ETH zürich

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA

fence.t: Hardware Support for Preventing Microarchitectural Timing Channels

seL4 Summit 2022

10.10.2022

Nils Wistoff nwistoff@iis.ee.ethz.ch

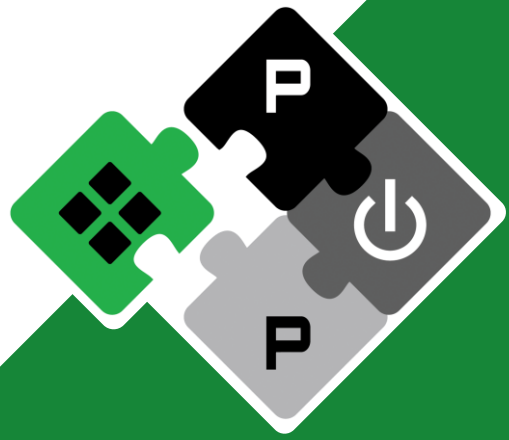
Professors:

Gernot Heiser gernot@unsw.edu.au

Luca Benini lbenini@iis.ee.ethz.ch

PULP Platform

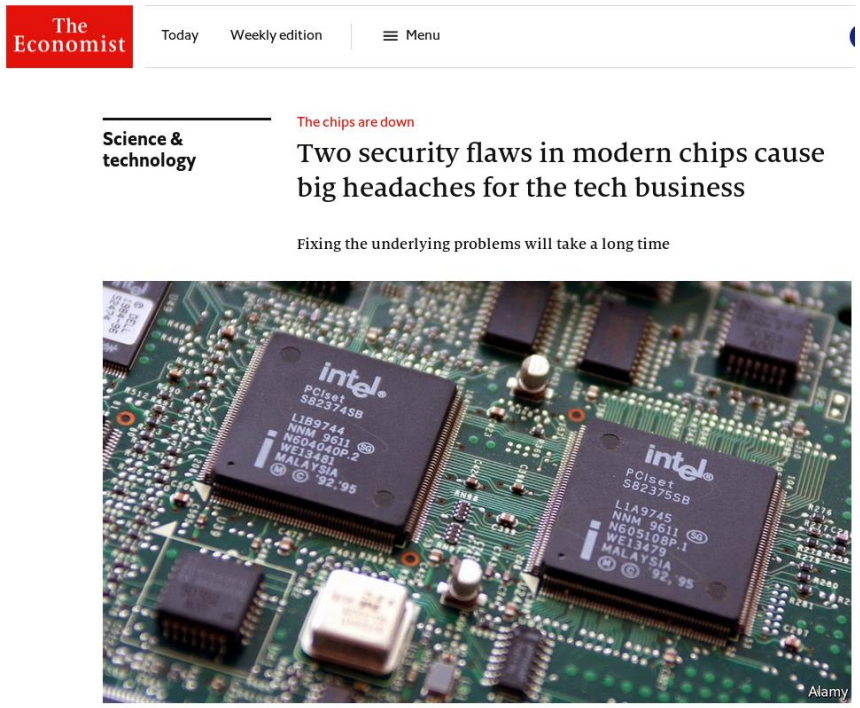
Open Source Hardware, the way it should be!



@pulp_platform

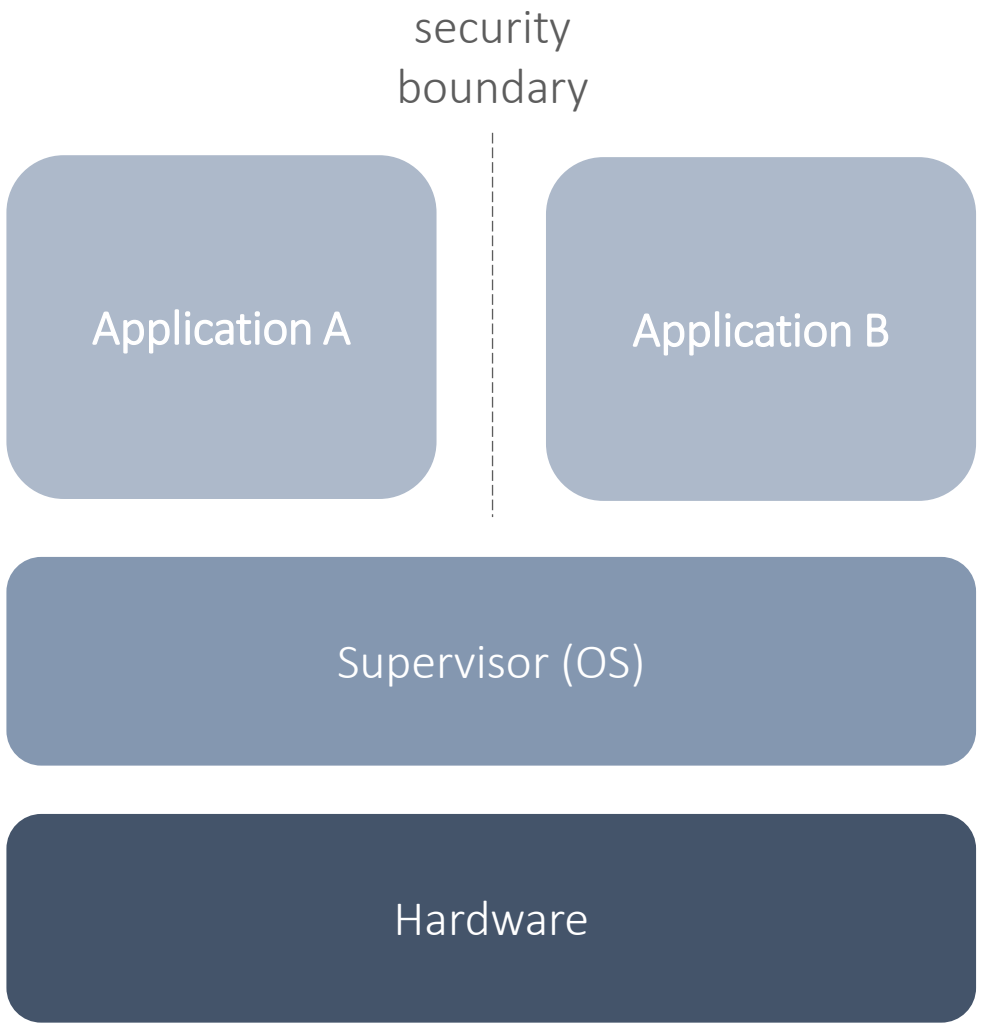
pulp-platform.org

youtube.com/pulp_platform

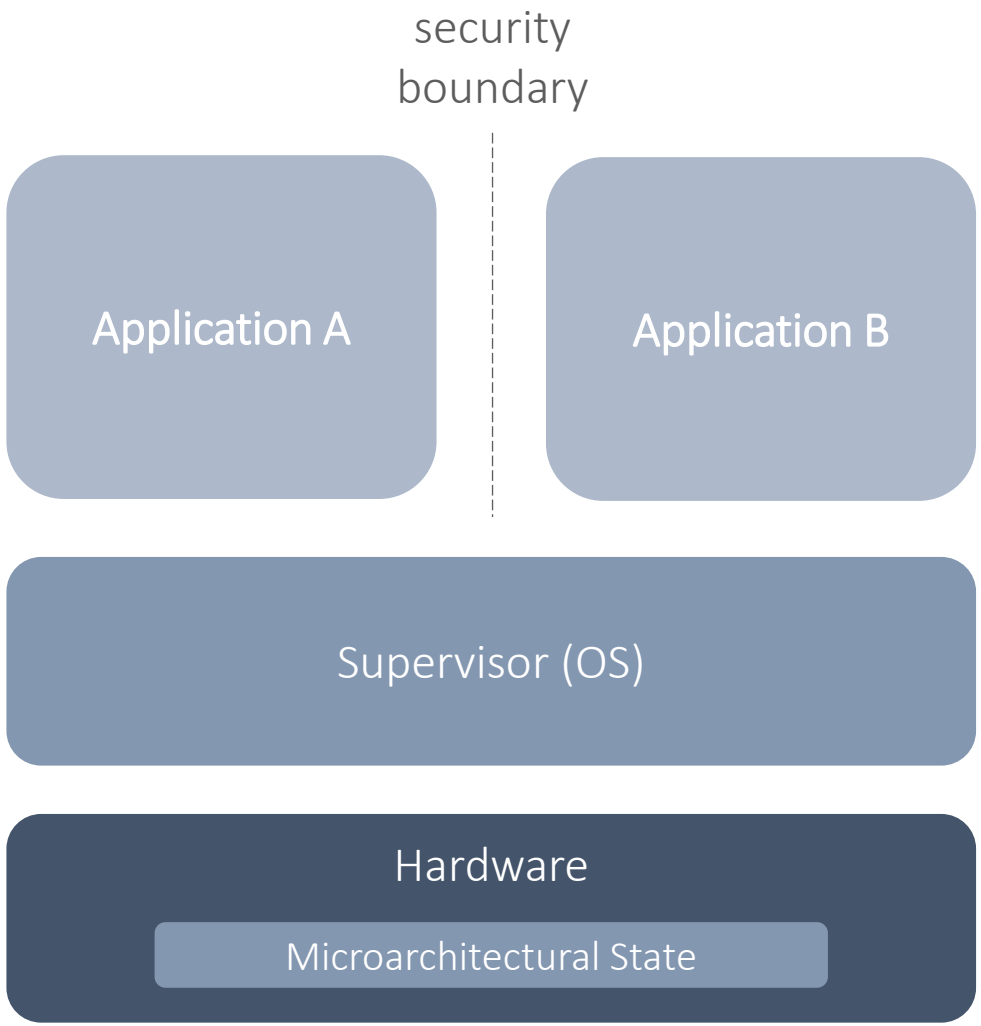


TCs
CVA6
fence.t
CS
Costs
End

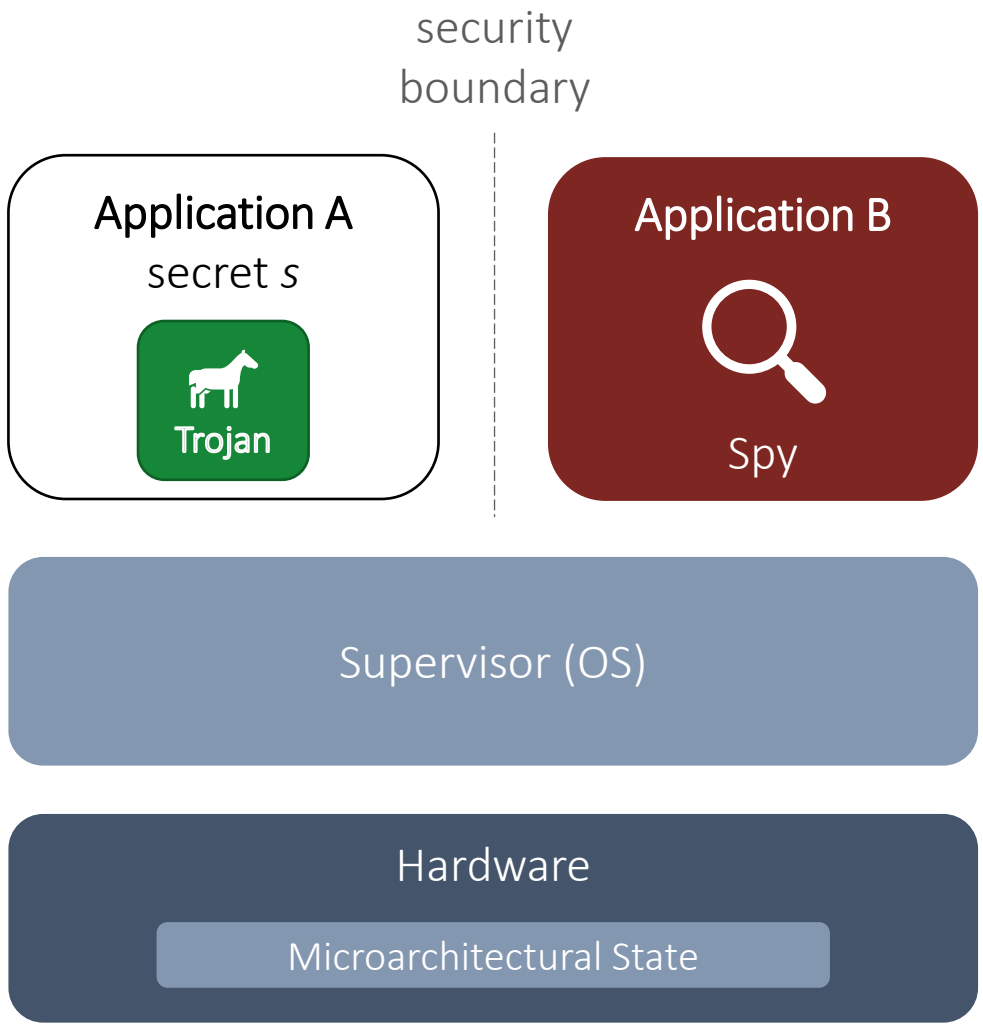
Timing Channel



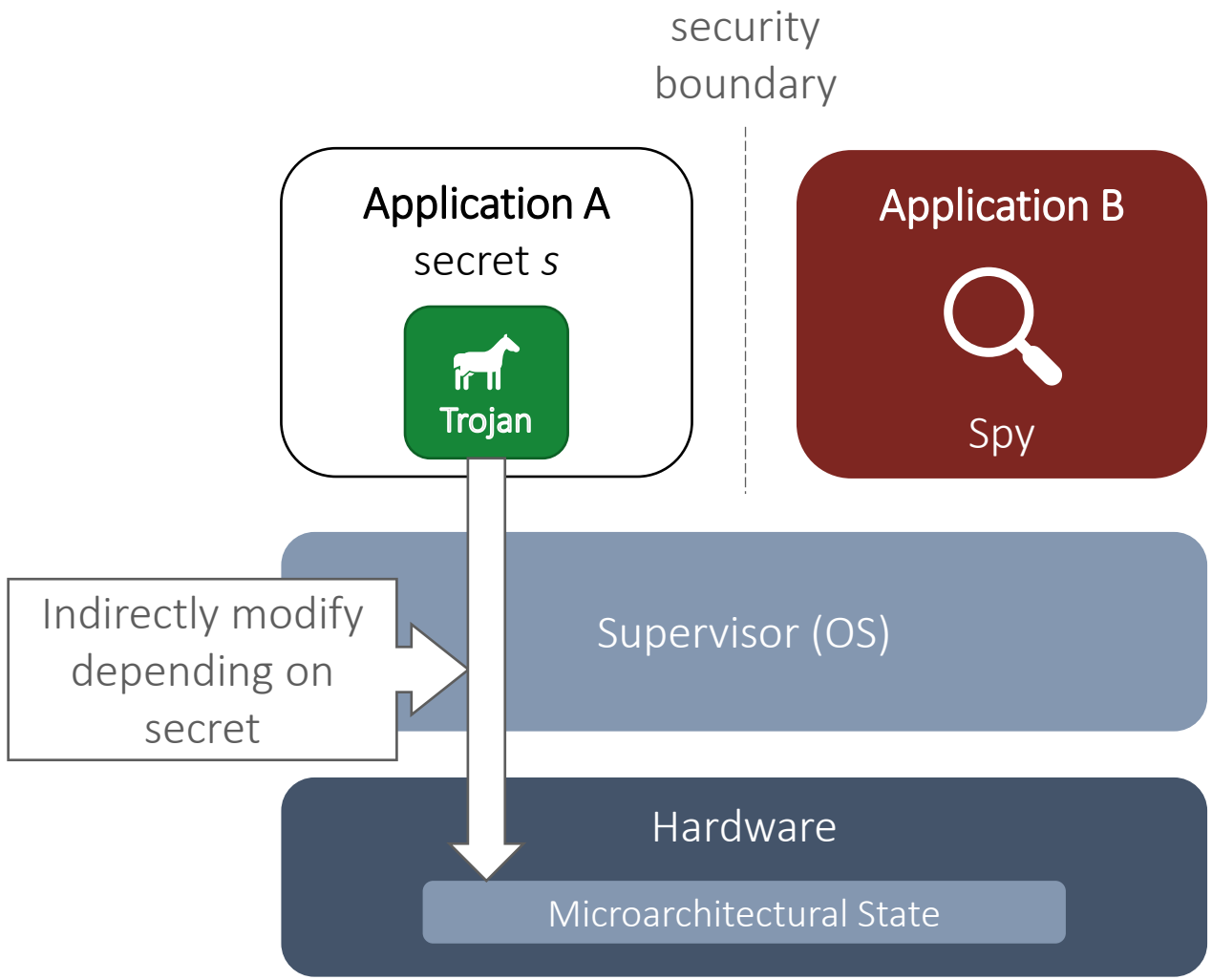
Timing Channel



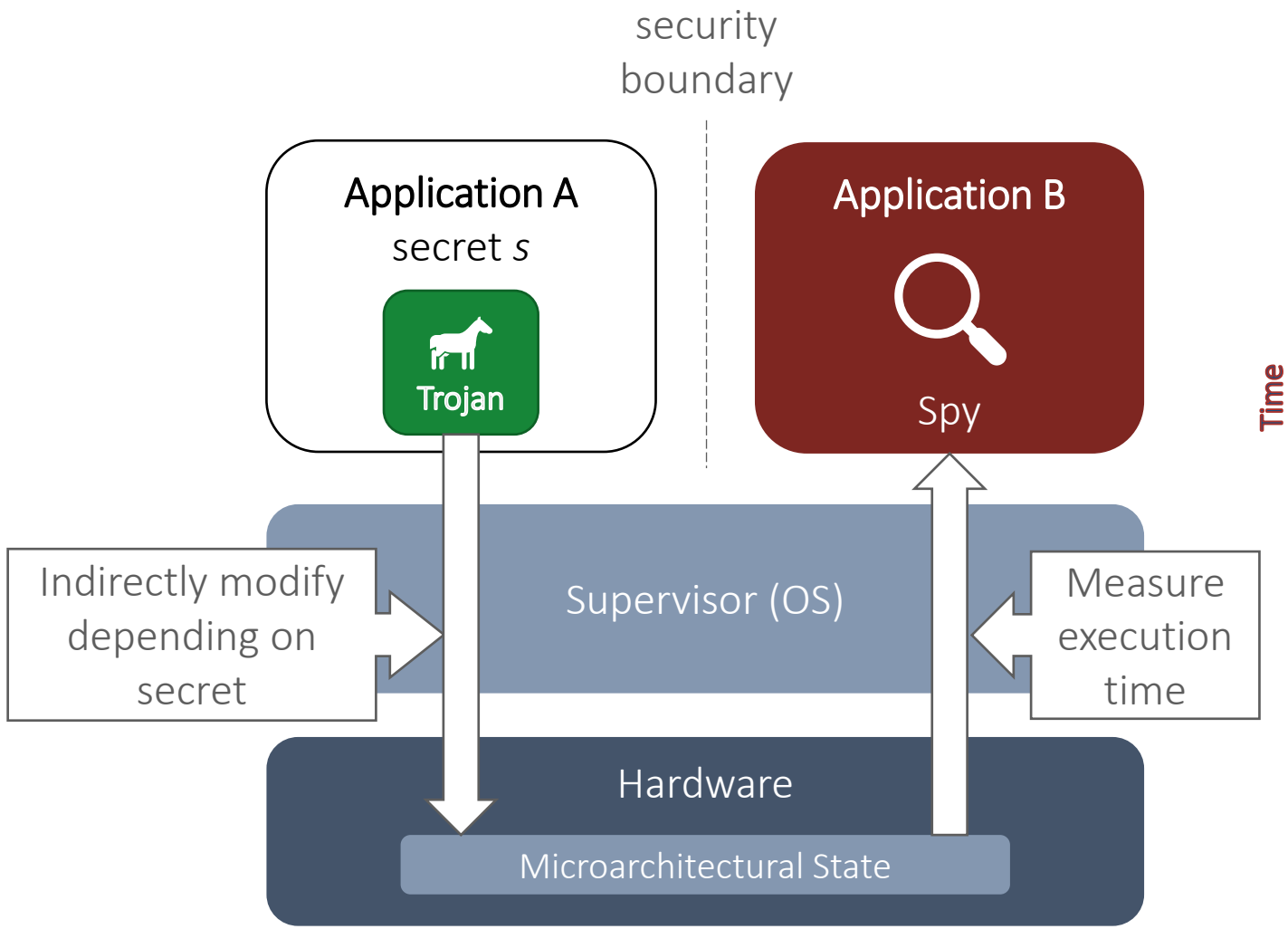
Timing Channel

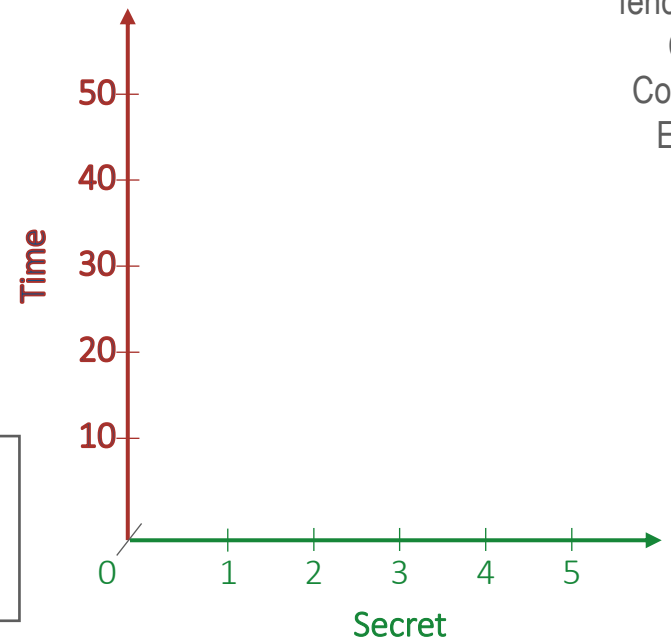


Timing Channel



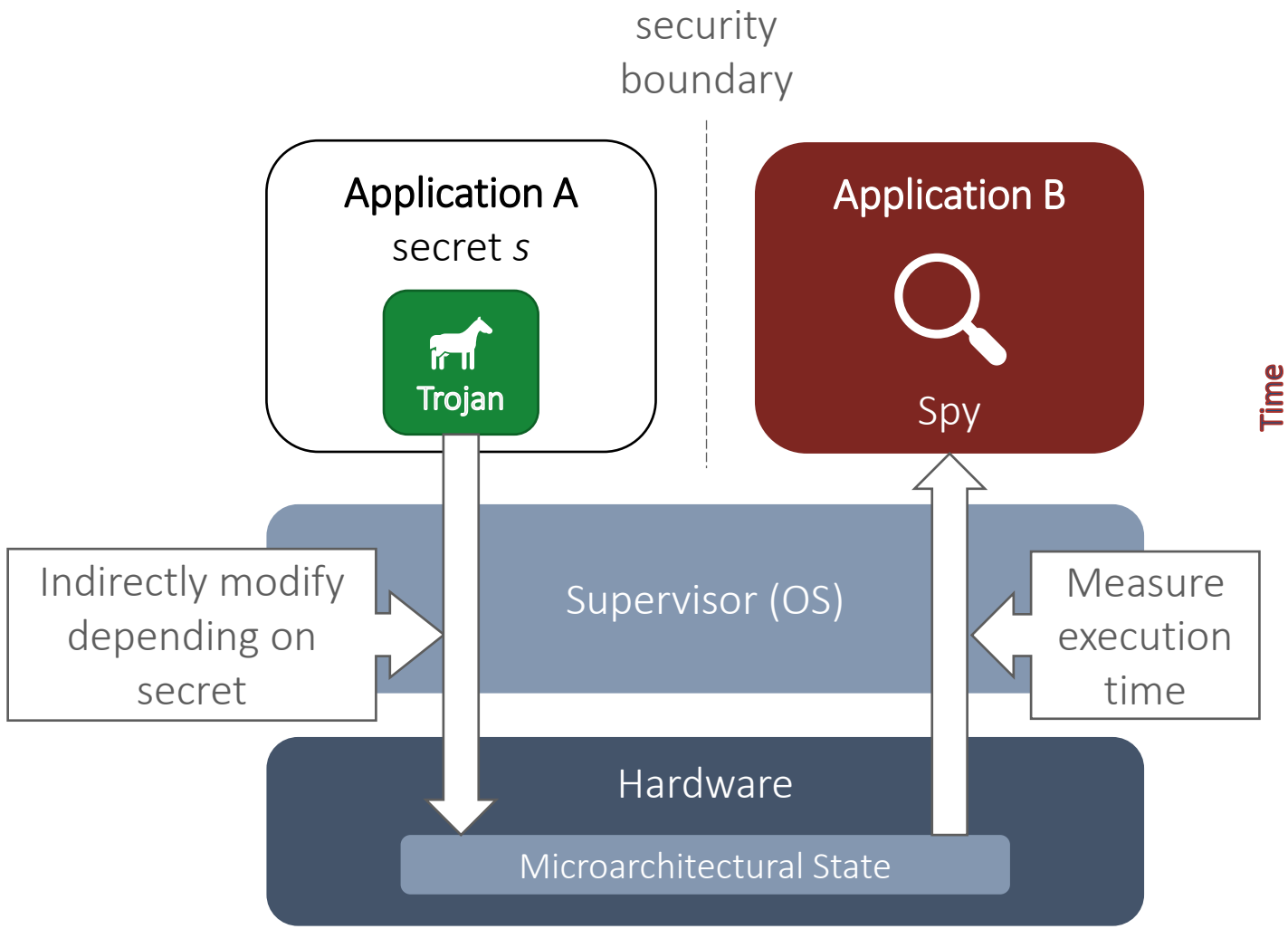
Timing Channel

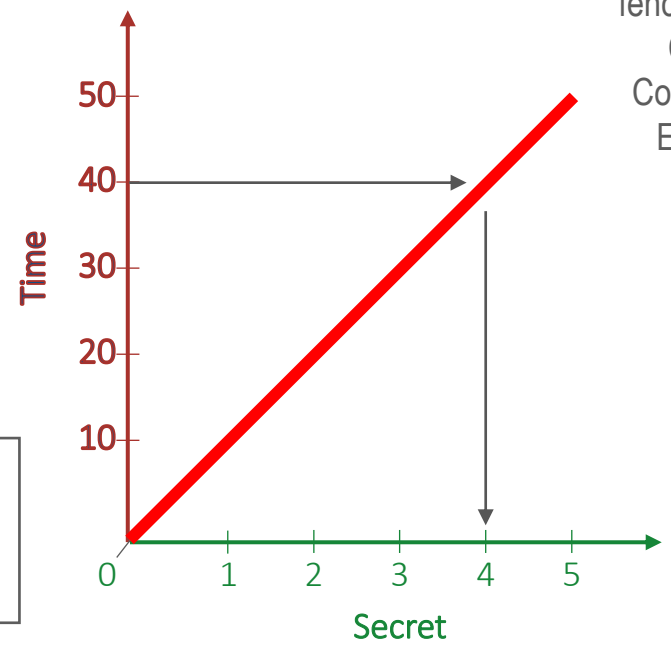




TCs
CVA6
fence.t
CS
Costs
End

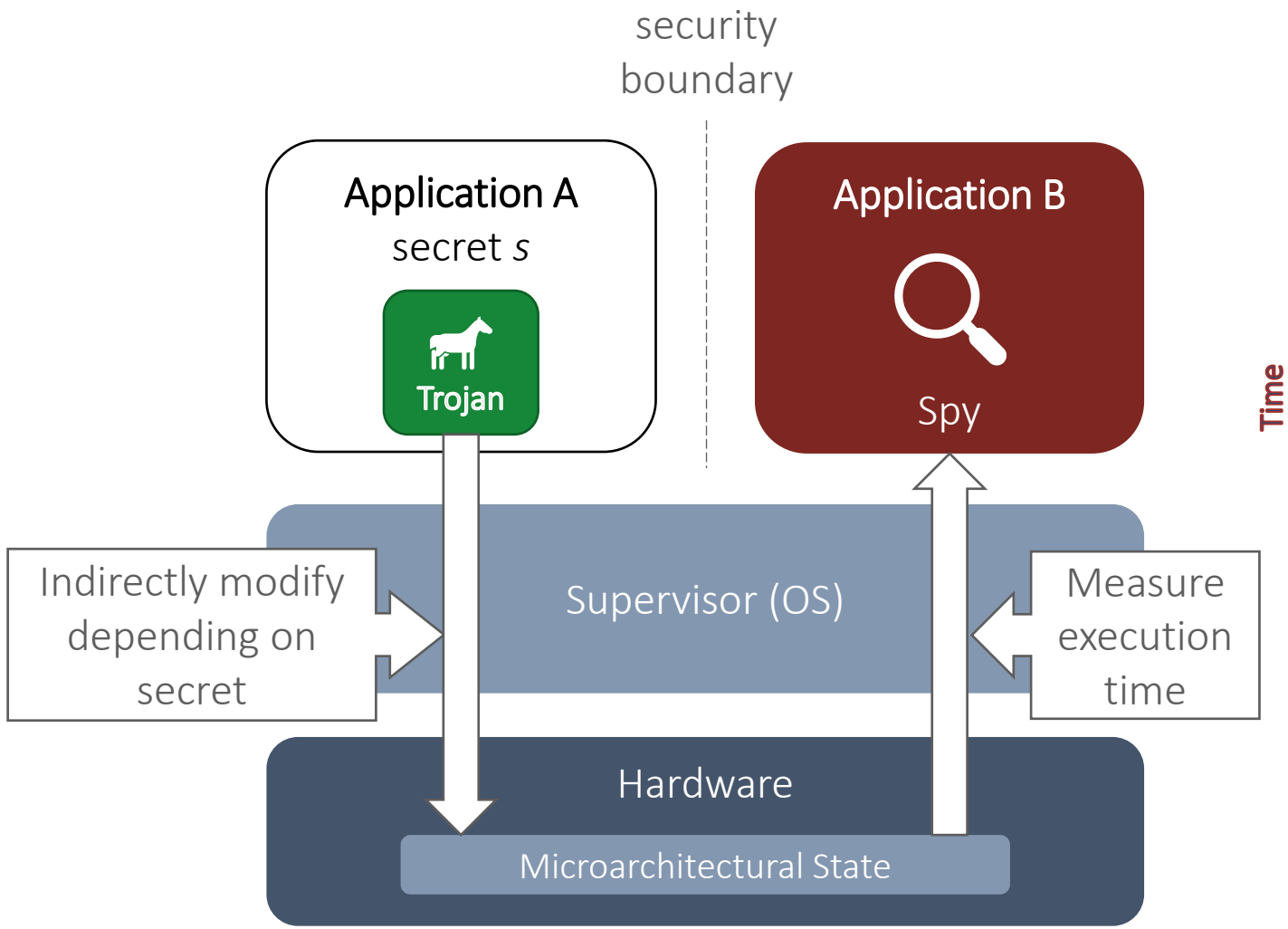
Timing Channel

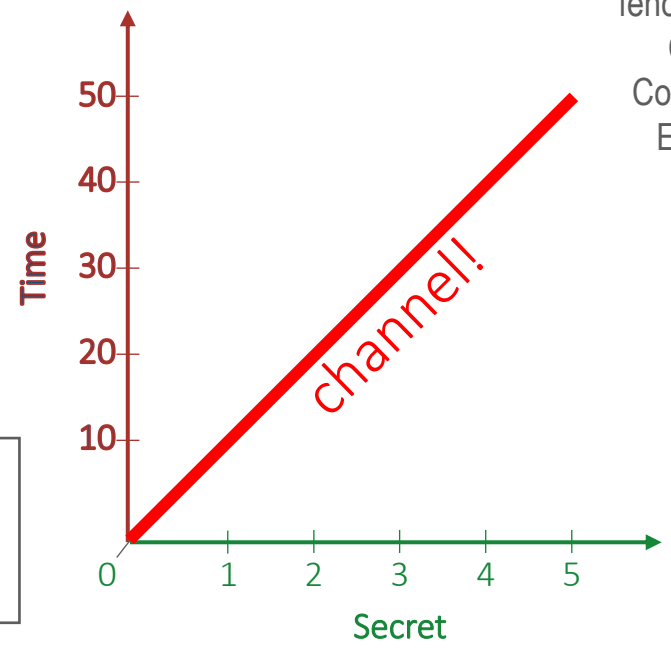




TCs
CVA6
fence.t
CS
Costs
End

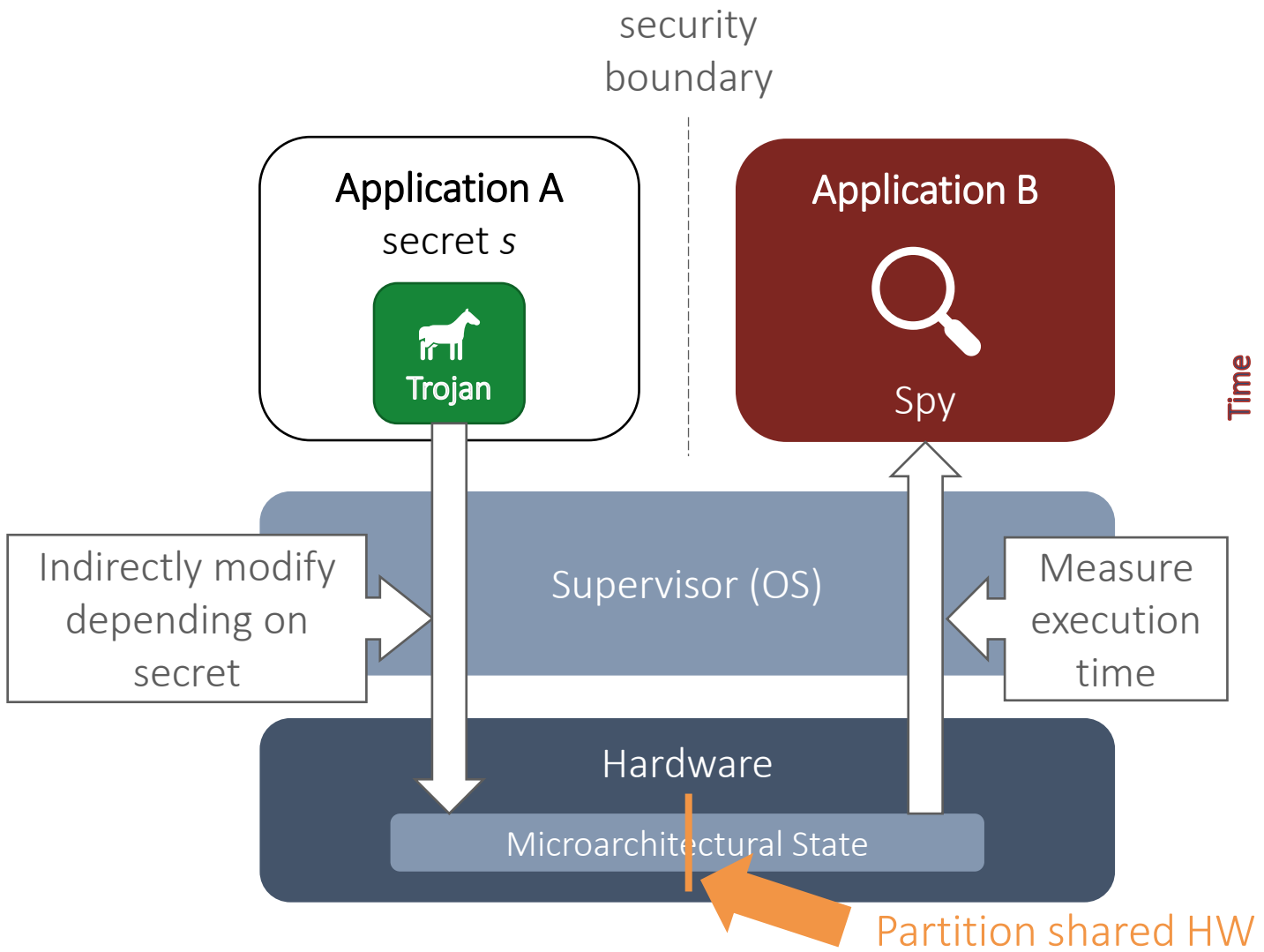
Timing Channel

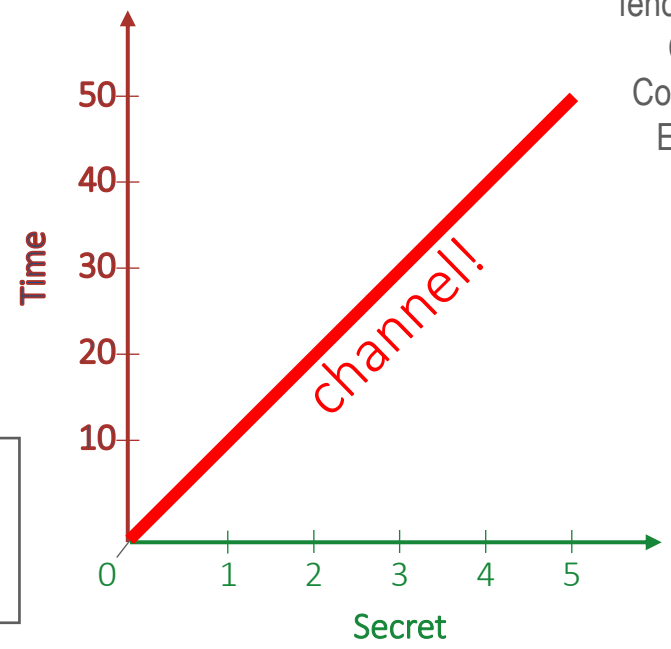




TCs
CVA6
fence.t
CS
Costs
End

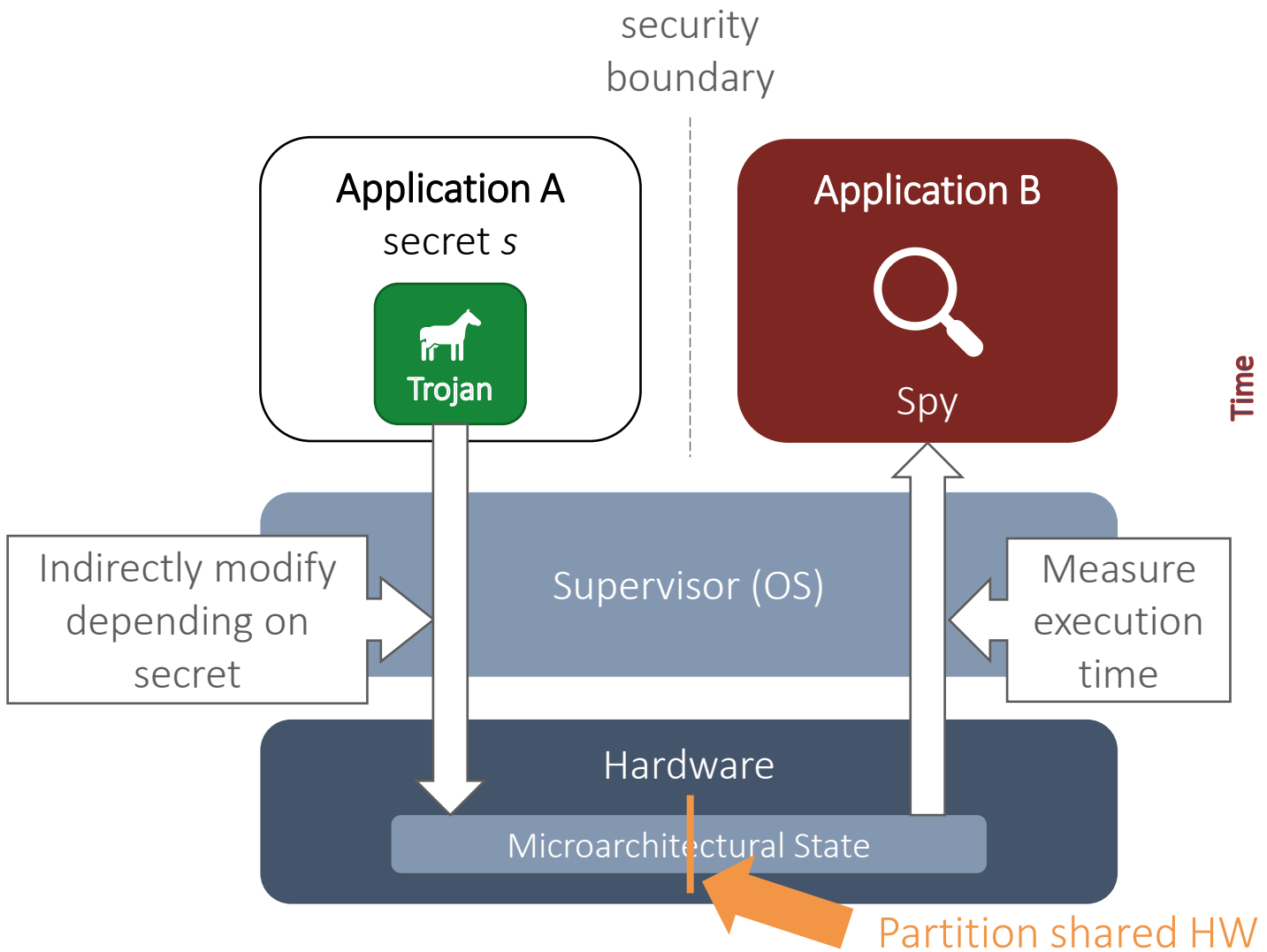
Timing Channel

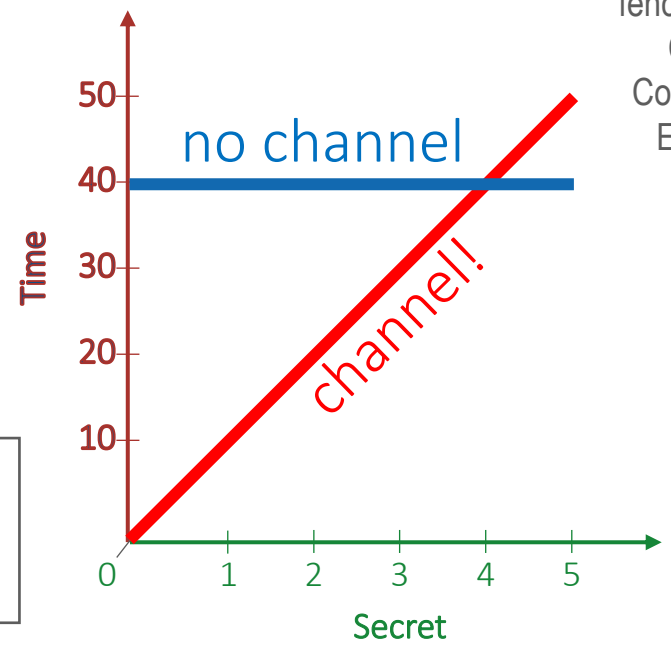




TCs
CVA6
fence.t
CS
Costs
End

Timing Channel



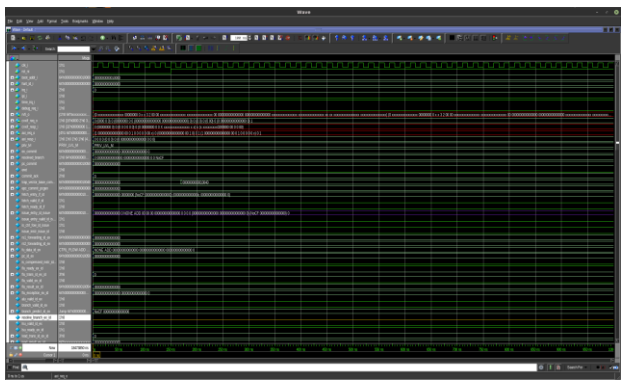


TCs
CVA6
fence.t
CS
Costs
End

CVA6 (Ariane)

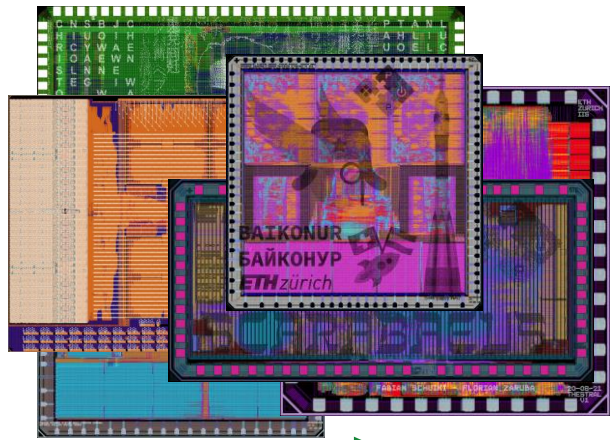
- Open-source 64-bit application-class RISC-V processor
- Boots Linux (or seL4)
- Developed by PULP team at ETH
- Now owned and maintained by OpenHW Group
- Widely used in academia and industry

RTL Simulation



FPGA Emulation

ASIC



speed, turn-around time, cost

Timing Channels on CVA6

Application

Channel bench

+

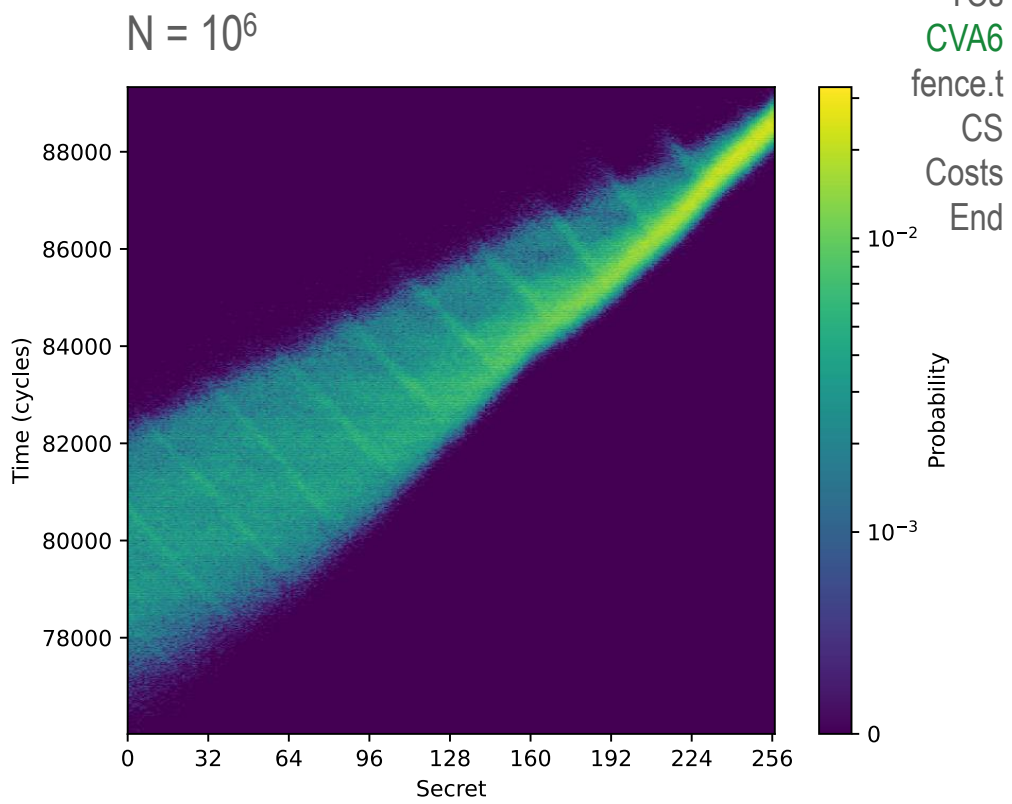
OS

seL4

Security. Performance. Proof.

+

Hardware Platform



Timing Channels on CVA6

Application

Channel bench

+

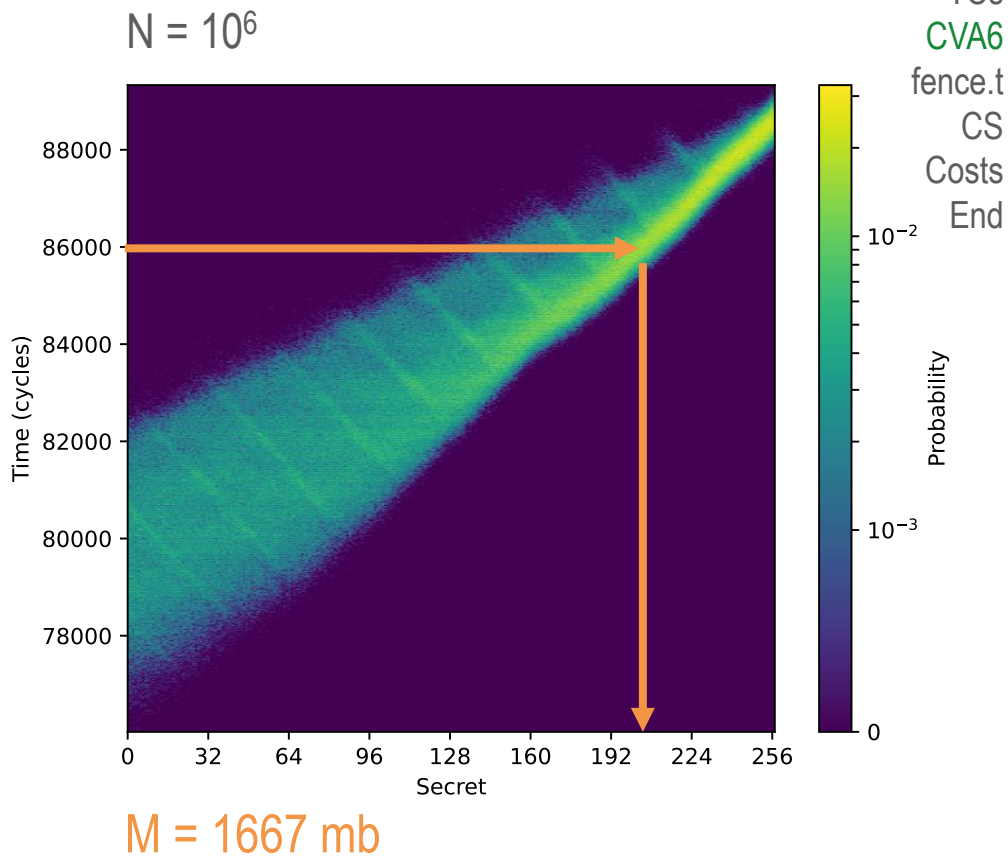
OS

seL4

Security. Performance. Proof.

+

Hardware Platform



Timing Channels on CVA6

Application

Channel bench

+

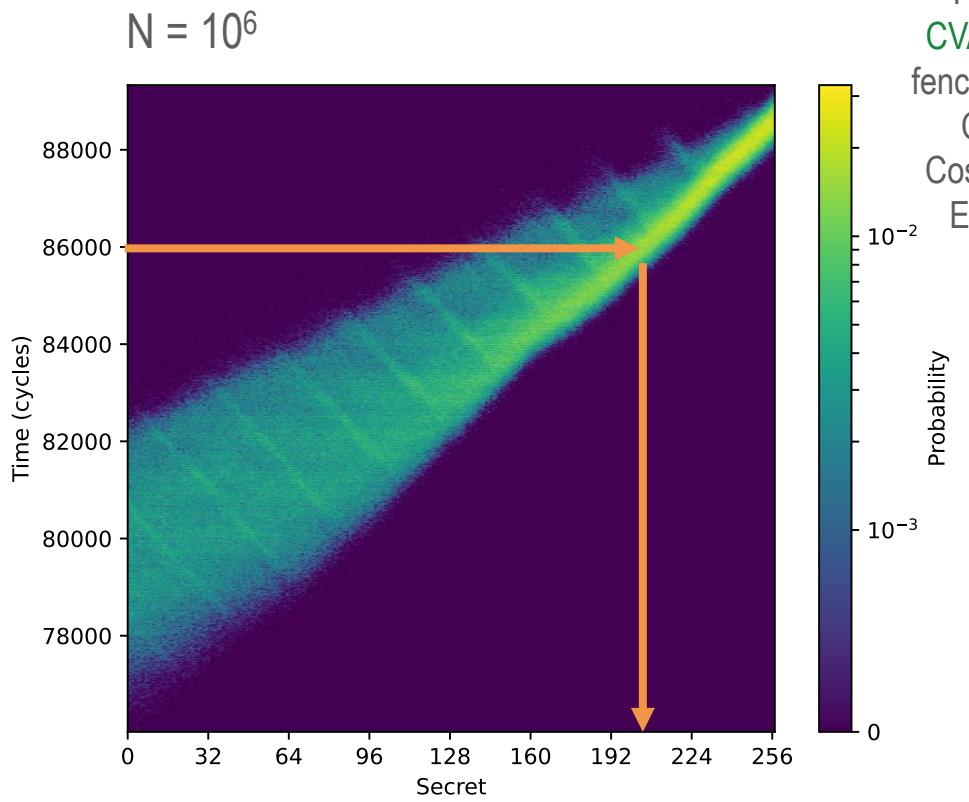
OS

seL4

Security. Performance. Proof.

+

Hardware Platform

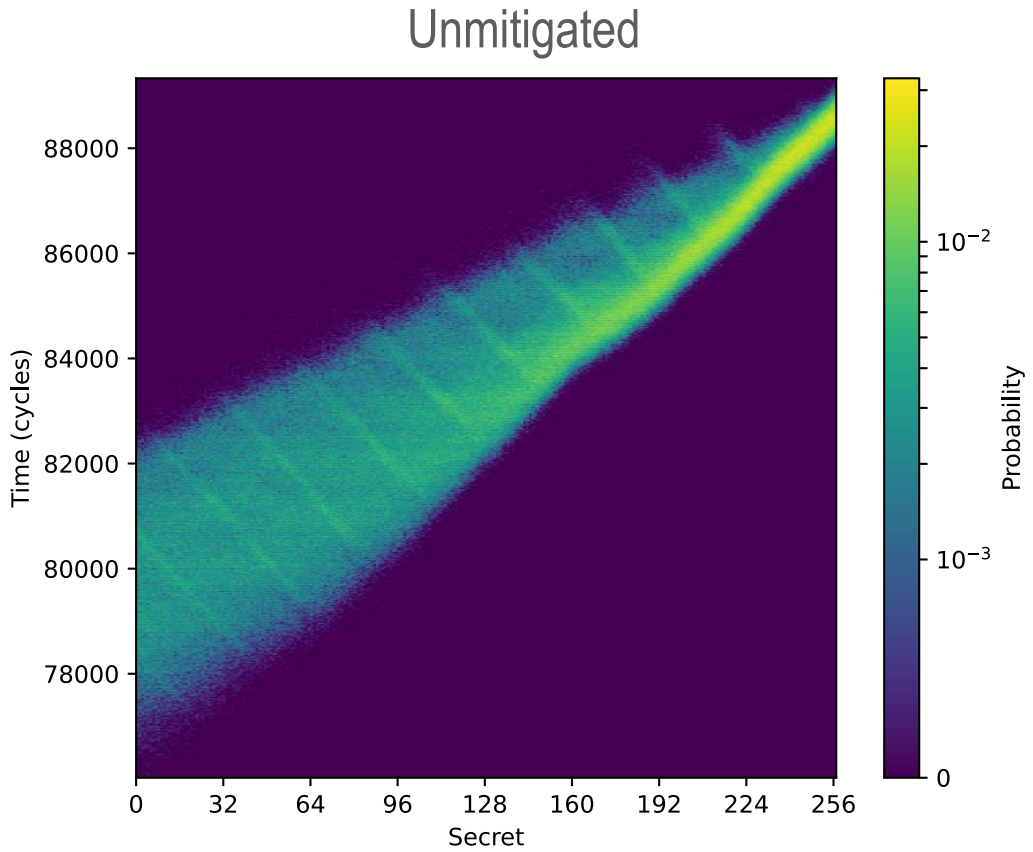


M = 1667 mb

M_0 = 0.5 mb

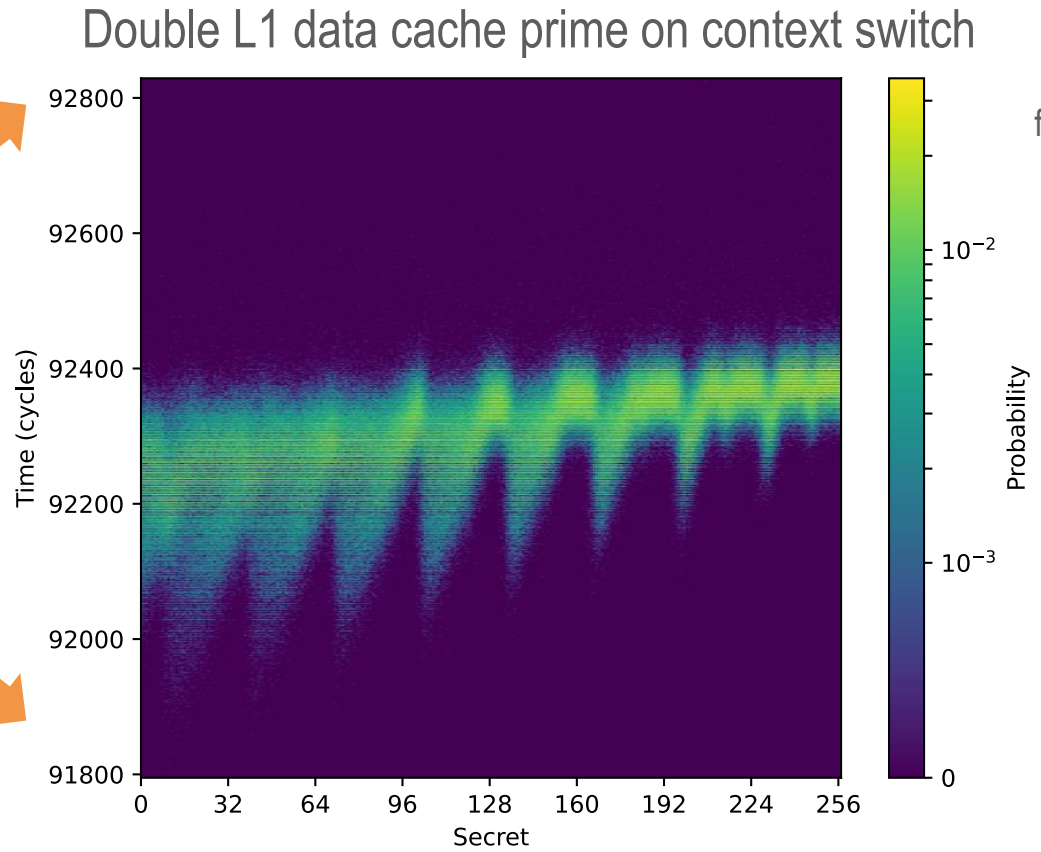
Characterises noise

L1 Data Cache: Software Mitigation



$N = 10^6$, $M = 1667.3$ mb, $M_0 = 0.5$ mb

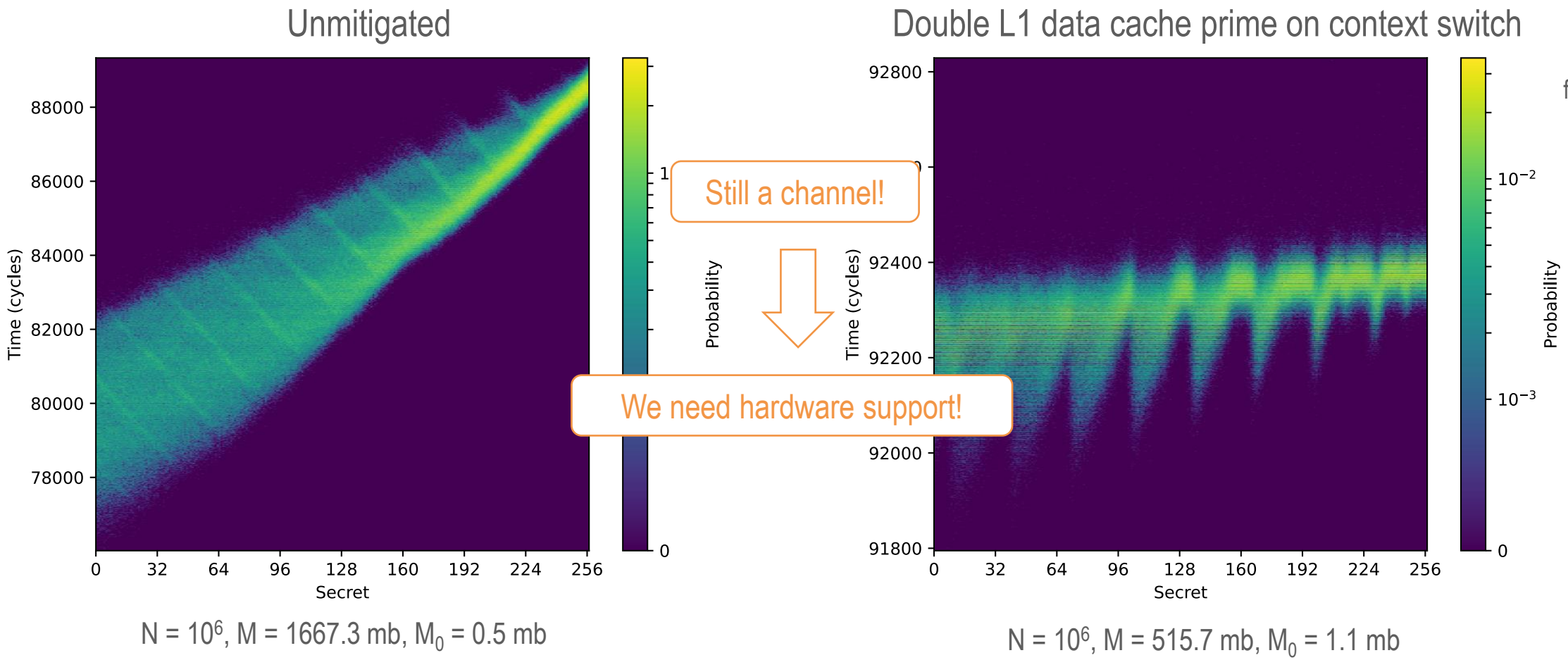
Reduced Range



$N = 10^6$, $M = 515.7$ mb, $M_0 = 1.1$ mb

TCs
CVA6
fence.t
CS
Costs
End

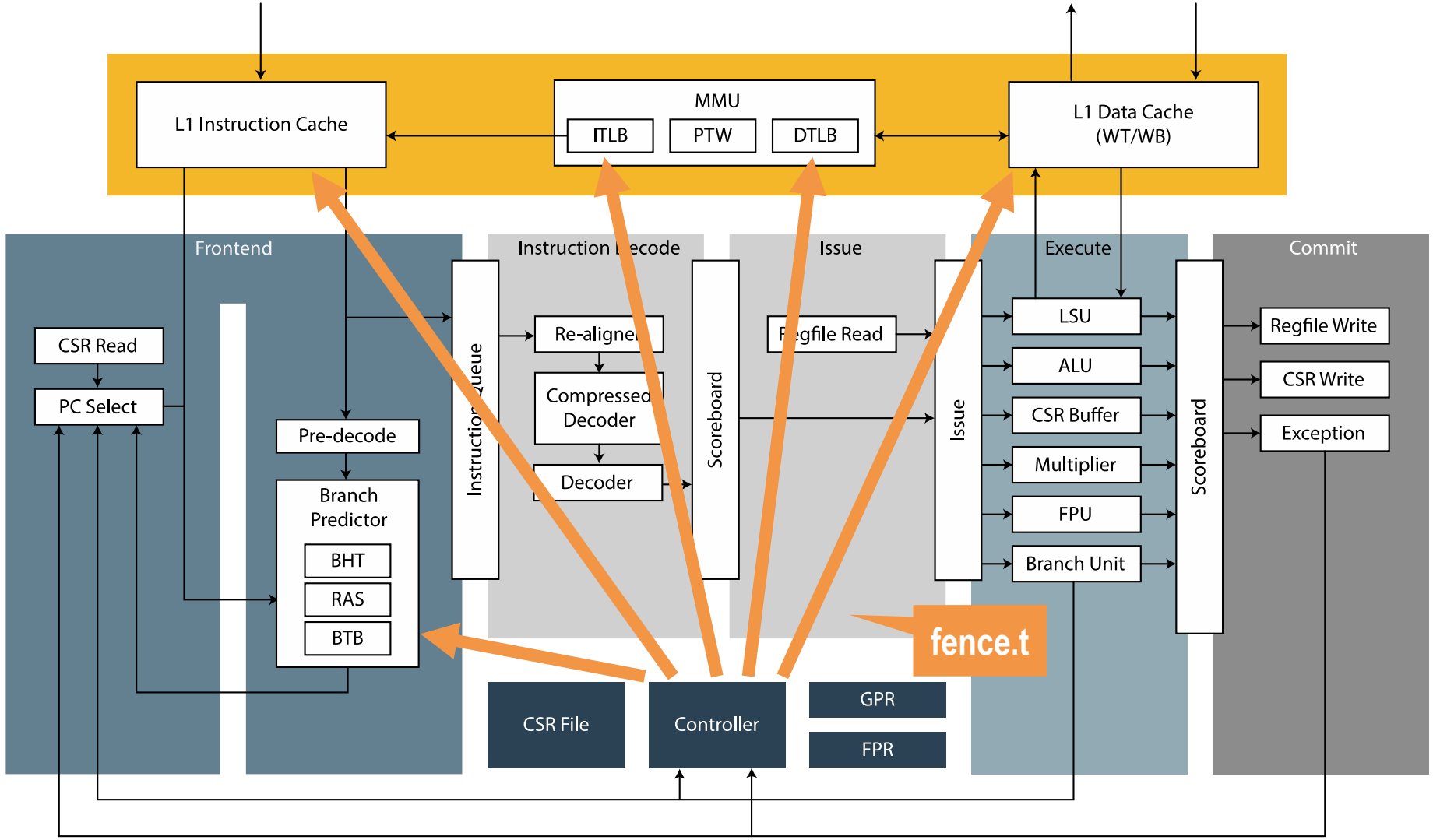
L1 Data Cache: Software Mitigation



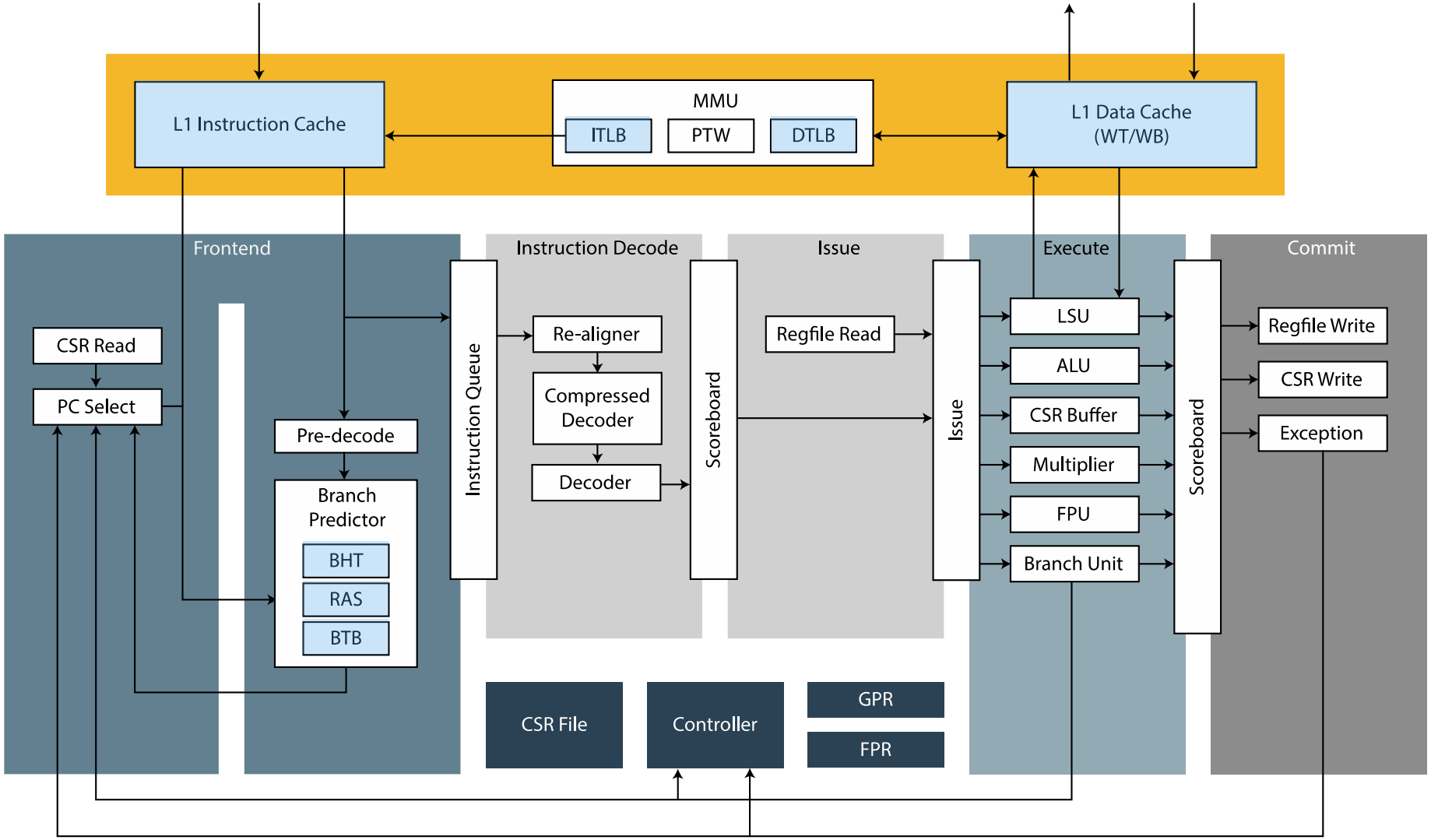
$N = 10^6$, $M = 1667.3$ mb, $M_0 = 0.5$ mb

$N = 10^6$, $M = 515.7$ mb, $M_0 = 1.1$ mb

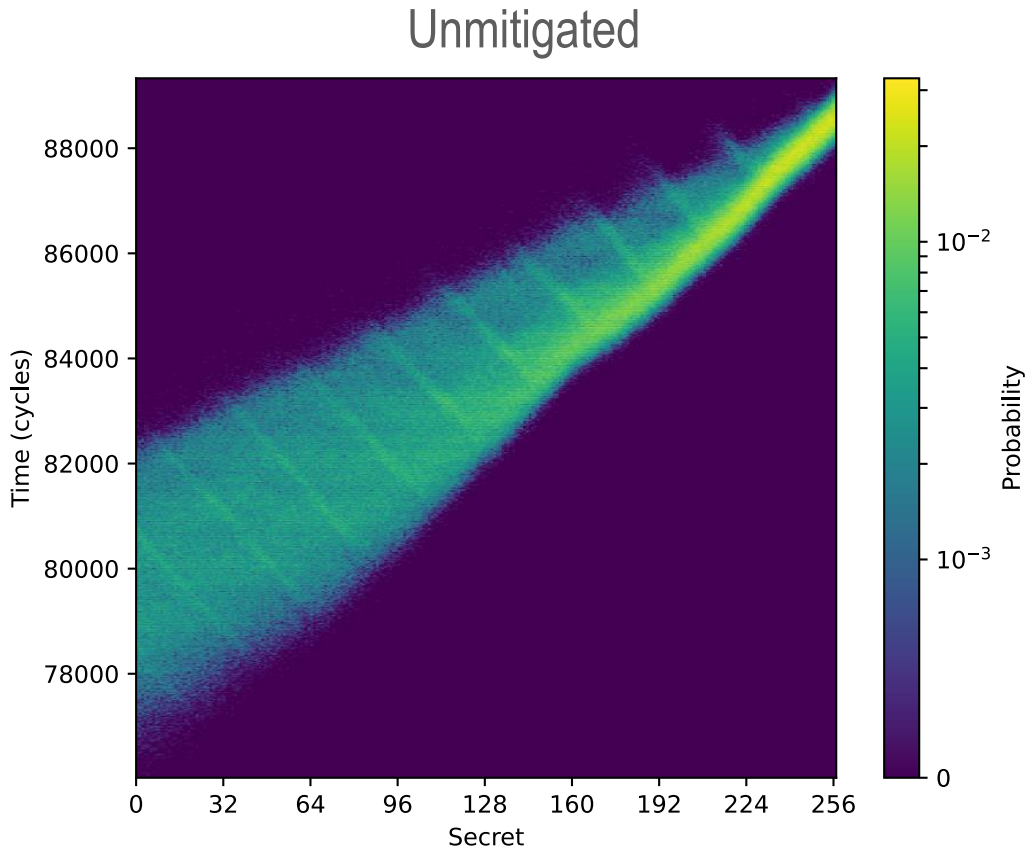
fence.t Instruction



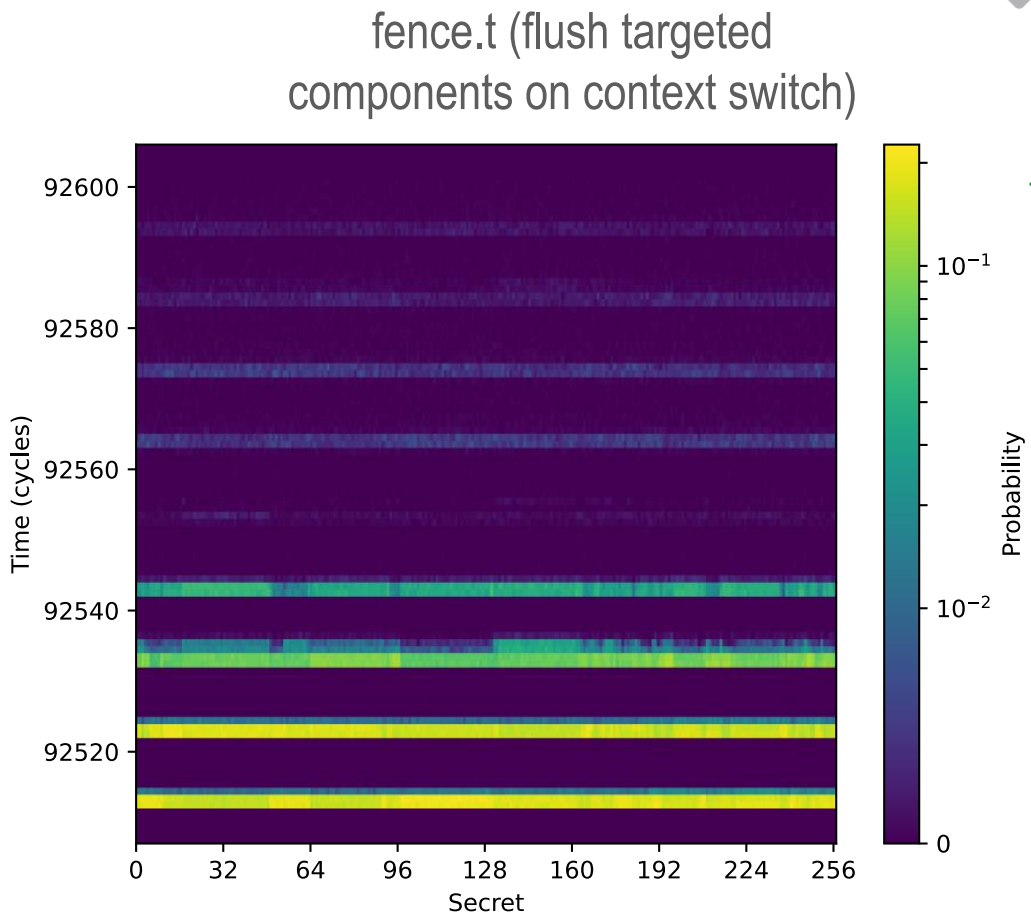
fence.t Implementation: Basic Flush



L1 Data Cache: Basic Flush



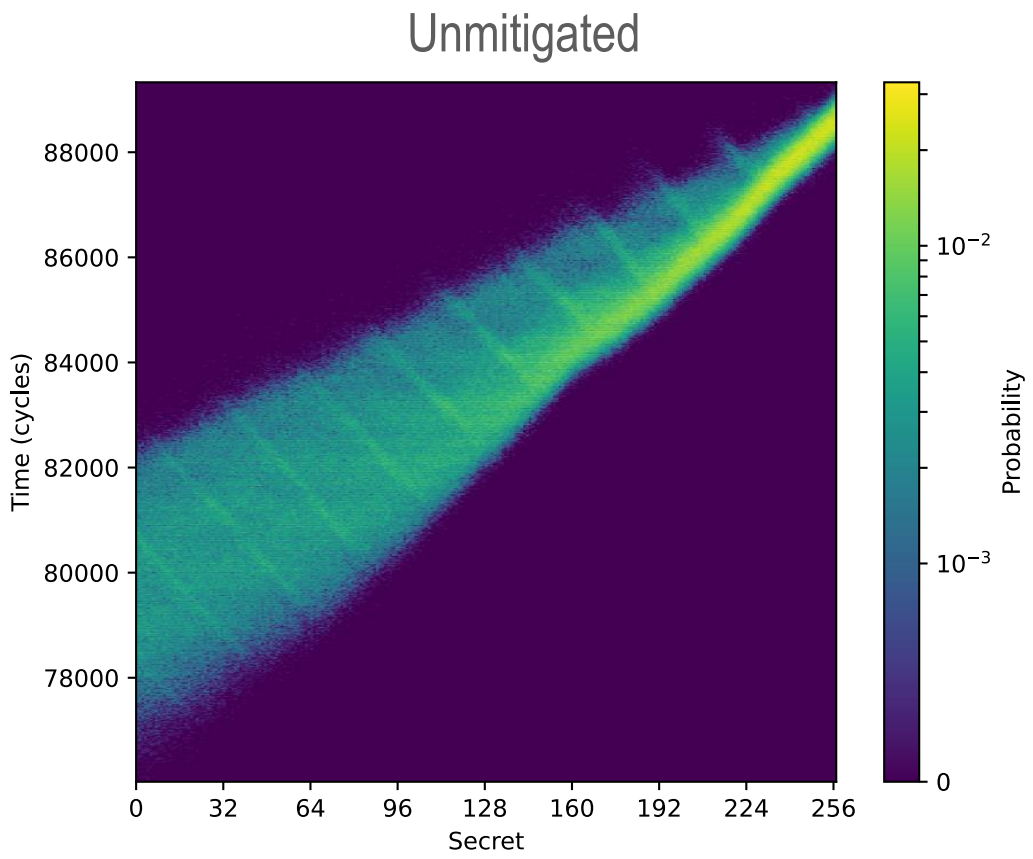
$N = 10^6$, $M = 1667.3$ mb, $M_0 = 0.5$ mb



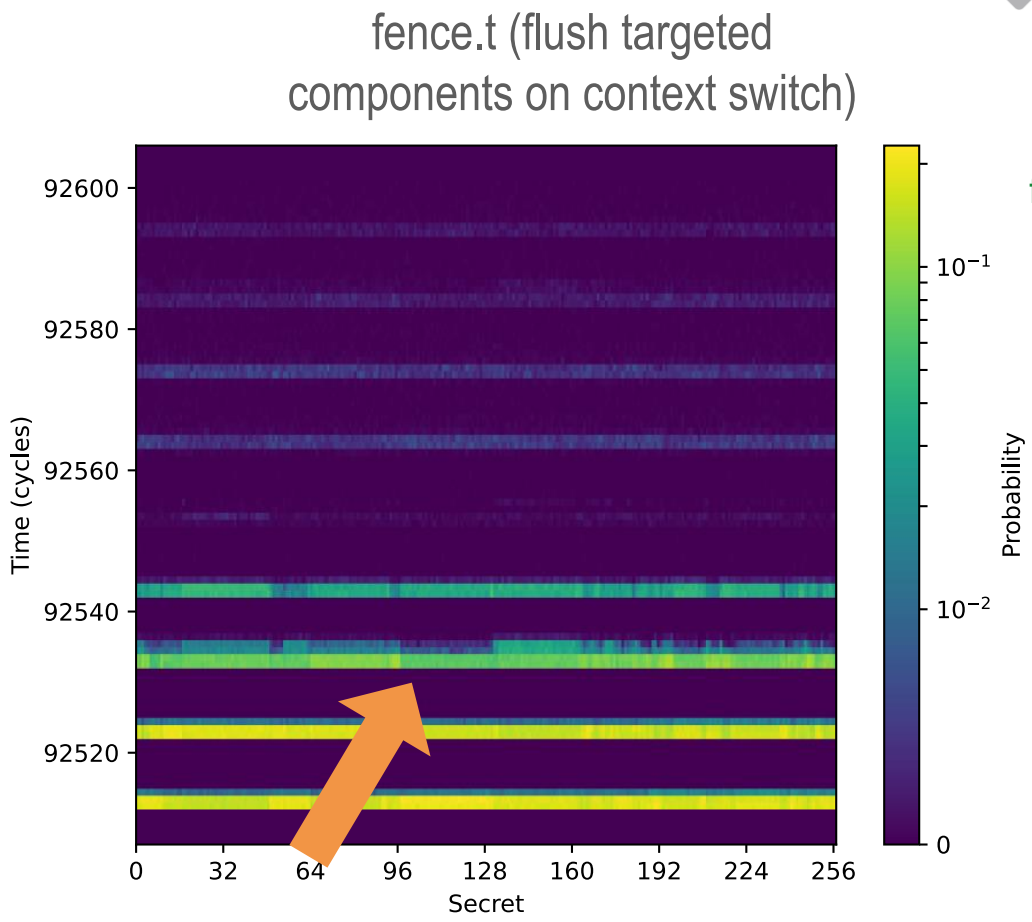
$N = 10^6$, $M = 7.7$ mb, $M_0 = 1.4$ mb

TCs
CVA6
fence.t
CS
Costs
End

L1 Data Cache: Basic Flush



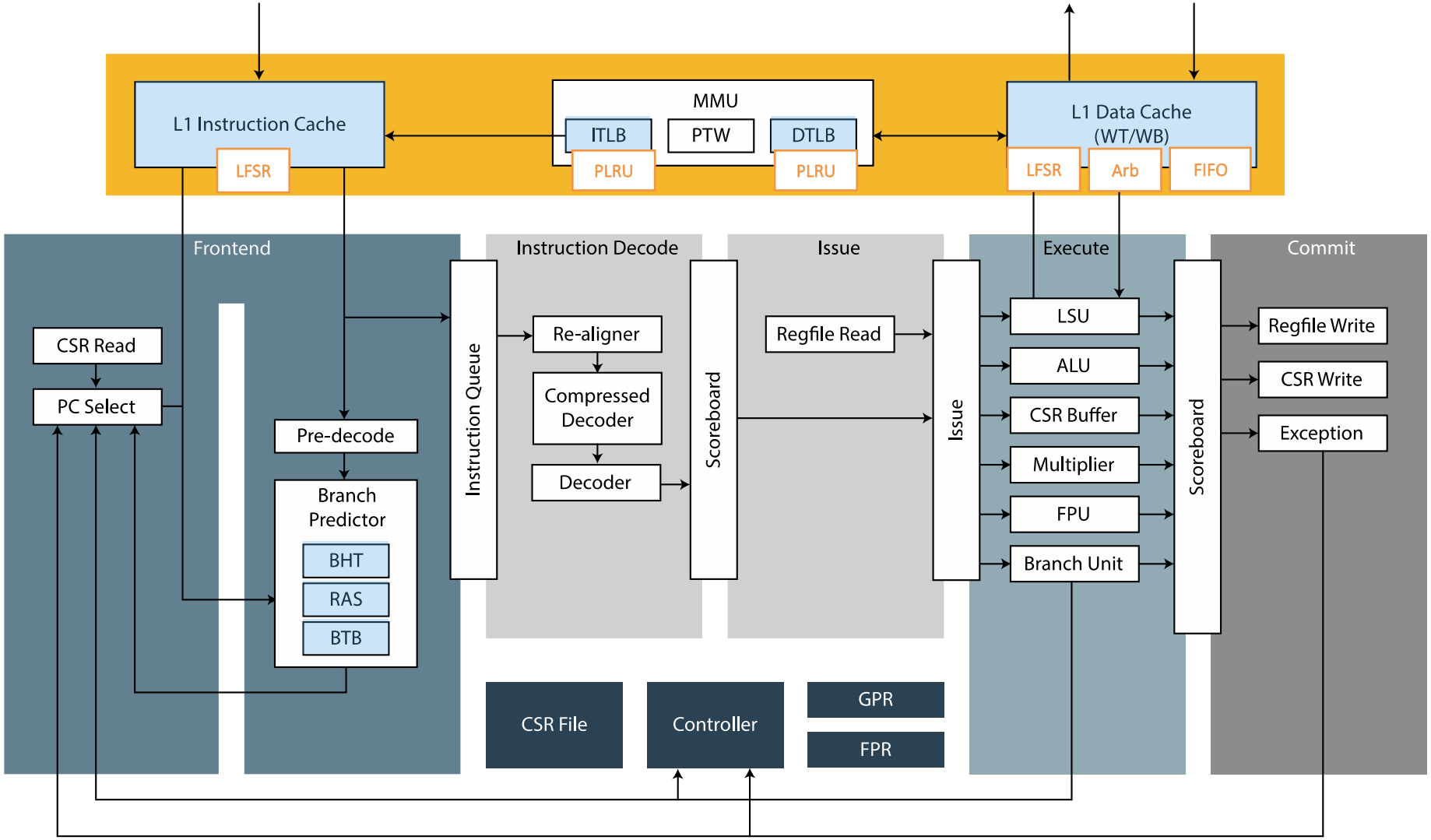
$N = 10^6$, $M = 1667.3$ mb, $M_0 = 0.5$ mb



$N = 10^6$, $M = 7.7$ mb, $M_0 = 1.4$ mb

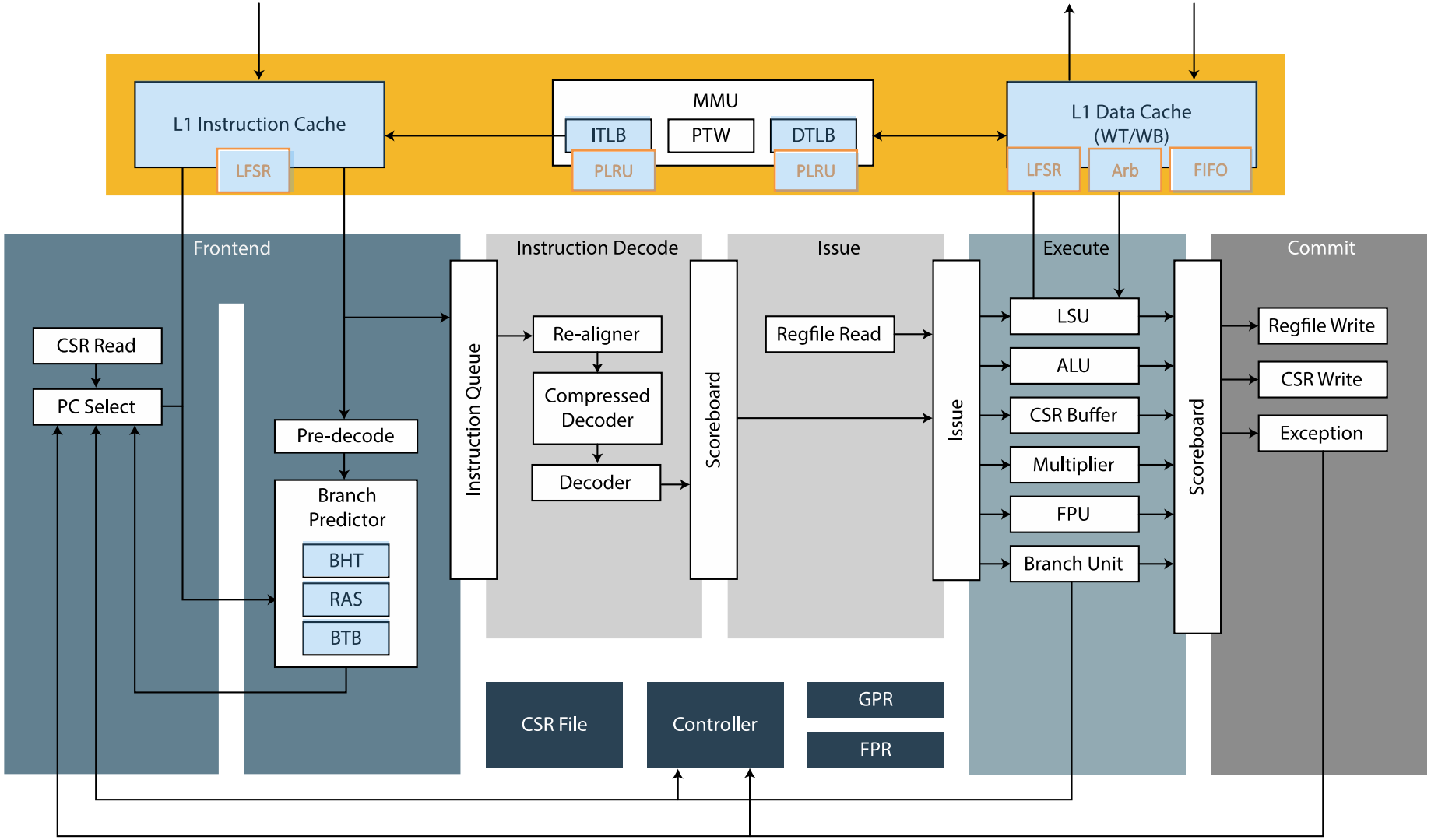
TCs
CVA6
fence.t
CS
Costs
End

2nd Order Microarchitectural State

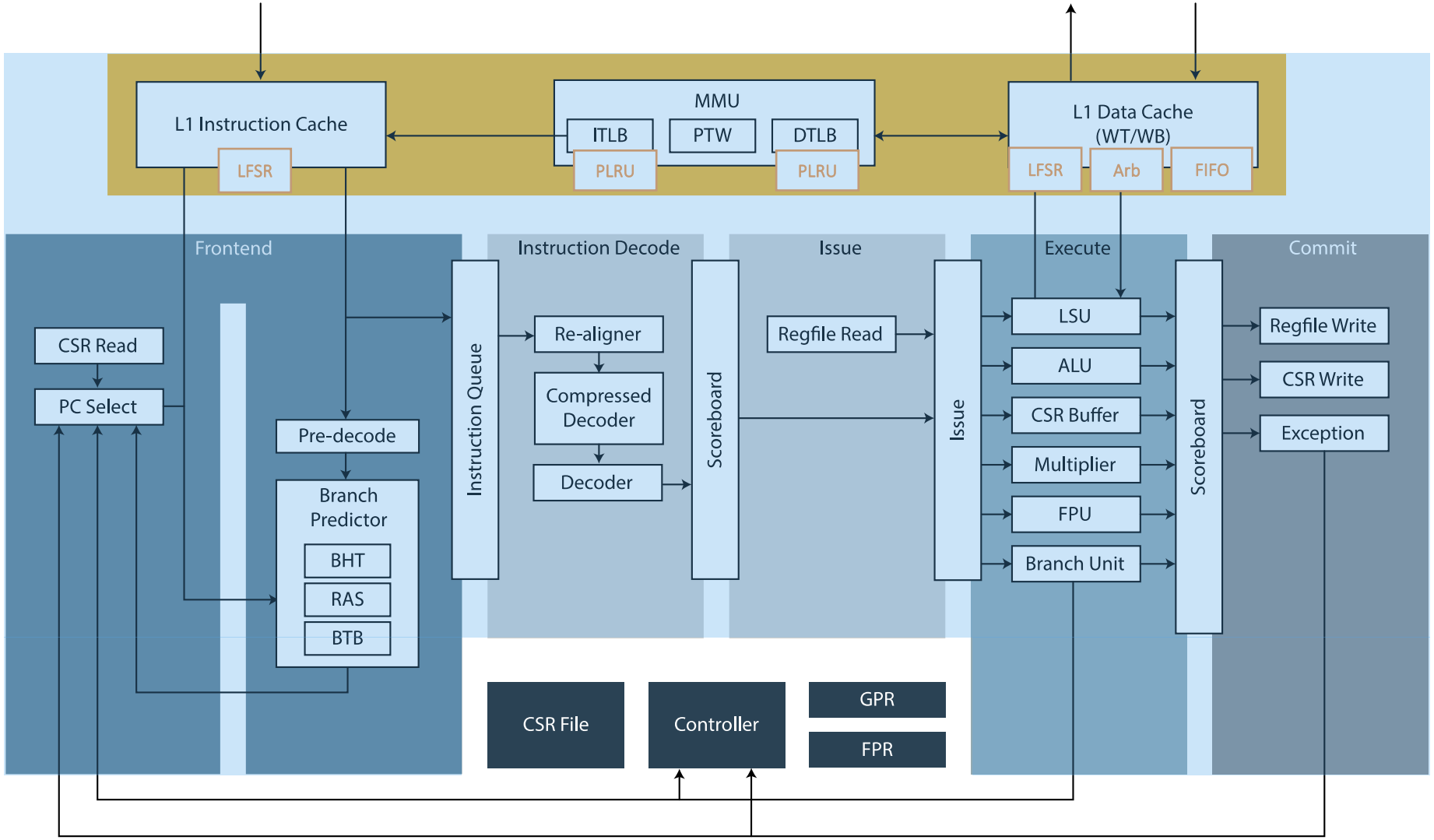


TCs
CVA6
fence.t
CS
Costs
End

fence.t Implementation: Extensive Selective Flush



fence.t Implementation: Microreset



fence.t Implementation: Microreset

Step 1: Save the program counter.

Step 2: Save locally modified (dirty) architectural state.
e.g. dirty cache lines in a write-back cache

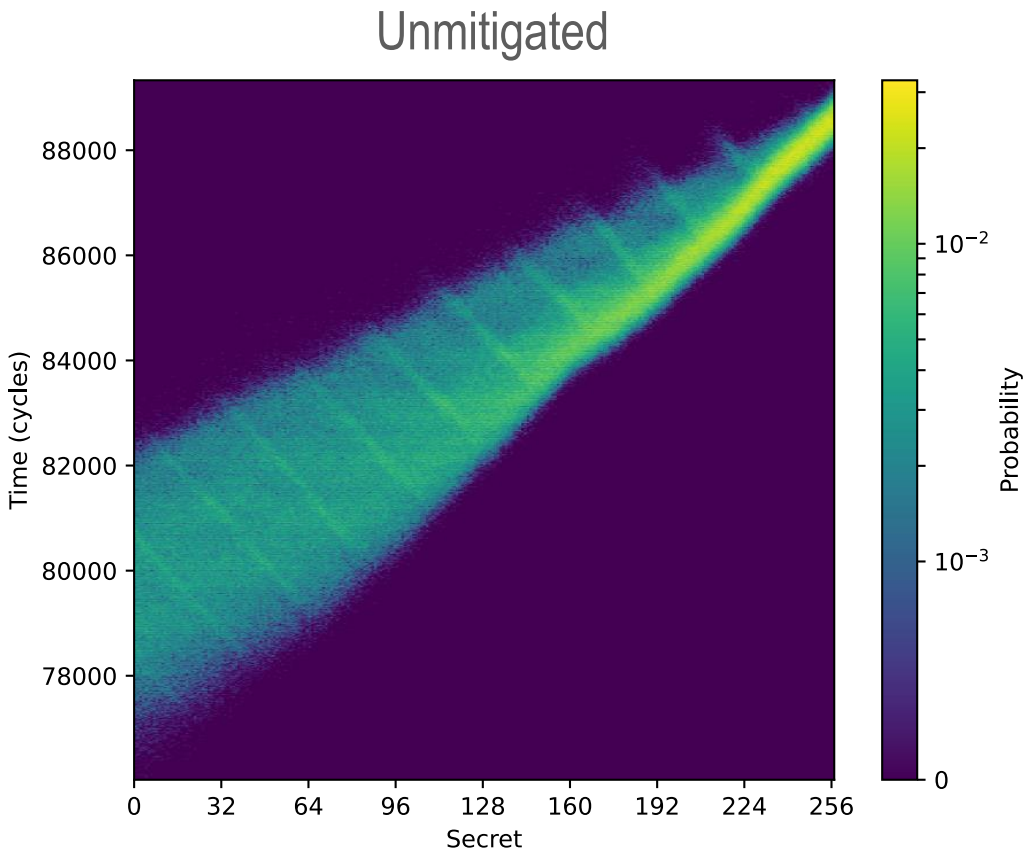
Step 3: Drain pending transactions.
e.g. memory transactions from previous write-back

Step 4: Clear components that cannot be cleared in a single cycle.
e.g. cache SRAMs

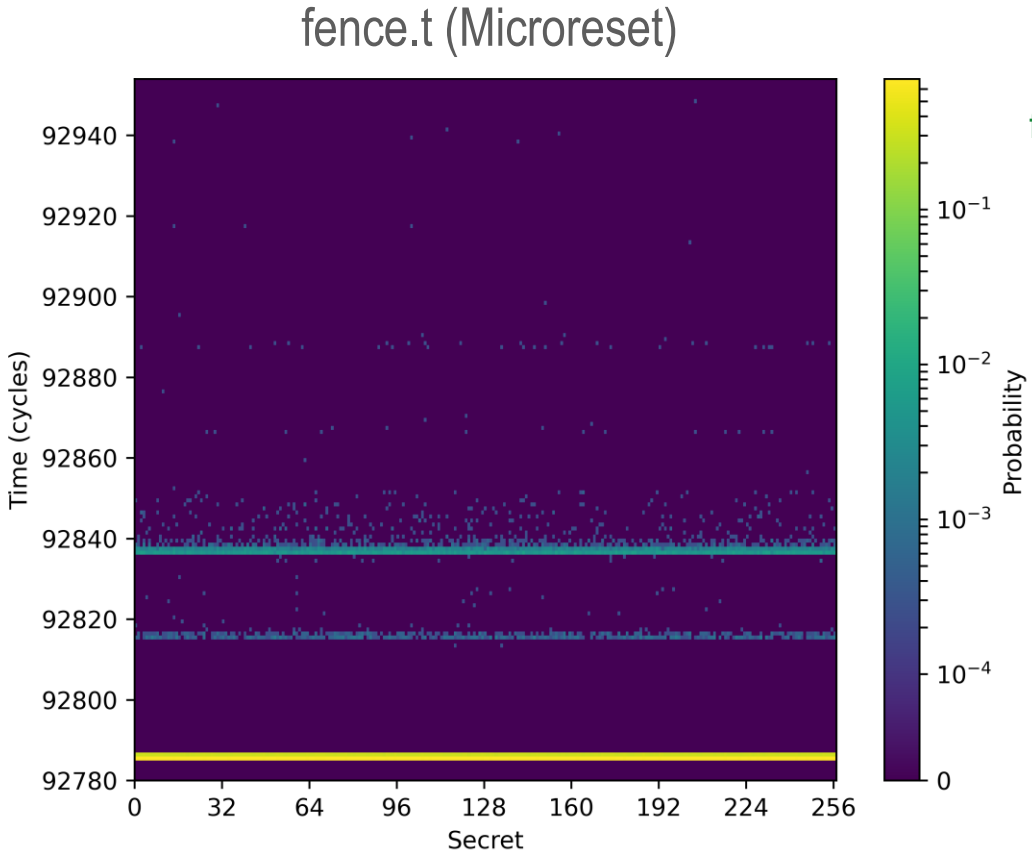
Step 5: Assert Microreset.

Step 6: Continue execution from saved program counter.

L1 Data Cache Channel: Microreset



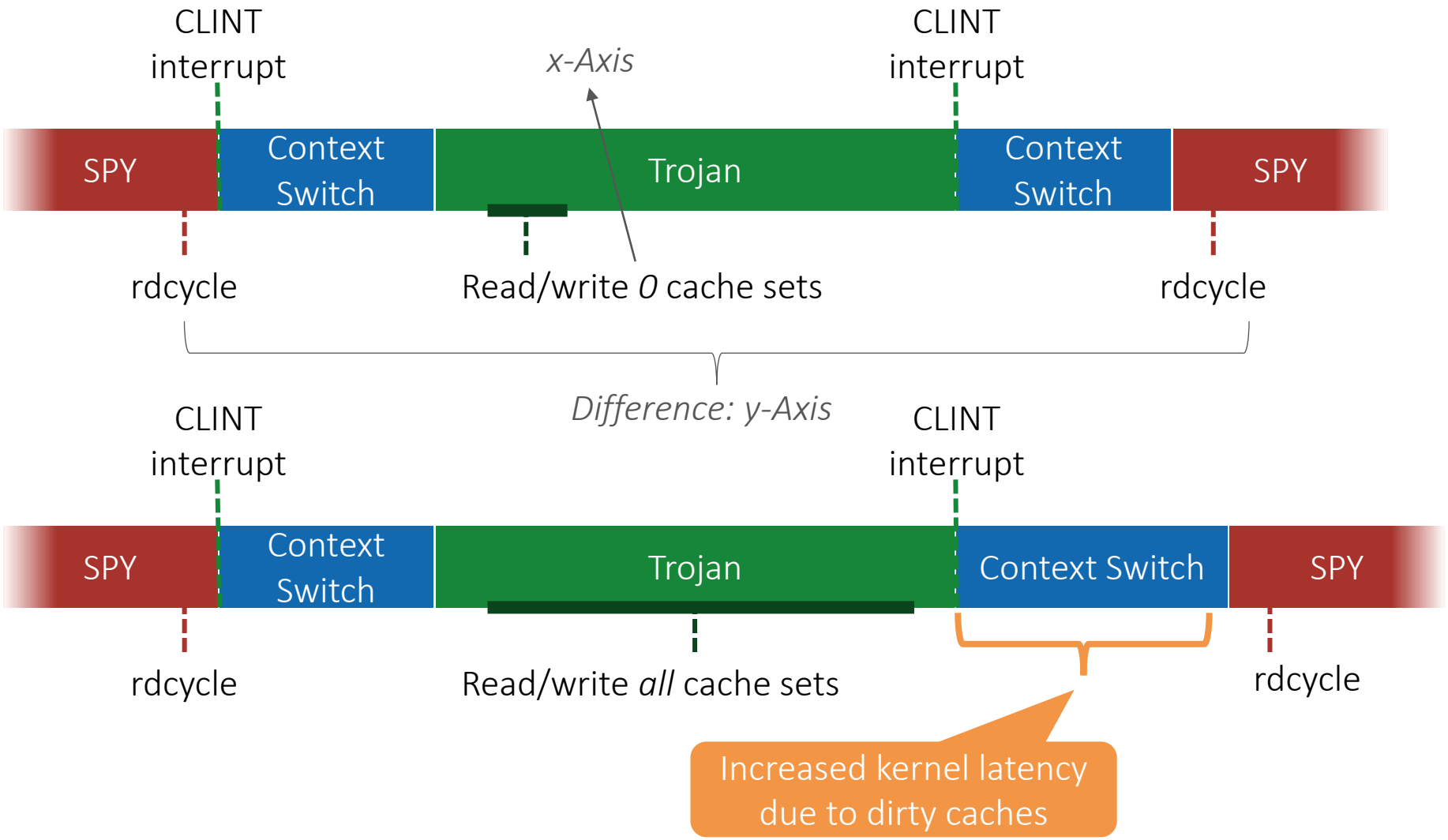
$N = 10^6$, $M = 1667.3$ mb, $M_0 = 0.5$ mb



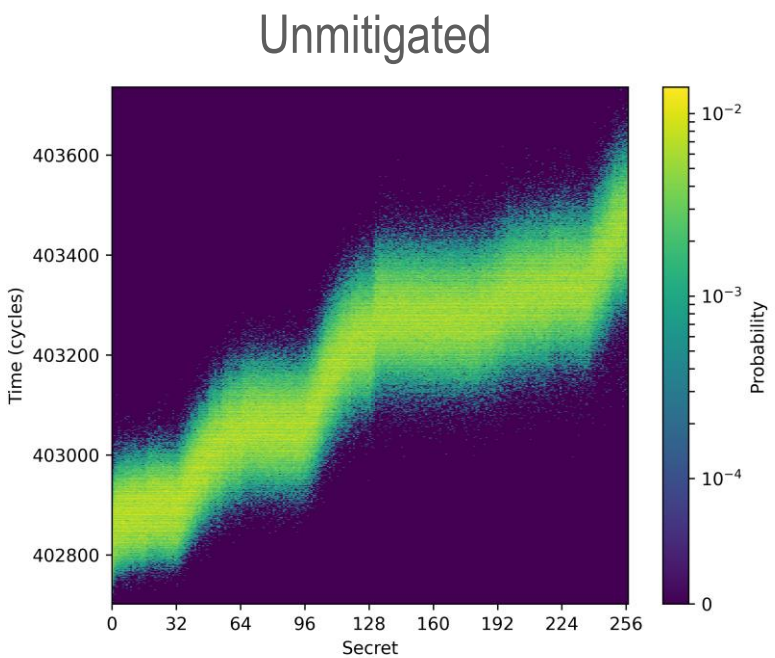
$N = 10^6$, $M = 21.7$ mb, $M_0 = 27.8$ mb

TCs
CVA6
fence.t
CS
Costs
End

Context-Switch Latency Channel

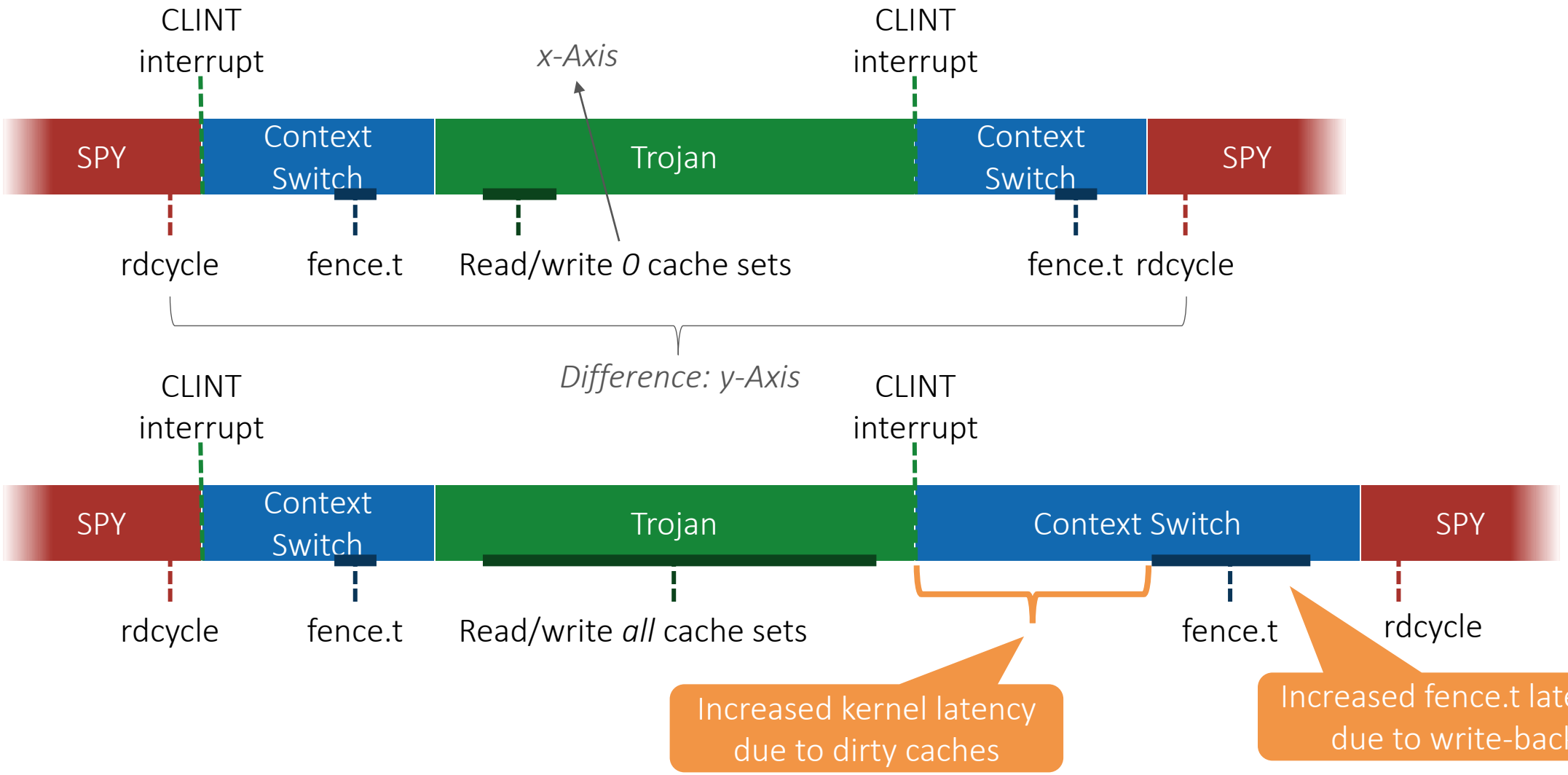


Context-Switch Latency Channel



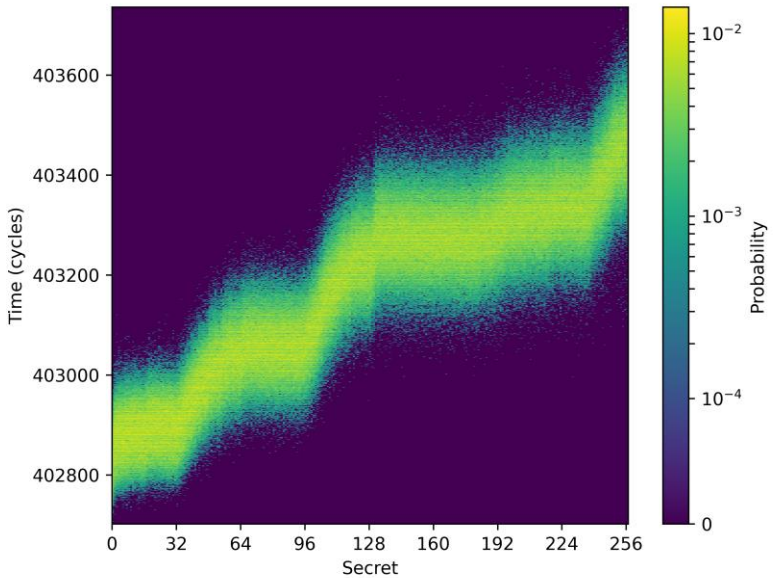
M = 1297.9 mb, M_0 = 0.5 mb

Context-Switch Latency Channel



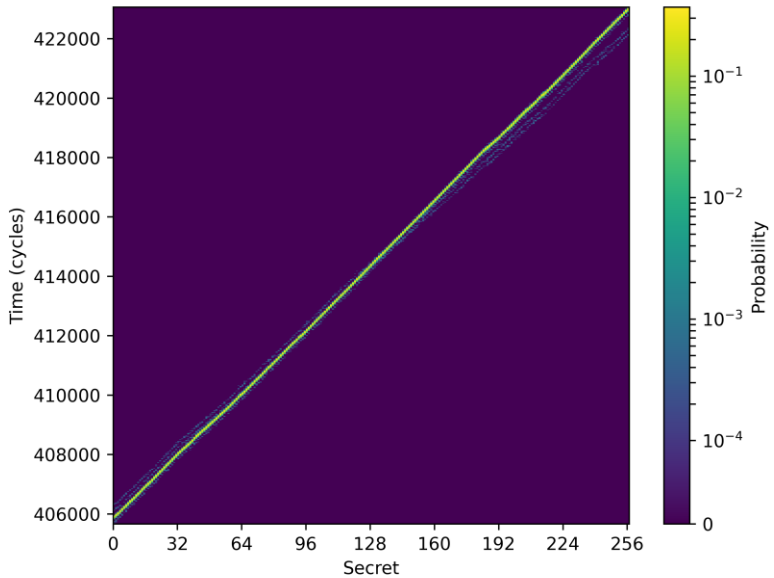
Context-Switch Latency Channel

Unmitigated



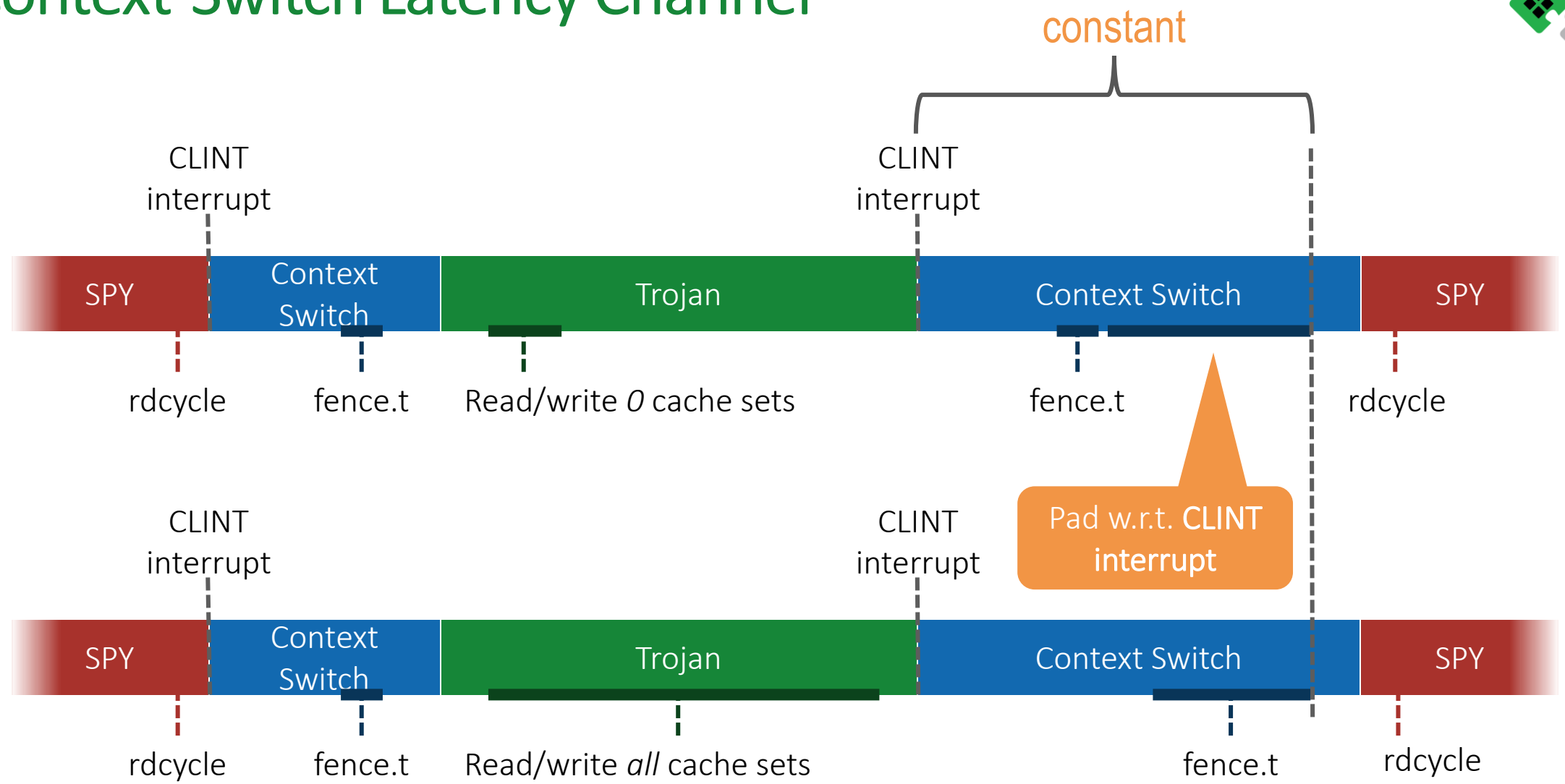
M = 1297.9 mb, M_0 = 0.5 mb

fence.t



M = 7257.2 mb, M_0 = 0.4 mb

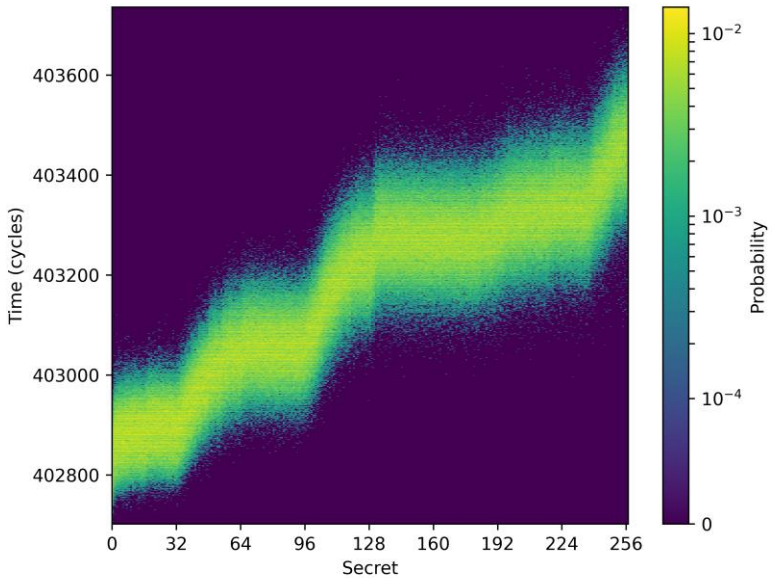
Context-Switch Latency Channel



TCs
CVA6
fence.t
CS
Costs
End

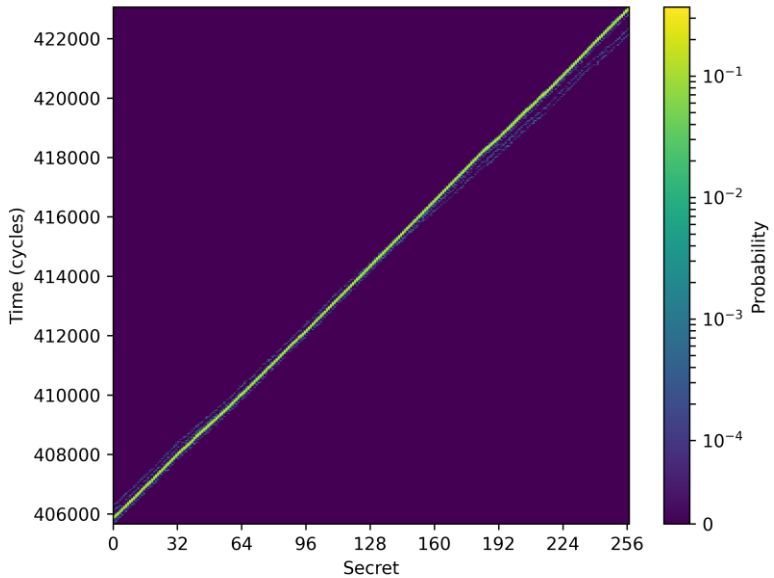
Context-Switch Latency Channel

Unmitigated



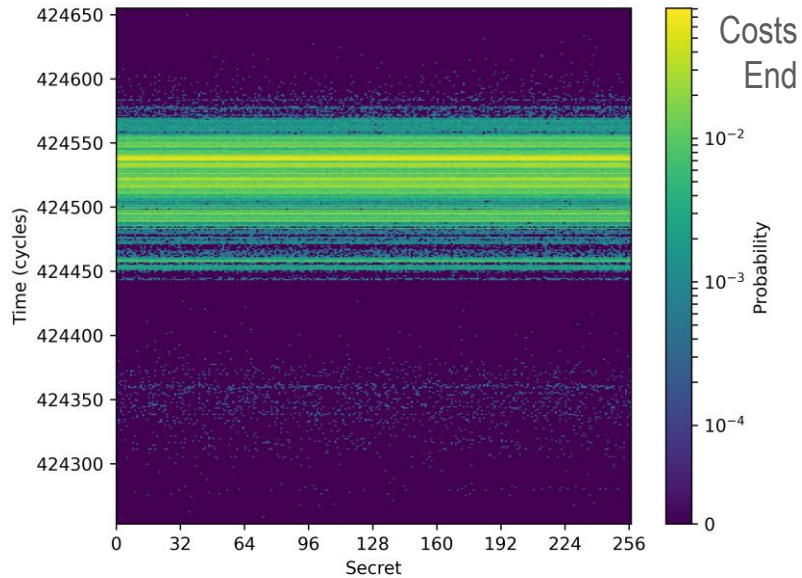
M = 1297.9 mb, M_0 = 0.5 mb

fence.t



M = 7257.2 mb, M_0 = 0.4 mb

fence.t + padding



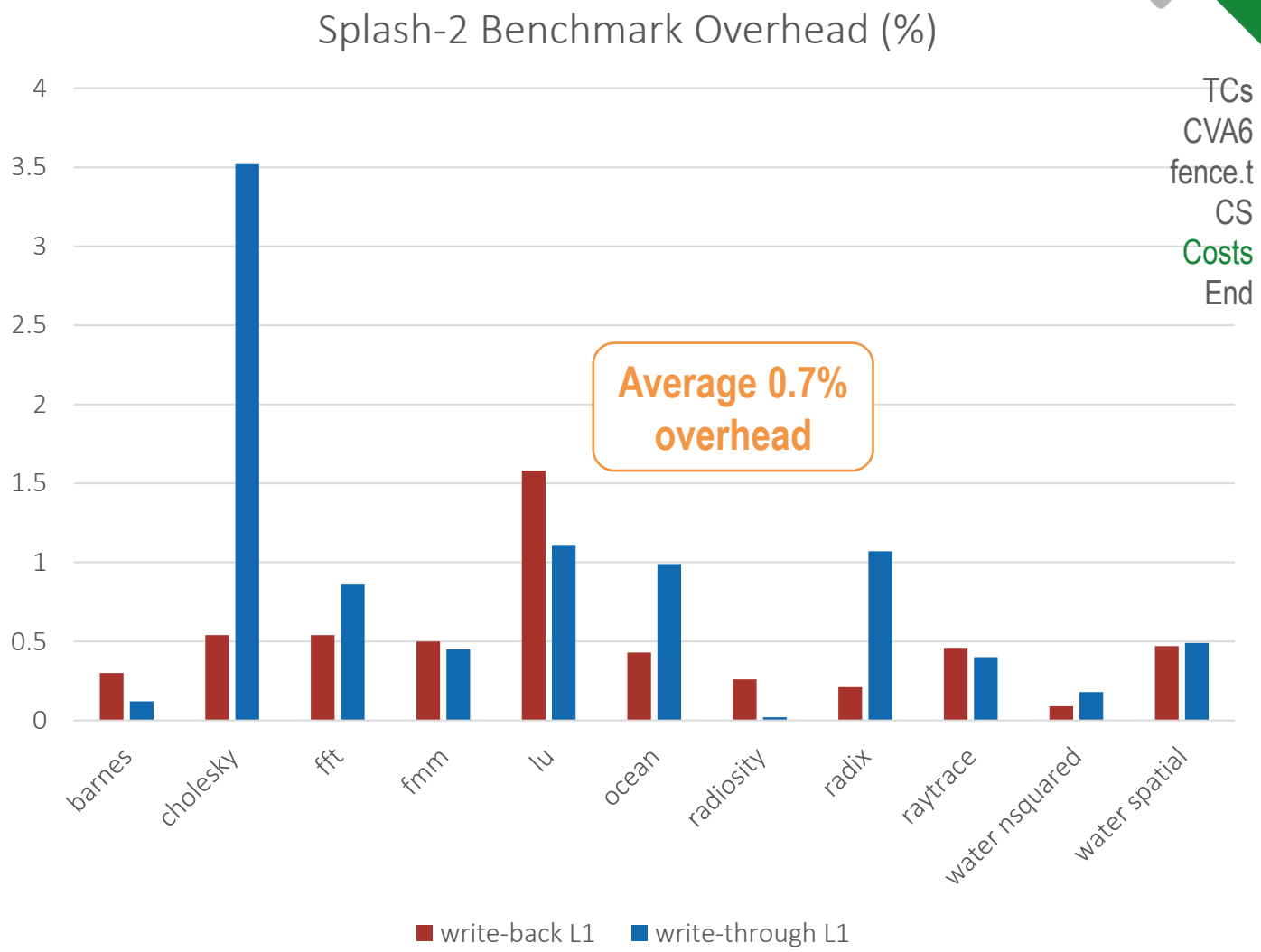
M = 1.4 mb, M_0 = 1.6 mb

Costs

- 2 threads: 1 benchmark + 1 idle
- 1GHz system clock
- 10ms timeslice
- Context switch every 10M cycles

- Synthesis in GF22FDX @1GHz

Negligible hardware costs



Conclusion

- Existing ISA **cannot prevent timing channels**.
- **fence.t** instruction enables temporal partitioning of hardware.
- **Microreset** is a **systematic and straight-forward** implementation of fence.t.
- Negligible hardware costs, 0.7% average performance impact.
- Time protection is a **system-wide challenge**.
- Contributing to the RISC-V spec!

uSC SIG / uSCR-IS TG