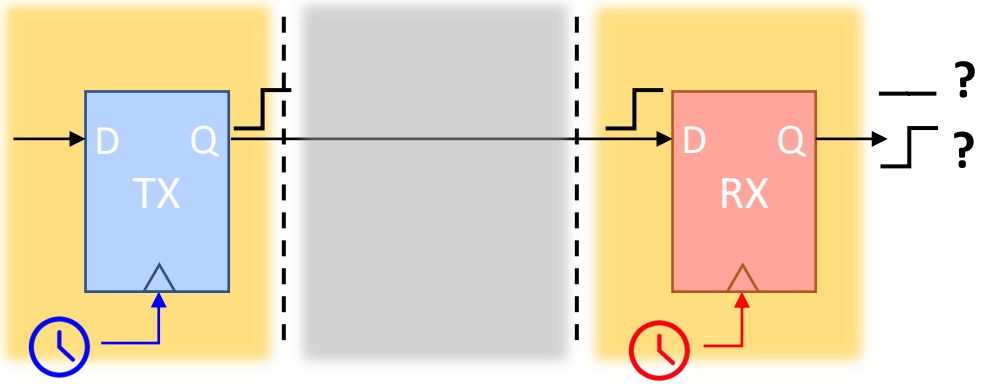


[Farjadrad et al.]

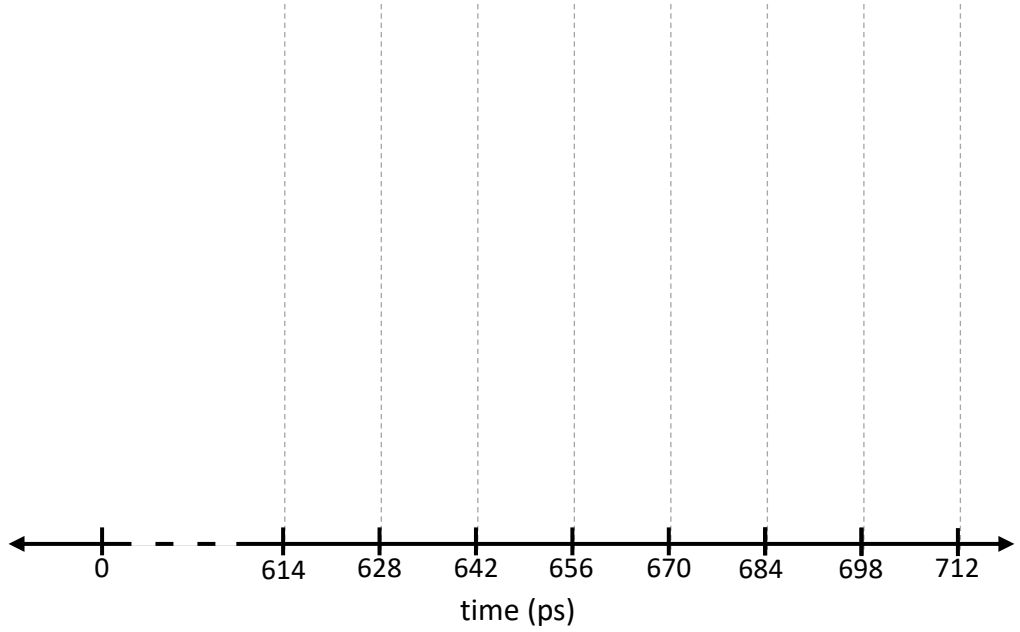
[1] David Kehlet. Accelerating innovation through a standard chiplet interface: The advanced interface bus (AIB). Intel White Paper, 2017
[2] Lin et al. A 7nm 4GHz Armcore-based CoWoS chiplet design for high performance computing. In 2019 Symp on VLSI Circuits, 2019
[3] D. Das Sharma, “Universal Chiplet Interconnect express (UCIe)® : Building an open chiplet ecosystem”, UCIe Consortium White paper, 2022
[4] R. Farjadrad et al., "A Bunch-of-Wires (BoW) Interface for Interchiplet Communication," IEEE Micro, 2020

Measuring Delay

- Use phase compensation to measure propagation delay of signal from neighboring chiplet
  - Transmit repeatedly
  - Sweep receiver phase
  - Find phase with 50% failure



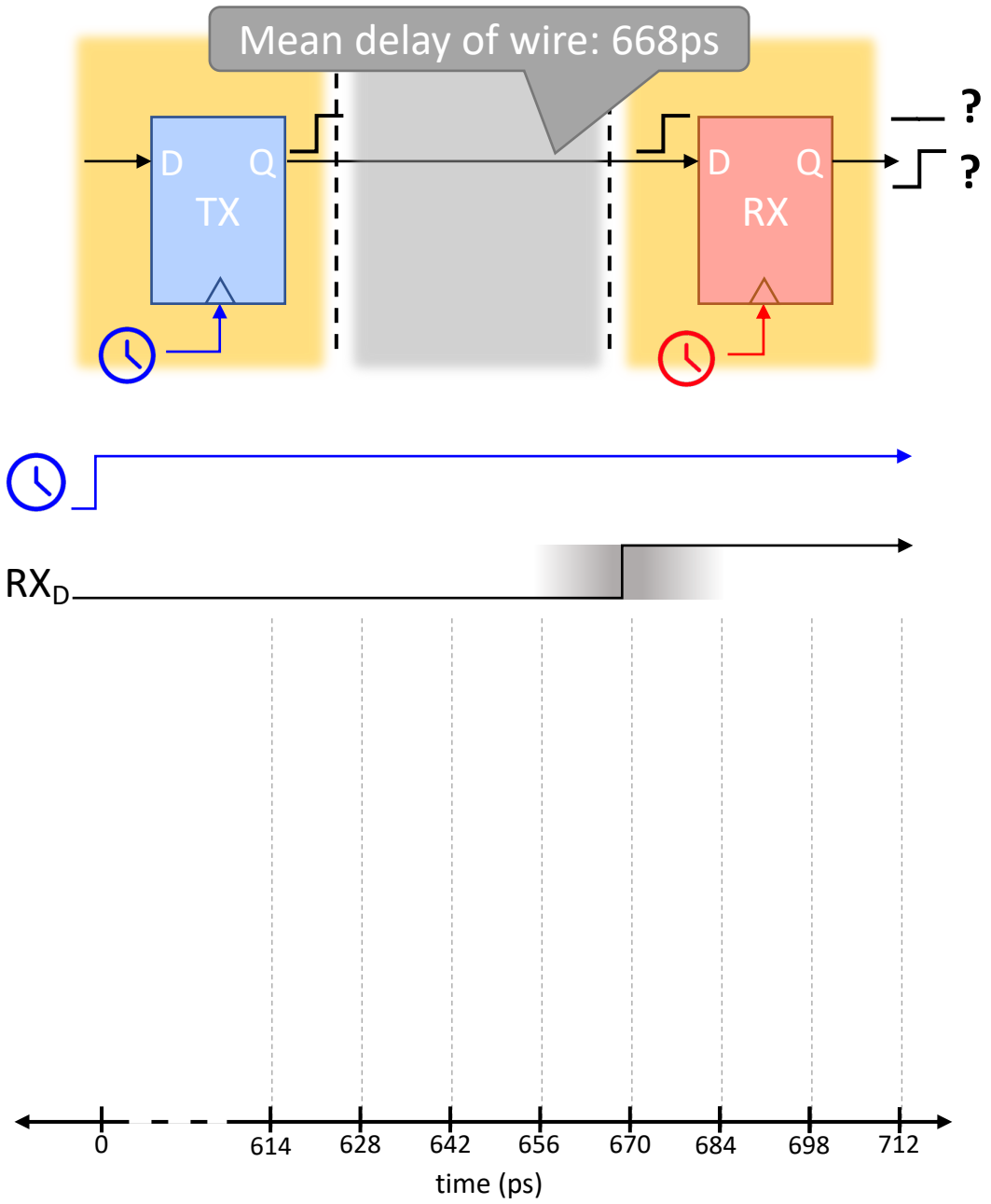
- Delay defined as the skew between TX and RX clock that causes rising transition to be received as 0 and 1 with equal probability



Measuring Delay

- Use phase compensation to measure propagation delay of signal from neighboring chiplet
  - Transmit repeatedly
  - Sweep receiver phase
  - Find phase with 50% failure

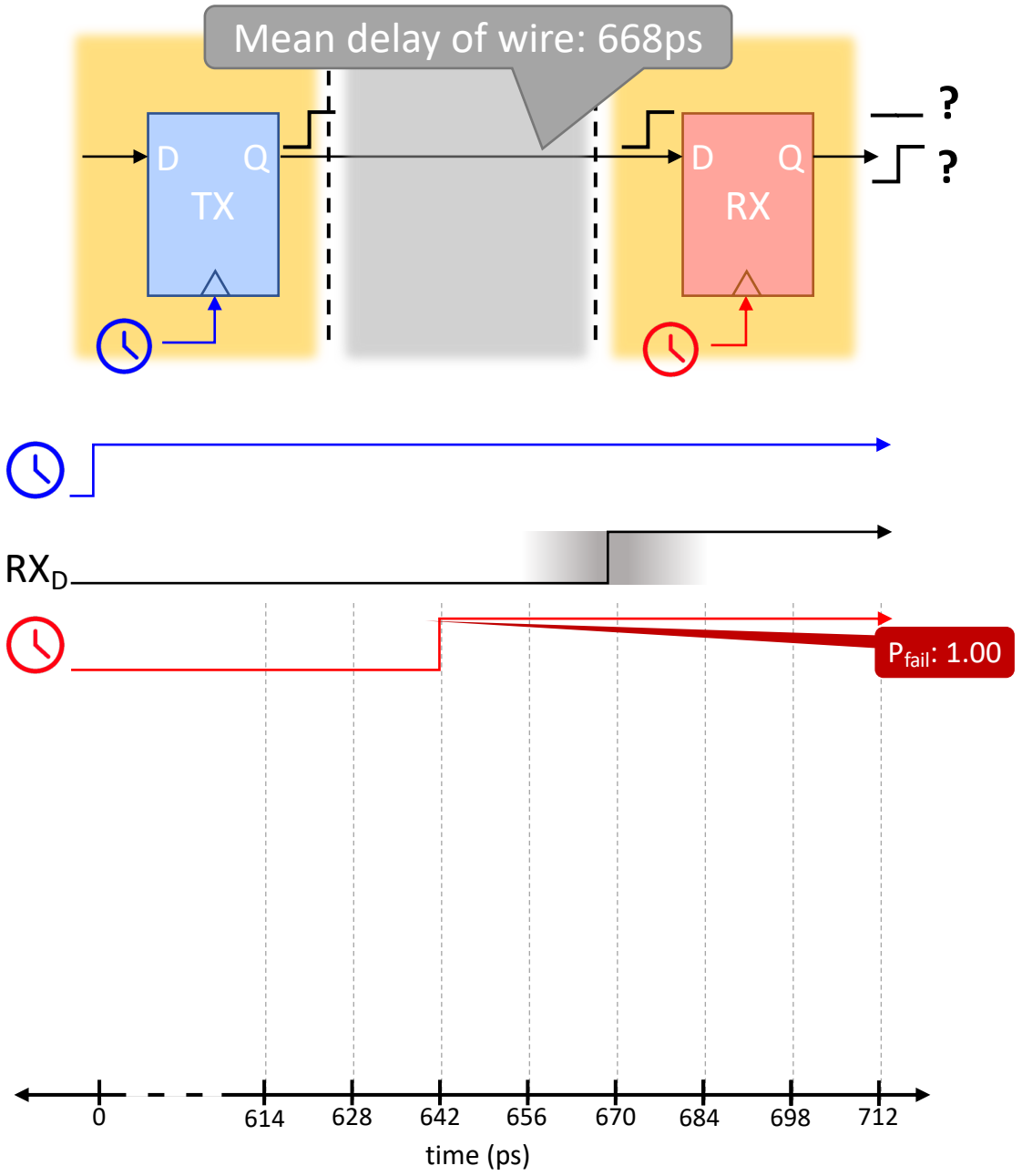
- Delay defined as the skew between TX and RX clock that causes rising transition to be received as 0 and 1 with equal probability



Measuring Delay

- Use phase compensation to measure propagation delay of signal from neighboring chiplet
  - Transmit repeatedly
  - Sweep receiver phase
  - Find phase with 50% failure

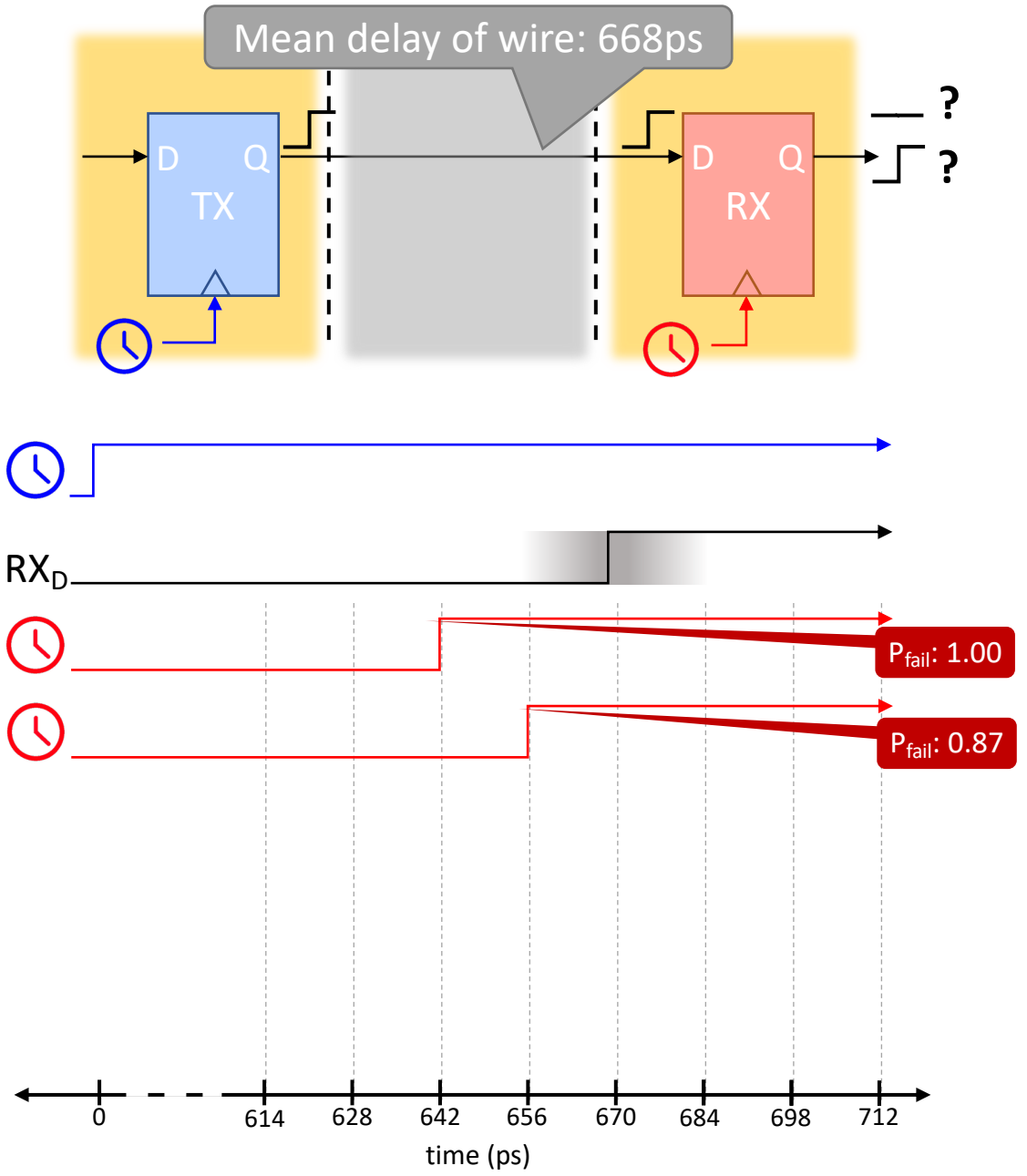
- Delay defined as the skew between TX and RX clock that causes rising transition to be received as 0 and 1 with equal probability



Measuring Delay

- Use phase compensation to measure propagation delay of signal from neighboring chiplet
  - Transmit repeatedly
  - Sweep receiver phase
  - Find phase with 50% failure

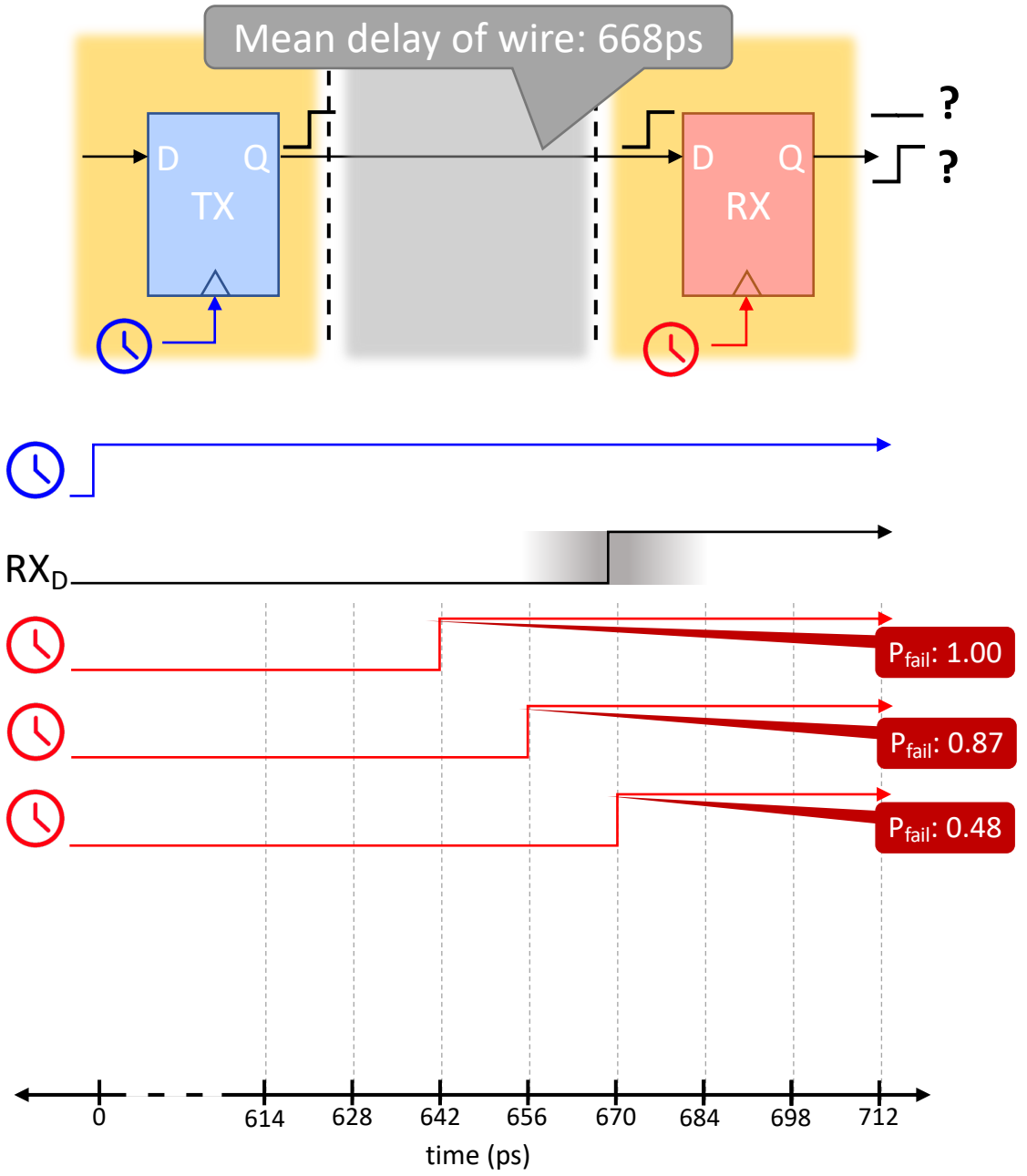
- Delay defined as the skew between TX and RX clock that causes rising transition to be received as 0 and 1 with equal probability



Measuring Delay

- Use phase compensation to measure propagation delay of signal from neighboring chiplet
  - Transmit repeatedly
  - Sweep receiver phase
  - Find phase with 50% failure

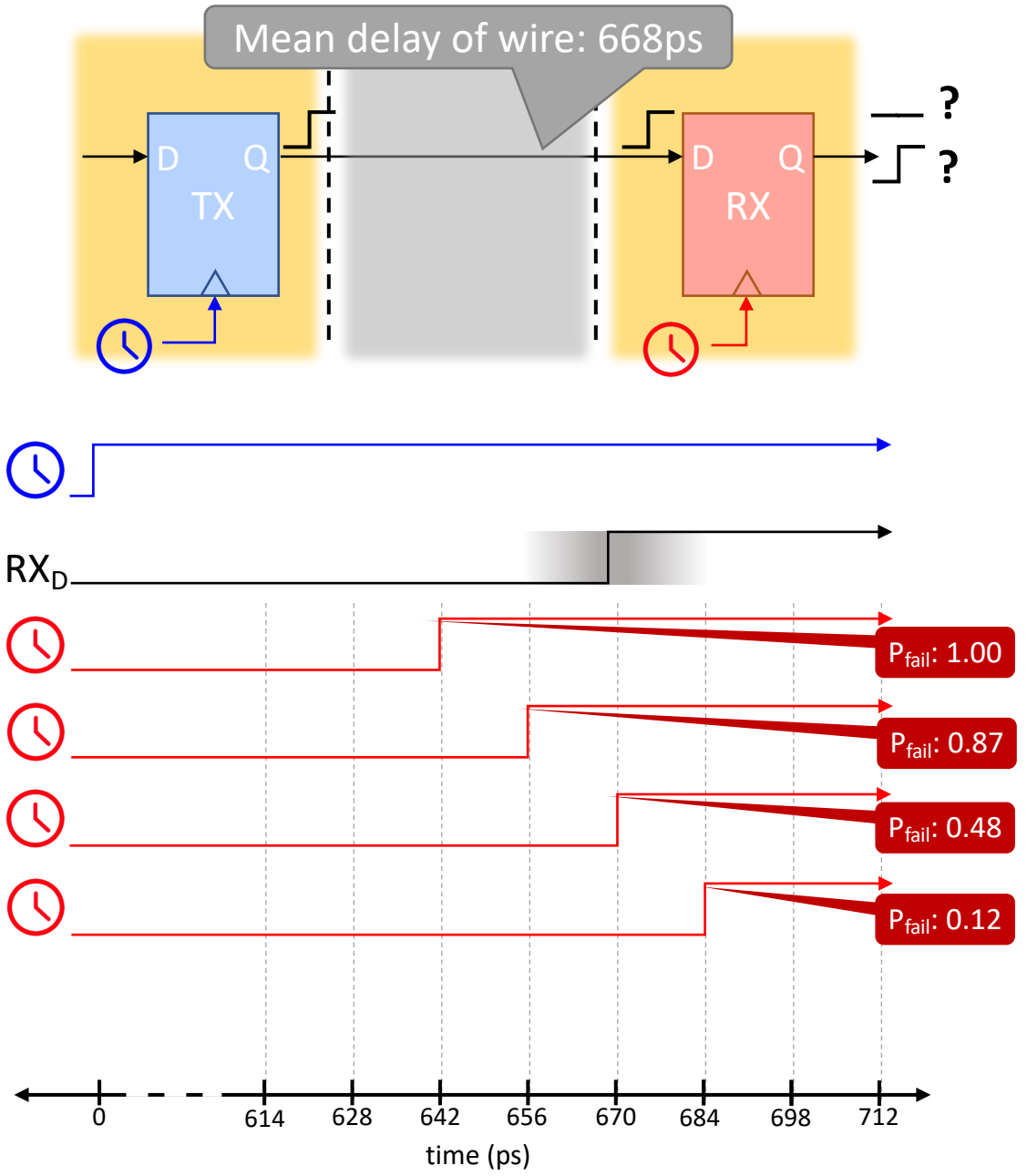
- Delay defined as the skew between TX and RX clock that causes rising transition to be received as 0 and 1 with equal probability



Measuring Delay

- Use phase compensation to measure propagation delay of signal from neighboring chiplet
  - Transmit repeatedly
  - Sweep receiver phase
  - Find phase with 50% failure

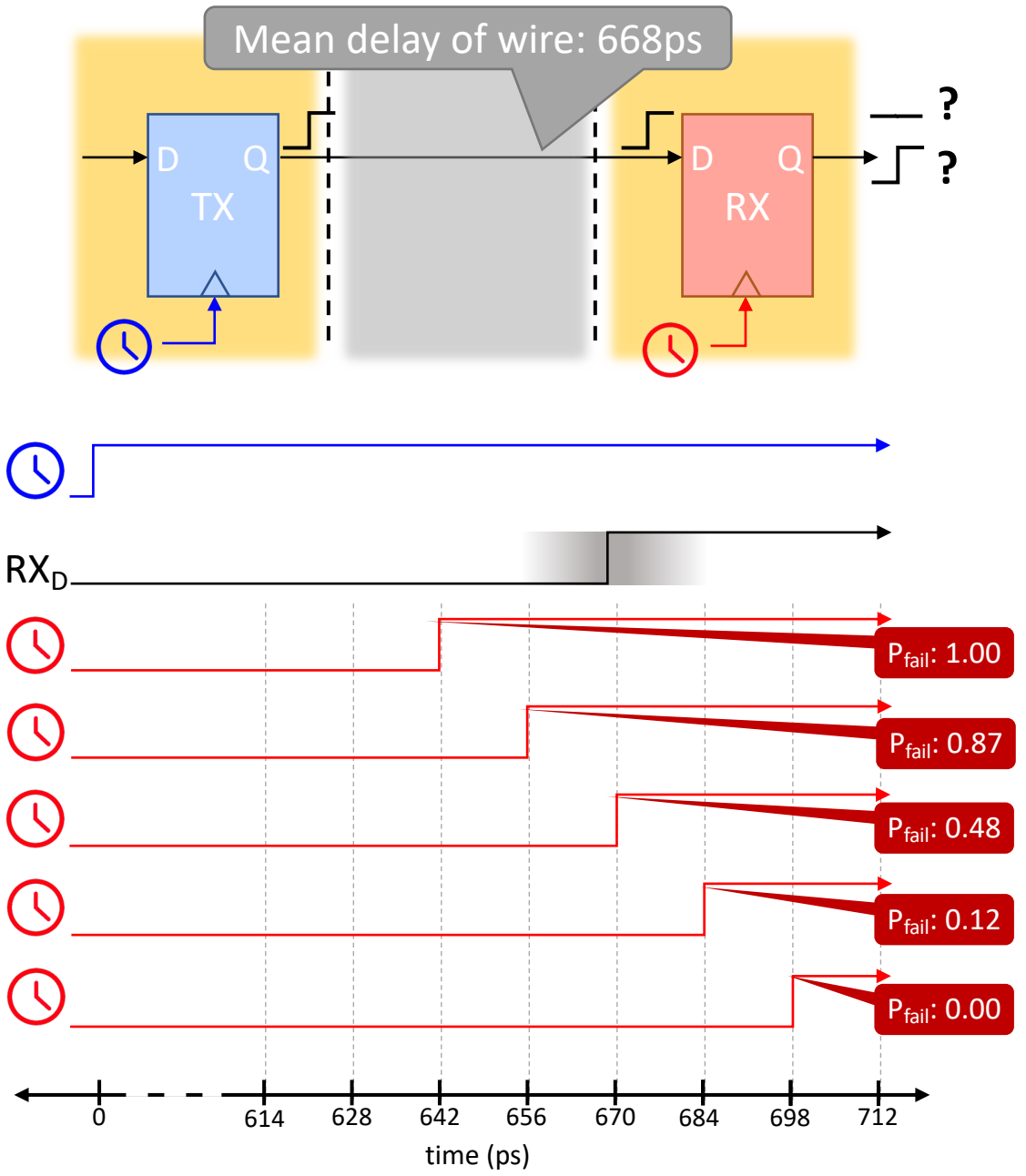
- Delay defined as the skew between TX and RX clock that causes rising transition to be received as 0 and 1 with equal probability



Measuring Delay

- Use phase compensation to measure propagation delay of signal from neighboring chiplet
  - Transmit repeatedly
  - Sweep receiver phase
  - Find phase with 50% failure

- Delay defined as the skew between TX and RX clock that causes rising transition to be received as 0 and 1 with equal probability

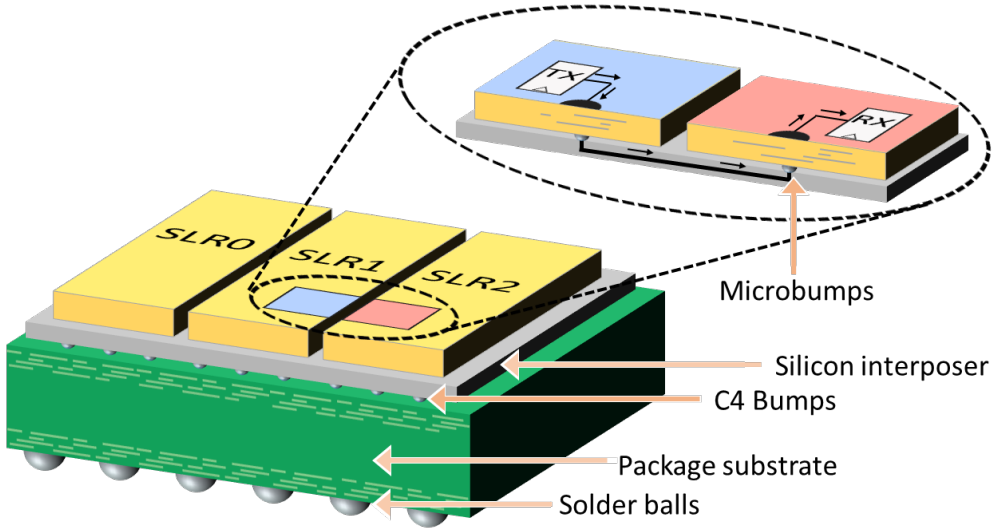


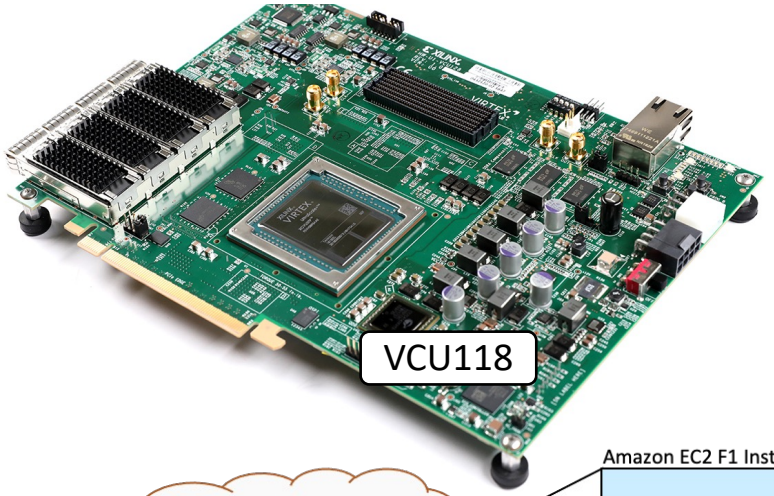
Experimentation Platforms

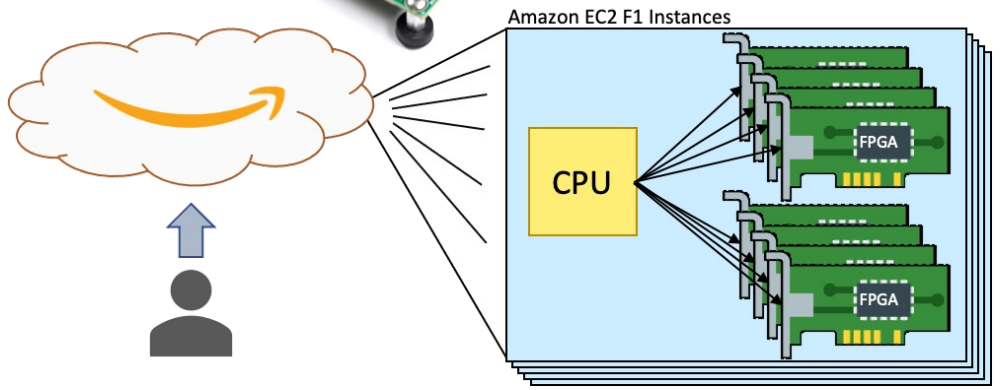
- FPGA as prototype and test platform
  - Provides control over clocking
  - Logic programming enables transmitting arbitrary patterns between chiplets

- Xilinx Virtex Ultrascale+ FPGAs
  - part# xcvu9p-flgb2104-2-i
  - Chiplets organized in Super Logic Regions (SLR)
  - Interposer wires called Super Long Lines (SLL)

- In-lab testing using VCU118 kit
- AWS EC2 F1 instances in cloud to test on larger population







Overview

Measuring Delay

Design & Implementation

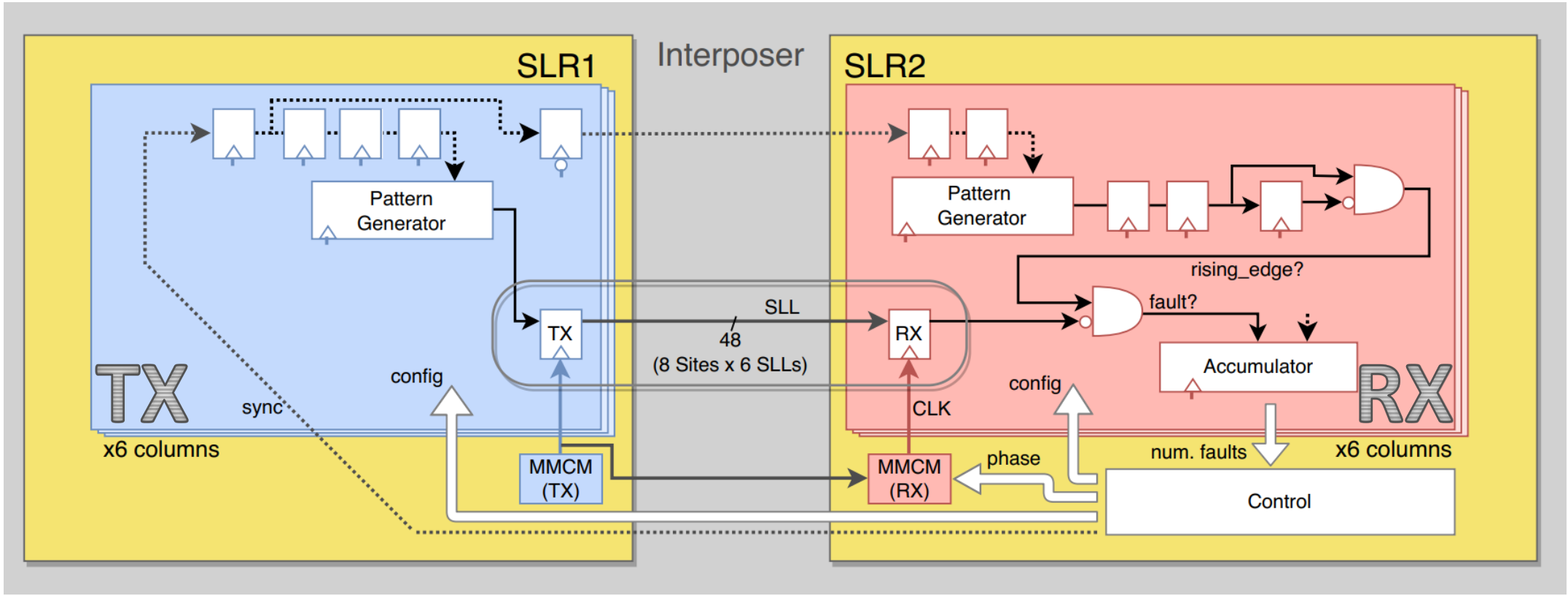
Statistics

Characterization Experiments

Summary

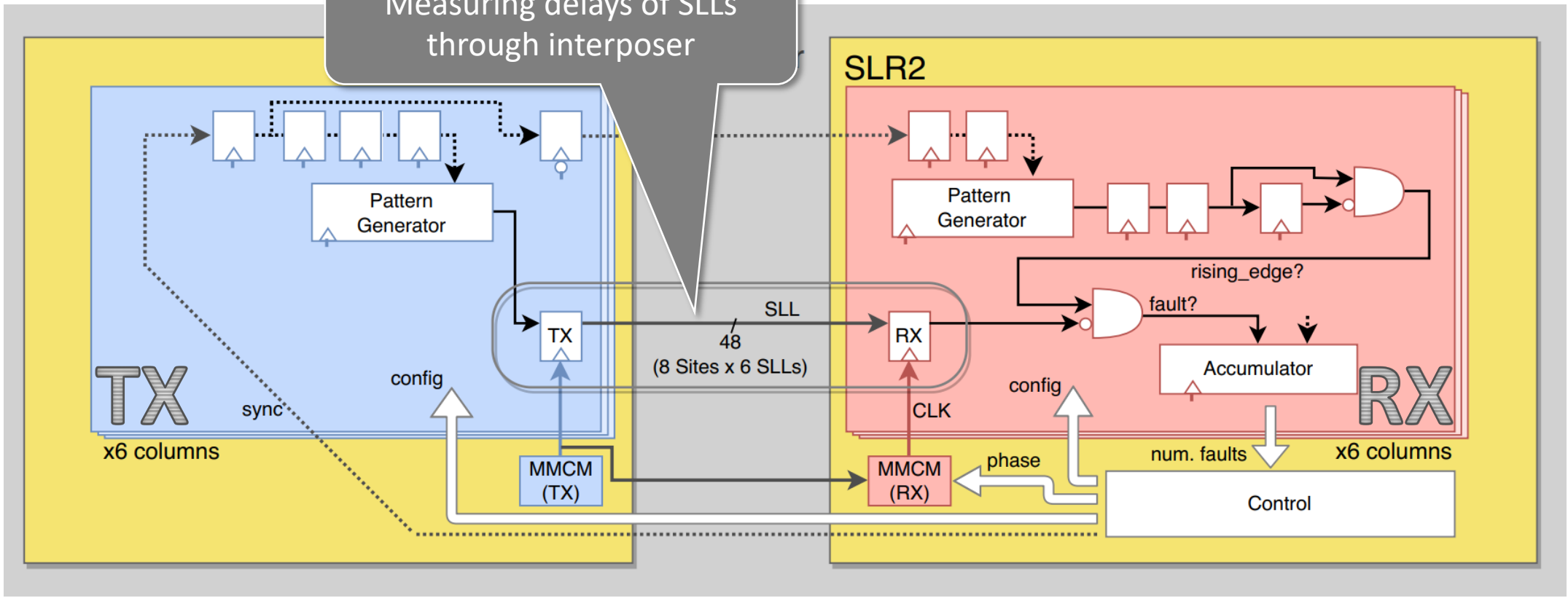
Chiplet PUF - Schematic

- Column-based design with 48 SLLs per column
- Instantiated on multiple columns
- Xilinx Mixed Mode Clock Manager (MMCM) to adjust sampling clock skew



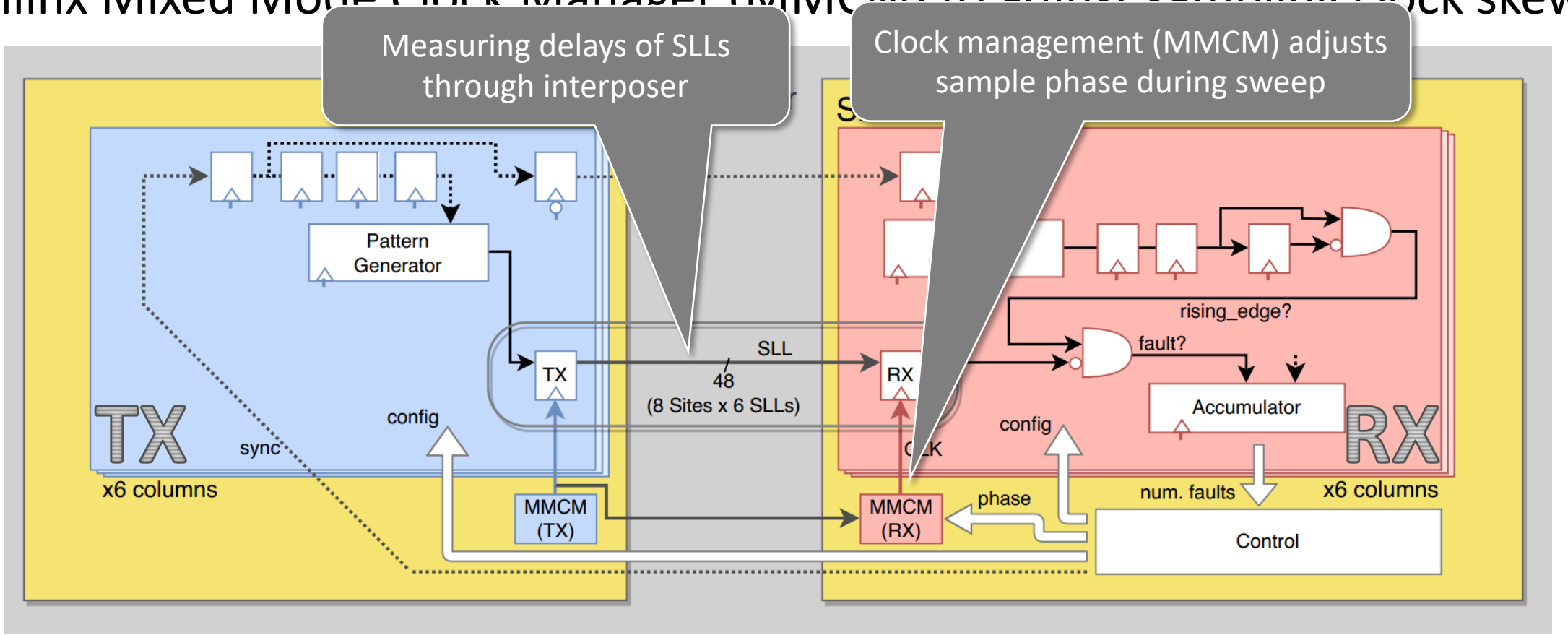
Chiplet PUF - Schematic

- Column-based design with 48 SLLs per column
- Instantiated on multiple columns
- Xilinx Mixed Mode Clock Manager (MMCM) to adjust sampling clock skew



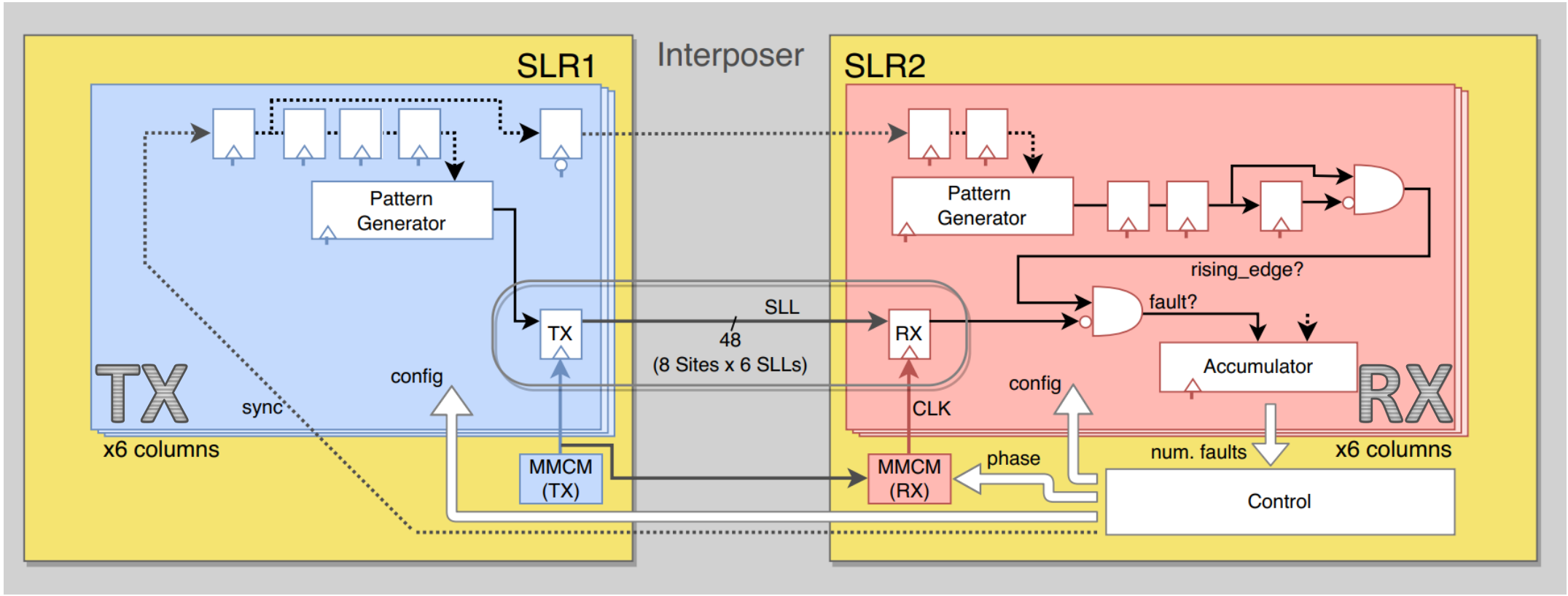
Chiplet PUF - Schematic

- Column-based design with 48 SLLs per column
- Instantiated on multiple columns
- Xilinx Mixed Mode Clock Manager (MMCM) to adjust sampling clock skew



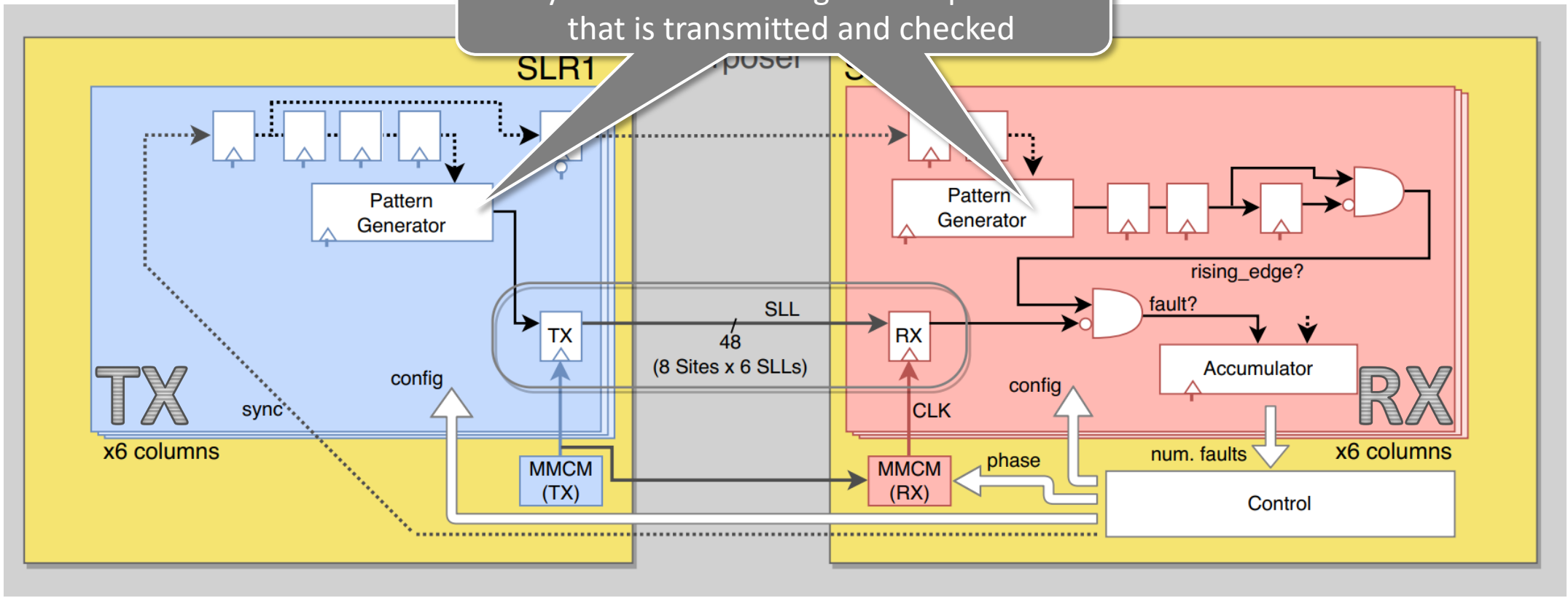
Chiplet PUF - Schematic

- Column-based design with 48 SLLs per column
- Instantiated on multiple columns
- Xilinx Mixed Mode Clock Manager (MMCM) to adjust sampling clock skew



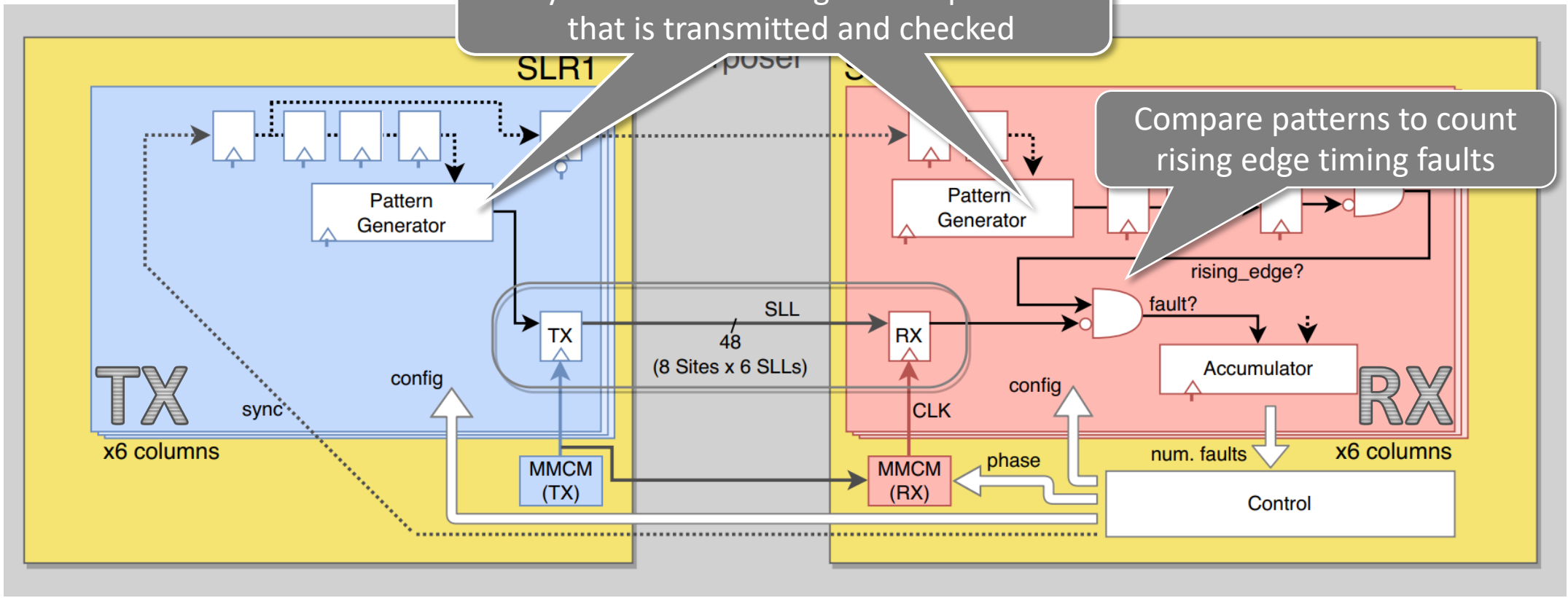
Chiplet PUF - Schematic

- Column-based design with 48 SLLs per column
- Instantiated on multiple columns
- Xilinx Mixed Mode Clock Manager (MMCM) to adjust sampling clock skew



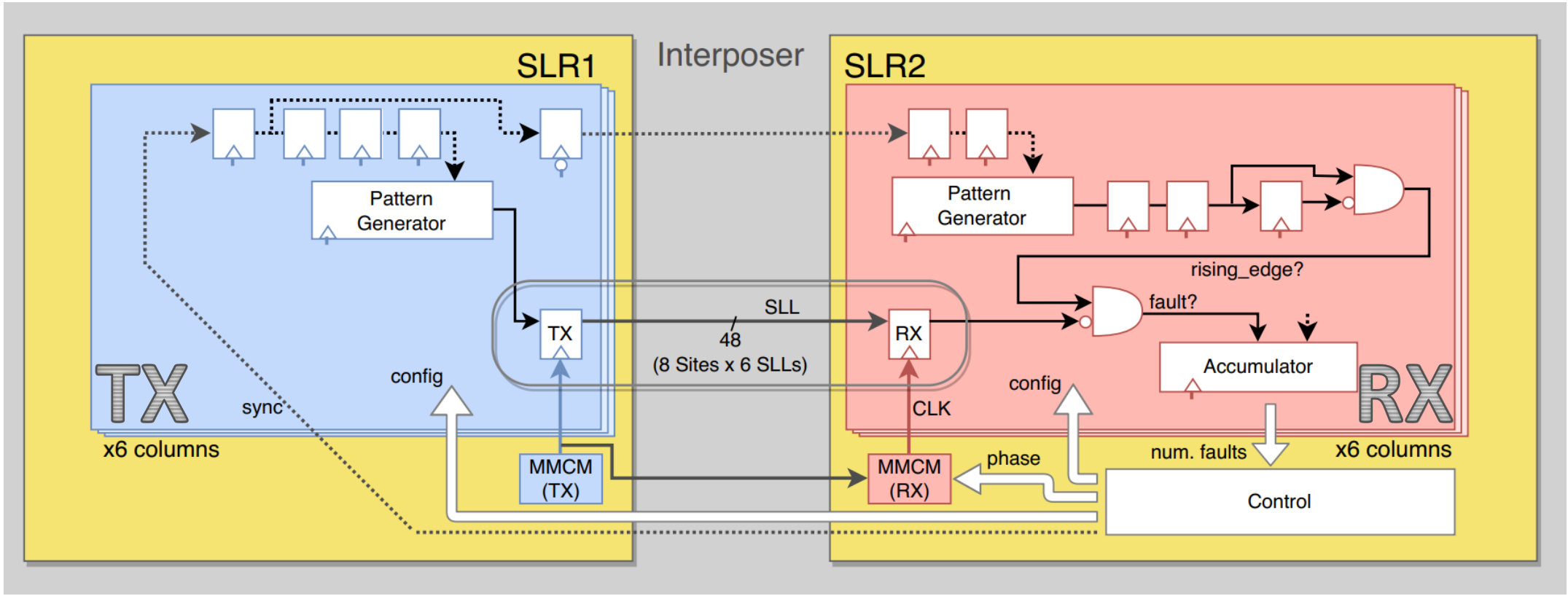
Chiplet PUF - Schematic

- Column-based design with 48 SLLs per column
- Instantiated on multiple columns
- Xilinx Mixed Mode Clock Manager (MMCM) to adjust sampling clock skew



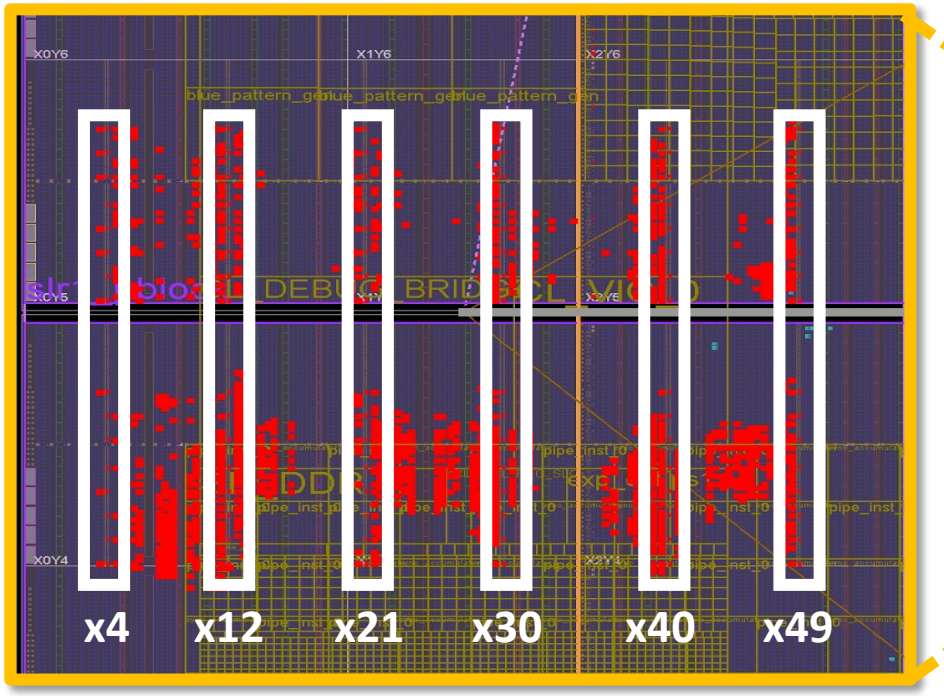
Chiplet PUF - Schematic

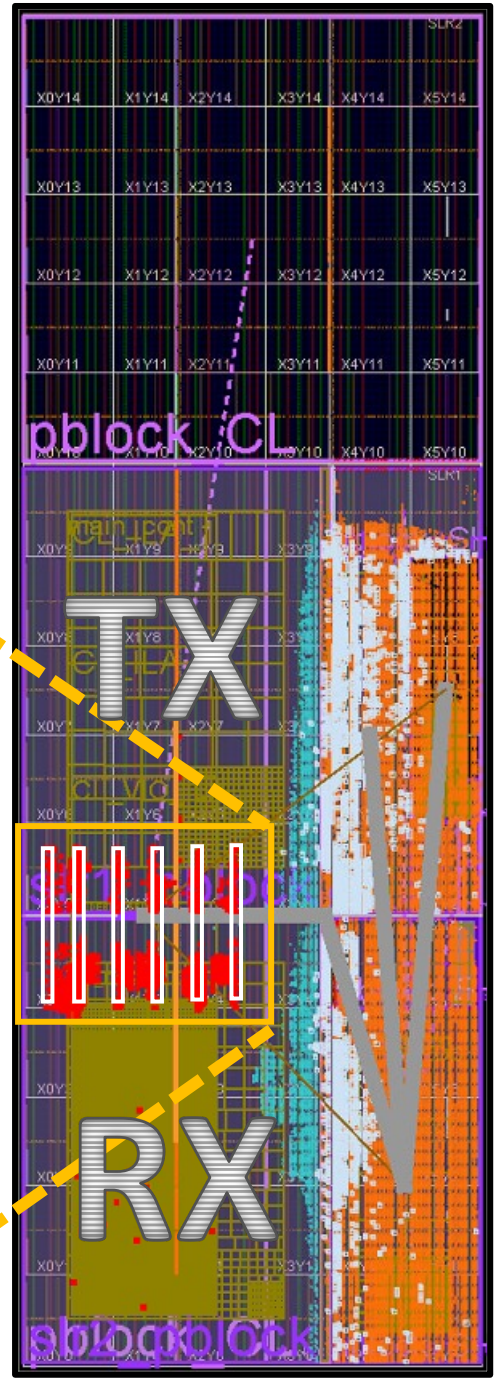
- Column-based design with 48 SLLs per column
- Instantiated on multiple columns
- Xilinx Mixed Mode Clock Manager (MMCM) to adjust sampling clock skew



Chiplet PUF - Implementation

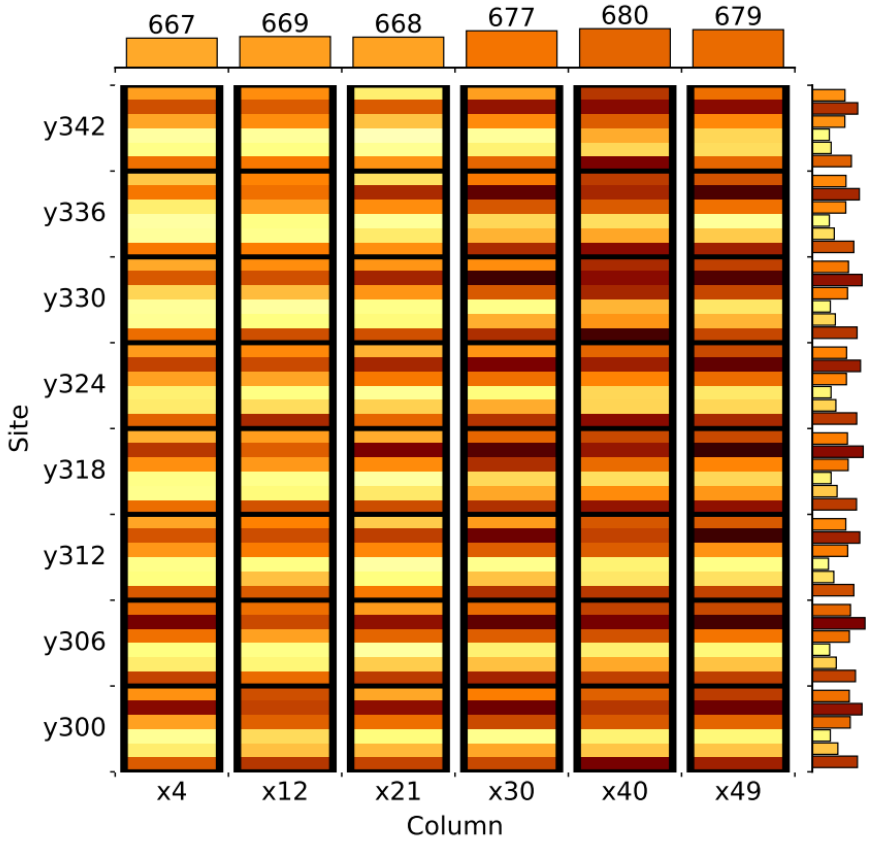
- 6 columns instantiated (288 SLLs)
- Using <2% of the 17,280 SLLs between the chiplets
- 0.27% LUT and 0.34% FF utilization

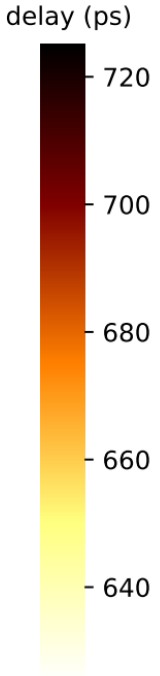


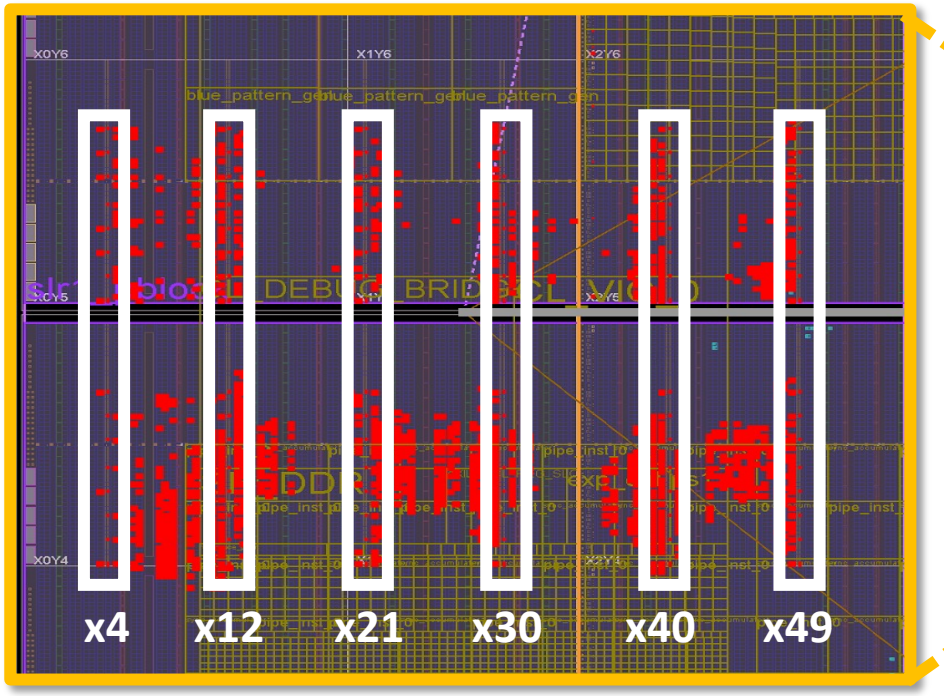


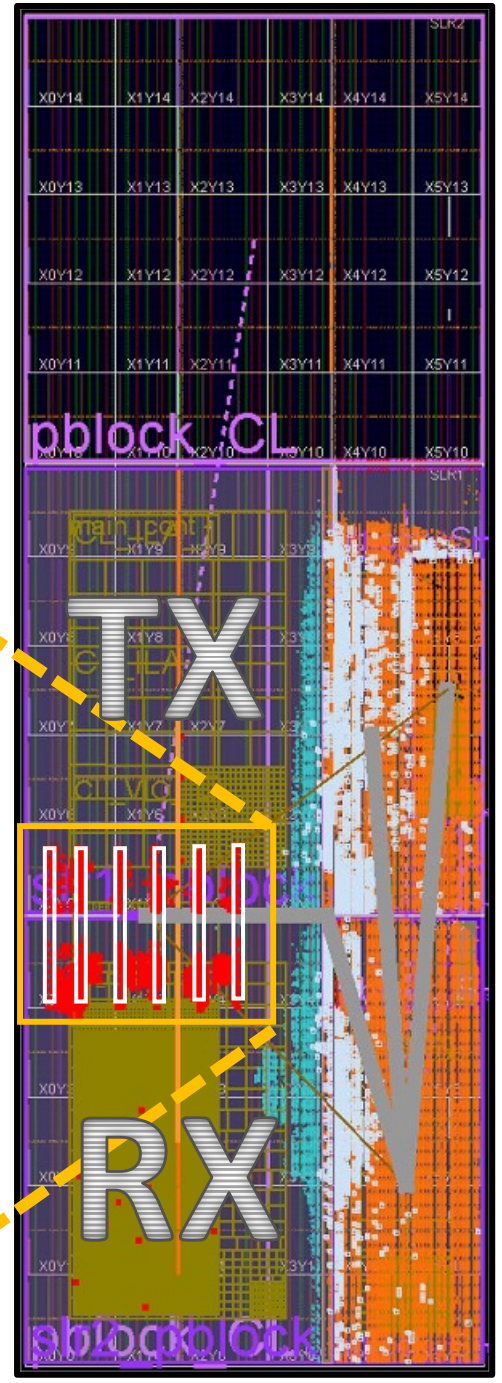
Chiplet PUF - Implementation

- 6 columns instantiated (288 SLLs)
- Using <2% of the 17,280 SLLs between the chiplets
- 0.27% LUT and 0.34% FF utilization



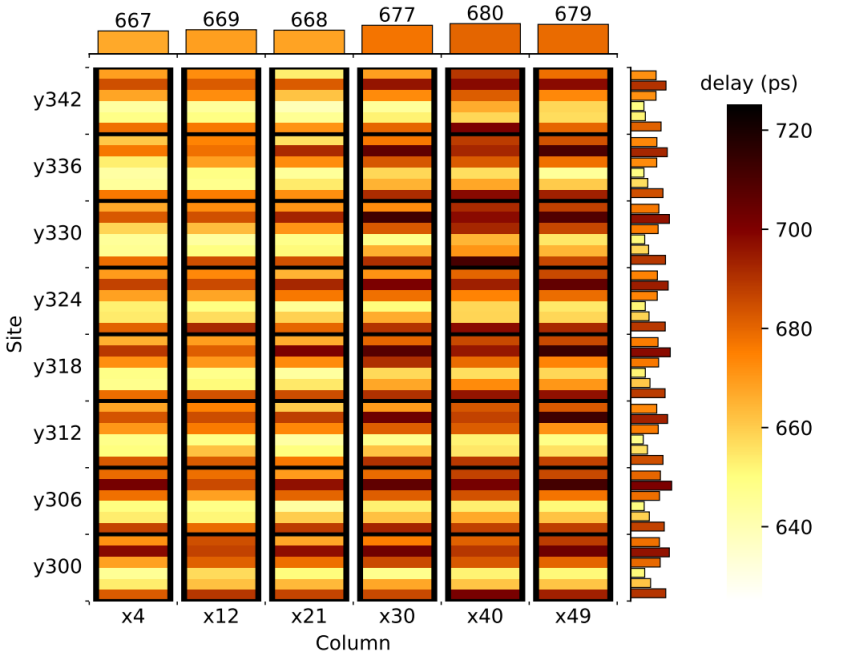


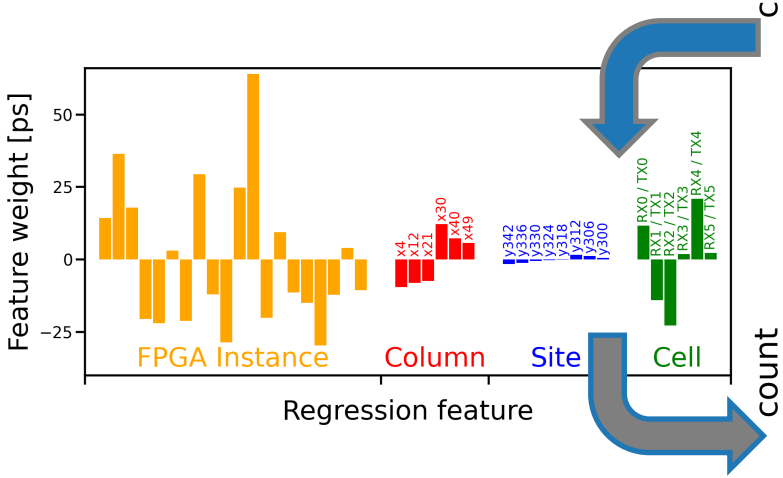


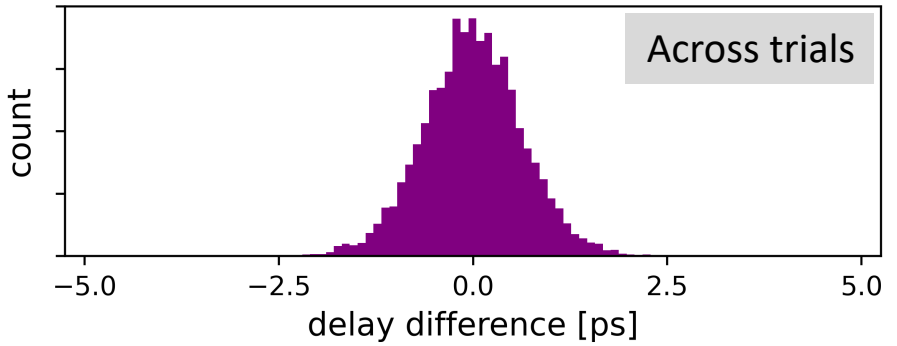


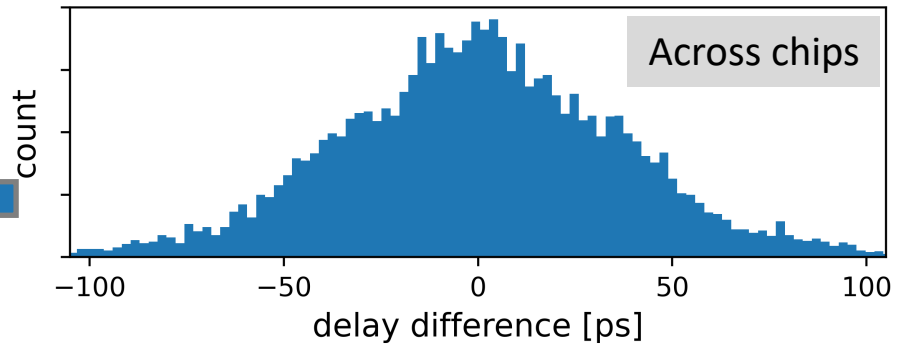
Measured SLL Delays

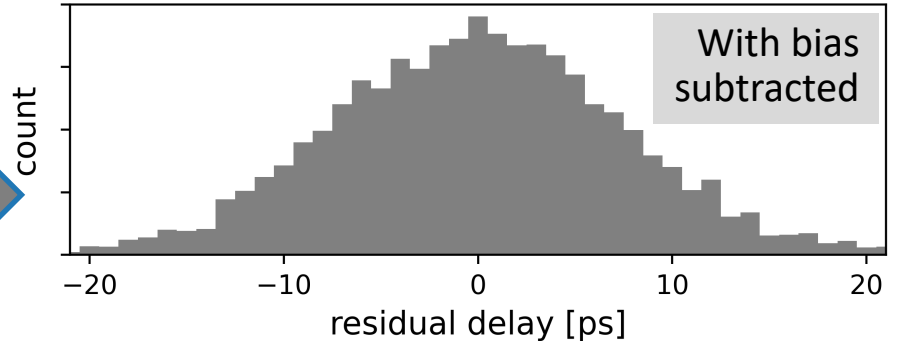
- SLL delay measurements in 630-720 ps range
- Reliable and instance-specific
  - 0.5 ps difference across trials
  - 29.4 ps difference across chips
  - 5.8 ps difference after removing biases





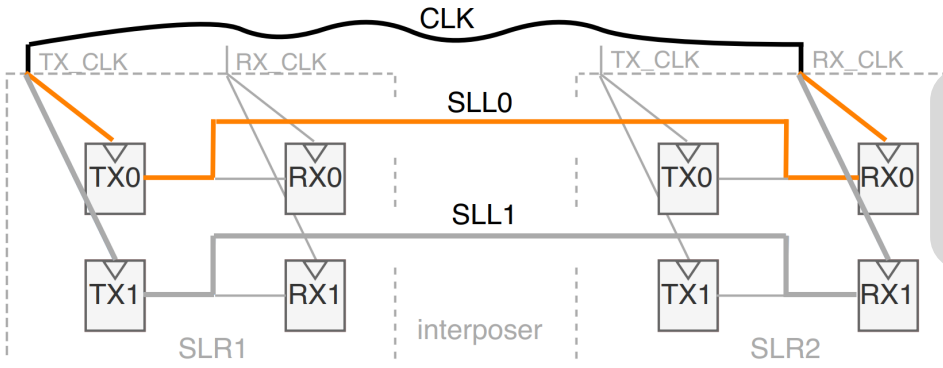




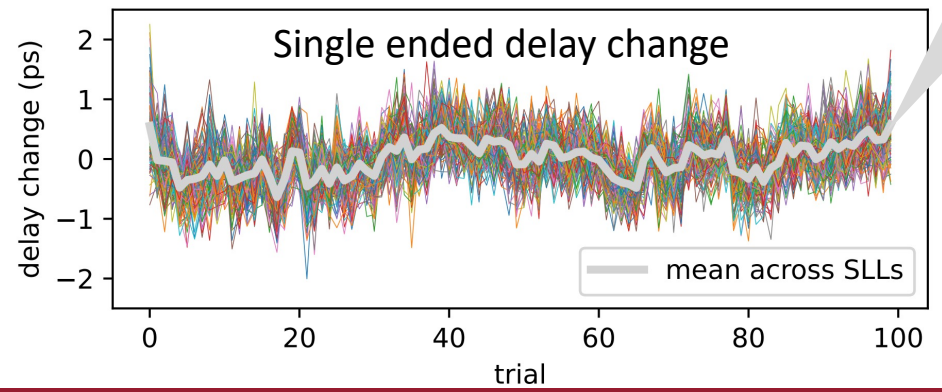


Robust Delay Measurement

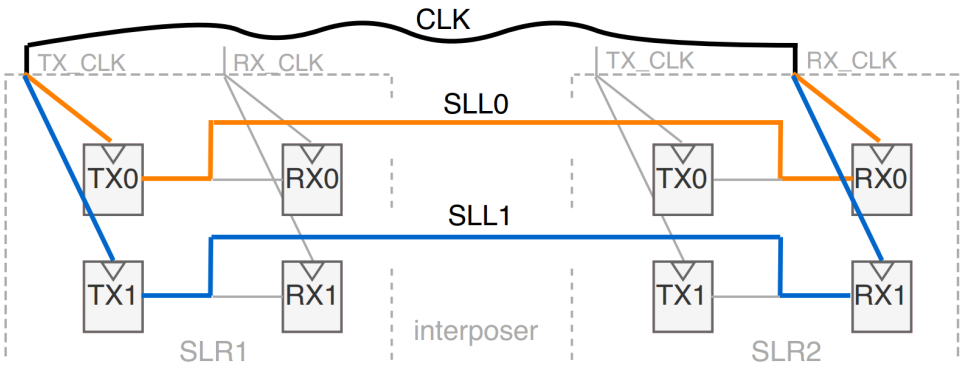
- Differential delays between SLL pairs outperform single-ended SLL delays
- Delay measurement becomes independent of clock path
  - Less delay drift because clock changes become common mode
  - Clock path reused across SLLs → Avoid miscounting skew variation as uniqueness

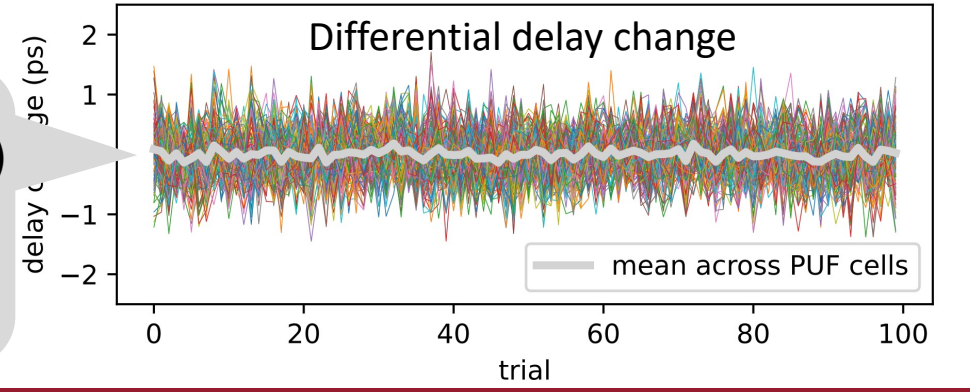


Delay = SLL0 – clk
Paths impacted differently by env.



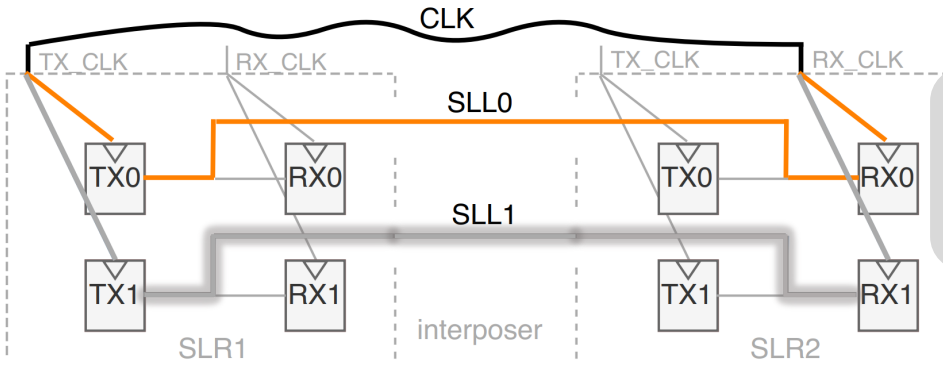
Delay = (SLL0 - clk)
– (SLL1 - clk)
= SLL0 - SLL1
Impacted similarly by env. changes



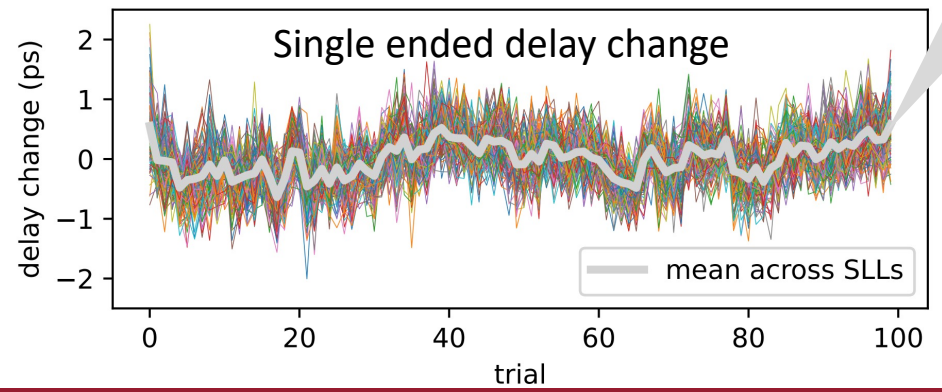


Robust Delay Measurement

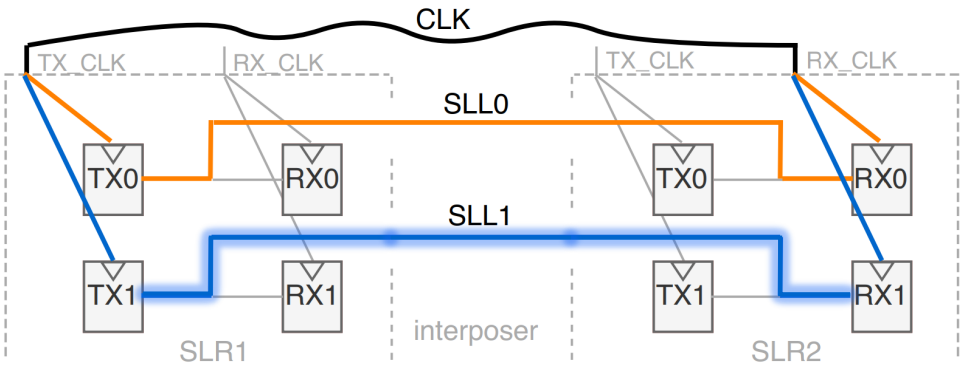
- Differential delays between SLL pairs outperform single-ended SLL delays
- Delay measurement becomes independent of clock path
  - Less delay drift because clock changes become common mode
  - Clock path reused across SLLs → Avoid miscounting skew variation as uniqueness

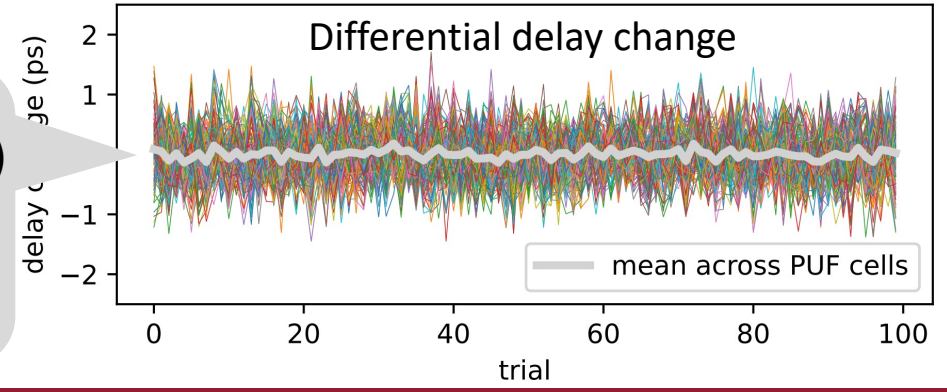


Delay = SLL0 – clk
Paths impacted differently by env.



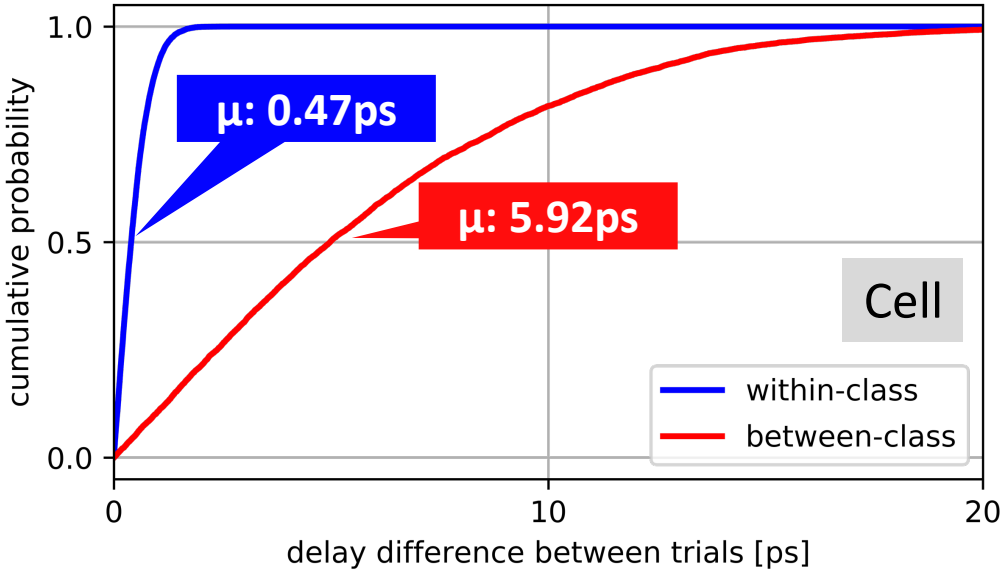
Delay = (SLL0 - clk)
– (SLL1 - clk)
= SLL0 - SLL1
Impacted similarly by env. changes

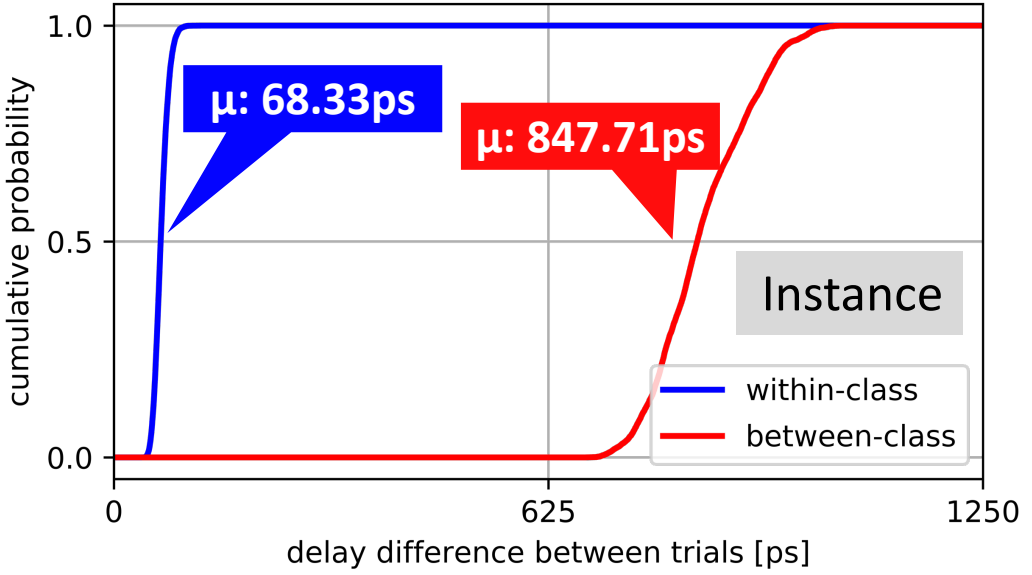




Within and Between-Class Distances

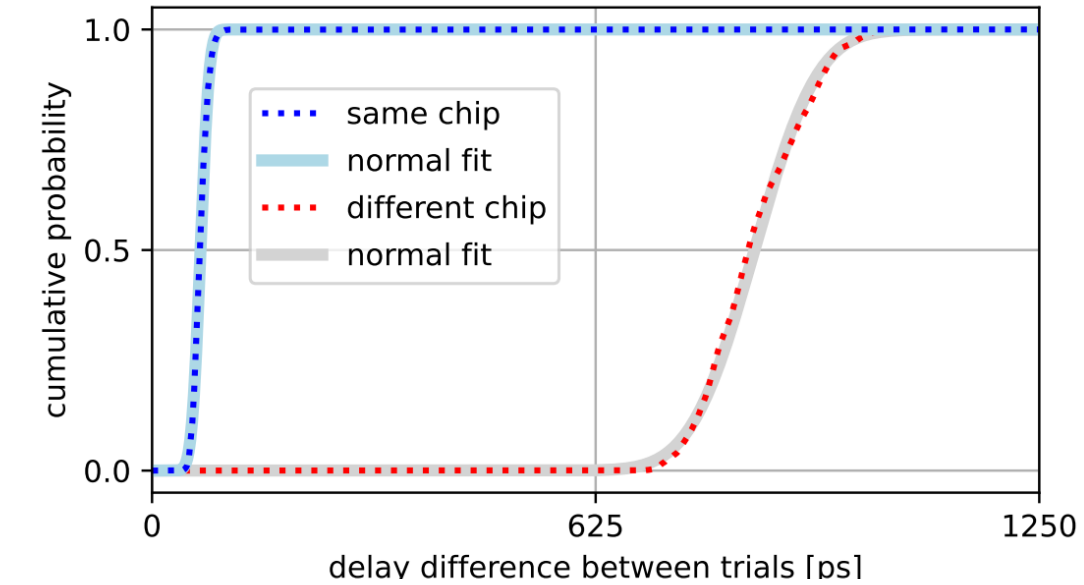
- Data from 20 AWS EC2 F1 instances
  - Same VU9P part used for local testing
- Cumulative distributions
  - PUF cell difference = $\left|D_{t,s} - D_{t',s}\right|$
  - Instance difference = $\sum_{s=1}^{144}\left|D_{t,s} - D_{t',s}\right|$
- Separation of within-class and between-class distances is consistent with the PUF being a reliable and unique fingerprint

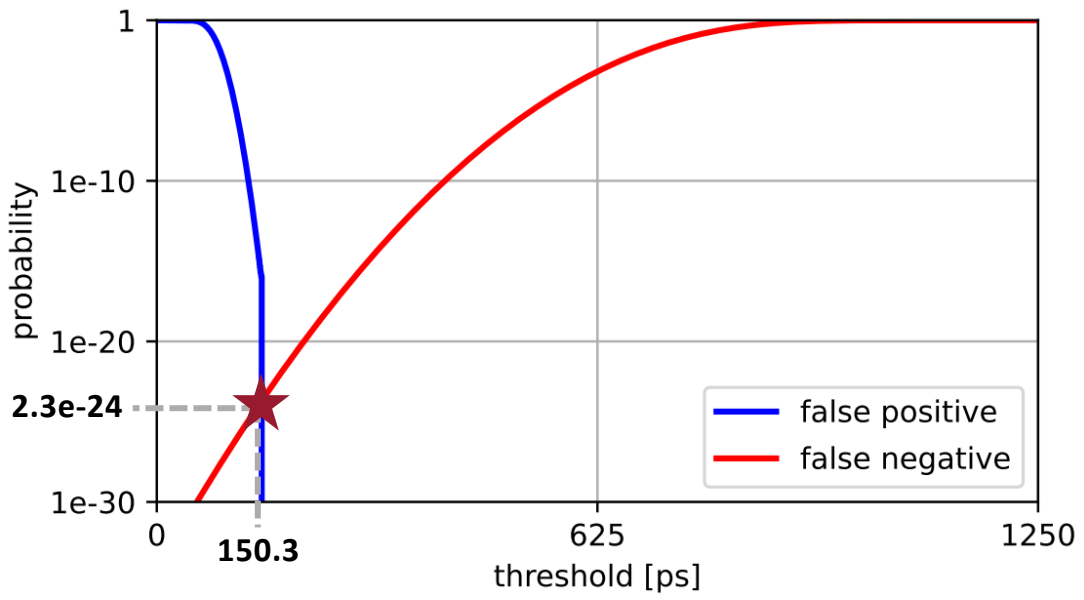




Type I and II errors

- Empirical data approximately normally distributed
- Fitted normal distributions used to estimate false positive and false negative rates in a larger population
- Equal error point occurs at threshold = 150.3 ps
- Type I and II error rates are 2.3e-24





Overview

Measuring Delay

Design & Implementation

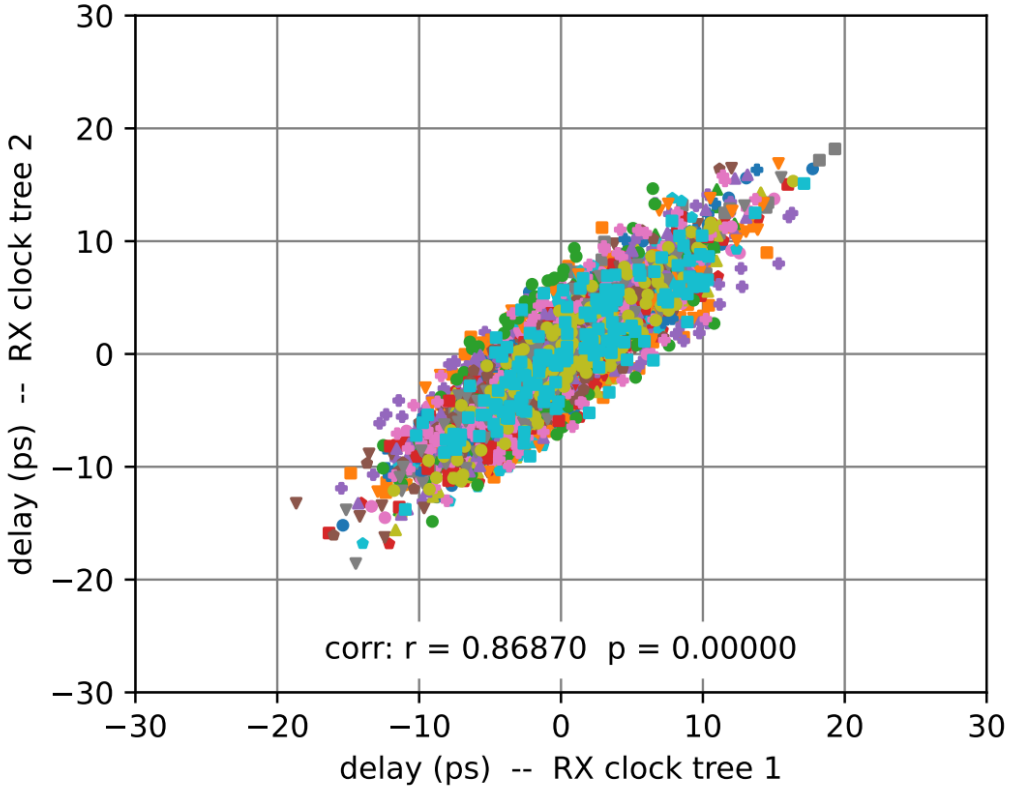
Statistics

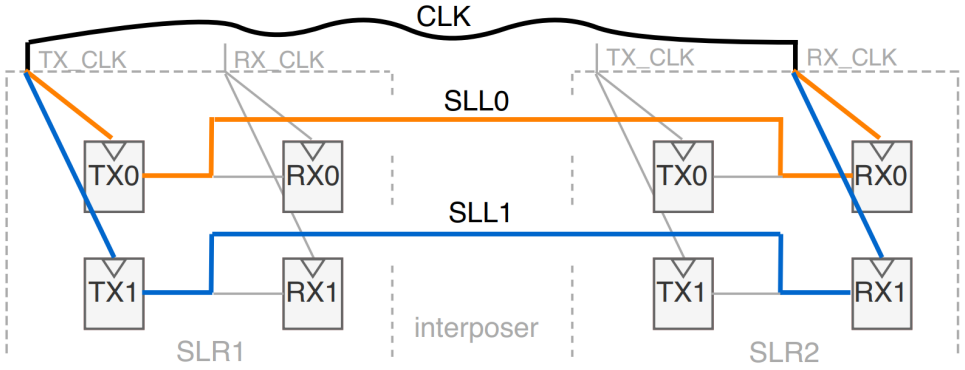
Characterization Experiments

Summary

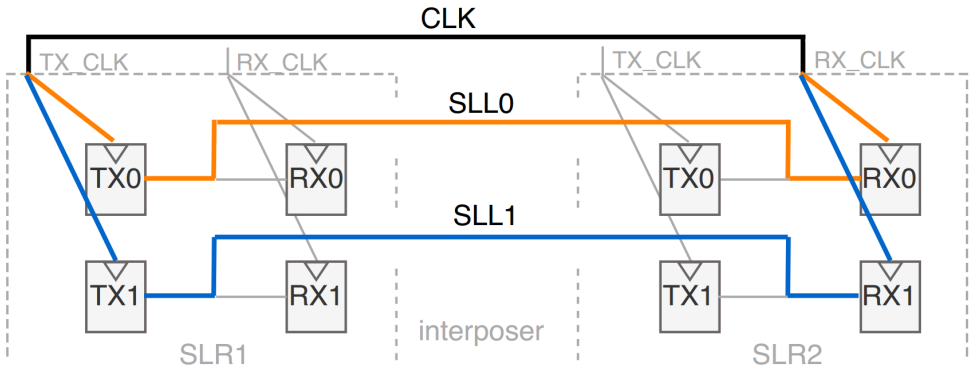
Characterization – Using different clock trees

- Testing whether differential PUF output is insensitive to clock
  - Crucial for minimizing impact of environmental fluctuations and of variation on clock tree
- Compare PUFs between two variants:
  - **Same** interposer wires, **same** drivers
  - **Different** clock distribution path
- Highly correlated outputs (r = 0.869) in experiments on 20 cloud instances x 144 cells
- Conclusion: PUF insensitive to clock, as intended



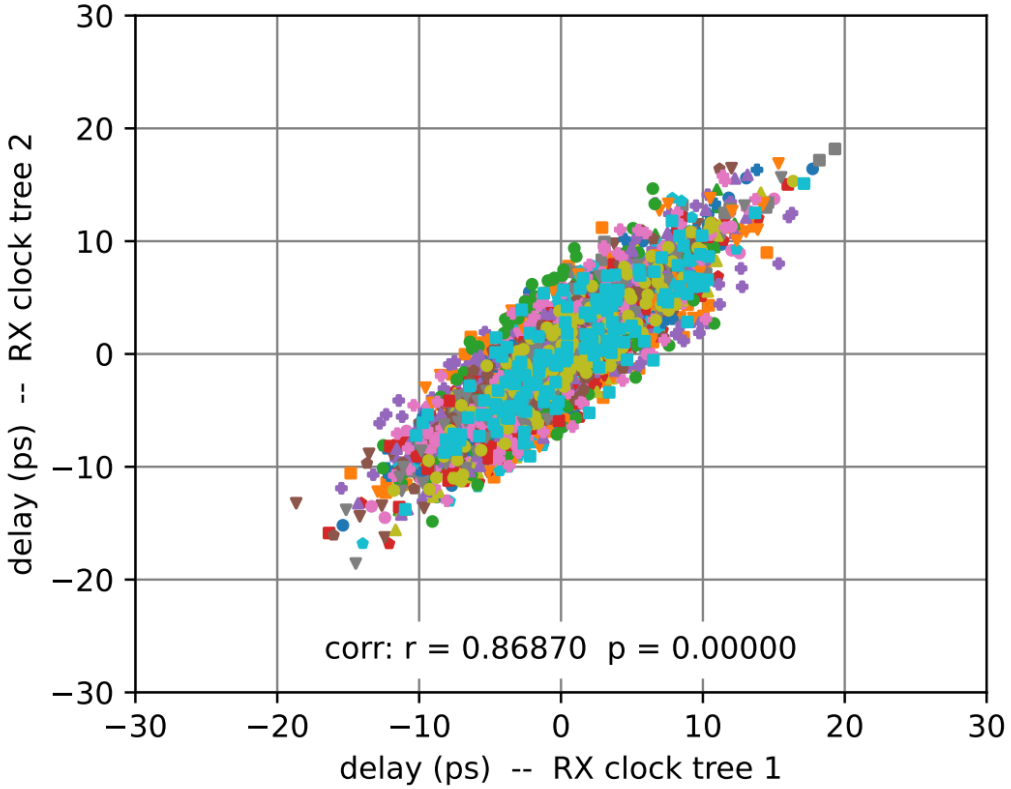


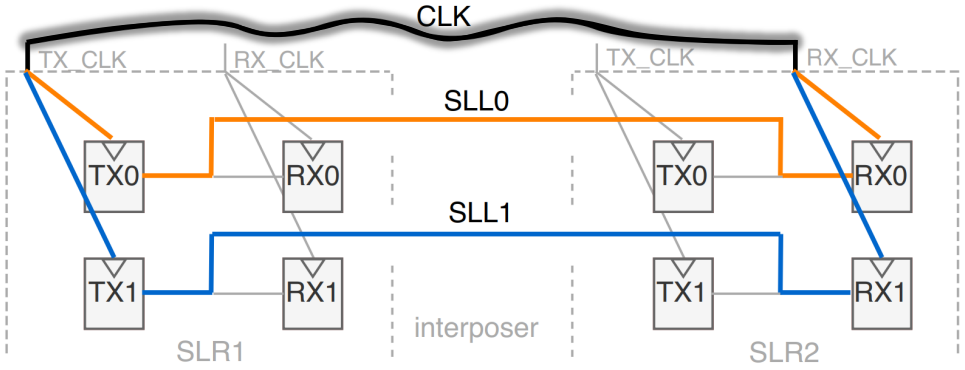
VS



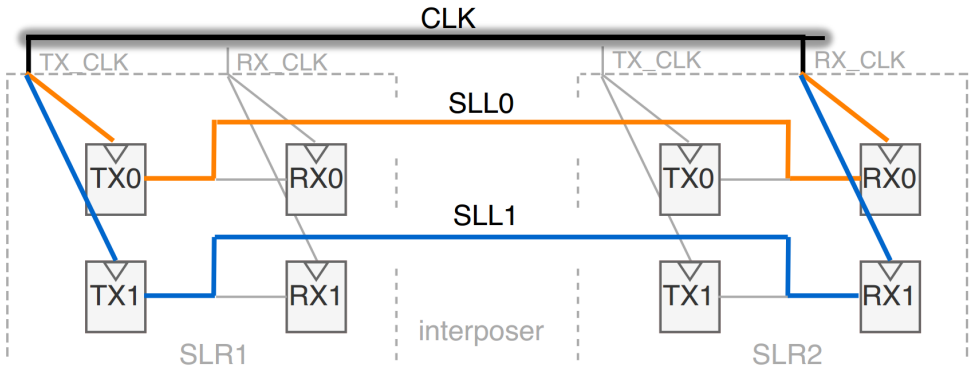
Characterization – Using different clock trees

- Testing whether differential PUF output is insensitive to clock
  - Crucial for minimizing impact of environmental fluctuations and of variation on clock tree
- Compare PUFs between two variants:
  - **Same** interposer wires, **same** drivers
  - **Different** clock distribution path
- Highly correlated outputs (r = 0.869) in experiments on 20 cloud instances x 144 cells
- Conclusion: PUF insensitive to clock, as intended



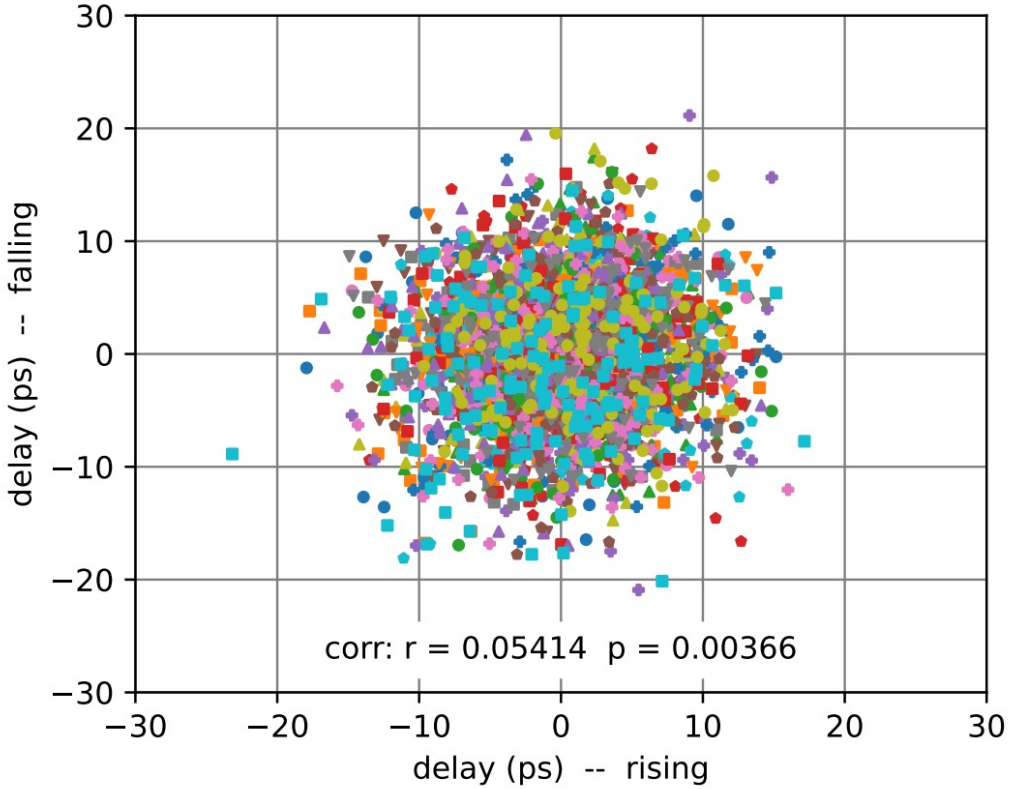


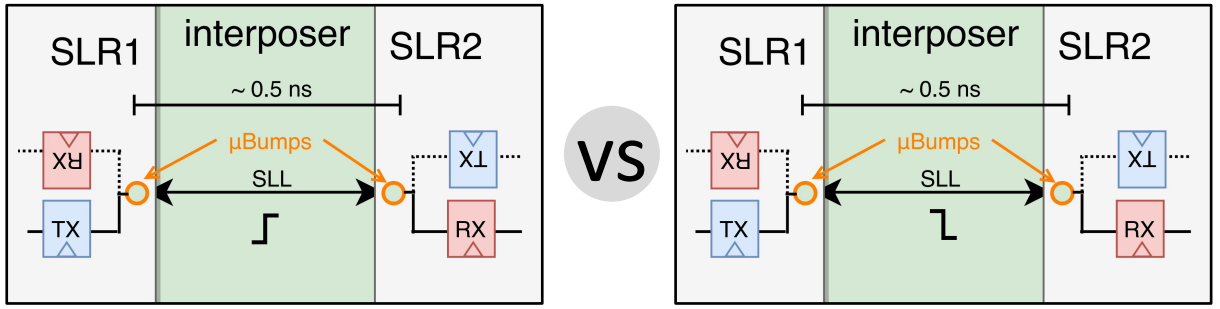
VS



Characterization – Rising vs Falling transition

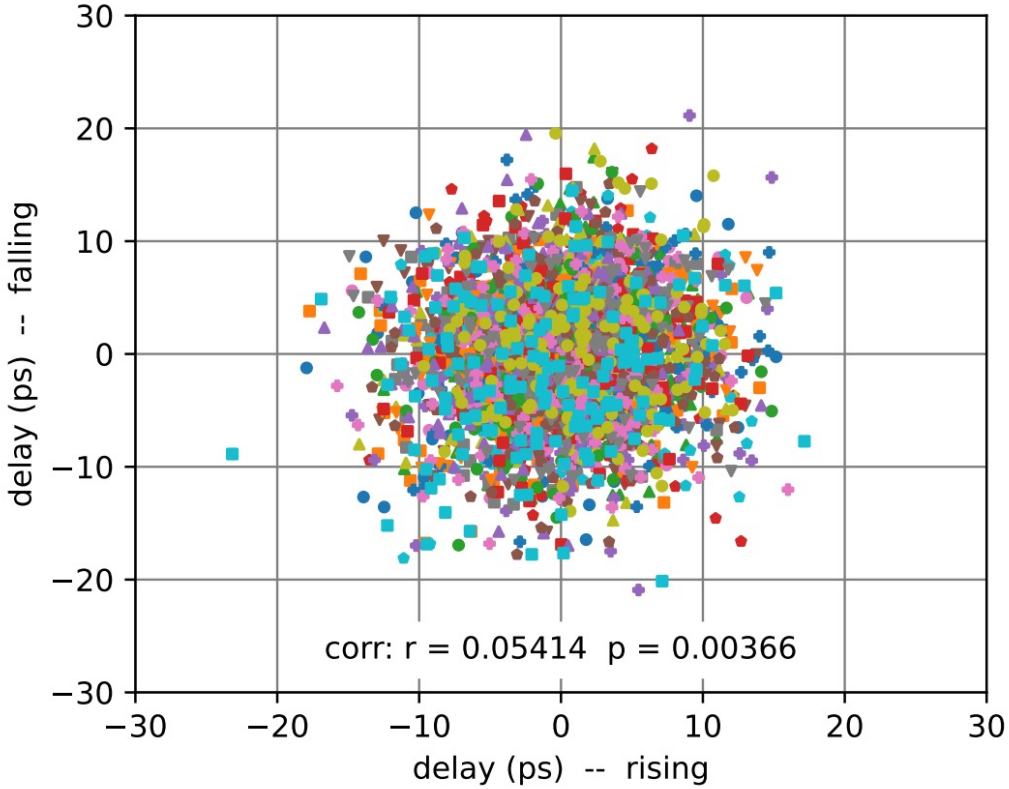
- Testing whether drivers or interposer wires dominate variability
- Compare PUF variants using rising or falling transition to measure delays:
  - **Same** interposer wires in both cases
  - **Different** transistors driving wires
  - **Different** transistors in sampling flops
- Weaker correlation (r=0.054) implies that variation of interposer wires is not dominant factor
- Conclusion: Transistor variation is a significant source of entropy

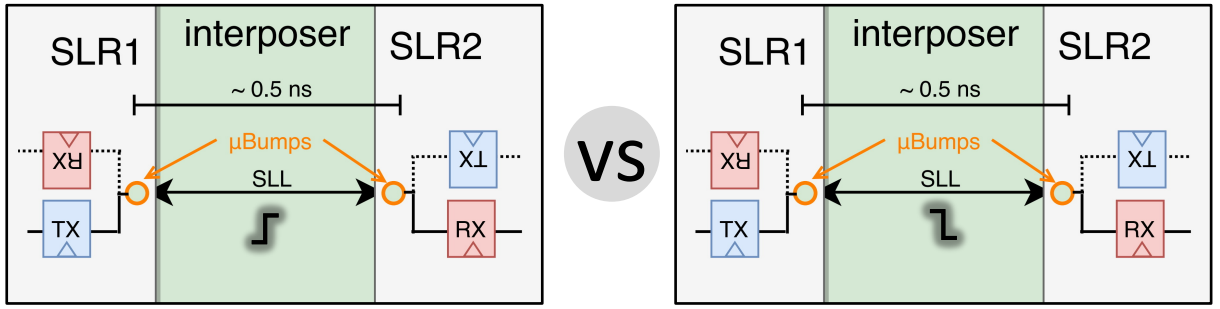




Characterization – Rising vs Falling transition

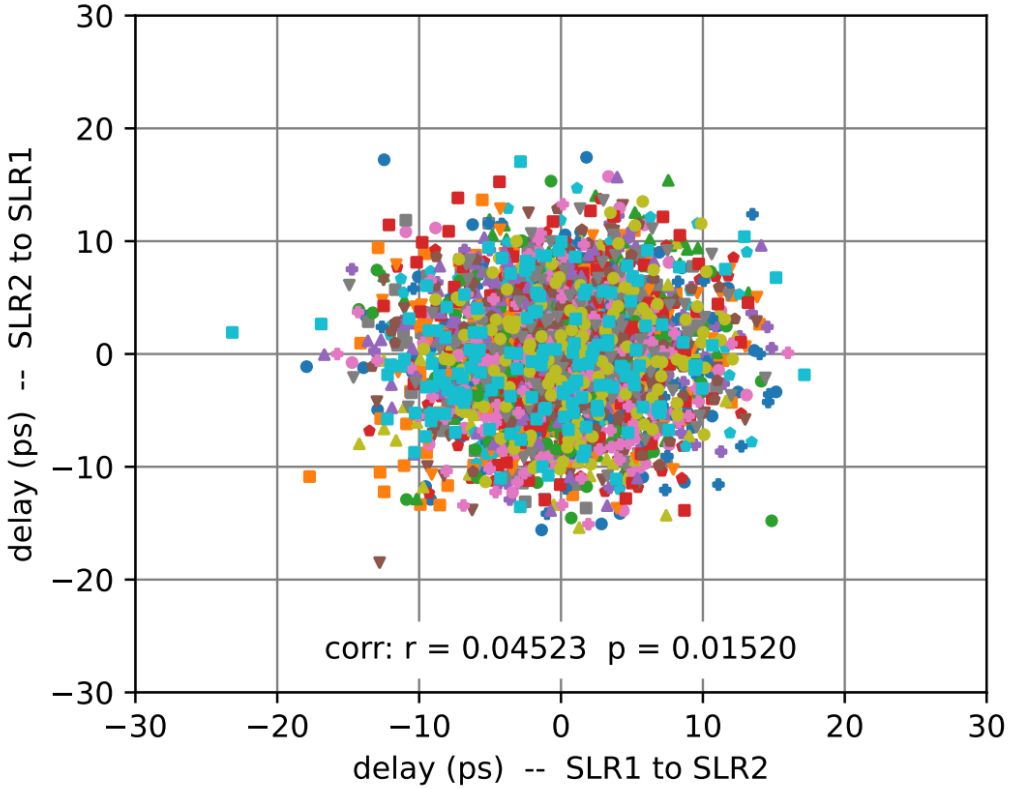
- Testing whether drivers or interposer wires dominate variability
- Compare PUF variants using rising or falling transition to measure delays:
  - **Same** interposer wires in both cases
  - **Different** transistors driving wires
  - **Different** transistors in sampling flops
- Weaker correlation (r=0.054) implies that variation of interposer wires is not dominant factor
- Conclusion: Transistor variation is a significant source of entropy

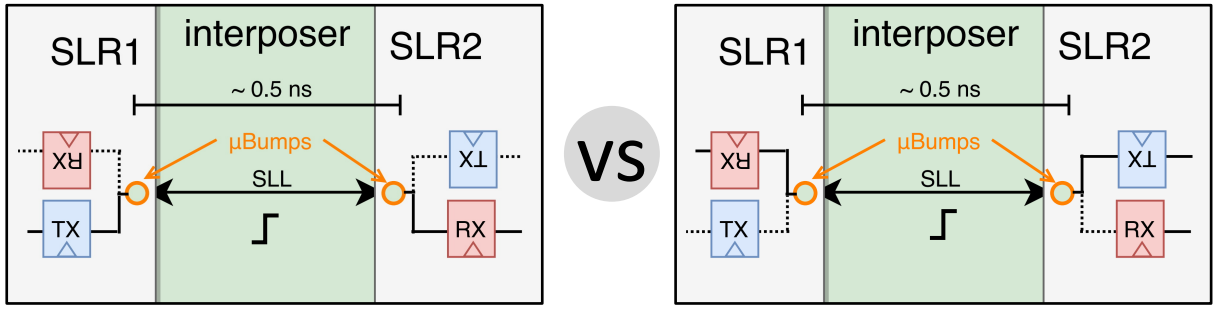




Characterization – Swapping TX and RX

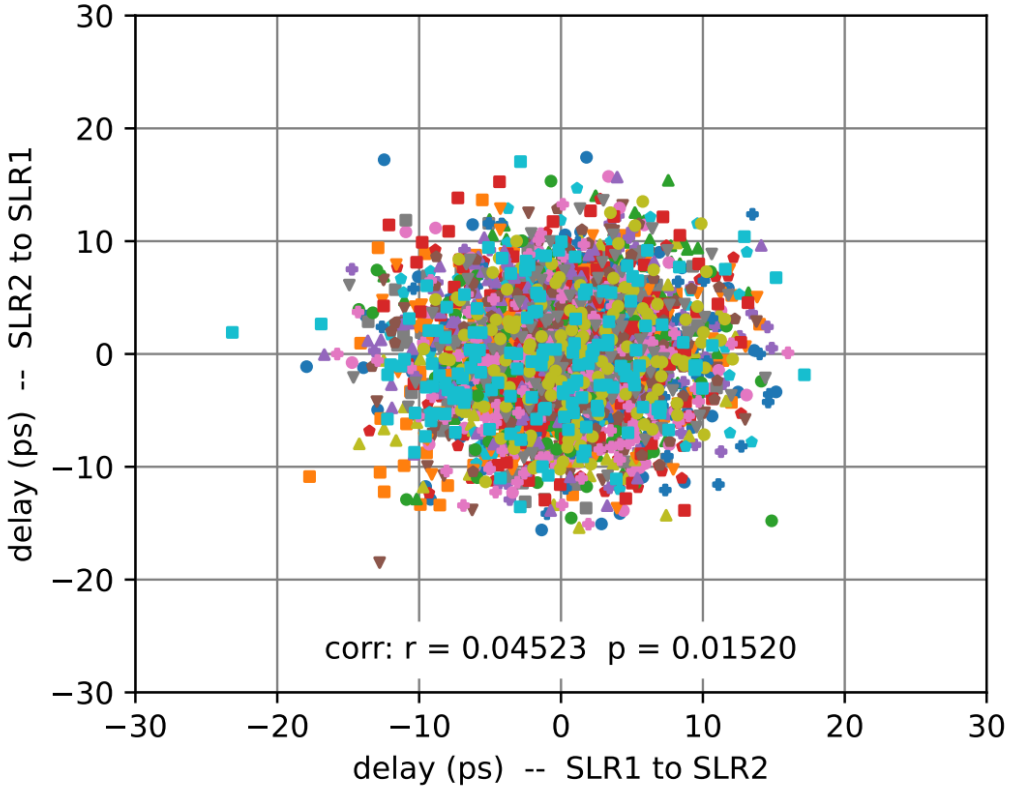
- Testing impact of driving same wire from each end
  - Possible in Xilinx architecture because SLLs are bidirectional
- Comparing two variants with:
  - **Same** interposer wires
  - **Different** transistor instances
  - **Different** environment for TX and RX
- Weak correlation (r=0.045) again implies that variation of interposer wires is not dominant factor
- Conclusion: Transistor variation is a significant source of entropy

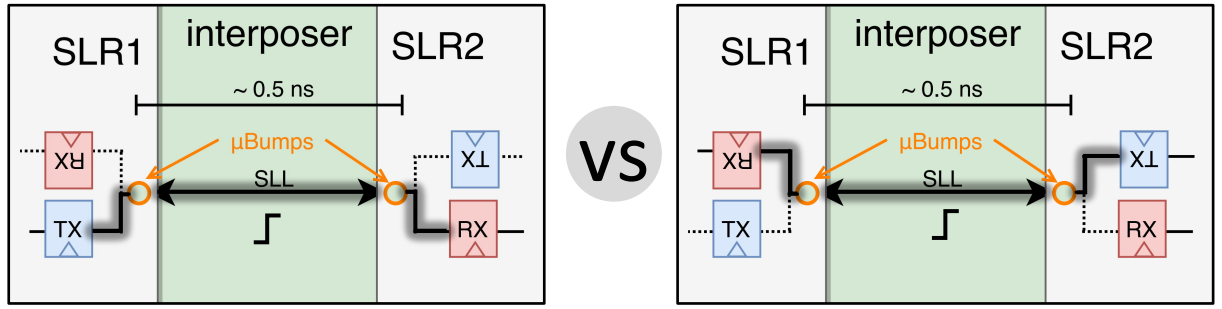




Characterization – Swapping TX and RX

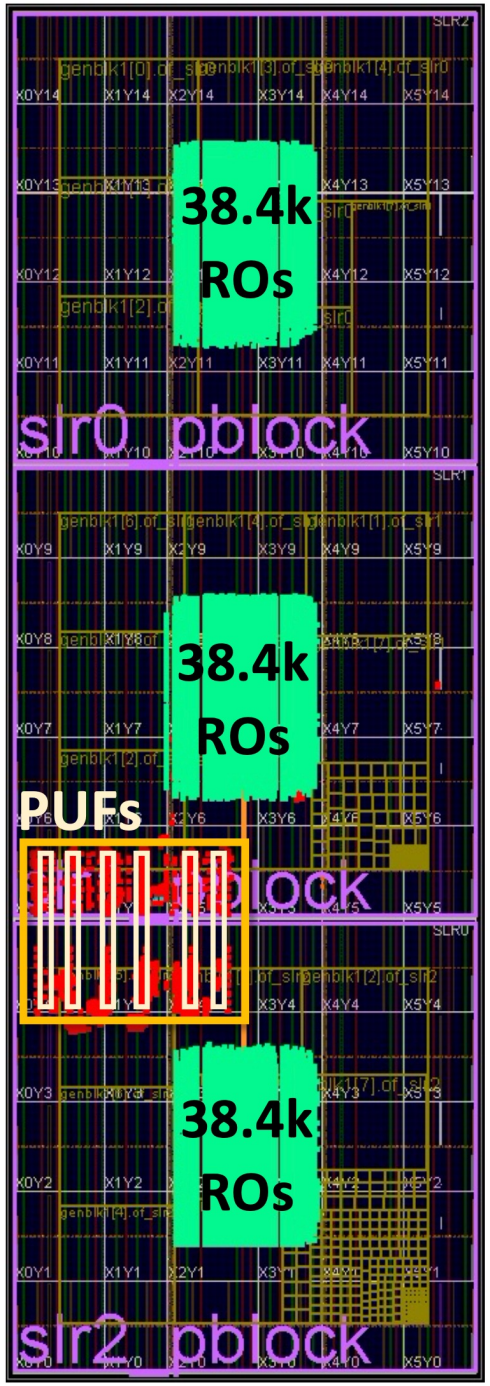
- Testing impact of driving same wire from each end
  - Possible in Xilinx architecture because SLLs are bidirectional
- Comparing two variants with:
  - **Same** interposer wires
  - **Different** transistor instances
  - **Different** environment for TX and RX
- Weak correlation (r=0.045) again implies that variation of interposer wires is not dominant factor
- Conclusion: Transistor variation is a significant source of entropy





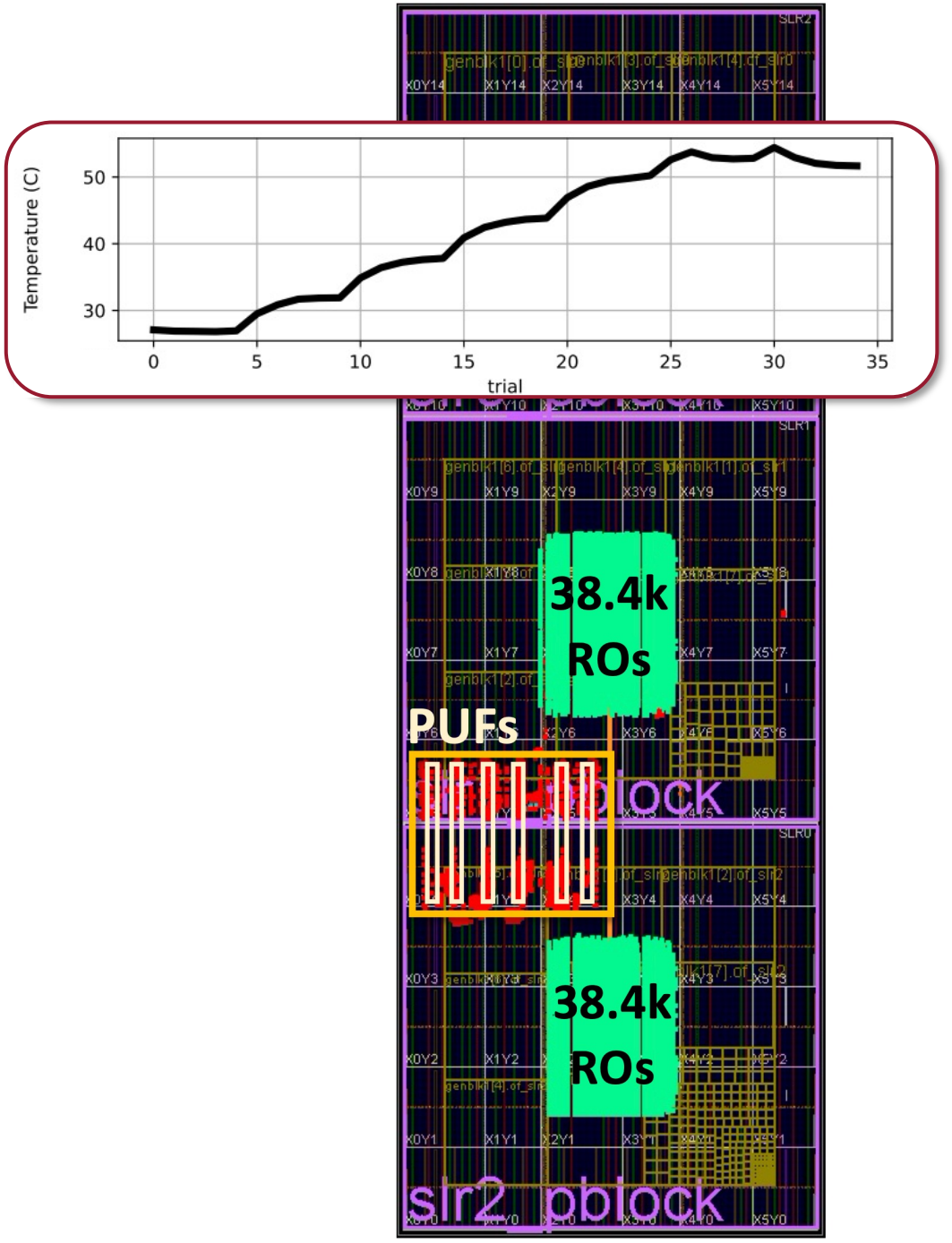
Heating & Compensation

- 38.4k power wasting ring oscillators (ROs) added to each SLR
  - Controlled in groups of 4.8k
- SLL delays increase proportional to die temperature
  - Sensitivity is non-uniform
  - Causes error in output of differential PUF cells
- Compensate delay by learning and applying per-SLL delay coefficient
  - Does not use temperature sensor



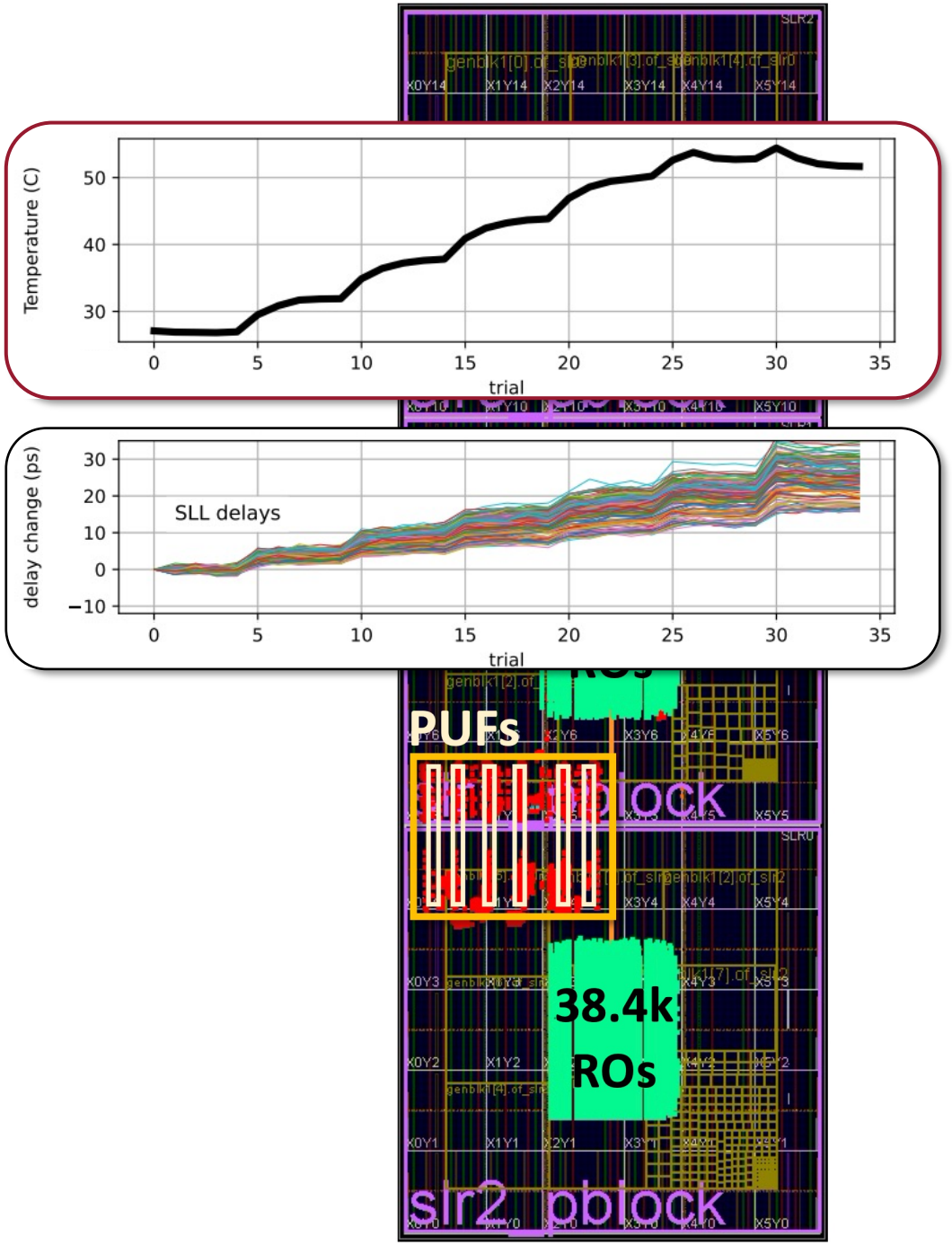
Heating & Compensation

- 38.4k power wasting ring oscillators (ROs) added to each SLR
  - Controlled in groups of 4.8k
- SLL delays increase proportional to die temperature
  - Sensitivity is non-uniform
  - Causes error in output of differential PUF cells
- Compensate delay by learning and applying per-SLL delay coefficient
  - Does not use temperature sensor



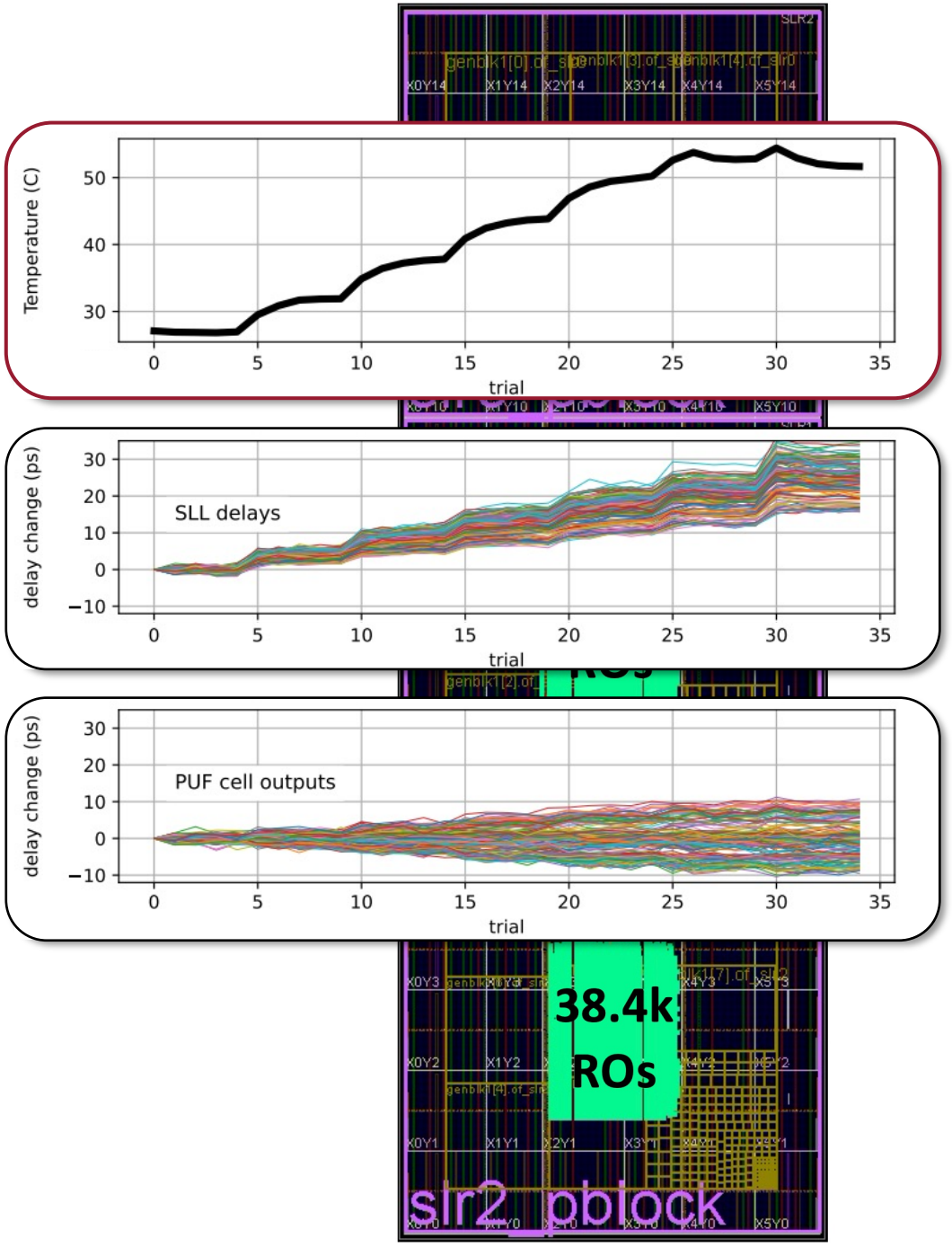
Heating & Compensation

- 38.4k power wasting ring oscillators (ROs) added to each SLR
  - Controlled in groups of 4.8k
- SLL delays increase proportional to die temperature
  - Sensitivity is non-uniform
  - Causes error in output of differential PUF cells
- Compensate delay by learning and applying per-SLL delay coefficient
  - Does not use temperature sensor



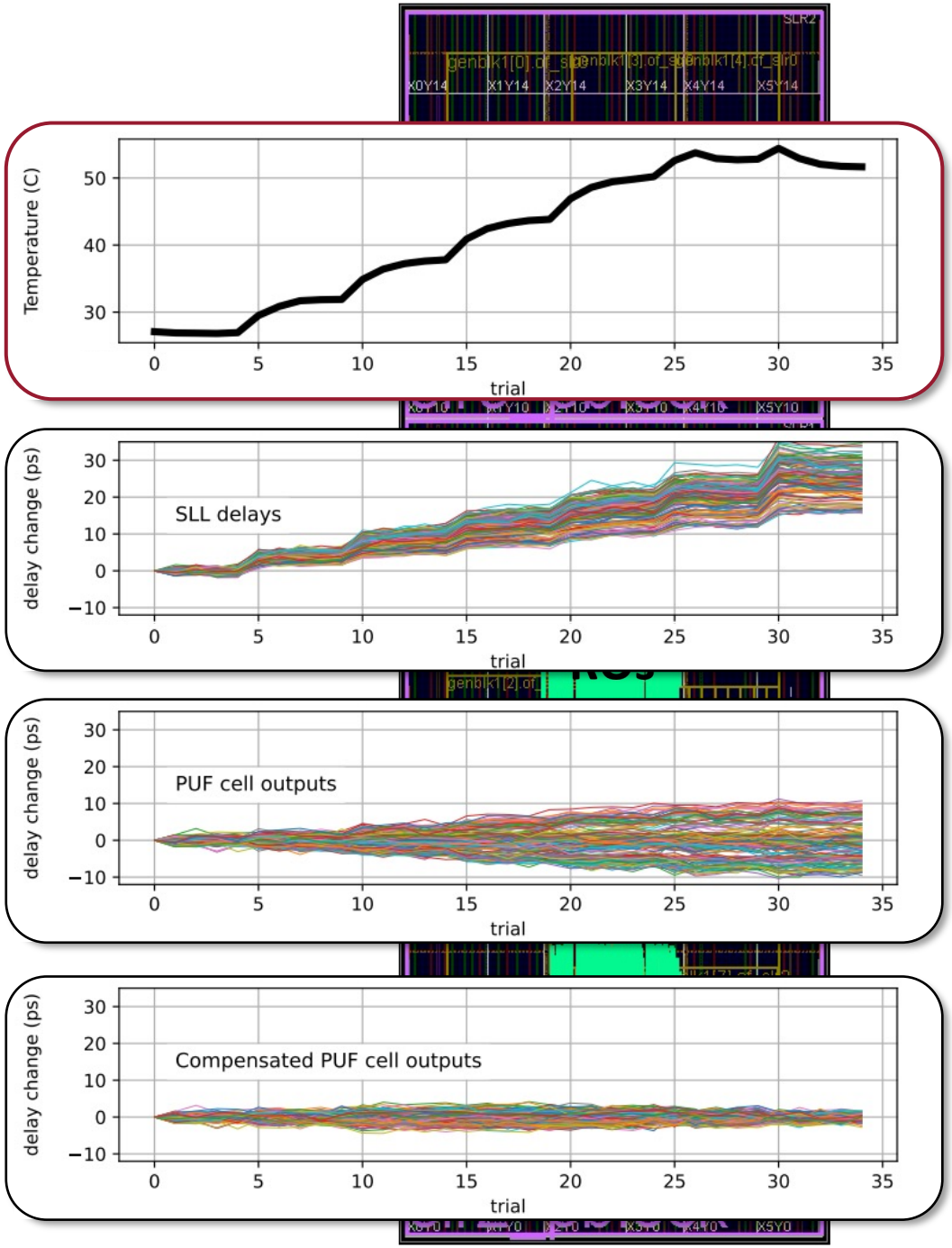
Heating & Compensation

- 38.4k power wasting ring oscillators (ROs) added to each SLR
  - Controlled in groups of 4.8k
- SLL delays increase proportional to die temperature
  - Sensitivity is non-uniform
  - Causes error in output of differential PUF cells
- Compensate delay by learning and applying per-SLL delay coefficient
  - Does not use temperature sensor



Heating & Compensation

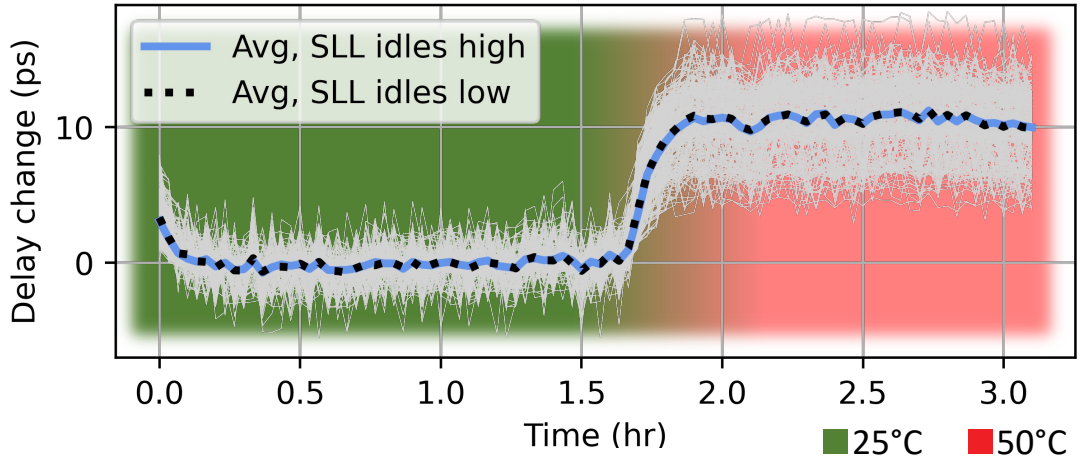
- 38.4k power wasting ring oscillators (ROs) added to each SLR
  - Controlled in groups of 4.8k
- SLL delays increase proportional to die temperature
  - Sensitivity is non-uniform
  - Causes error in output of differential PUF cells
- Compensate delay by learning and applying per-SLL delay coefficient
  - Does not use temperature sensor



Testing for Impact of Aging

- Aging can change circuit delay
- Potentially detrimental to PUF response stability

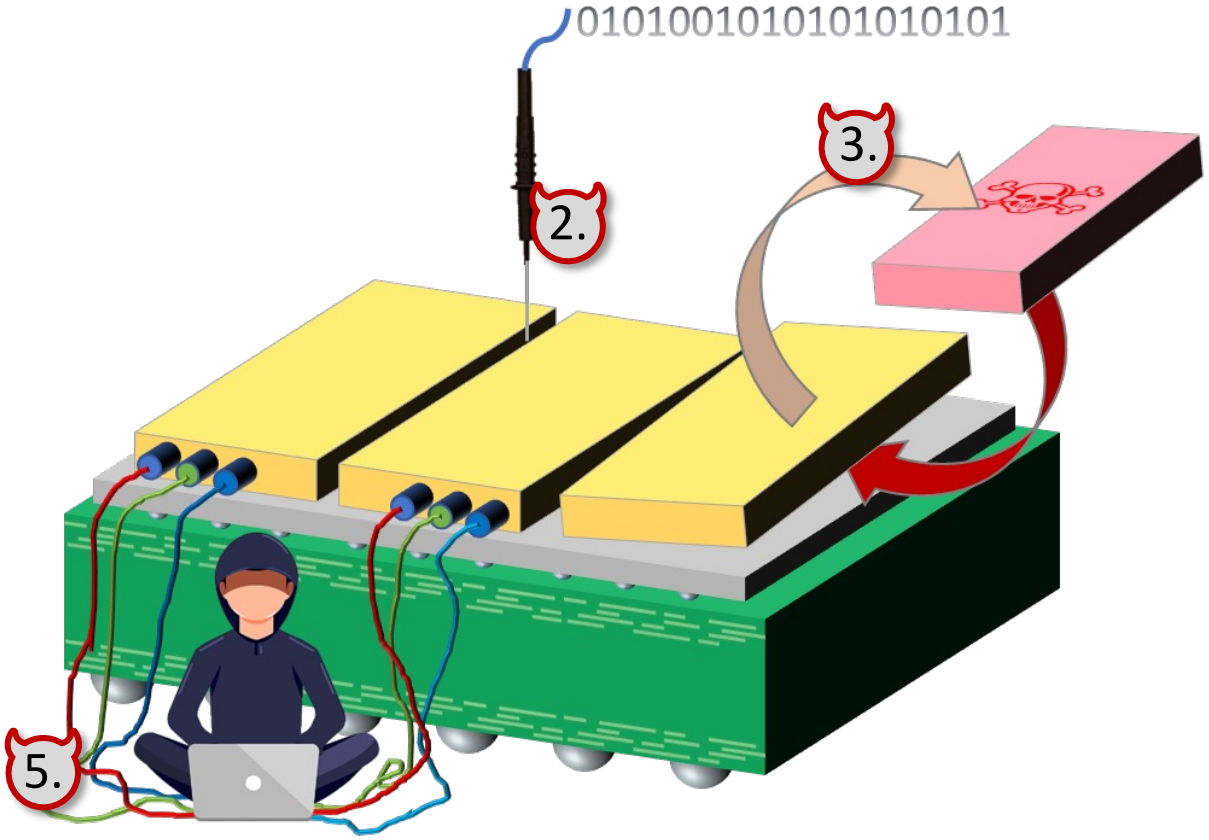
- Test: randomly assign SLLs to two groups, which are aged in opposite directions
  - Pull-high vs pull-low when idling between measurements
- Conclusion: groups do not diverge, implying that aging has little to no effect



Threats Addressed by Chiplet PUF

- PUF responses stable at picosecond level
  - Provides evidence of package integrity

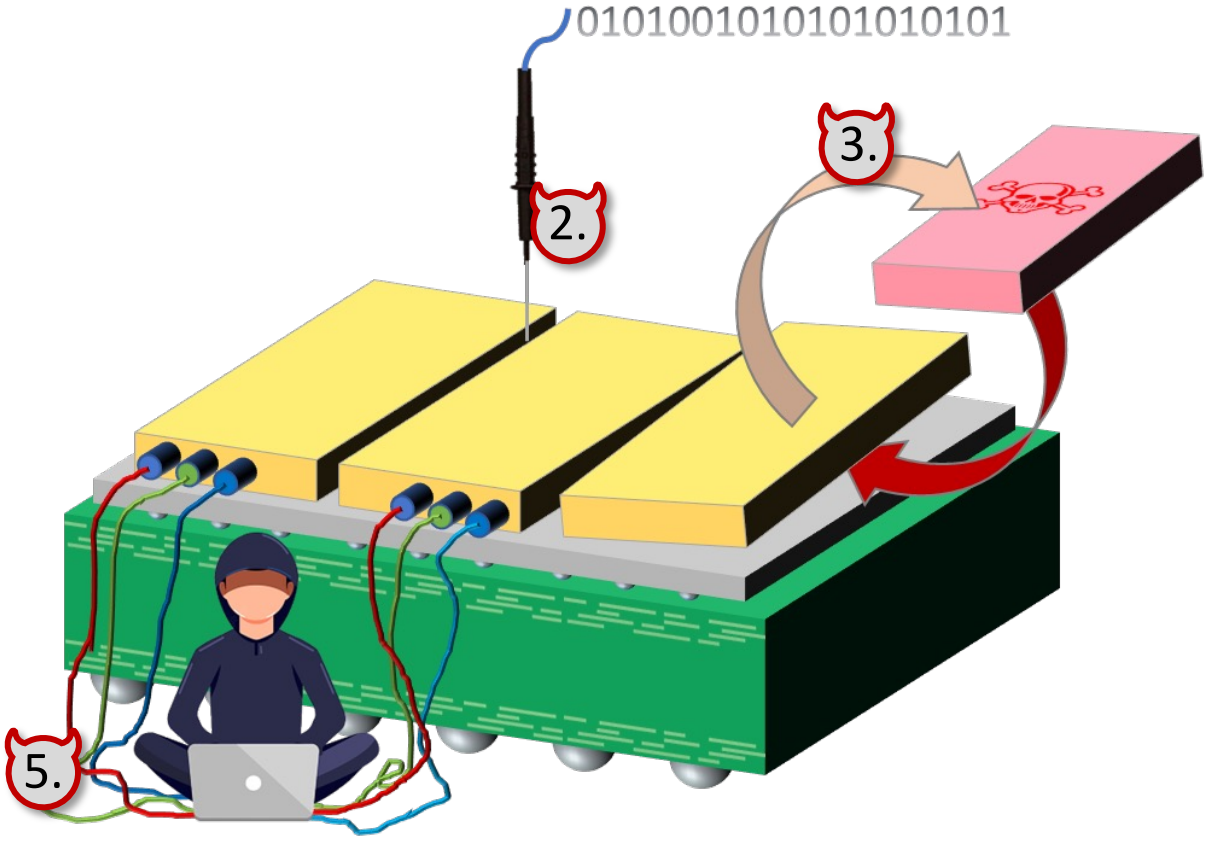
1. Trojans in co-packaged chiplets
2. Probing exposed interposer wires
3. Die-swapping
4. Side-channels from within package
5. Man-in-the-middle



Threats Addressed by Chiplet PUF

- PUF responses stable at picosecond level
  - Provides evidence of package integrity

1. Trojans in co-packaged chiplets
2. Probing exposed interposer wires
3. Die-swapping
4. Side-channels from within package
5. Man-in-the-middle

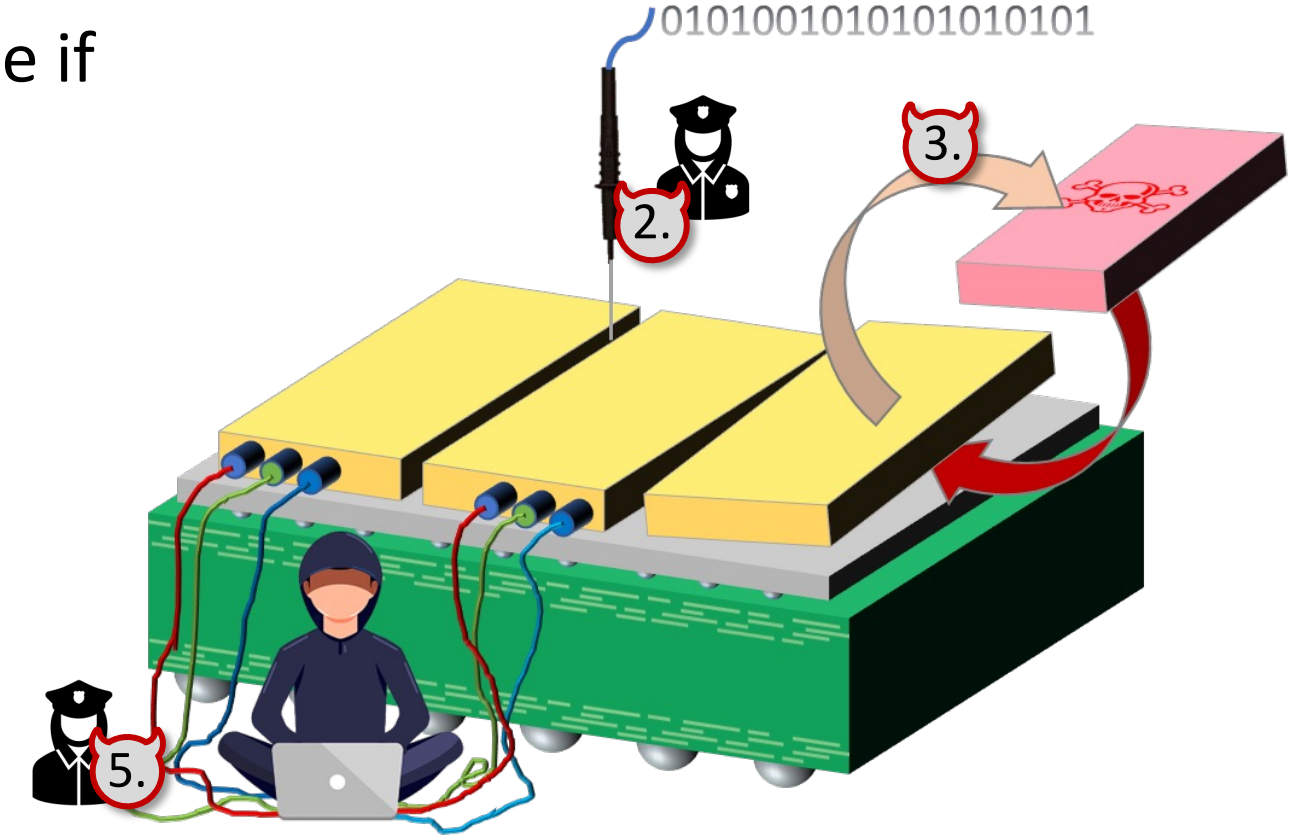


Threats Addressed by Chiplet PUF

- PUF responses stable at picosecond level
  - Provides evidence of package integrity

1. Trojans in co-packaged chiplets
2. Probing exposed interposer wires
3. Die-swapping
4. Side-channels from within package
5. Man-in-the-middle

- Physical probes and MITM detectable if causing delay changes that exceed within-class distances



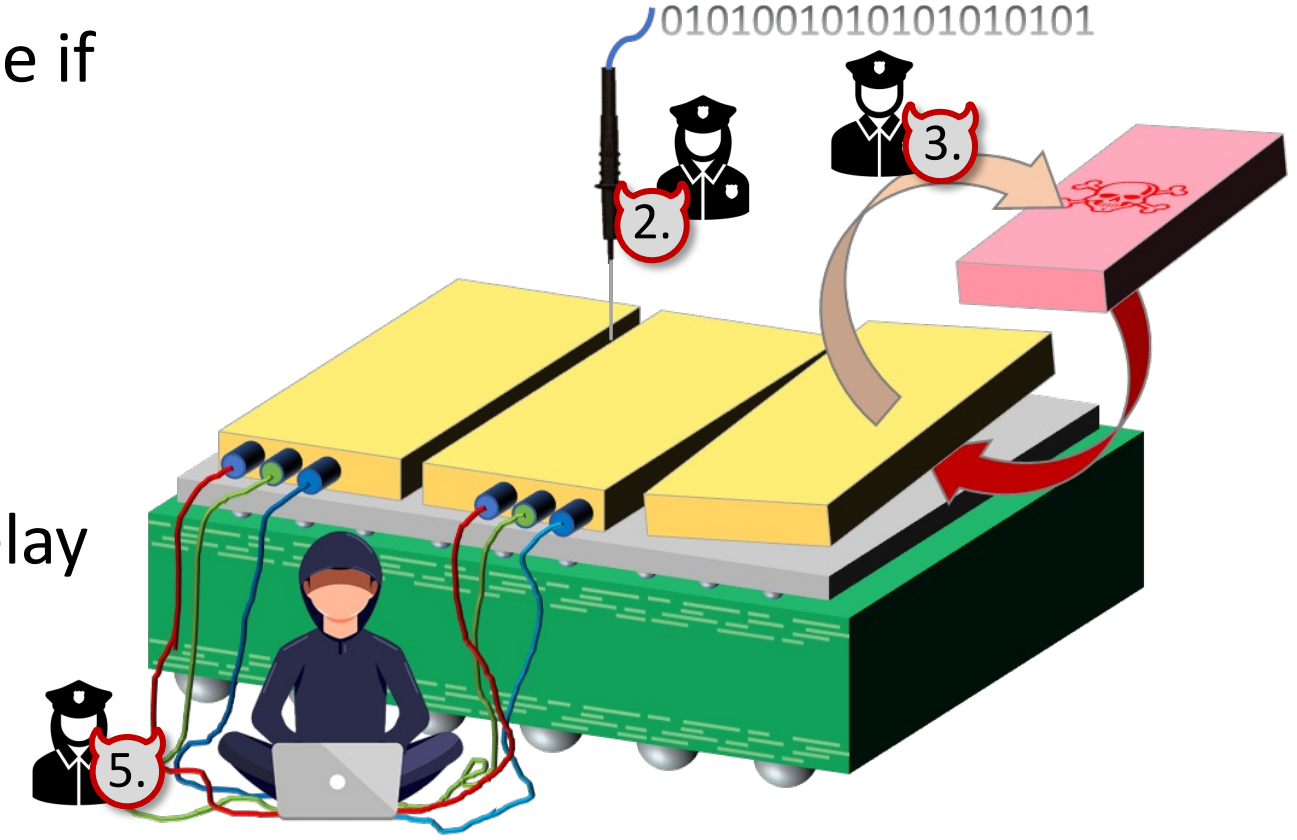
Threats Addressed by Chiplet PUF

- PUF responses stable at picosecond level
  - Provides evidence of package integrity

1. Trojans in co-packaged chiplets
2. Probing exposed interposer wires
3. Die-swapping
4. Side-channels from within package
5. Man-in-the-middle

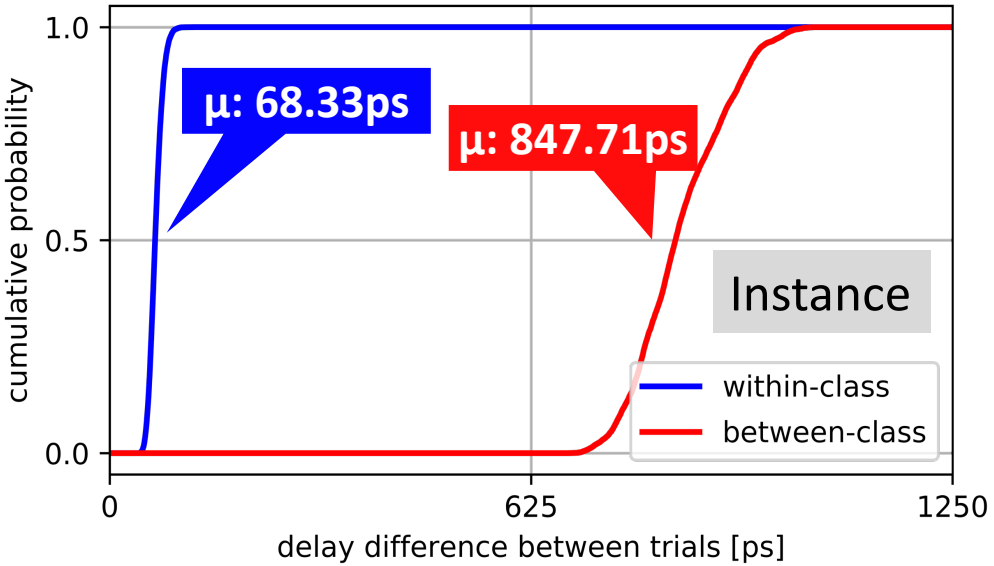
- Physical probes and MITM detectable if causing delay changes that exceed within-class distances

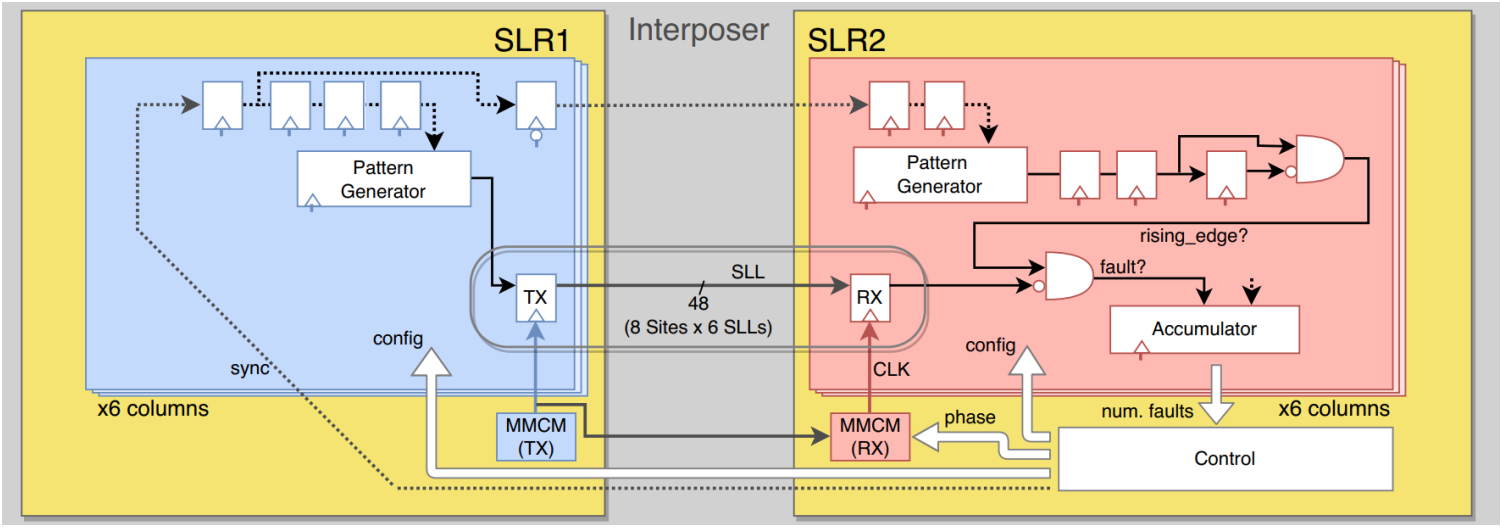
- Die swapping detectable because drivers contribute to entropy and delay measurements exist only on RX die

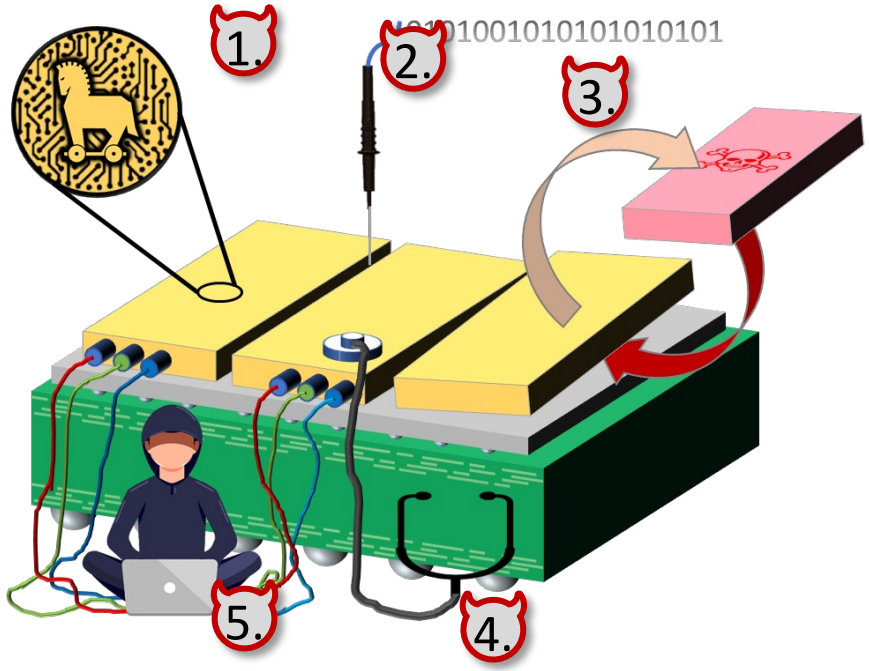


Conclusion

- Presented a security primitive to extract delay fingerprints from connections between chiplets
- Prototyped using Xilinx Ultrascale+ FPGAs locally and across a population on AWS EC2 F1
- Performed analysis across a variety of design manipulations to identify the specific sources of entropy in the system

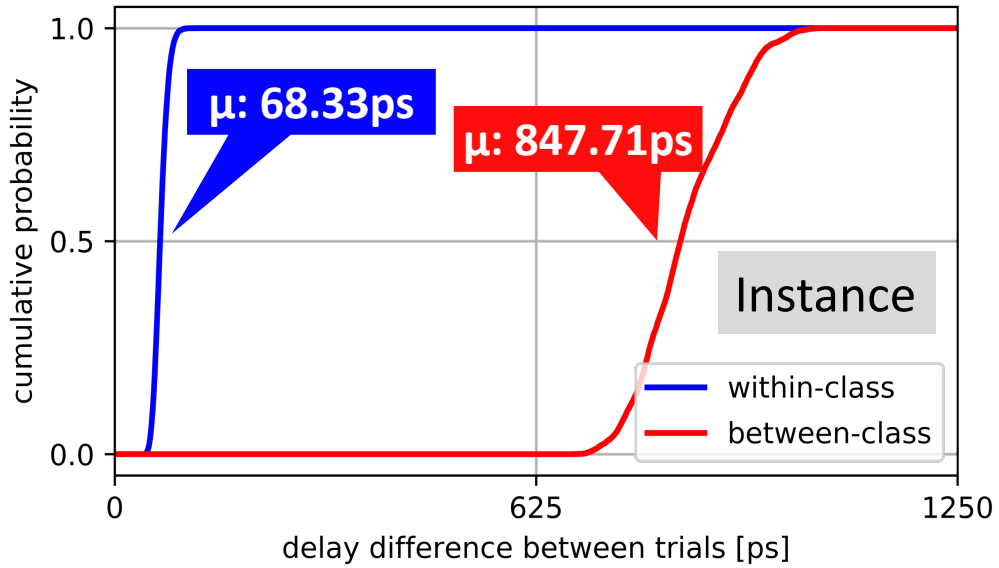






Conclusion

- Presented a security primitive to extract delay fingerprints from connections between chiplets
- Prototyped using Xilinx Ultrascale+ FPGAs locally and across a population on AWS EC2 F1
- Performed analysis across a variety of design manipulations to identify the specific sources of entropy in the system



Thank you! Questions?