White Paper

Information Technology
Cybersecurity

Fault-Injection Countermeasures, Deployed at Scale

Authors:

Daniel Nemiroff
Sr. Principal Engineer
Intel Corporation

Carlos Tokunaga, Ph.D
Principal Engineer
Intel Corporation

Abstract

This paper details the design, calibration, and validation methodology for a fault-injection detection circuit that was first productized in Intel 12th gen Intel® Core™ processors. We will describe how fault-injection attacks can impact circuit timing, the high-level design of the Tunable Replica Circuit (TRC), data gathering phase that occurs in HVM (high volume manufacturing), the methodology to create a calibration recipe, false positive testing, fault-injection testing and the final HVM production calibration flow. Additionally, this paper will illustrate the feedback loop where the circuit is tuned from data gathered from the false positive and attack-lab testing.

1 Introduction

With the hardening of software (SW) vulnerabilities through the use of virtualization, stack canaries, authenticating code before execution, etc., attackers have turned their attention to physically attacking computing platforms. A favorite tool of these attackers is fault injection attacks via glitching voltage, clock pins, EM [1-3] to cause circuits to fail timing, resulting in the execution of malicious instructions, exfiltration of secrets, etc.

Several techniques have been added to detect these attacks such as multi-bit encoding of fuses, ECC on busses/fabrics/registers, hardware (HW) enforced program execution control as well as other methods [4-6]. However, there is a need to complement these mitigations with HW-based sensors that explicitly detect circuit-based timing failures that occur as the result of an attack.

A crucial aspect for productizing such HW sensors is calibration. If the sensor is calibrated too aggressively, it would detect normal workload voltage droops as false positives. If false positives occur in the field the resulting platform instability may lead to unnecessary recalls, a catastrophic event for Intel. While we prioritize the elimination of false positives, we need to make sure all real fault-injection attacks that will induce an internal error are detected. Minimizing the false negatives, successful fault events is also very important. To mitigate false positives, we developed a feedback-based calibration flow. This feedback loop uses results from false positive and false negative testing along with margin data from the HW sensor that indicates how close the sensor was to detecting a glitch.

The TRC was developed to detect dynamic variation (voltage droop, temperature, aging) in circuits [7-8], which the TRC detects as timing violations. Since the TRC can be calibrated to a point where such timing violations could only be the result of an attack, we chose the TRC as the HW sensor to mitigate fault-injection attacks in Intel security processors and calibrated it using the process described above.

Starting with the Intel 12th gen Intel Core processor, we have successfully integrated, characterized and validated the TRC into the Intel® Converged Security and Manageability Engine (Intel® CSME) sub-system. In this initial instantiation the Intel CSME TRC was calibrated and entered false positive and false negative testing.

Table of Contents

Data from initial testing indicated we had set the guardbands too high as we missed some glitches in false negative testing. Following our methodology for calibration feedback, we generated a revised calibration recipe, fused new units and re-ran the testing, successfully tuning the recipe on the 2nd pass. We performed several TRC configuration validation runs to extensively and successfully pass the false negative and false positive tests for each product.

2 TRC Properties and Behavior

As stated in the introduction, the TRC was initially developed to detect aging in CPU circuitry, but its characteristics allowed it to be used to detect fault-injection attacks.

The TRC consists of a launching flip-flop (FF), a tunable delay chain, and a capture FF (as shown below), allowing the TRC to detect timing violations:

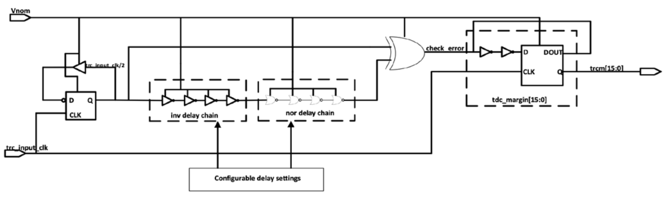

Figure 1. TRC block diagram

The TRCs delay chain is per-device calibrated during high-volume manufacturing. It is the calibration of this delay chain that allows the TRC to determine if the timing of each cycle is correct and not been tampered with via fault-injection attacks. Simply said, the delay chain of the TRC has a 1:1 relationship with the voltage and clock frequency that are used to power Intel CSME and other devices on the 12th gen Intel Core processor.

2.1 Example Setup Time Attacks using Clock and Voltage

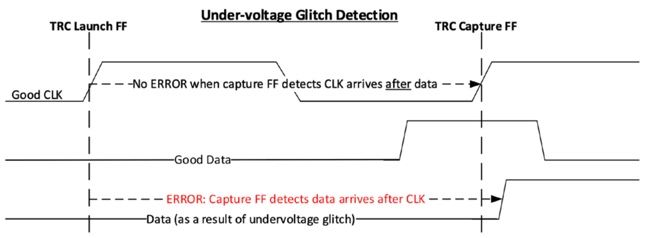
Under-voltage Glitch Detection
TRC Launch FF
TRC Capture FF
Good CLK
No ERROR when capture FF detects CLK arrives after data
Good Data
ERROR: Capture FF detects data arrives after CLK
Data (as a result of undervoltage glitch)

Figure 2. Under-voltage glitch detection

The above diagram illustrates how the TRC detects an undervoltage attack. In the case of a good clock and voltage, the TRC Capture FF detects that the clock arrives after the data. However, if the attacker has caused a voltage droop, the TRC Capture FF will detect that the data arrives after the clock.

- Using this example diagram, in the case of a successful attack, the attacker will have succeeded in latching a 0 instead of a 1.
- Practically, if all data lines latched a zero, this would result in the Intel CSME processor execution a NOP instruction.

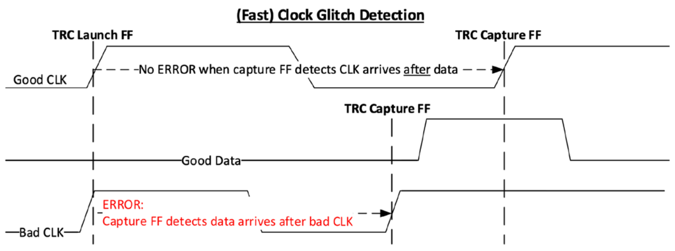
(Fast) Clock Glitch Detection
TRC Launch FF
TRC Capture FF
Good CLK
No ERROR when capture FF detects CLK arrives after data
TRC Capture FF
Good Data
ERROR:
Capture FF detects data arrives after bad CLK
Bad CLK

Figure 3. (Fast) clock glitch detection

Similar to a voltage droop attack, the TRC will detect an over-clocking attack as the Capture FF detects that data arrives after the clock because the clock has been sped up.

- While clock and voltage are the most predictable attack vectors (and easiest to illustrate), temperature and EM will cause timing to fail as well, resulting in the same conditions within the TRC. However, we have limited data in this area.

2.2 TRC Detecting a FI Attack

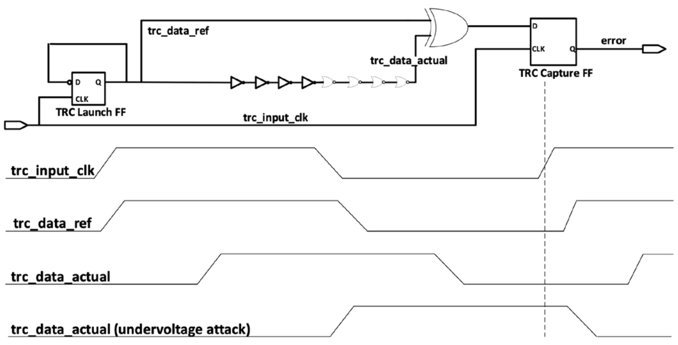

Figure 4. TRC FI attack detection

The above diagram illustrates how the TRC detects an undervoltage fault injection (FI) attack. The first rising edge of the trc_input_clk causes the Launch FF to drive the trc_data_actual through a delay chain to an XOR gate. The other input to the XOR gate, is the trc_data_ref signal which has no delays in it. The second rising edge of the trc_input_clk causes the Capture FF to latch the output of the XOR gate.

The delay chain is calibrated such that, when no attack is occurring, the Capture FF will latch a 0b as the output of the XOR gate. However, if the data lines have been slowed down due an undervoltage glitch, the XOR gate will output a 1b at the time the Capture FF latches the error signal.

It should be noted that trc_data_ref will also be slowed down by a undervoltage glitch, however voltage has more of an impact to the trc_data_actual signal due to the number of circuits in the delay chain.

2.3 TRC vs. Other Solutions

The industry has seen a variety of other fault-injection detection solutions which primarily consist of analog voltage and clock and temperature sensors. We chose to utilize the TRC as it can detect attacks from multiple environmental conditions. Additionally, the TRC is a digital device, making it easy to port from one process node to another and it consumes a smaller die area than building both an analog clock monitor and analog voltage level detector.

2.4 Integration into Intel CSME

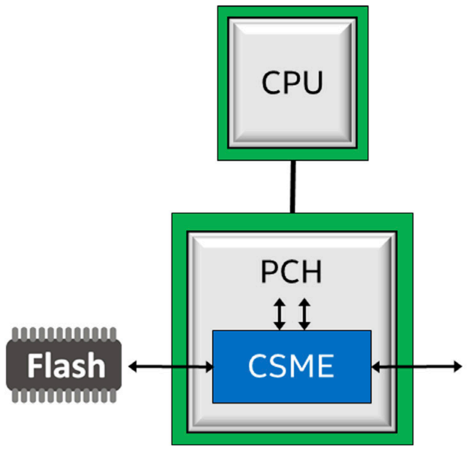

Figure 5. Intel CSME integration

Intel CSME is an embedded subsystem in the Platform Controller Hub (PCH) of Intel client platforms. It is a standalone low power Intel processor with dedicated Hardware (HW), the Root of Trust of the platform that provides an isolated execution environment protected from host SW running on main CPU. Intel CSME contains a HW mast ROM that authenticates and executes Intel CSME FW.

In the Intel 12th gen Intel Core processors, we integrated a TRC into the System Agent of Intel CSME as shown below:

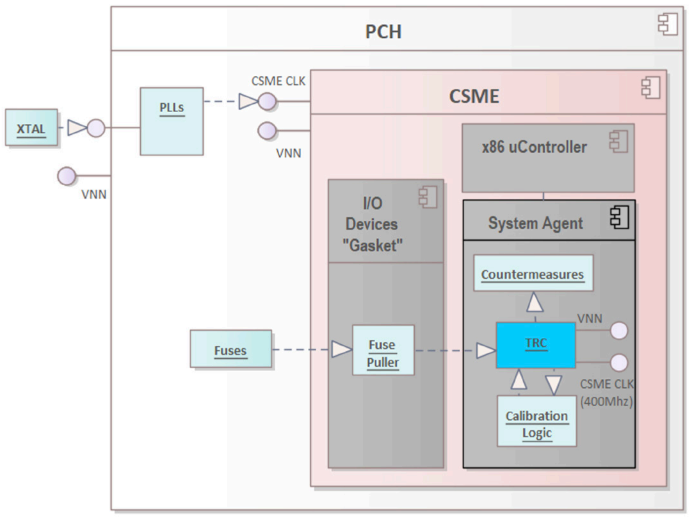

Figure 6. TRC integration into the system agent of Intel CSME

The Intel CSME TRC monitors the power and clock coming into Intel CSME, to help protect all portions of Intel CSME from an attack. When the TRC detects a glitch, it invokes countermeasures that result in a Intel CSME reset. The rest of the SoC is not impacted.

The TRC is on the same reset line as all Intel CSME HW, and if Intel CSME is on, the TRC is monitoring this power. If Intel CSME is power-gated, the TRC is also power-gated.

3 TRC Calibration

Calibration occurs on every part in HVM (high volume manufacturing), by converting a voltage glitch (Vglitch) to a device-unique delay, which is then programmed into TRC fuses. At reset-exit, Intel CSME pulls TRC fuses before beginning execution.

3.1 Generating the Calibration Recipe

Determining Vglitch:

Vglitch is determined on a per-product basis, based on the power delivery integrity and workload guardbands, then subtracting the guardband from Vnom.

The Need for Slope:

On the manufacturing line, to convert Vglitch to a programmable delay each part exports the delay that generates a TRC error at Vnom. The manufacturing tester then calculates the corresponding delay at Vglitch using a linear equation, containing a global slope. Slope is determined using three coordinates, where each coordinate is a voltage/delay pair. By capturing data from thousands of devices across voltage, temperature and process corners we are able to calculate a slope that is used globally for a product.

TRC Calibration FSM (Finite State Machine):

Delays at Vnom, Vmax and Vmin are determined by setting the voltage, then schmooing the TRC delay until it generates an error. To simplify the tester code and improve test time, we implemented an FSM in Intel CSME, allowing HW to automate data schmooing, such that the HVM tester only needs to trigger the FSM and record its output.

Gathering Data in HVM to Calculate a Global Slope:

In HVM, we gather data on thousands of DuTs (device under test) across slow, normal and fast process corners. Class and sort testers run the TRC FSM at Vmin, Vnom and Vmax at hot and cold temperatures, capturing these 6-data points for each DuT. With all these points we are able to calculate a slope that covers an entire product line.

Calibrating in HVM, Using Slope:

Restating, the recipe takes the delay at Vnom (read by the production tester) and outputs the delay at Vgiltch, via: *DELAY@Vglitch = DELAY@Vnom – (Vglitch <divided by> slope/GLOBAL_SLOPE)*.

- DELAY@Vnom is read by the tester from the TRC FSM.
- Vglitch is a constant for the product.
- GLOBAL_SLOPE is calculated above.

3.2 False Positive Testing

False positive testing is executed to ensure that no false positives in the field occur that could result in system stability issues. As such, the false positive testing exercises a workload that could result in standard operating voltage droops. See the Results section for details on the workload run.

Static False Positive Testing:

Following execution of the workload, engineers query the TRC to determine:

1) If the TRC detected an error.
2) If the TRC is close to triggering an error, done by reading the margin register in the TRC.

If any parts show either the TRC detecting an error or being close to margin, recalibration is required as the risk for field-based false positives would be high.

Dynamic False Positive Testing:

Following completion of static testing, the IP-SV team runs the same workloads, schmooing the delays of the TRC using JTAG. This is done to determine at which point voltage droops that occur during false positive testing trigger a TRC error, creating an additional mechanism to indicate how much margin to a false positive a calibrated TRC contains.

3.3 False Negative Testing

False negative testing induces voltage glitches into the product to determine if the TRC catches them. This is done by connecting a voltage generator (VC glitcher, manufactured by Riscure Inc: https://www.riscure.com/product/vc-glitcher) to the external pins associated with the VNN power rail which the TRC and the security IP it is protecting are connected. This requires bypassing the standard motherboard voltage regulator by reworking the board.

Static False Negative Testing:

After each voltage glitch, the engineers query the TRC to determine if it detected the glitch as well as how close to an error the TRC was (by reading the margin register). Missing a glitch does not necessarily indicate the TRC is poorly calibrated because SoC circuitry masks glitches before they can negatively impact the security engines. However, a glitch that causes the SoC to reset that is not detected, is a failure.

Dynamic False Negative Testing:

Following static testing engineers shmoo the delays of the TRC to determine at which point the TRC detected a glitch, that it might have otherwise missed. This information, along with false positive data is fed back to the architecture team during the next phase of TRC calibration.

3.4 Analyzing Results

With all data collected from false negative and positive testing, the architecture team determines if a change needs to be made to the recipe and the cycle starts again.

3.5 HVM: Lock in Recipe for Production

Once the recipe is high quality, the HVM class team locks in the recipe into their production class test tapes. From this point on, all production silicon will be fused with the TRC enabled and calibrated by using the TRC FSM to read the delay at Vnom and use the recipe to calculate and program the Vglitch delay.

4 Results

4.1 False Positive Results

False positive testing was done in a standard post-silicon environment. In order to analyze the TRC behavior under end-user conditions, the voltage droop was replicated via a TDP workload setup. The goal is to maximize the number of disks and devices connected, together with a script to generate transactions between them and stress audio core after windows booted up. It includes 2 x Gen 4.0 NVMe devices, a Gen 3.0 NVMe device, a SATA disk and 2 x USB 3.2 external storage units.

Static False Positive Testing:

Over 50 parts were tested at this stage. None of the parts showed TRC asserting false positive errors, and via margin readings it was determined that TRC was not close to triggering an error.

Dynamic False Positive Testing:

From the same set of units, several were manually selected to go into the Dynamic testing stage. Using the delay override capabilities of TRC, the delay was increased gradually until finding the points where the margin indicated that TRC is close to triggering the error and when it actually detected an error. When TDP workload is running and PCH is being stressed, the required delay value to reach the error is usually lower.

4.2 Threat Model

False negative testing on the Intel 12th gen Intel Core processors PCH encompassed testing multiple parts, by connecting a voltage generator to the board-level power rail that feeds the VNN rail. Reference negative voltages were driven from the generator for varying pull-widths, resulting in voltages measured at the board-level power rail to be below vMin. After injecting each glitch, the states of TRC and the SoC were queried and recorded into one of the following categories:

1) TRC and SoC state did not change.
2) SoC reset, and the state of the TRC was unknown.
3) SoC state did not change, and the TRC detected an error.

In the graph below, the categorized platform response is plotted for each attempted glitch on a specific part at the calibrated delay code, in terms of the glitch's voltage and length. The presence of the band of red Xs highlights the TRC's detection capability in response to transient voltage glitches. These are the glitches that did not reset the SoC but were detected by the TRC. The region of functionality just before the SoC begins crashing due to glitches is often where a fault injection attack will focus, as the attacker's objective is to change the state of the system in some way without crashing everything. As such, it is important that this red X band be present and that the TRC is calibrated to detect voltage glitches before they actually begin crashing the SoC.

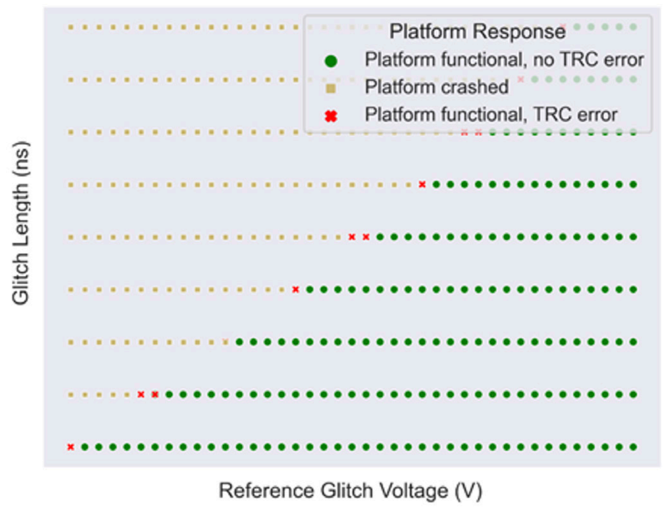

Figure 7. Platform response to voltage glitch

On the part-specific results plotted above, the red X band does not exist at every glitch length, indicating that the calibrated delay code for this part could be set too low to be effective, due to too much guardband in the calibration formula. To observe the effect of the delay code on the TRC's detection capability, the same glitch scan and response categorization was repeated with various delay codes, starting from the calibrated delay code and increasing. For a given glitch length, the width of the red X band was measured and plotted, as shown below:

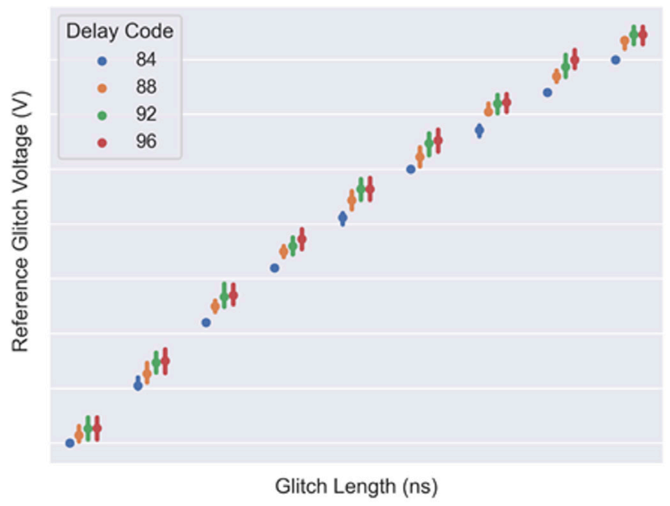

Figure 8. Glitch scan and response

As can be seen, increasing the delay from 84 to 92 allowed the TRC to detect glitches in a larger range for a given glitch length, where there was no meaningful detection capability when going to a delay of 96.

4.3 Resulting Change in the Recipe

From these results we inferred the TRCs were calibrated too conservatively and reduced the guardband, thereby changing the Vglitch constant. The guardband was adjusted to improve our false negative coverage.

5 Conclusion

Following the change in recipe, additional DuTs were manufactured by HVM and sent for false positive and false negative testing. This 2nd pass of testing proved the change in recipe allowed the TRC to catch the glitches it previously missed. Additionally, false positive testing passed, indicating no parts were close to triggering a false positive. At this point the team was confident to lock in this final recipe and begin HVM.

5.1 External Testing at Riscure

To further gain confidence in the TRC and gain additional insight into FI testing, we contracted with Riscure to evaluate the TRC. We submitted multiple 12th gen Intel Core processor parts with the TRC to Riscure for clock, voltage and EMFI testing. In the end, Riscure was unable to successfully execute a FI attack against , concluding, “In all cases the successful glitches were detected by the implemented countermeasures”.

Acknowledgements

Thanks to the following Intel collaborators for their contributions to their work on the TRC:

- Matias Leonetti
- Swetha Basani
- Parthiv Trivedi
- Sivakumar Ramakrishnan
- Joseph Friel
- Mohamad Faiz Mohd Faridh
- Nanda G Kumar Kalavai
- Masahide Kakeda
- Avinash Varna
- Habib Shawal
- Jim Tschanz

6 References

[1] A. Barenghi, et al., "Low Voltage Fault Attacks on the RSA Cryptosystem," IEEE 2009 Workshop on Fault Diagnosis and Tolerance in Cryptography (FDTC), 2009.

[2] M. Alam, et al., "RAM-Jam: Remote Temperature and Voltage Fault Attack on FPGAs using Memory Collisions," FDTC, 2019

[3] M. ELmohr, et al., "EM Fault Injection on ARM and RISC-V,"International Symposium on Quality Electronic Design (ISQED)", 2020

[4] N. Wiersma, et al., "Safety != Security: On the Resilience of ASIL-D Certified Microcontrollers against Fault Injection Attacks," FDTC, 2017

[5] J. Sakamoto, et al., "How to Code Data Integrity Verification Secure Against Single-Spot-Laser-Induced Instruction Manipulation Attacks," IEEE/ACS 17th International Conference on Computer Systems and Applications (AICCSA), 2020

[6] M. Werner, et al., "Protecting RISC-V Processors against Physical Attacks," Design, Automation & Test in Europe Conference & Exhibition (DATE), 2019

[7] K. Bowman, et al., " The “Palisades” Resilient Processor for Improved Performance and Energy Efficiency," DTTC, 2010.

[8] K. Bowman, et al., " All-Digital Dynamically Adaptive Clock Distribution for Voltage Droop Tolerance," DTTC 2012.