WPI

WOOT 2025

GlitchGlück: Enabling Software Vulnerabilities through Guided Hardware Fault Injection

Zhenyuan (Charlotte) Liu, Dillibabu Shanmugam and Patrick Schaumont
{zliu12, dshanmugam, pschaumont}@wpi.edu
Worcester Polytechnic Institute Vernam Lab, Worcester, MA, USA

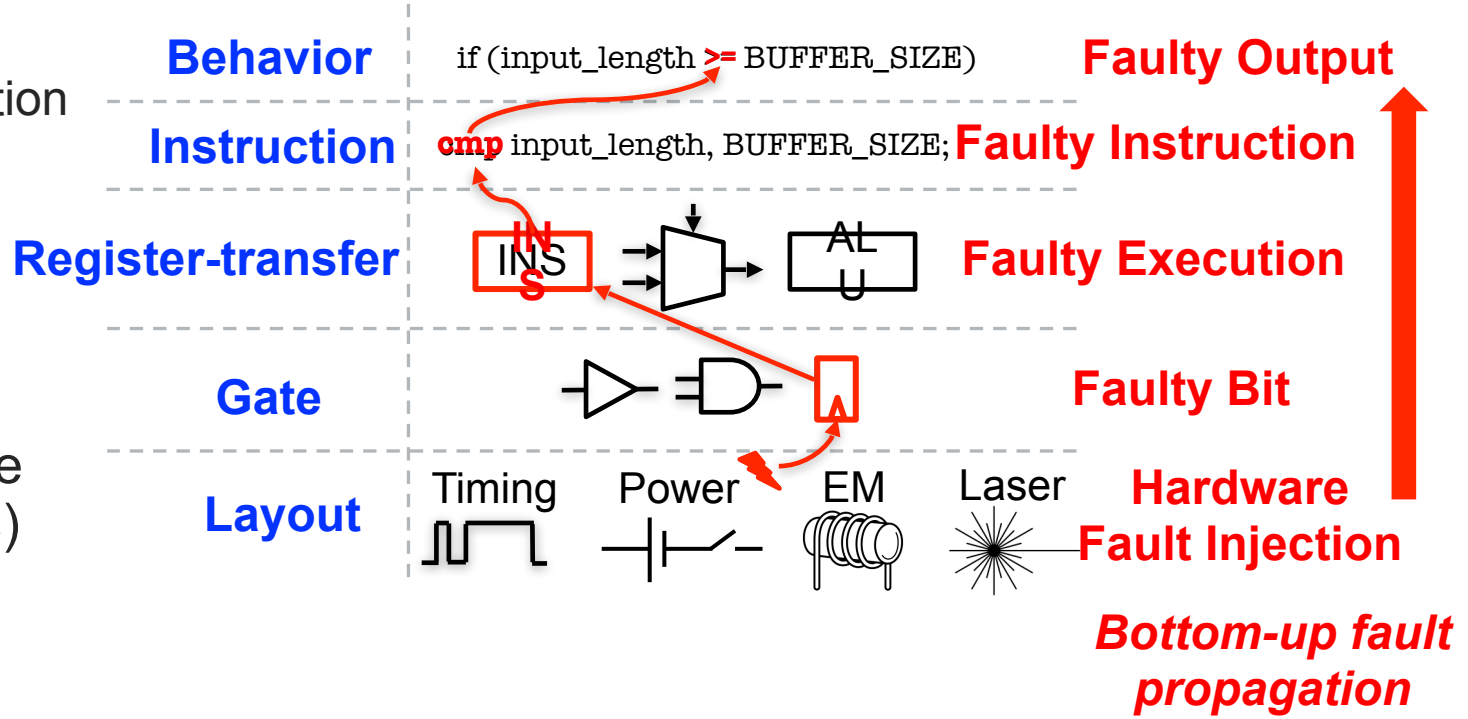
Artifact Available: https://github.com/Secure-Embedded-Systems/woot2025-GlitchGluck

August 2025

National Science Foundation

Background

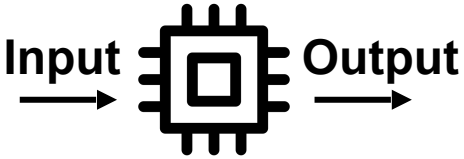
- **Software vulnerabilities**
  - Bugs in the software logic
  - Software-level countermeasures
    - Buffer overflow[1] – Input size validation
    - Could still be compromised by hardware fault injection
    - Unaware of hardware details
- **Hardware fault injection**
  - Use physical disturbances (voltage glitches or electromagnetic pulses) in hardware to disrupt intended circuit behavior



[1]Input data exceeds a buffer's size and causes overwriting adjacent memory (return address), altering control flow

Limitations

- **Black-box approach (blind fault attacks)**
  - Little visibility into hardware details
  - Classic fault models (e.g., instruction skipping): inadequate, as shown in [1, 2, 3]



- **Exhaustive fault parameter searches**
  - Thousands of trials to identify few meaningful fault effects (Elmohr [4], Bozzato [5], and Roscian [6])
  - Attack methodology is only driven from observed faulty outcomes (trial-and-error approach)
- **High-precision setups** (e.g., low-cost laser [7], focused EM [8]) improve targeting but remain environment-dependent and non-portable.

Large number of trials acceptable
Automated flow
Slow and non-portable

Attackers

Manual analysis
Fault impact verification
Fault space coverage

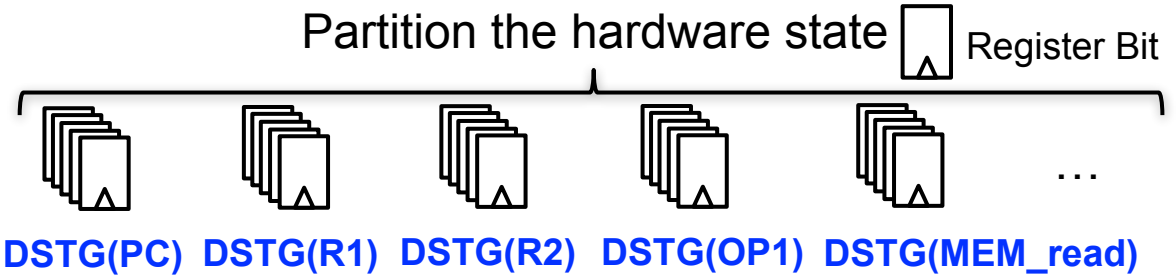
Countermeasure Designers

GlitchGlück – Guided Hardware Fault Injection

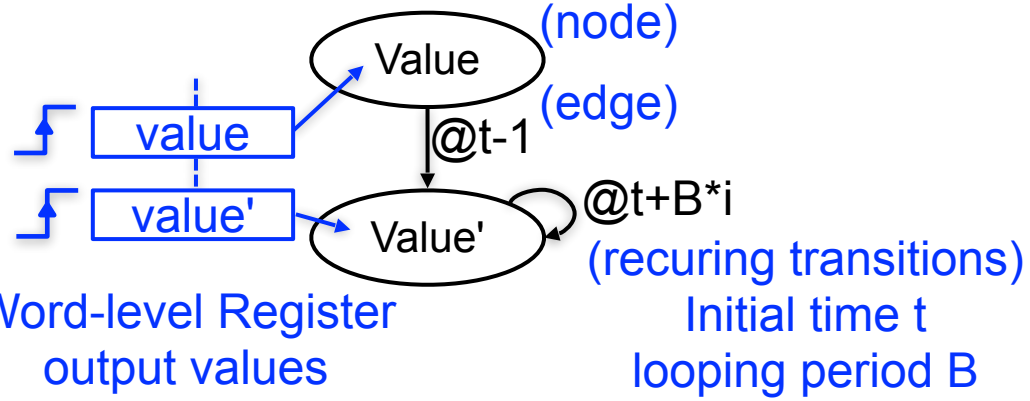
- **A thief trying to steal a jewelry hidden inside a large building**
  - Guesses blindly, wastes time, triggers alarms, and majority of the time it fails
  - Fault injection without guidance (blind process)
- **A detailed blueprint**
  - Turns guesswork into targeted action
  - Creates a plan, faster, more accurate strategy
  - Fault injection with guidance (**GlitchGlück**)
    - Uses pre-silicon layout data as the blueprint
    - More accurate attacks (one single glitch) and stronger defenses in design

GLITCHGLÜCK – Tool

- Simulation testbench
- No changes to RTL or gate-level netlist
- For large DSTGs, use time window selection and state filtering
- Based on non-speculative data from register outputs
- Captures modeled effects (e.g., dynamic inputs, simulated noise) reflected in flops
- Unmodeled physical phenomena (e.g., environmental noise) are excluded unless observable in flops



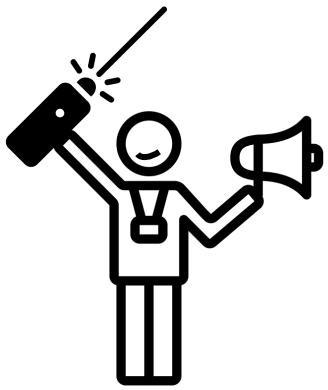
Dynamic State Transition Graph (DSTG)



Temporal visualization of software-hardware interactions

GLITCHGLÜCK – Methodology (Example Code)

Buffer overflow mitigation via input size validation (assert)



GlitchGlück

Analysis the DSTG
One single glitch
Minimal efforts

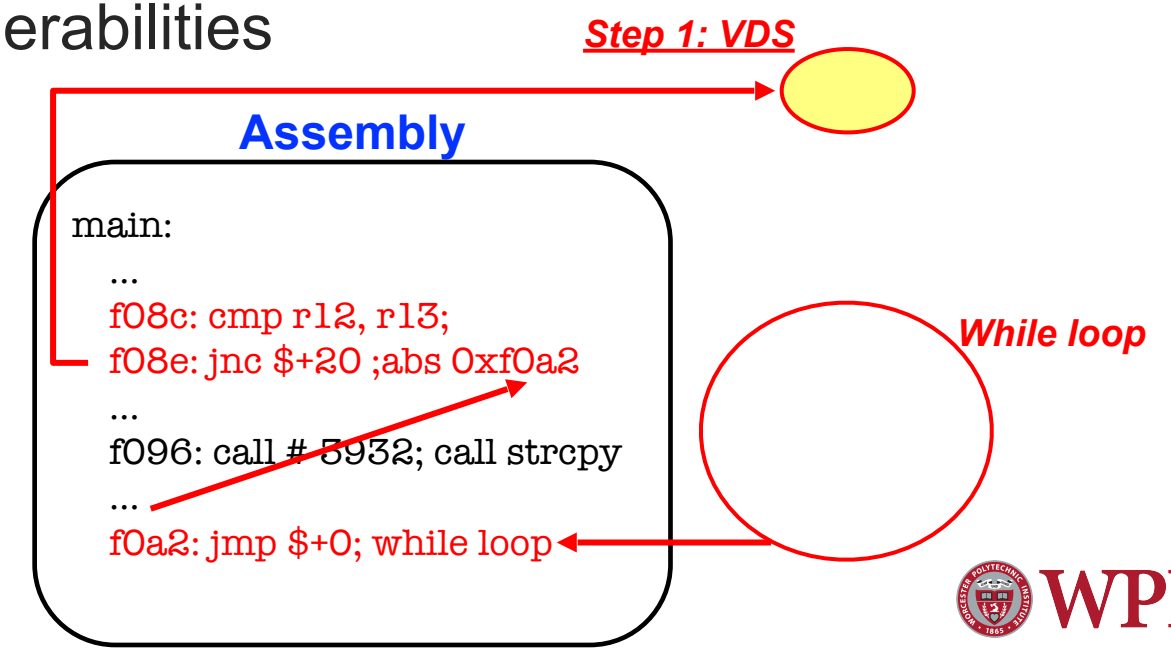
```
1  #define ASSERT(condition)
2      if (!(condition)) {
3          handle_error(__FILE__, __LINE__);
4      }
5
6  void handle_error(const char *file, int line) {
7      while(1);
8  }
9
10 char string[19] = "AAAAAAAAAAAAAAAA\x22\xf0\x0";
11
12 int main() {
13     char buffer[16];
14     ASSERT(sizeof(buffer) > strlen(string));
15     strcpy(buffer, string);
16     return 0;
17 }
```

```
1  void malicious(void) {
2      memcpy((uint8_t *)0x200, "it is broken", 13);
3  }
```

GLITCHGLÜCK – Methodology

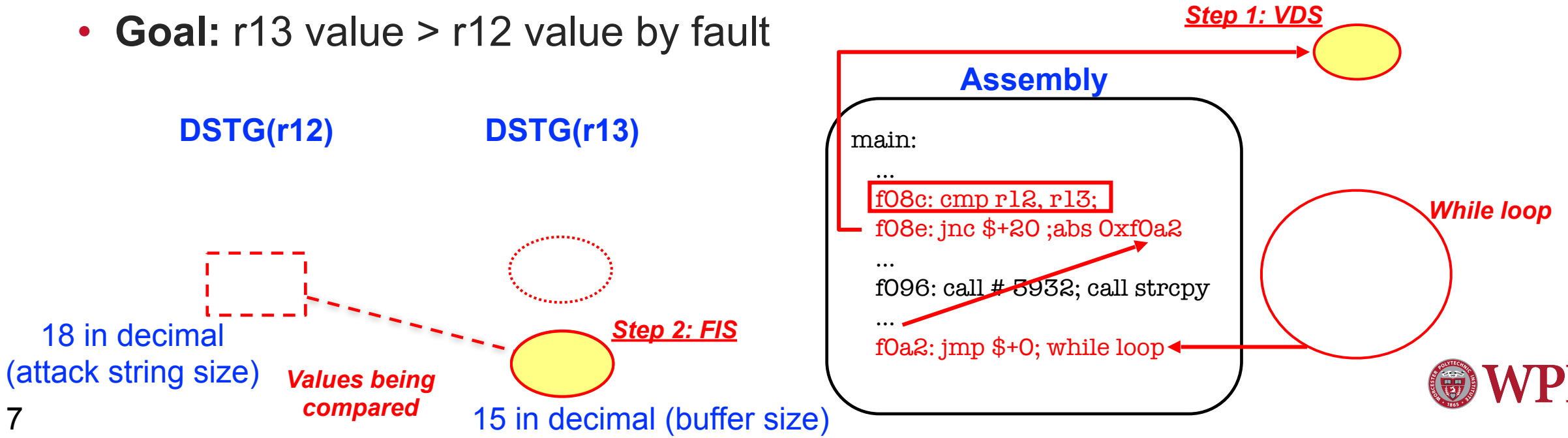
- **Step 1: Identify the Vulnerable Decision State (VDS)**
- **Definition:** A critical state that influences the system's behavior, representing a decision point in the execution flow (branch operation)
- By manipulating a VDS through fault, the execution path can be altered, leads to software vulnerabilities
- Program counter register, **DSTG(PC)**
- **Goal:** want to execute the strcpy

DSTG(PC)



GLITCHGLÜCK – Methodology

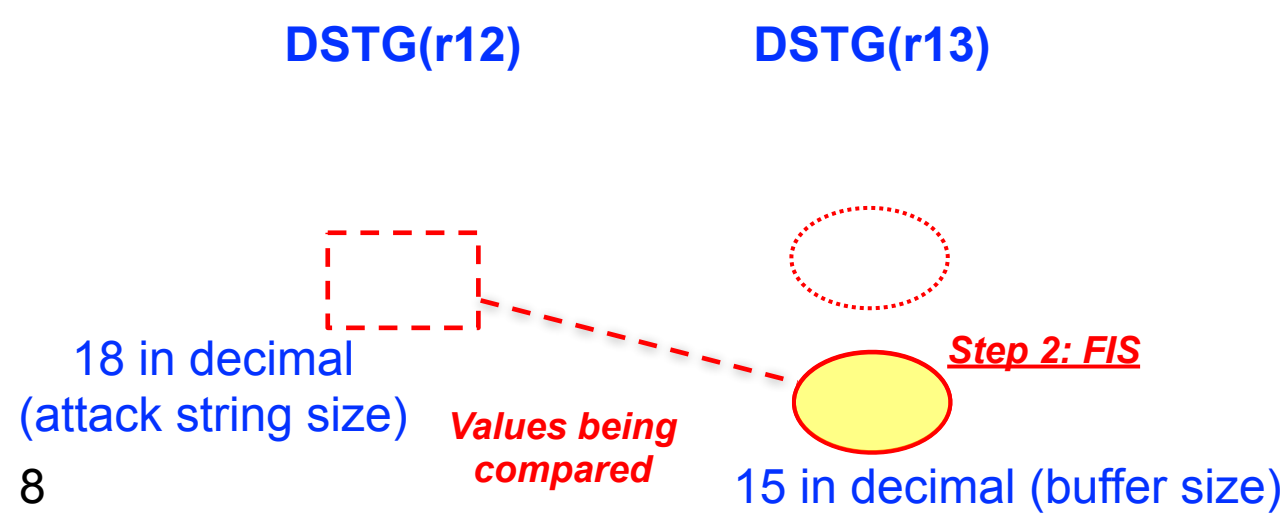
- **Step 2: Determine the Fault Injection State (FIS)**
- **Definition:** The state where the fault is injected, and it must have an impact dependency into the VDS
- A single VDS may be associated with multiple FISs (the one closest in time in always selected)
- **Goal:** r13 value > r12 value by fault



GLITCHGLÜCK – Methodology

- **Step 3: Establish the Fault Injection Time (FIT)**
- **Definition**: The exact clock cycle when a fault is injected – any random timing will not cause a direct impact to the VDS
- DSTGs reflects register output values
- Register input values are computed from upstream logic and registers

The input value to the FIS must be faulted



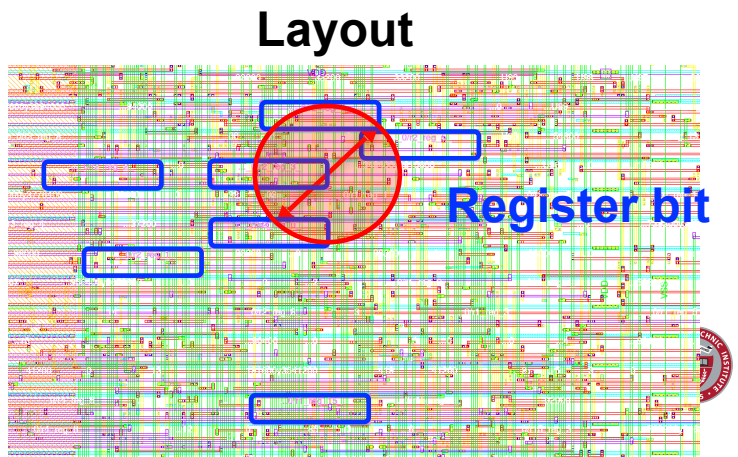
@923
r13[9] 1 → 0
r13[4] 1 → 0
r13[3] 0 → 1
r13[2] 0 → 1

Stabilize within 6ns (logic propagation time)

Step 3: FIT @923 6ns

GLITCHGLÜCK – Methodology

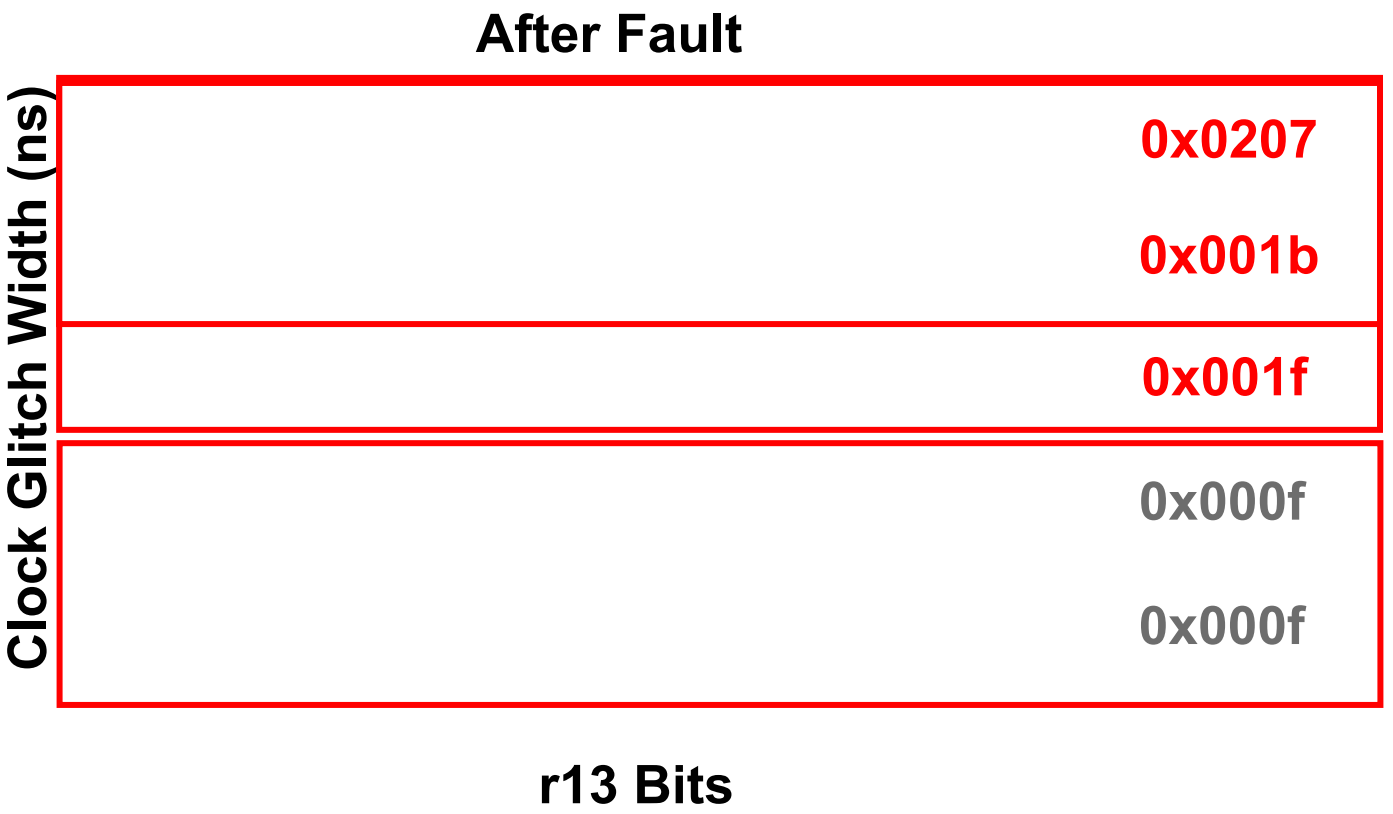
- VDS → branch instruction 0xf08e
- FIS → r13 (0x213 → 0xf)
- FIT → @923 with 6ns
- **Step 4: Validate the attack parameters (fault simulation)**
- **Global fault injection (clock glitching)**
  - Glitch width must be <6ns
  - Do not fault critical registers (e.g., PC_Next, Instruction Fetch), stabilize in 4.7ns
  - 4.7ns ~ 6ns
- **Localized fault Injection (laser injection)**
  - Targets specific bits within the FIS
  - Layout-aware laser injection
  - Laser radius size / bit-flip



WPI

GLITCHGLÜCK – Results

- **Fault simulation validation**
  - A 5ns clock glitch
  - ASCII string "it is broken" at 0x200
- **Post-silicon validation**
  - ASIC (OpenMSP430, 180nm with scan-chain support)
  - D. Shanmugam, Z. Liu, and P. Schaumont. "*Capri6: A solution for fault root cause detection*," Proceedings of the 34th Microelectronics Design and Test Symposium (IEEE MDTS), 2025.



Other Examples and Conclusion

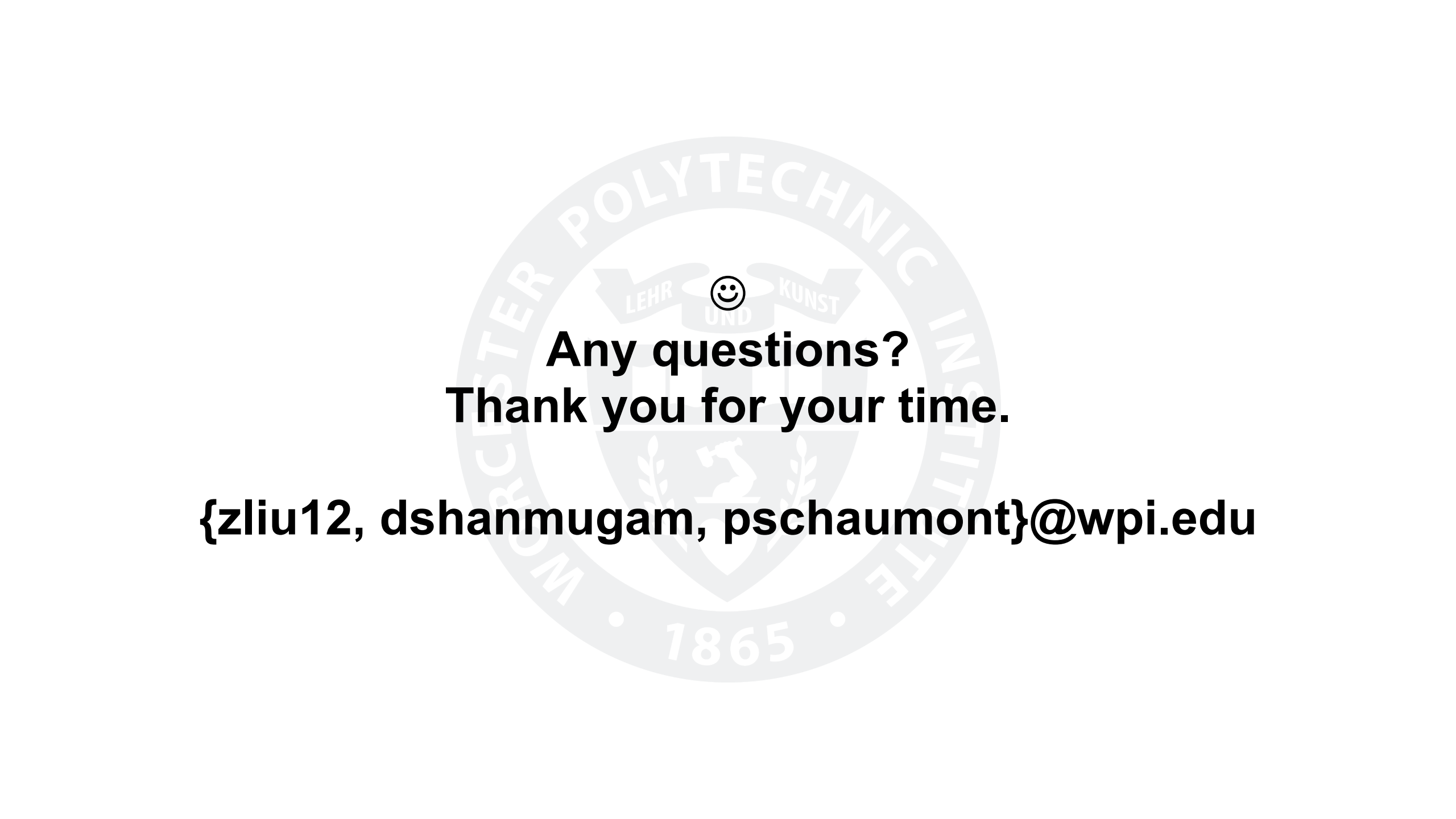
- **Fault simulation validation**
  - IBEX (instruction duplication) using clock glitching
  - PicoRV32 (verifyPIN5) using laser injection
- **Conclusion**
  - DSTG – a novel data structure visualize software-hardware interactions
  - Guided fault attacks without fault models (data-driven attack methodology)
  - Avoids blind attacks and reduces fault parameter search spaces
  - Layout-aware fault simulation + scan-chain → bridging the gap between simulation and measurement fault experiments
- **Future works**
  - IBEX ASIC chip with scan-chain support (including memory modules)
  - Automation of fault parameter extraction

WPI

References

- [1] Johan Laurent, Vincent Beroulle, Christophe Deleuze, Florian Pebay-Peyroula, and Athanasios Papadimitriou. On the importance of analysing microarchitecture for accurate software fault models. In *2018 21st Euromicro Conference on Digital System Design (DSD)*, pages 561– 564. IEEE, 2018.
- [2] Sébastien Michelland, Christophe Deleuze, and Laure Gonnord. From low-level fault modeling (of a pipeline attack) to a proven hardening scheme. In *Proceedings of the 33rd ACM SIGPLAN International Conference on Compiler Construction*, CC 2024, page 174–185, New York, NY, USA, 2024. Association for Computing Machinery.
- [3] Simon Tollec, Mihail Asavoae, Damien Couroussé, Karine Heydemann, and Mathieu Jan. Exploration of fault effects on formal risc-v microarchitecture models. In *2022 Workshop on Fault Detection and Tolerance in Cryptography (FDTC)*, pages 73–83, 2022.
- [4] Mahmoud A. Elmohr, Haohao Liao, and Catherine H. Gebotys. Em fault injection on arm and risc-v. In *2020 21st International Symposium on Quality Electronic De- sign (ISQED)*, pages 206–212, 2020.
- [5] Claudio Bozzato, Riccardo Focardi, Francesco Pal- marini, et al. Shaping the glitch: optimizing voltage fault injection attacks. *IACR transactions on crypto- graphic hardware and embedded systems*, 2019(2):199– 224, 2019.
- [6] Cyril Roscian, Jean-Max Dutertre, and Assia Tria. Frontside laser fault injection on cryptosystems - ap- plication to the aes' last round -. In *2013 IEEE Interna- tional Symposium on Hardware-Oriented Security and Trust (HOST)*, pages 119–124, 2013.
- [7] Martin S Kelly and Keith Mayes. High precision laser fault injection using low-cost components. In *2020 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 219–228. IEEE, 2020.
- [8] Lionel Riviere, Zakaria Najm, Pablo Rauzy, Jean-Luc Danger, Julien Bringer, and Laurent Sauvage. High precision fault injections on the instruction cache of armv7-m architectures. In *2015 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pages 62–67. IEEE, 2015.

WPI



☺

Any questions?
Thank you for your time.

{zliu12, dshanmugam, pschaumont}@wpi.edu