Stanford Synthesizing Formal Speculation Contracts

Xiaofu (Kaia) Li, Yao Hsiao, Caroline Trippel

Background

Hardware Side-Channel Attacks

In hardware side-channel attacks, a *transmitter* in the victim program modulates a *channel*, creating a μpath variability in a *transponder* instruction, which the attacker can observe to infer about data processed by the transmitter.

Transient execution attacks exploit speculation schemes implemented in hardware to improve performance. For example, the processor may predict branch targets and speculatively execute instructions that leak secret data and flush them if the the prediction is wrong. We call instructions that may introduce speculative execution speculative primitives.

Leakage Contracts Synthesize

Leakage contracts capture potential observations that leak secret information, which is an important foundation for developing defenses against side-channel attacks.

Prior work [1] developed RTL2MμPATH, which uncovers possible μpaths for instructions. SynthLC uses the μpaths to synthesize leakage signatures from the μpaths.

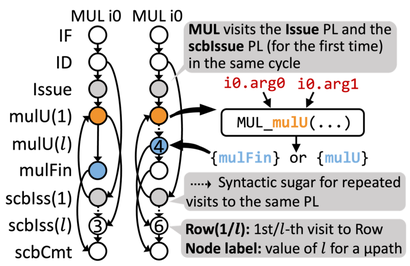


Figure 1. μpaths of MUL in CVA6 [1]

Project Overview

We aim to develop an automated procedure for synthesizing formally verified *speculative execution contracts*. In this portion, we focus on control-flow mis-predictions.

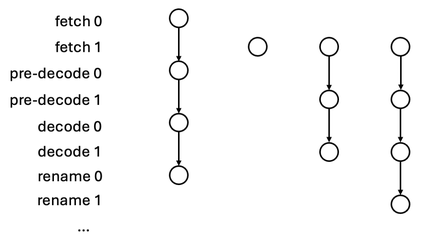
Methods

We experimented on RSD, an open-source RISC-V out-of-order superscalar processor.

Figure 2. Pipeline stages of RSD processor [2]

We implemented a SystemVerilog Assertions template to determine if a candidate speculation primitive instruction i_{sp} can cause a transponder instruction i_p to get flushed at a certain decision point. The approach is to issue symbolic i_{sp} and i_p instructions down the processor pipeline without the interference of other instructions. For a given decision source and destination, if the following are met

1. i_p always proceeds from the source to the destination when issued alone.
2. i_p may not proceed from the source to the destination when i_{sp} is also issued at some point.



Then we know i_{sp} caused i_p to flush at this decision point and is indeed a speculation primitive.

Figure 3. Example of flush/non-flush μpaths

Results

We experimented with a few candidate speculation primitives and observed several ways they can introduce flushes.

Branch and jump instructions mis-predicted as taken can cause a flush when they are resolved. The flush can also happen earlier in the pipeline when the decoded branch target doesn't match the predicted address.

Non-branch instructions can also introduce speculative execution when they are predicted as a branch. This induces a flush early in the pipeline when the instruction is decoded.

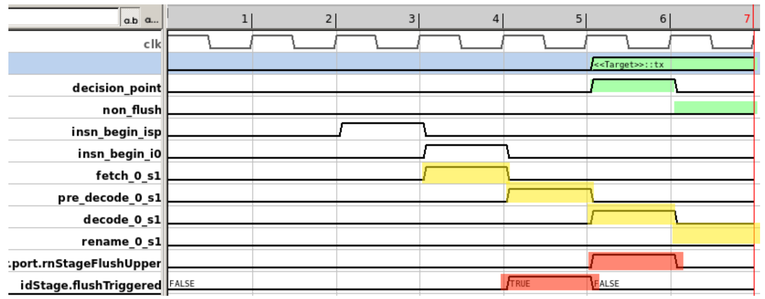


Figure 4. Waveform of a flush signal triggered by i_{sp} invalidating i_p

Next Steps

1. Extend current work to automatically enumerate all instructions that introduce speculation windows and when the speculation happens
2. Explore the synthesis of a data-flow contract

References

[1] Yao Hsiao, Nikos Nikoleris, Artem Khyzha, Dominic P. Mulligan, Gustavo Petri, Christopher W. Fletcher, and Caroline Trippel.
Rtl2mμpath: Multi-μpath synthesis with applications to hardware security verification.
In *Proceedings of the 57th IEEE/ACM International Symposium on Microarchitecture (MICRO '24)*, November 2024.

[2] Susumu Mashimo, Akifumi Fujita, Reoma Matsuo, Seiya Akaki, Akifumi Fukuda, Toru Koizumi, Junichiro Kadomoto, Hidetsugu Irie, Masahiro Goshima, Koji Inoue, et al.
An open source fpga-optimized out-of-order risc-v soft processor.
In *2019 International Conference on Field-Programmable Technology (ICFPT)*, pages 63–71. IEEE, 2019.