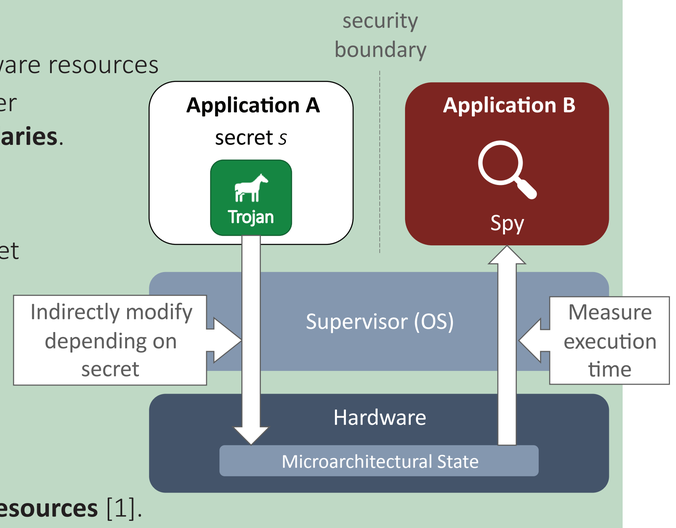
Towards Full Time Protection of an Open-Source, Out-of-Order RISC-V Core

Nils Wistoff[1], Gernot Heiser[2], Luca Benini[1,3]
[1]ETH Zürich, [2]UNSW Sydney, [3]University of Bologna

1. Timing Channels

General Concept:
- Applications **compete** for shared hardware resources
- Levarage timing interferences to transfer information, **bypassing security boundaries**.

Example:
- Trojan: utilise cache depending on secret
- Spy: measure execution time.
- Spy's measured execution time depends on Trojan's cache utilisation depends on secret.



Prevention:
- Let OS **partition all shared hardware resources** [1].
- Spatial partitioning: **divide** hardware resource and allocate different parts.
- Temporal partitioning: **flush** hardware resource when switching between applications [2].

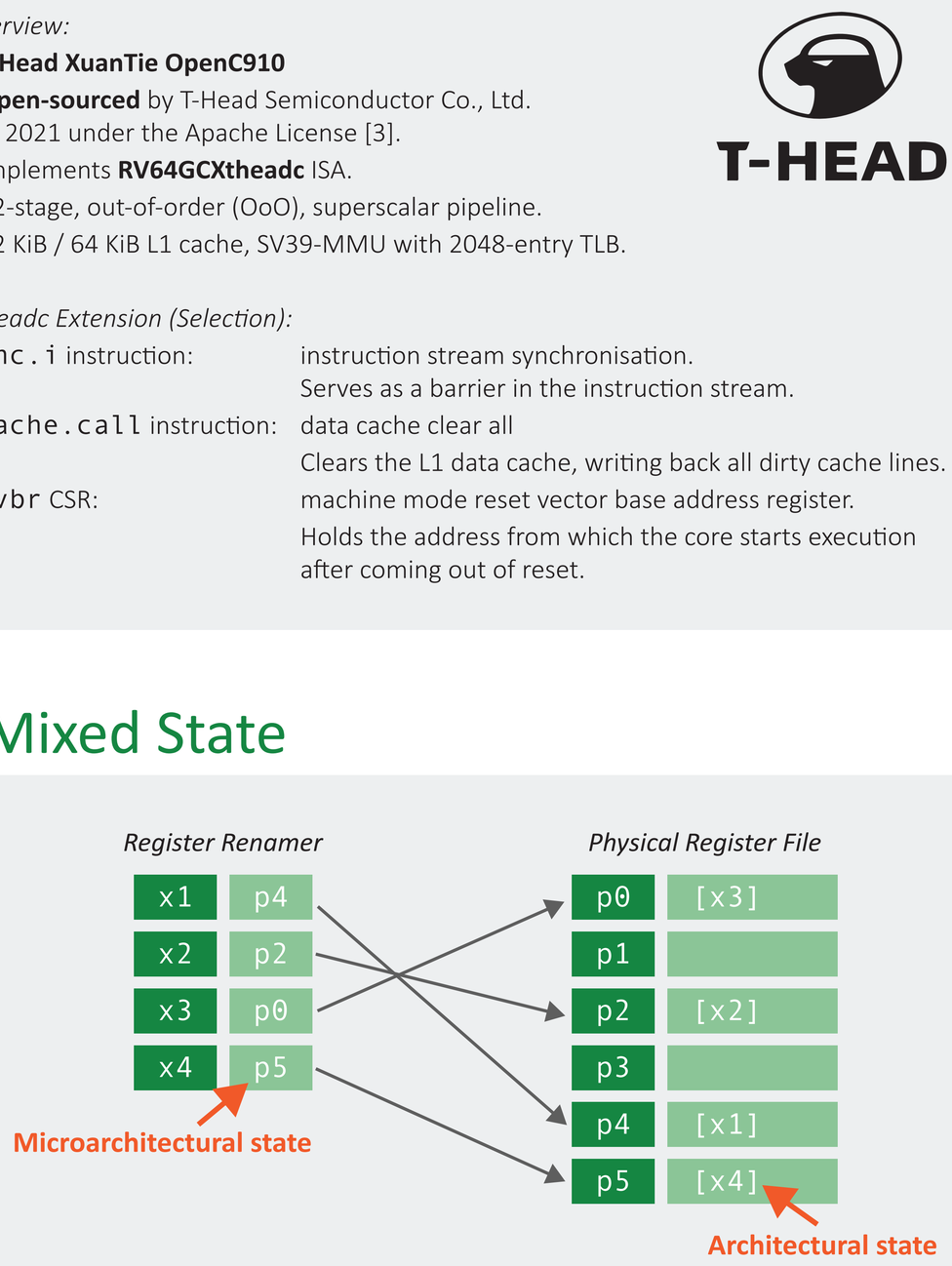
2. OpenC910

Overview:
- **T-Head XuanTie OpenC910**
- **Open-sourced** by T-Head Semiconductor Co., Ltd. in 2021 under the Apache License [3].
- Implements **RV64GCXtheadc** ISA.
- 12-stage, out-of-order (OoO), superscalar pipeline.
- 32 KiB / 64 KiB L1 cache, SV39-MMU with 2048-entry TLB.

Xtheadc Extension (Selection):

`sync.i` instruction:	instruction stream synchronisation. Serves as a barrier in the instruction stream.
`dcache.call` instruction:	data cache clear all. Clears the L1 data cache, writing back all dirty cache lines.
`mrvbr` CSR:	machine mode reset vector base address register. Holds the address from which the core starts execution after coming out of reset.

3. Mixed State



Problem:
Clearing microarchitectural state (renamer) causes loss of architectural state!

Solution:
Save architectural registers onto stack. Clear renamer **and** physical register file on `fence.t`.

4. fence.t in OpenC910

`fence.t`:
Temporal fence instruction that **flushes on-core microarchitectural state** for full temporal partitioning [1].
Experimental integration into OpenC910 where it clears **all** on-core state (except for CSRs).

Step 1: **Save context.**
Save the stack pointer (`sp`) at a known location that is not affected by `fence.t`, e.g. the `mscratch` CSR. Write the architectural registers onto the stack.

Step 2: **Define reset vector.**
Write the address of the instruction following fence.t into the `mrvbr` CSR. Execution after `fence.t` will resume from here.

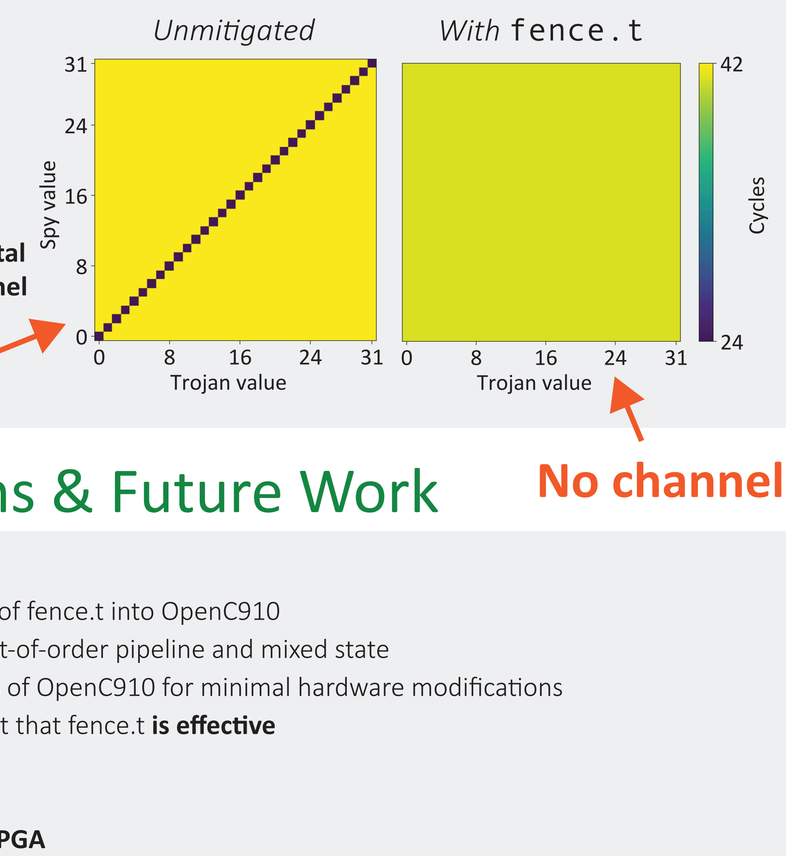
Step 3: **Clear the L1 data cache.**
Execute `dcache.call` to write back dirty cache lines.

Step 4: **Execute `fence.t`.**
This resets the entire core except for the CSR files. We guard the `fence.t` instruction by `sync.i` instructions to ensure that all previous steps have beem completed.

Step 5: **Restore context.**
Restore the stack pointer and load the architectural registers from the stack.

5. Preliminary Results

- Preliminary evaluation using bare-metal **timesec-bench** [4]
- Runs **prime-and-probe** attack on L1 data cache
- **Changes** along a **horizontal cut** indicate **timing channel**



6. Conclusions & Future Work

Conclusions:
- Experimental **integration** of fence.t into OpenC910
- New challenges due to out-of-order pipeline and mixed state
- **Reuse custom extensions** of OpenC910 for minimal hardware modifications
- Preliminary results suggest that fence.t **is effective**

Future work:
- Port modified system to **FPGA**
- Run timing channel benchmarks in **presence of OS**

References

[1] Qian Ge, Yuval Yarom, and Gernot Heiser. "No Security Without Time Protection: We Need a New Hardware-Software Contract". In: APSys'18. ACM, 2018, 1:1–1:9. doi: 10.1145/3265723.3265724.
[2] Nils Wistoff, Moritz Schneider, Frank K. Gürkaynak, Gernot Heiser, and Luca Benini.. "Systematic Prevention of On-Core Timing Channels by Full Temporal Partitioning".In: IEEE Trans. Comput. 72.5 (2023), pp. 1420–1430. doi: 10.1109/TC.2022.3212636.
[3] T-Head Semiconductor Co., Ltd. OpenC910 Core. 2021. url: https://github.com/T-head-Semi/openc910.
[4] Mathieu Escouteloup, Ronan Lashermes, Jacques Fournier, and Jean-Louis Lanet. "Under the dome: preventing hardware timing information leakage". In: CARDIS'21. Nov. 2021, pp. 1–20. url: https://hal.archives- ouvertes.fr/hal- 03351957.

e-mail: nwistoff@iis.ee.ethz.ch