

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

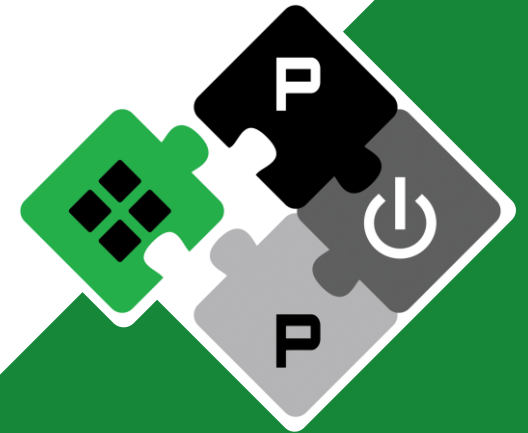
gernot@unsw.edu.au

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





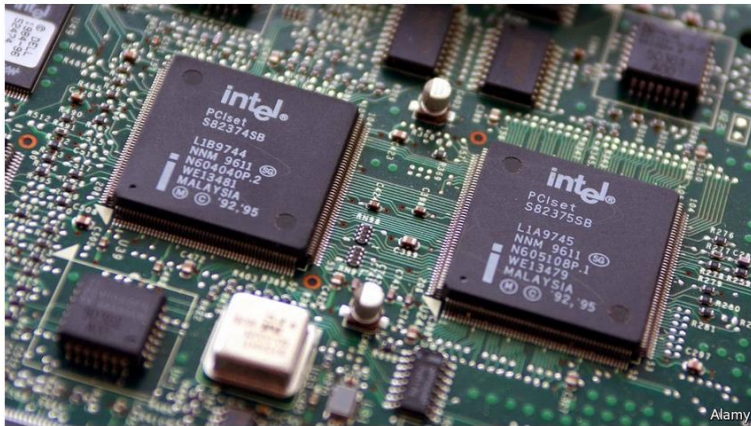
TCs
CVA6
fence.t
CS
Costs
End

Science & technology

The chips are down

Two security flaws in modern chips cause big headaches for the tech business

Fixing the underlying problems will take a long time



Alamy

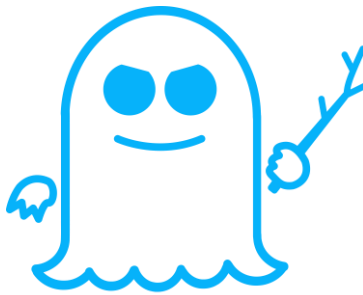
Jan 4th 2018

IT WAS a one-two punch for the computer industry. January 3rd saw the disclosure of two serious flaws in the design of the processors that power most of the world's computers. The first, appropriately called Meltdown, affects only chips made by Intel, and makes it possible to dissolve the virtual walls between the digital memory used by different programs, allowing hackers to steal sensitive data, such as passwords or a computer's encryption keys. The second,

ANDY GREENBERG SECURITY 01.03.2018 03:00 PM

A Critical Intel Flaw Breaks Basic Security for Most Computers

A Google-led team of researchers has found a critical chip flaw that developers are scrambling to patch in millions of computers.



SPECTRE

Speculative Execution + Covert Channel

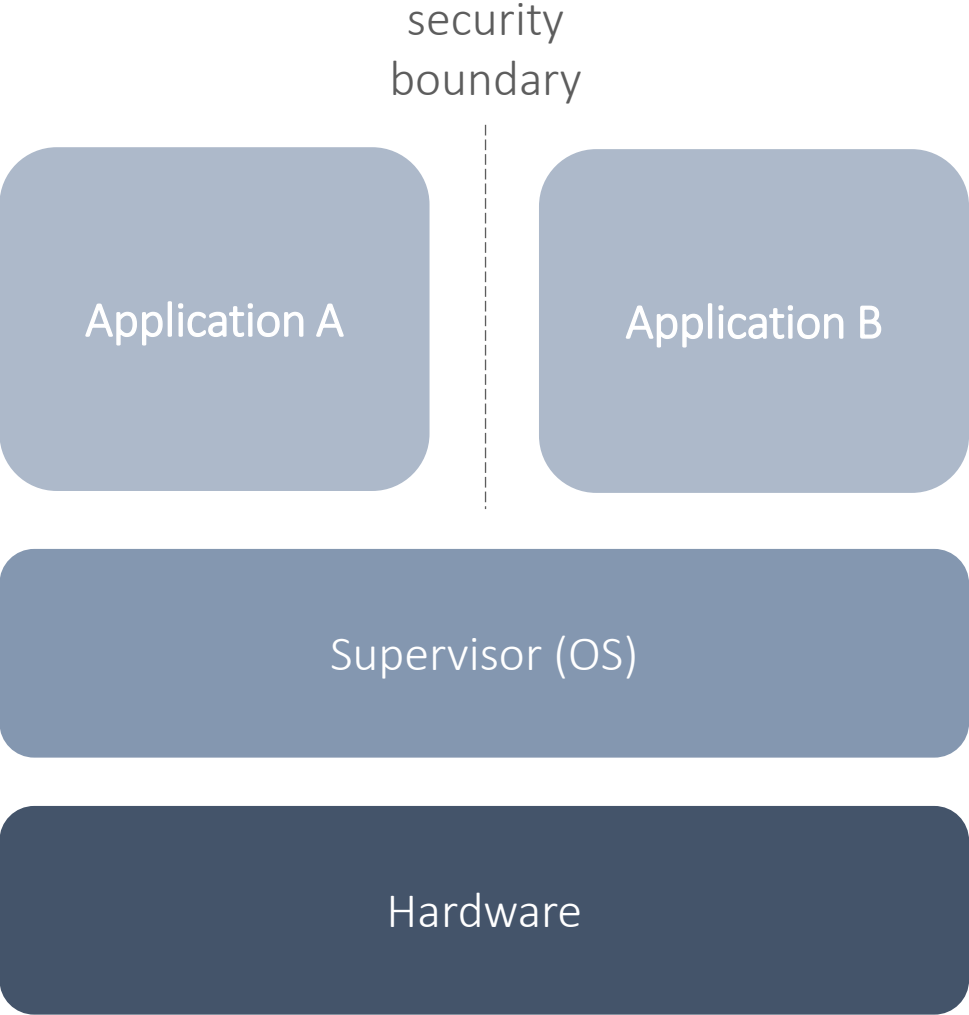


Intel's processors have a security bug and the fix could slow down PCs

By Tom Warren | @tomwarren | Jan 3, 2018, 8:45am EST

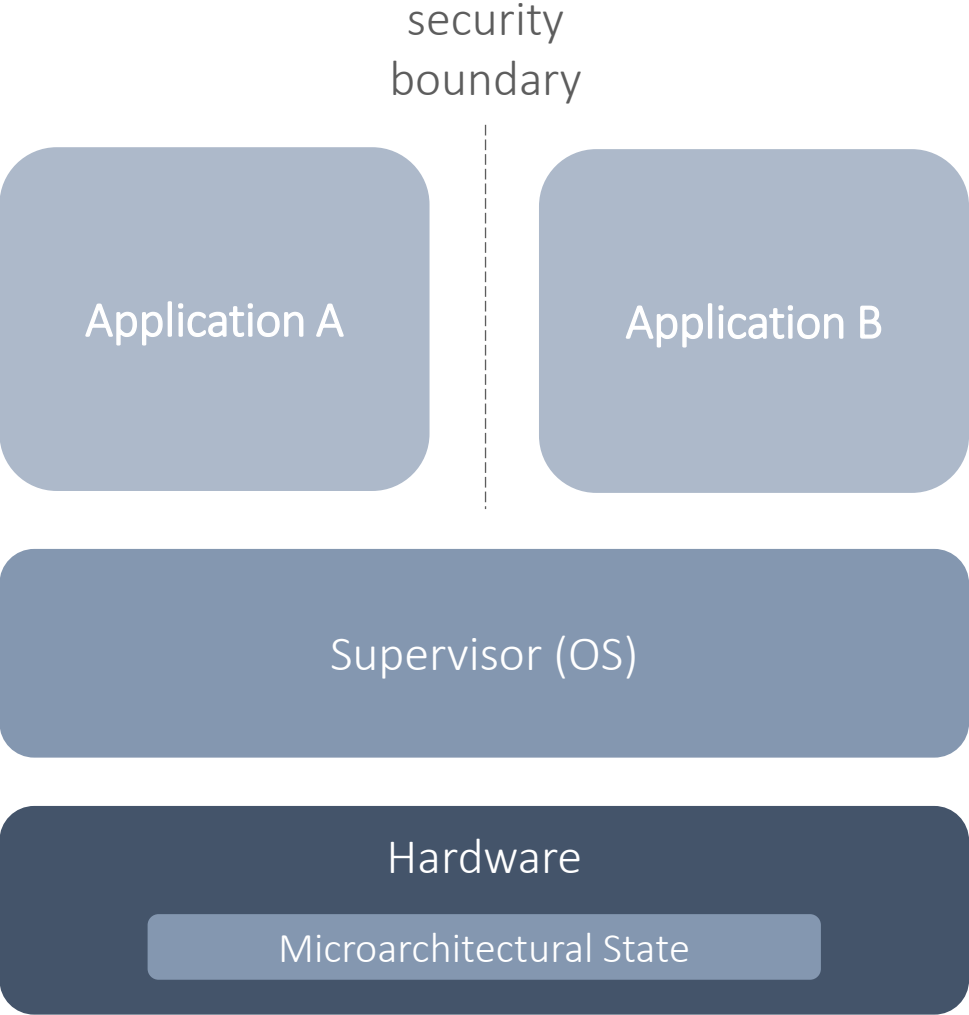
116

Timing Channel



TCs
CVA6
fence.t
CS
Costs
End

Timing Channel

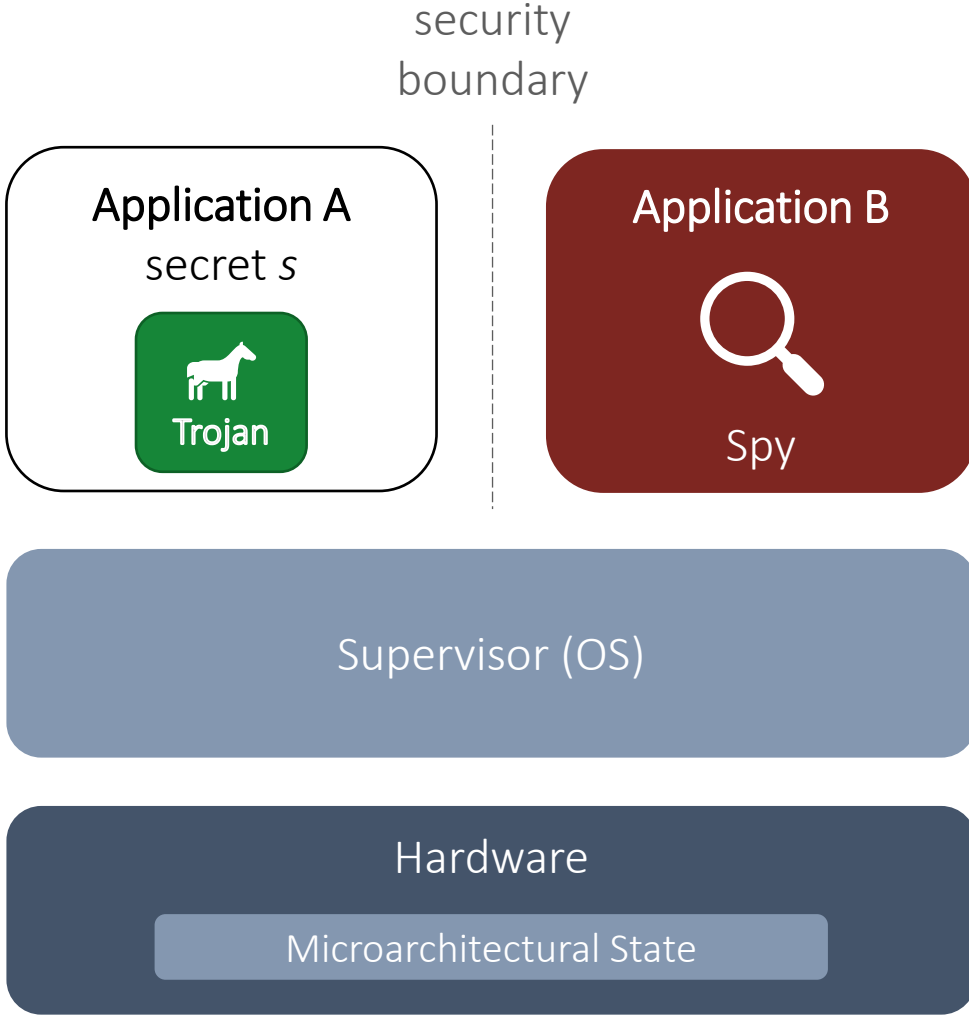


TCs
CVA6
fence.t
CS
Costs
End

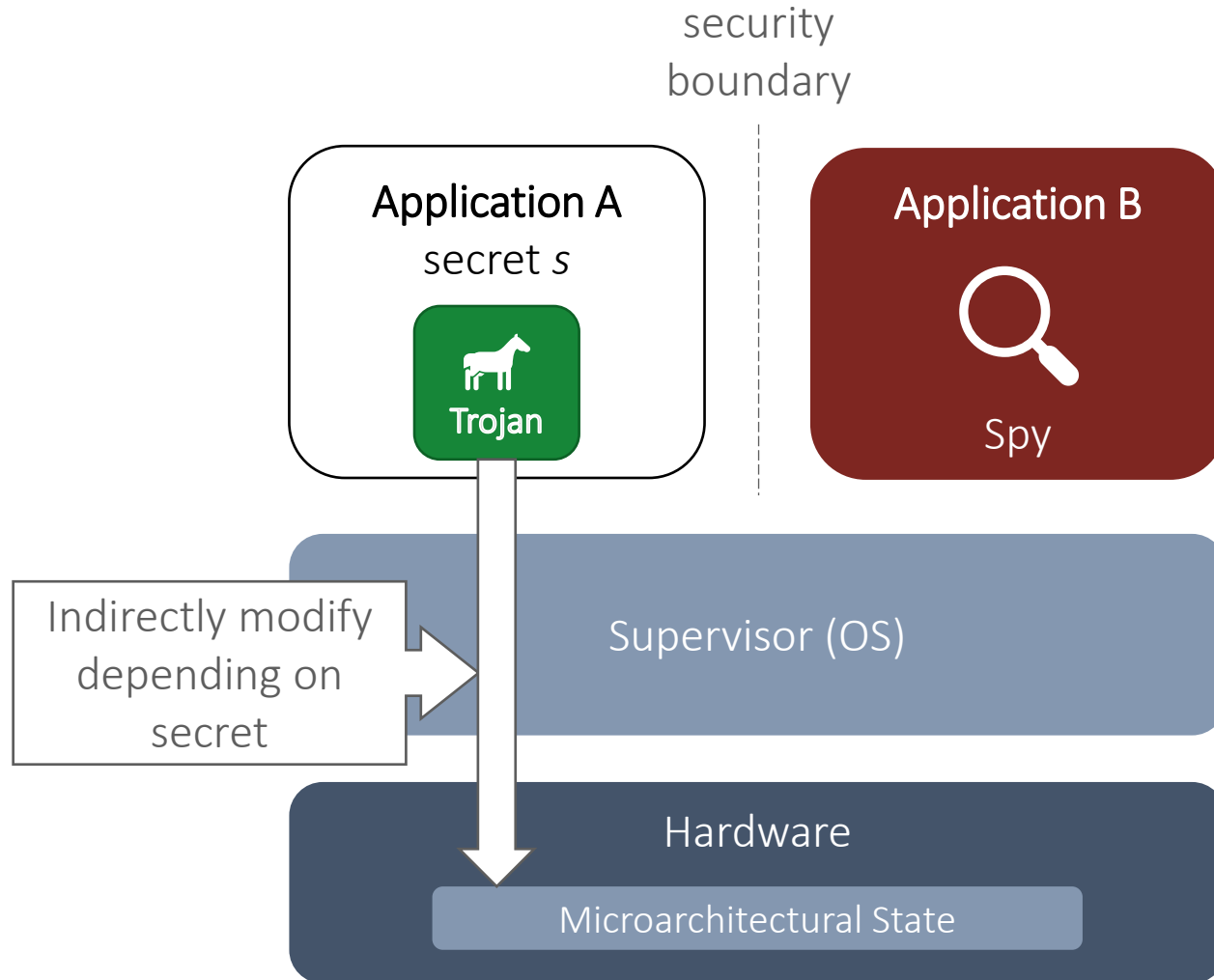
Timing Channel



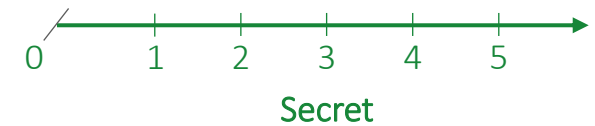
TCs
CVA6
fence.t
CS
Costs
End



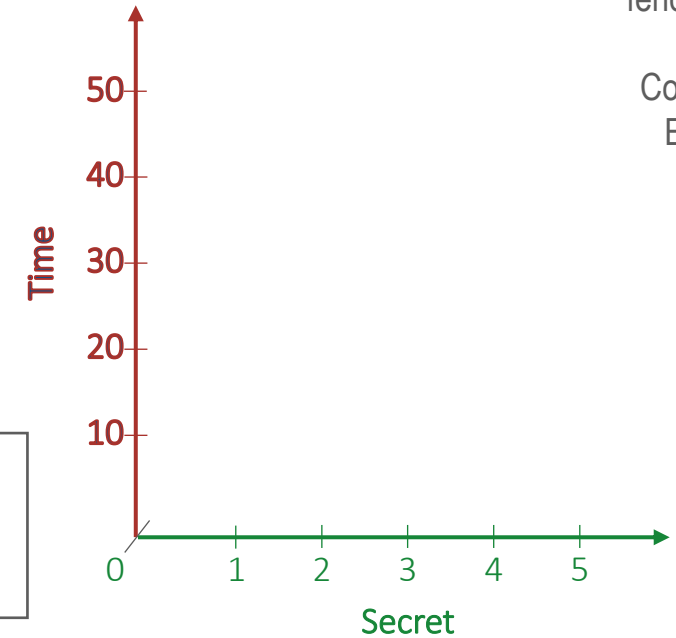
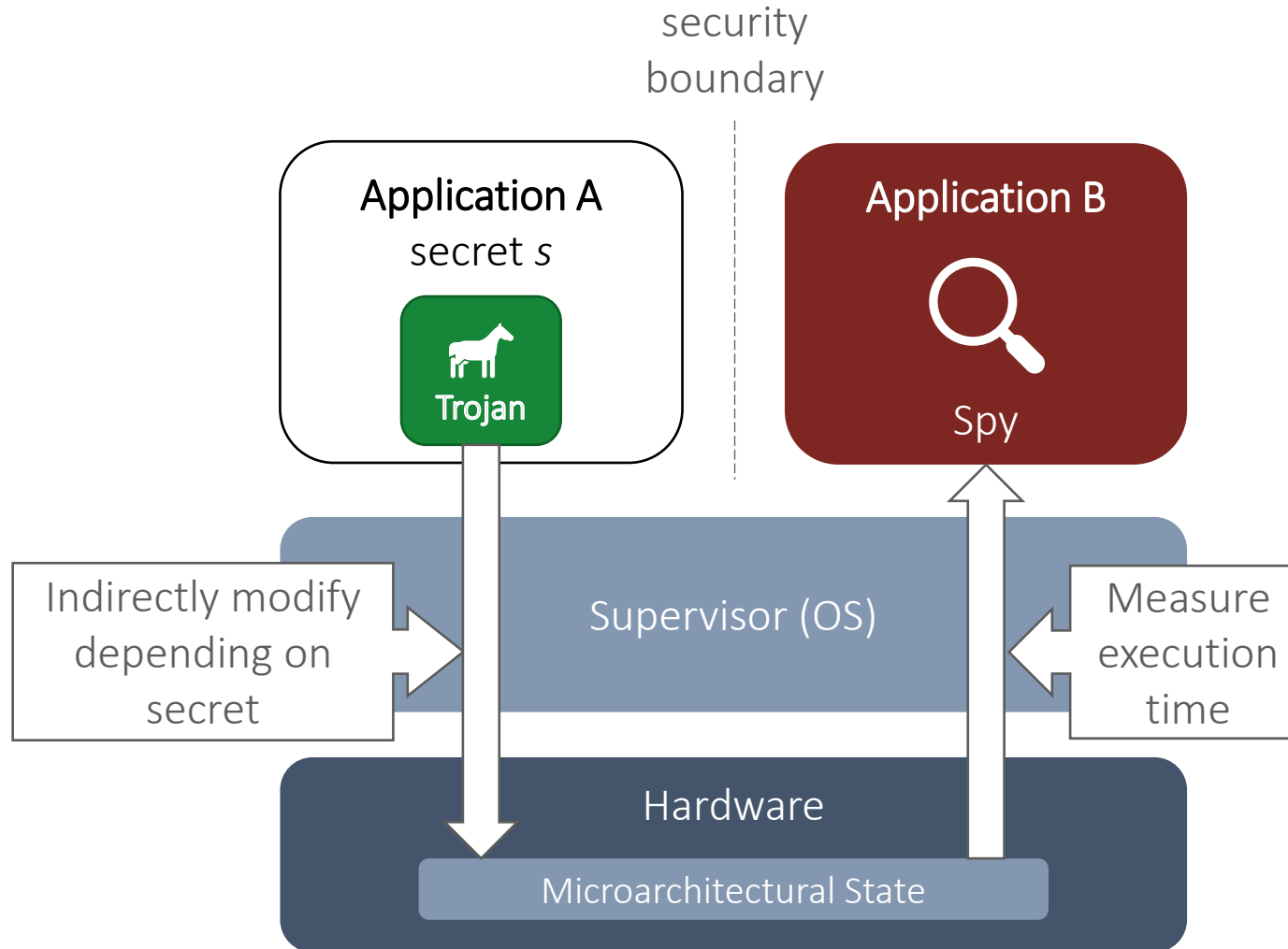
Timing Channel



TCs
CVA6
fence.t
CS
Costs
End

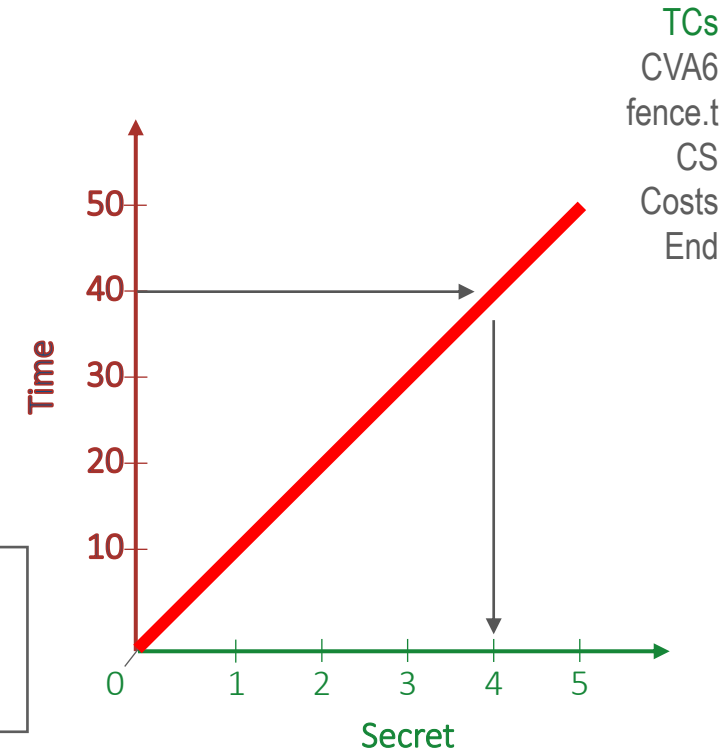
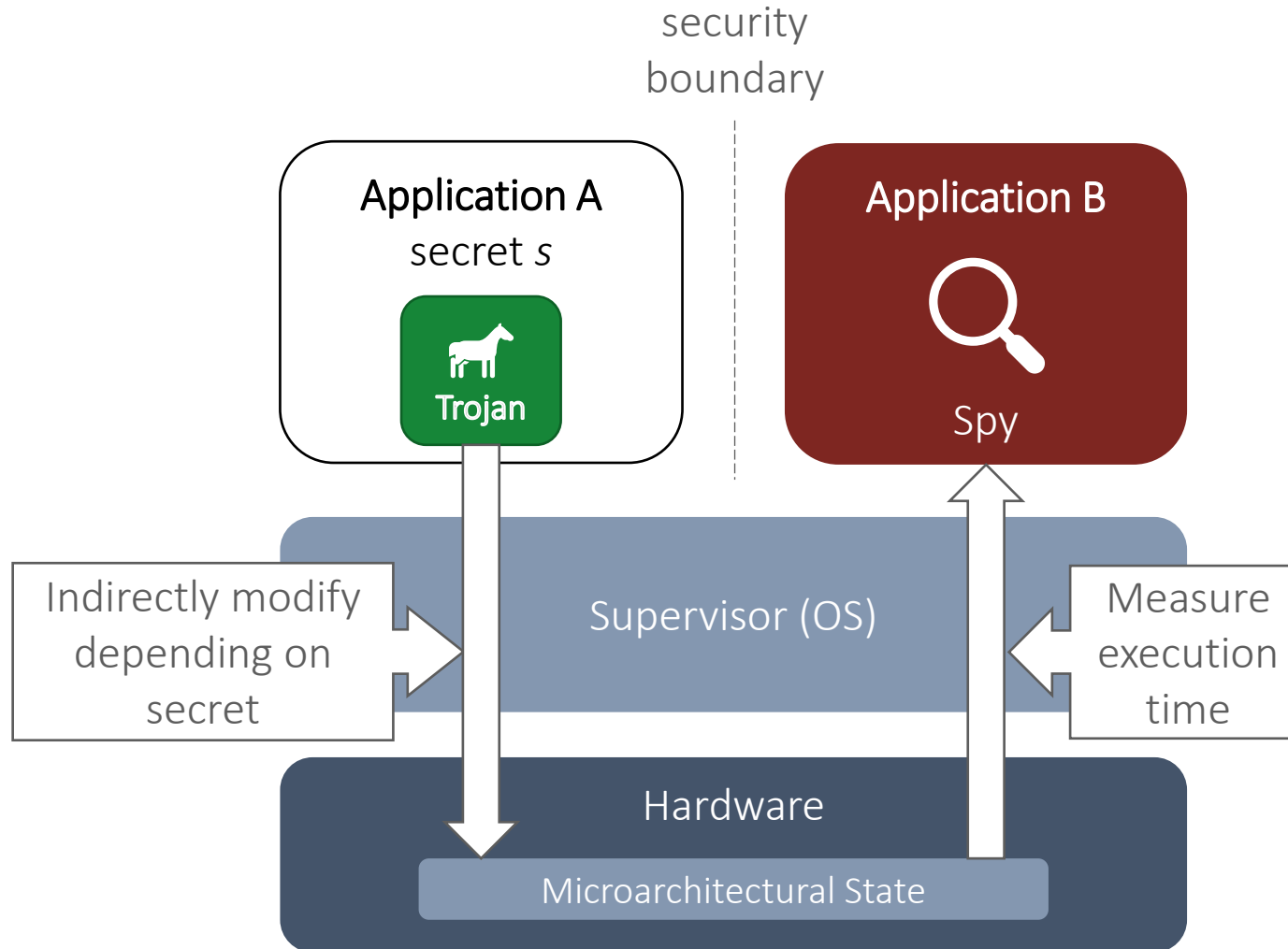


Timing Channel

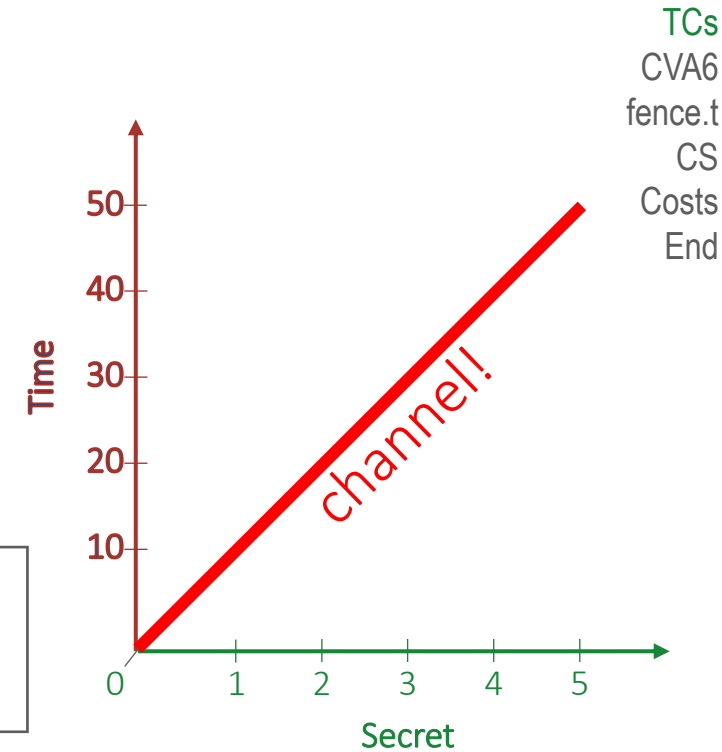
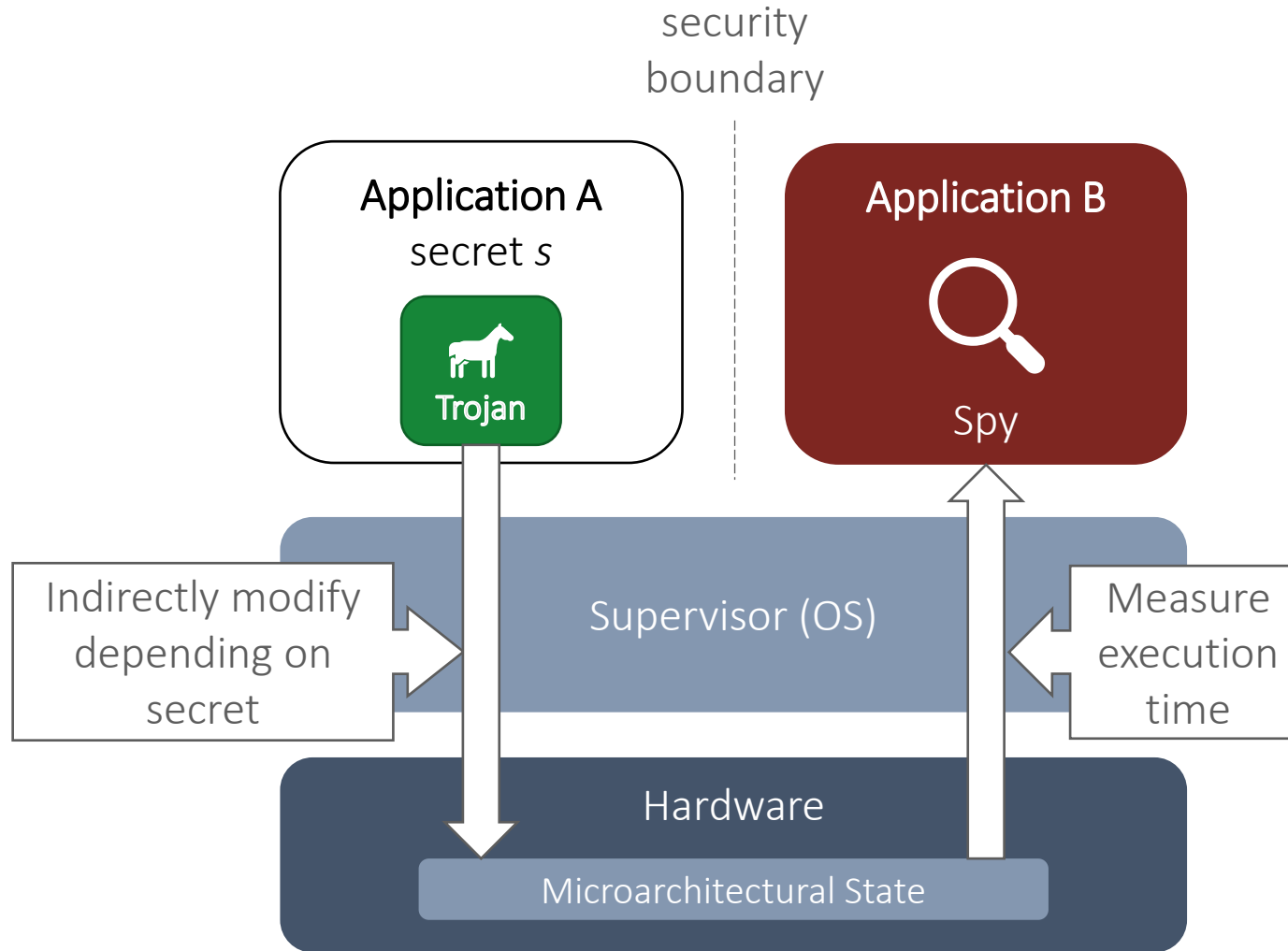


TCs
CVA6
fence.t
CS
Costs
End

Timing Channel

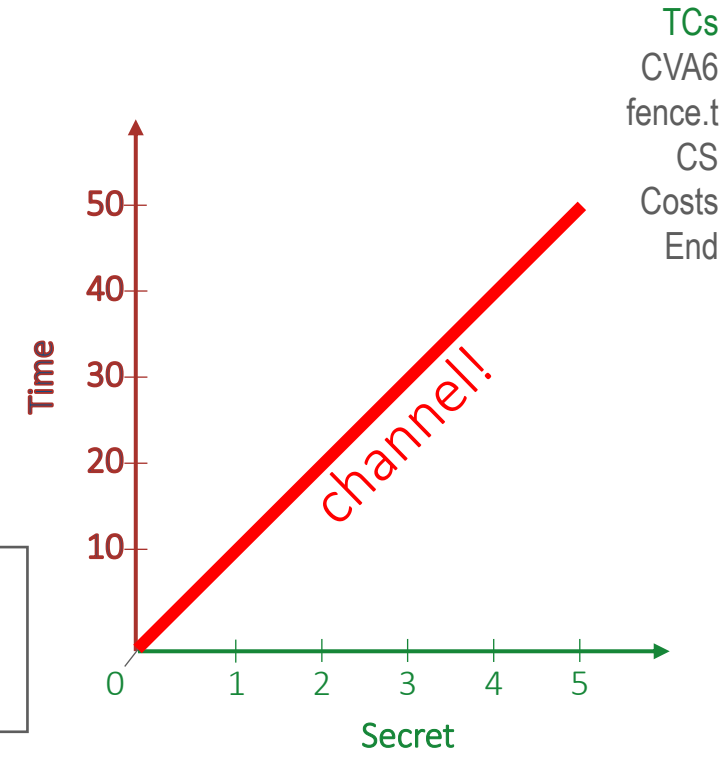
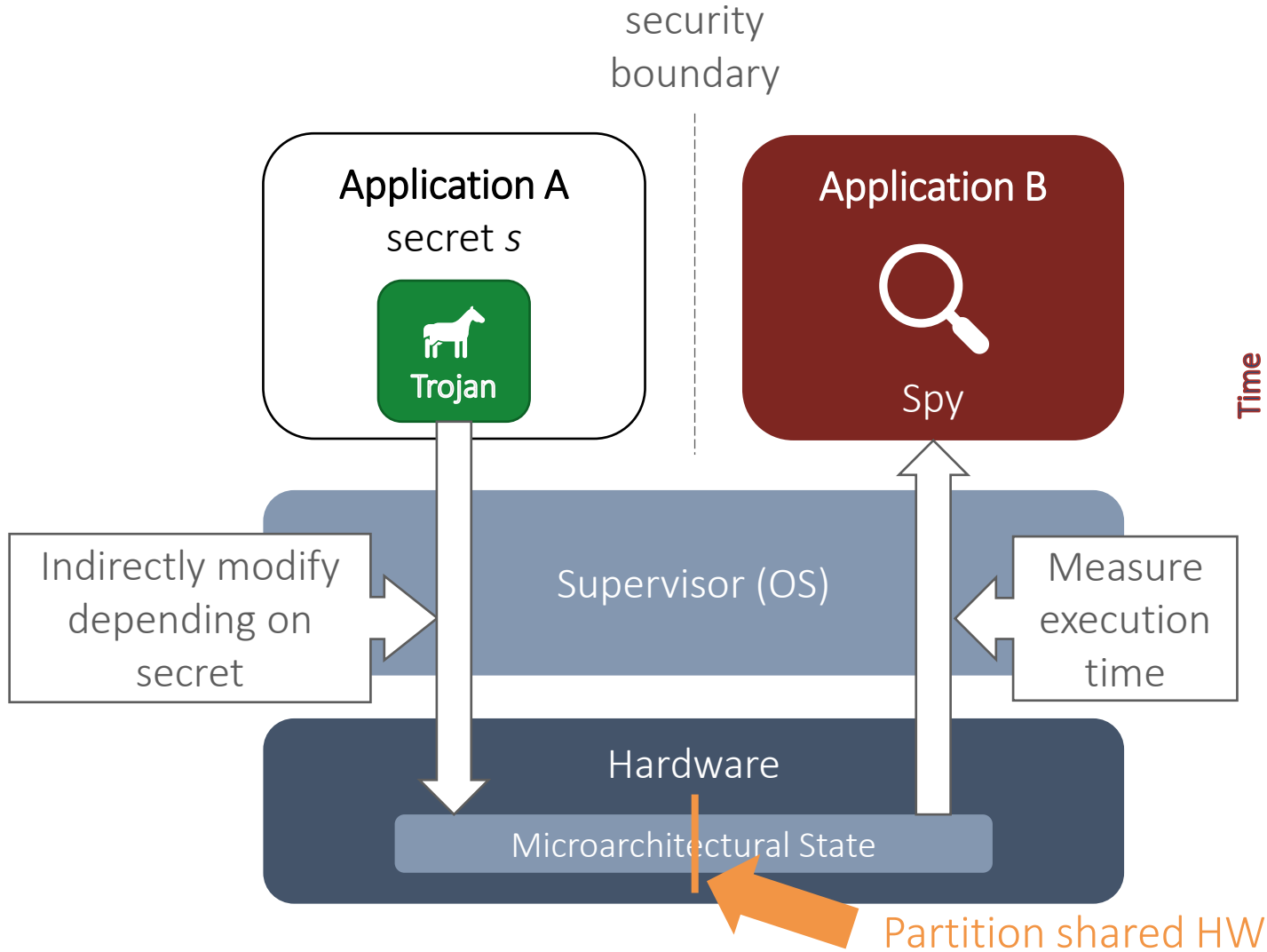


Timing Channel

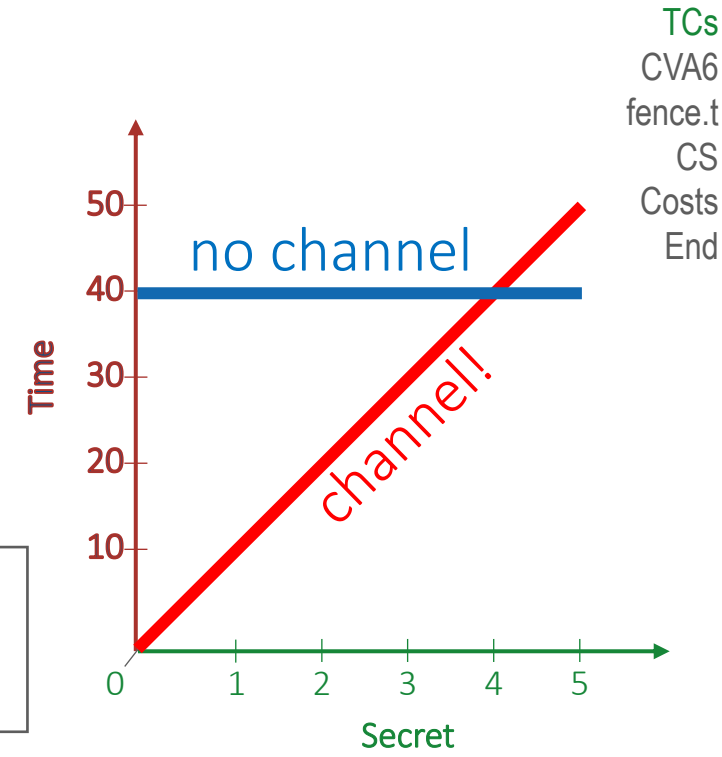
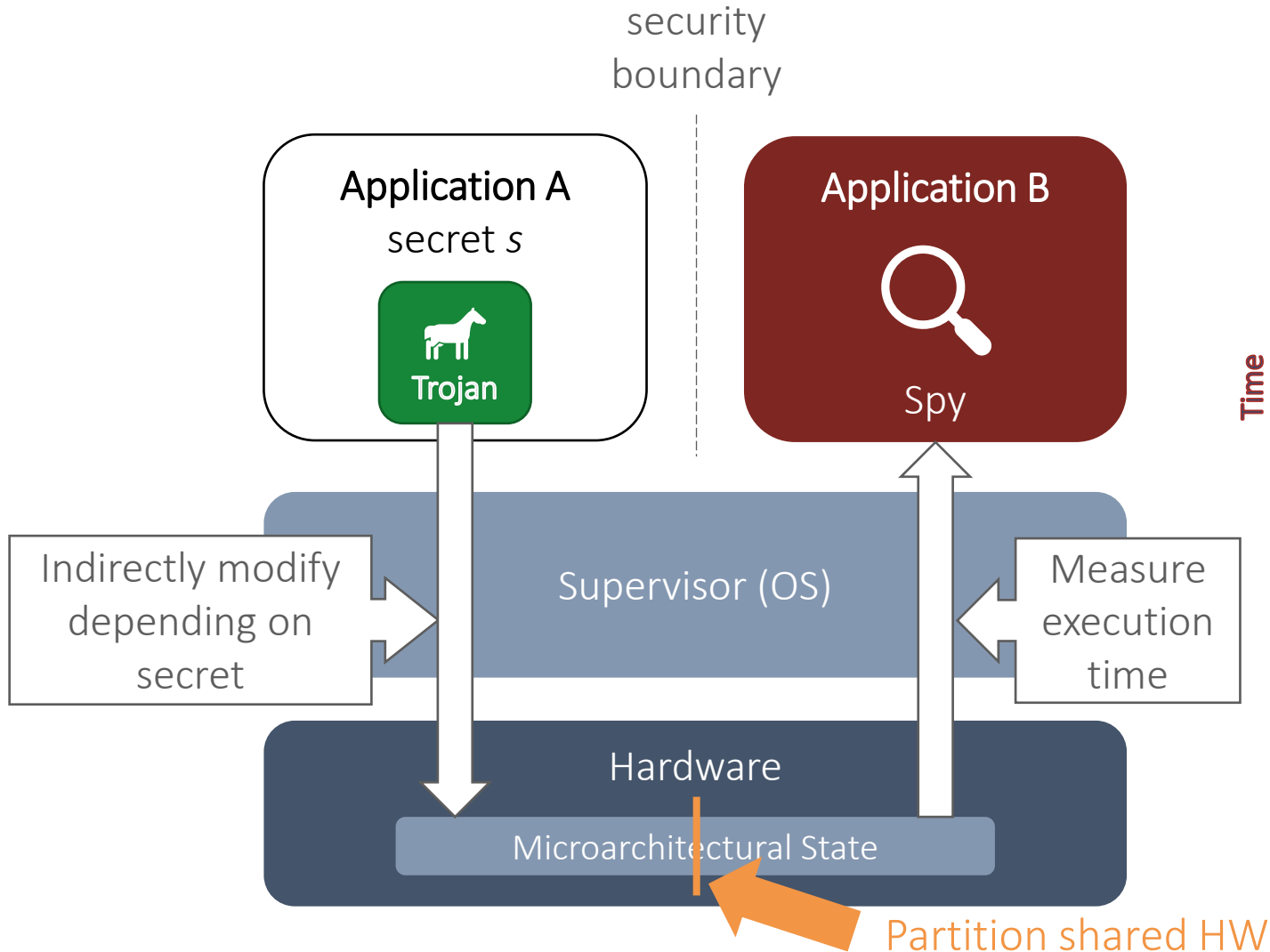


TCs
CVA6
fence.t
CS
Costs
End

Timing Channel



Timing Channel



CVA6 (Ariane)

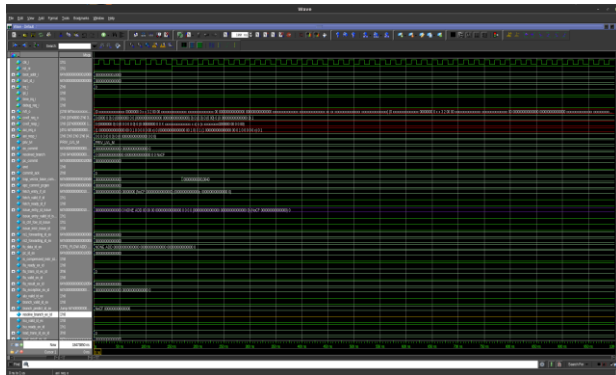


- Open-source 64-bit application-class RISC-V processor
- Boots Linux (or seL4)
- Developed by PULP team at ETH
- Now owned and maintained by OpenHW Group
- Widely used in academia and industry

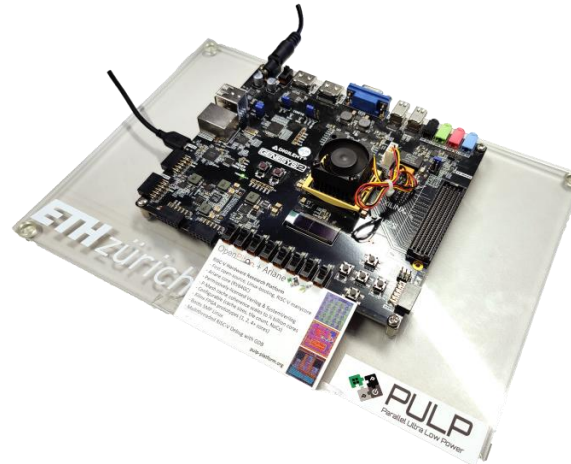


TCs
CVA6
fence.t
CS
Costs
End

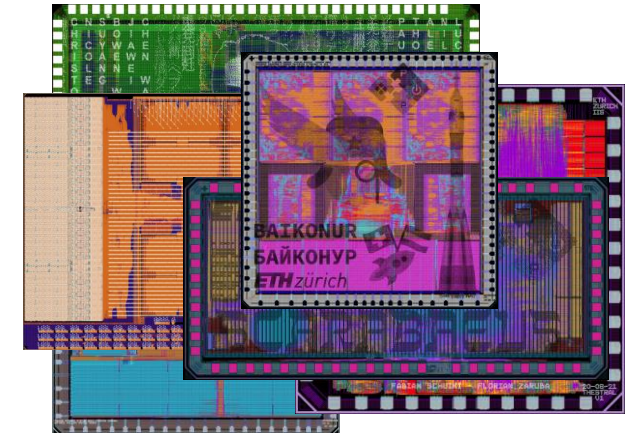
RTL Simulation



FPGA Emulation



ASIC



speed, turn-around time, cost

Timing Channels on CVA6



Application

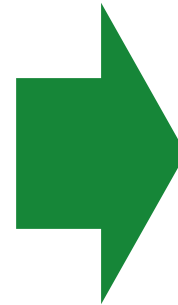
Channel bench



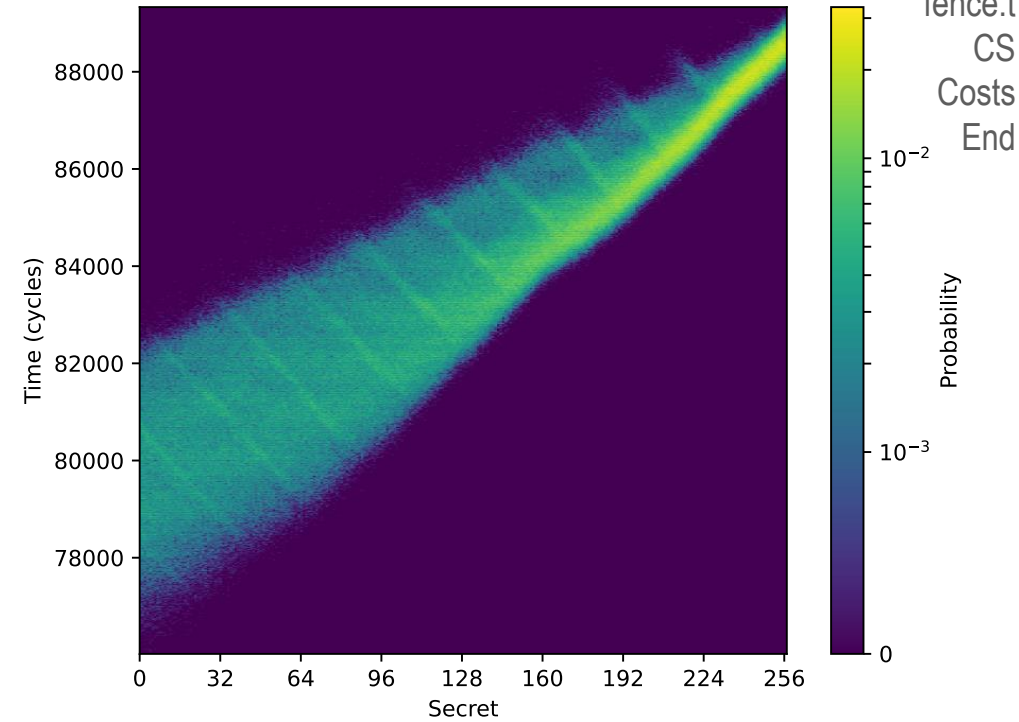
OS



Hardware Platform



$N = 10^6$



Timing Channels on CVA6



Application

Channel bench



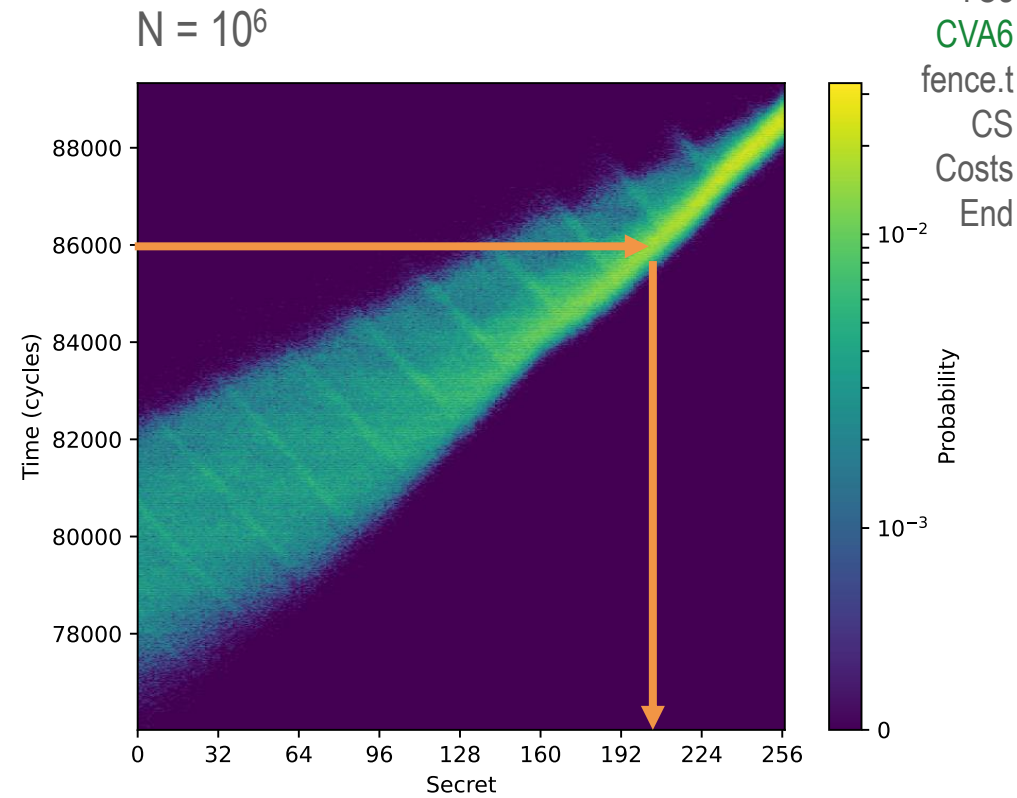
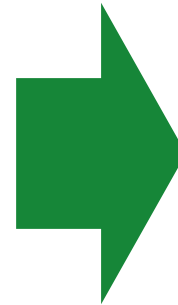
+

OS



+

Hardware Platform



$M = 1667$ mb

Timing Channels on CVA6



Application

Channel bench



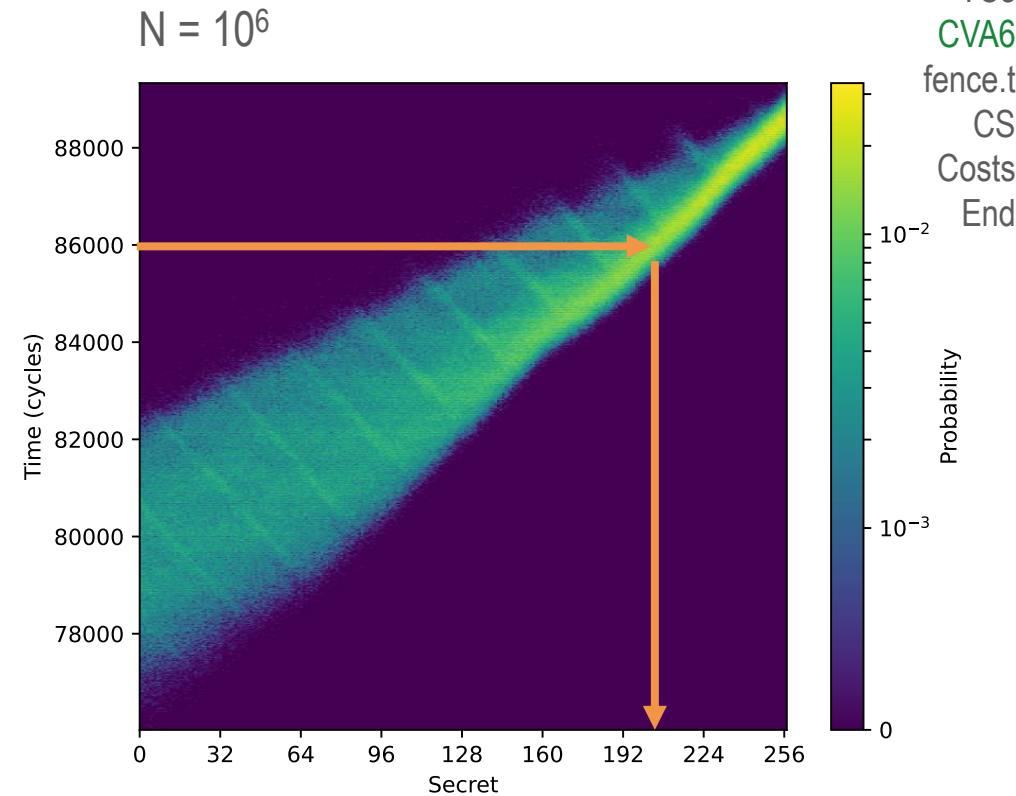
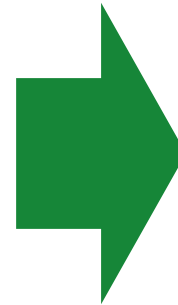
+

OS



+

Hardware Platform



$M = 1667$ mb

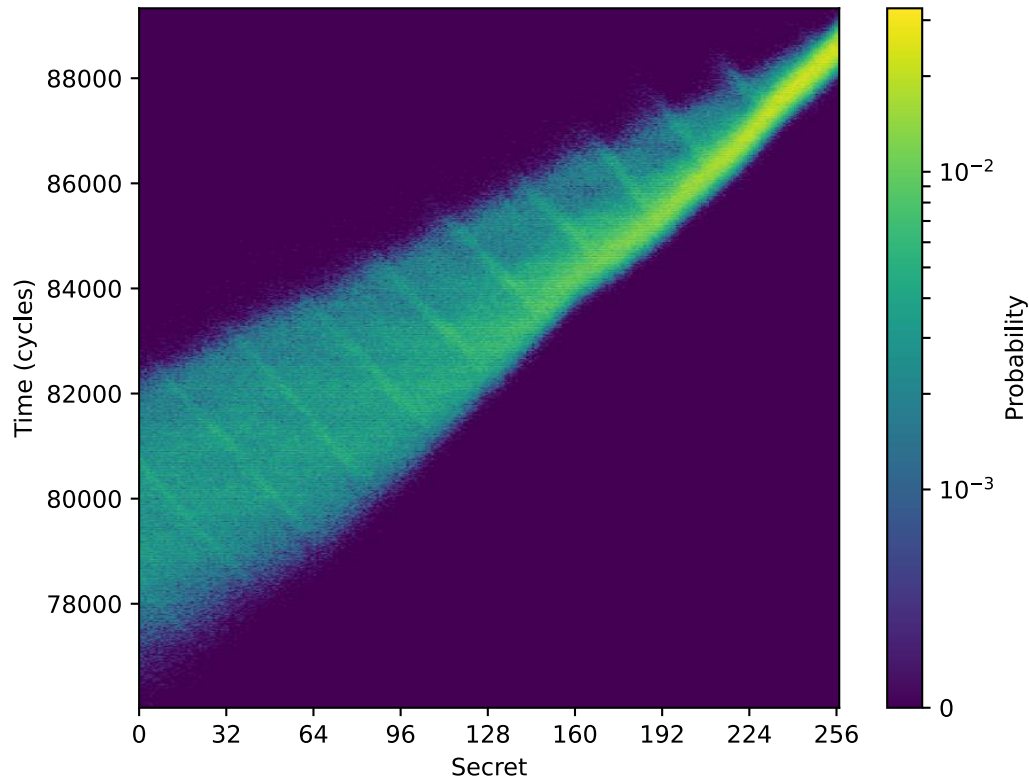
$M_0 = 0.5$ mb

Characterises noise

L1 Data Cache: Software Mitigation



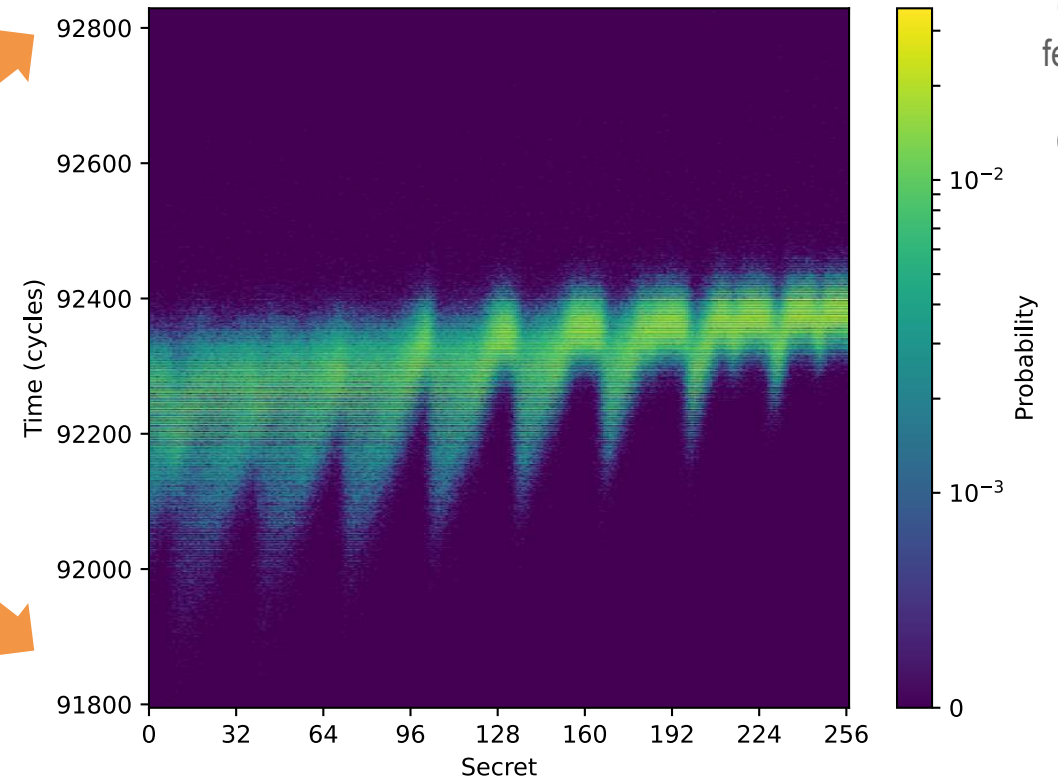
Unmitigated



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

Double L1 data cache prime on context switch

Reduced Range



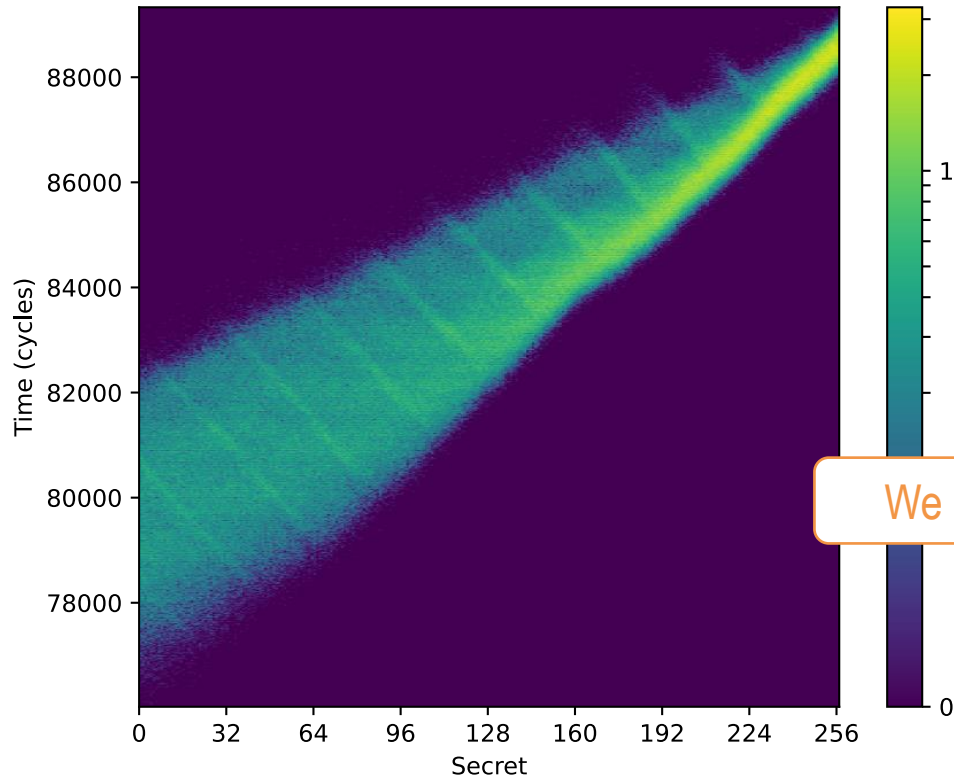
$N = 10^6$, $M = 515.7$ mb, $M_0 = 1.1$ mb

TCs
CVA6
fence.t
CS
Costs
End

L1 Data Cache: Software Mitigation



Unmitigated



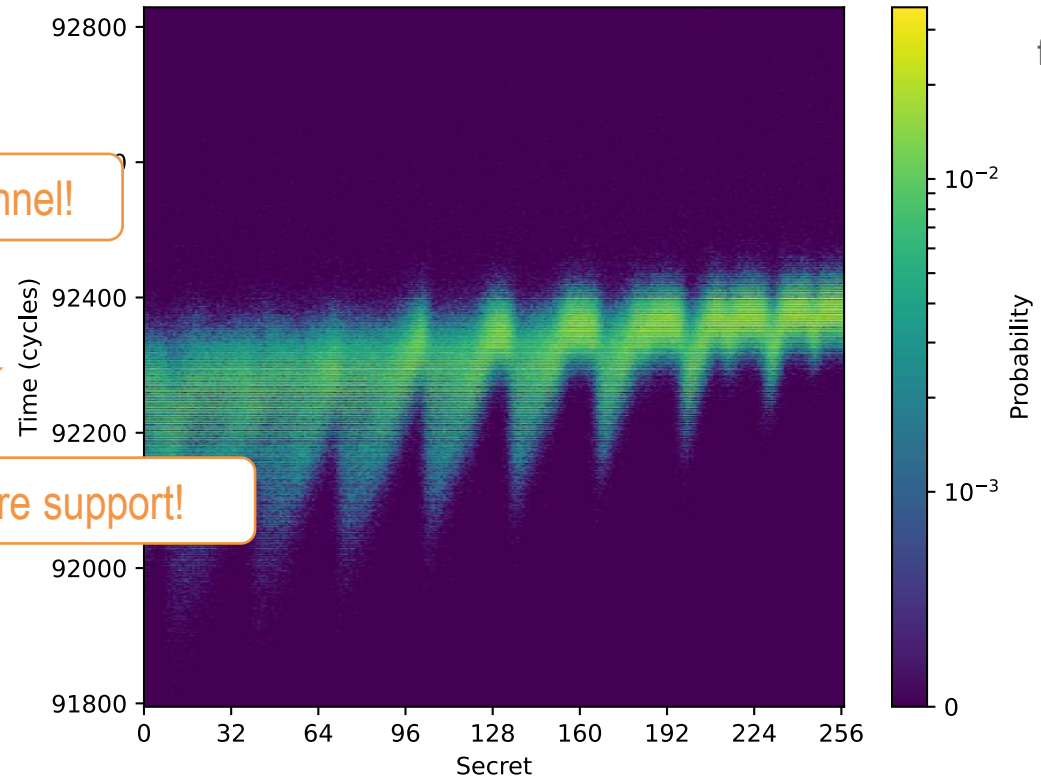
$N = 10^6$, $M = 1667.3$ mb, $M_0 = 0.5$ mb

Double L1 data cache prime on context switch

Still a channel!

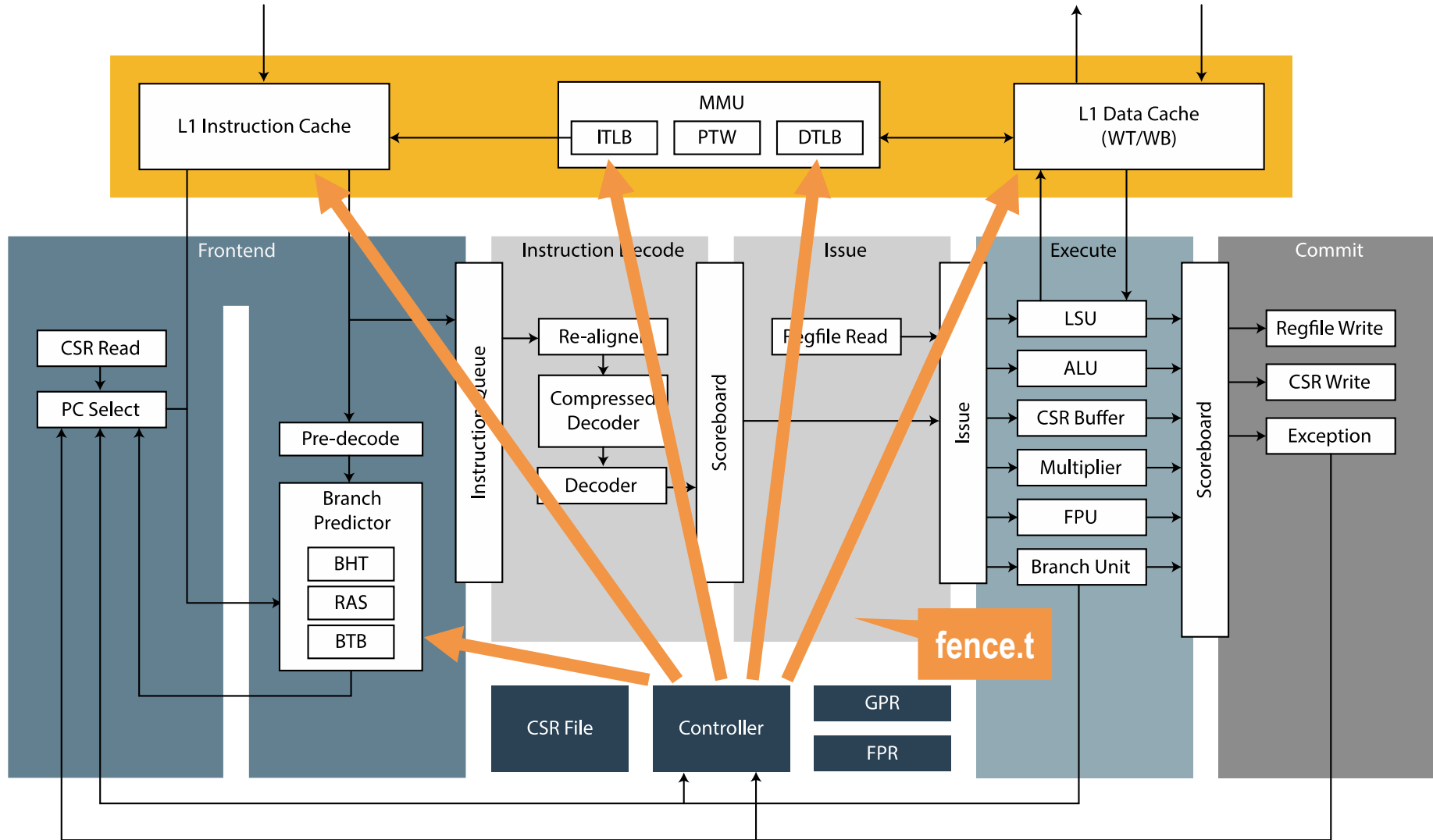


We need hardware support!



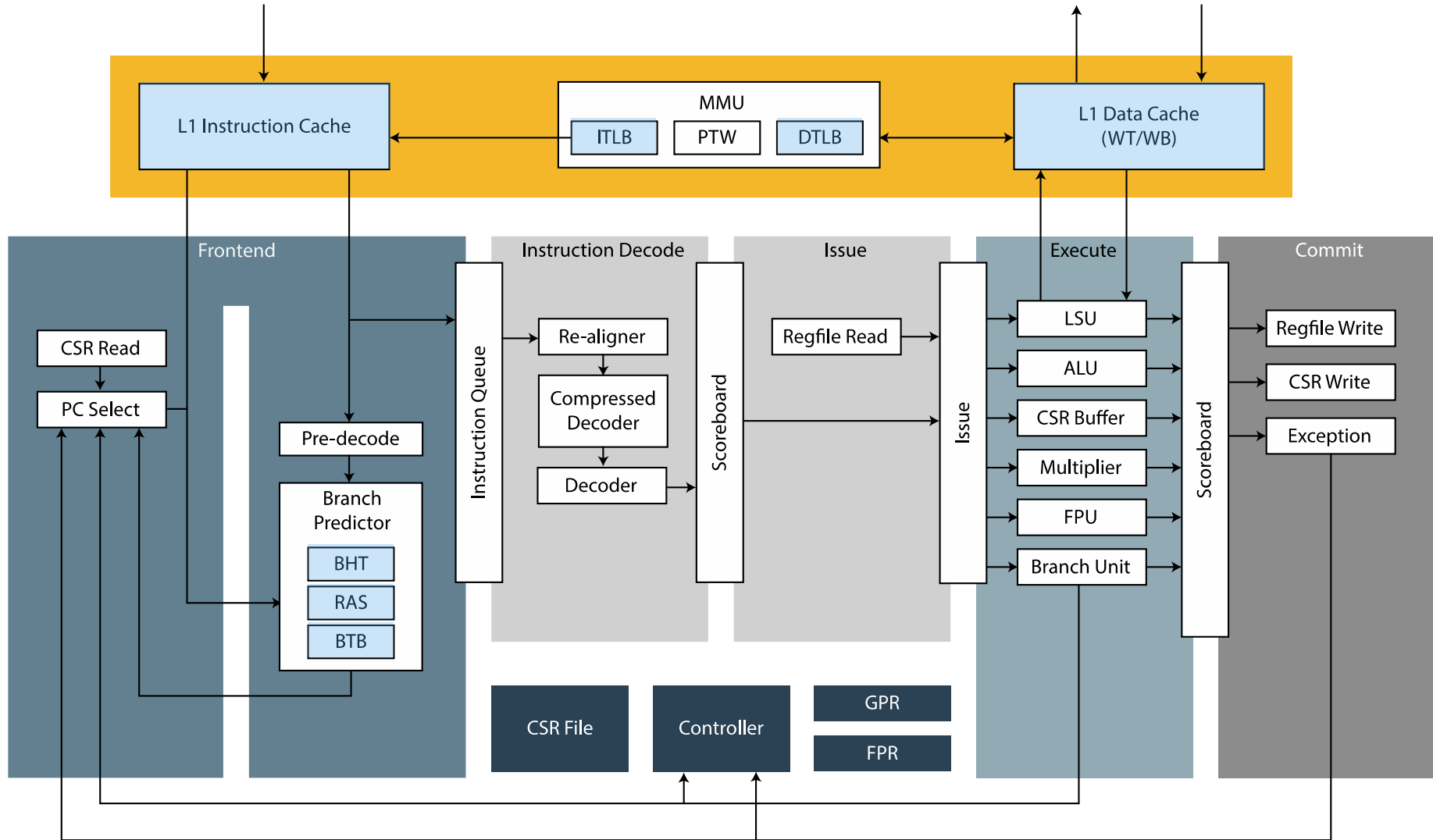
$N = 10^6$, $M = 515.7$ mb, $M_0 = 1.1$ mb

fence.t Instruction



TCs
CVA6
fence.t
CS
Costs
End

fence.t Implementation: Basic Flush

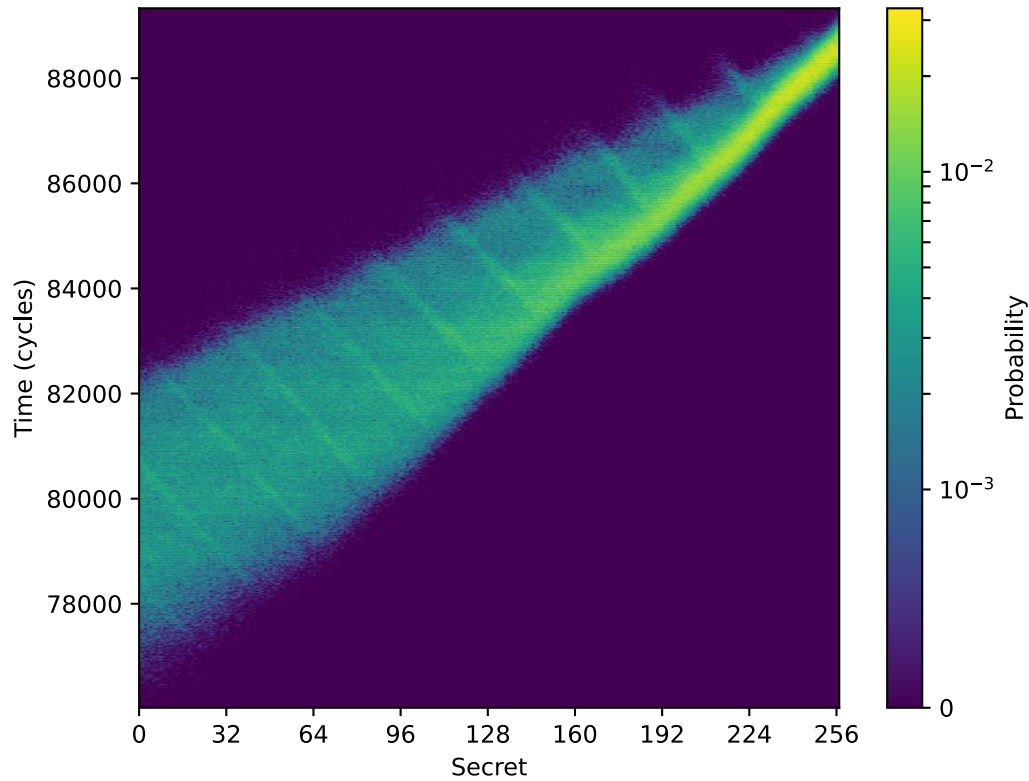


TCs
CVA6
fence.t
CS
Costs
End

L1 Data Cache: Basic Flush

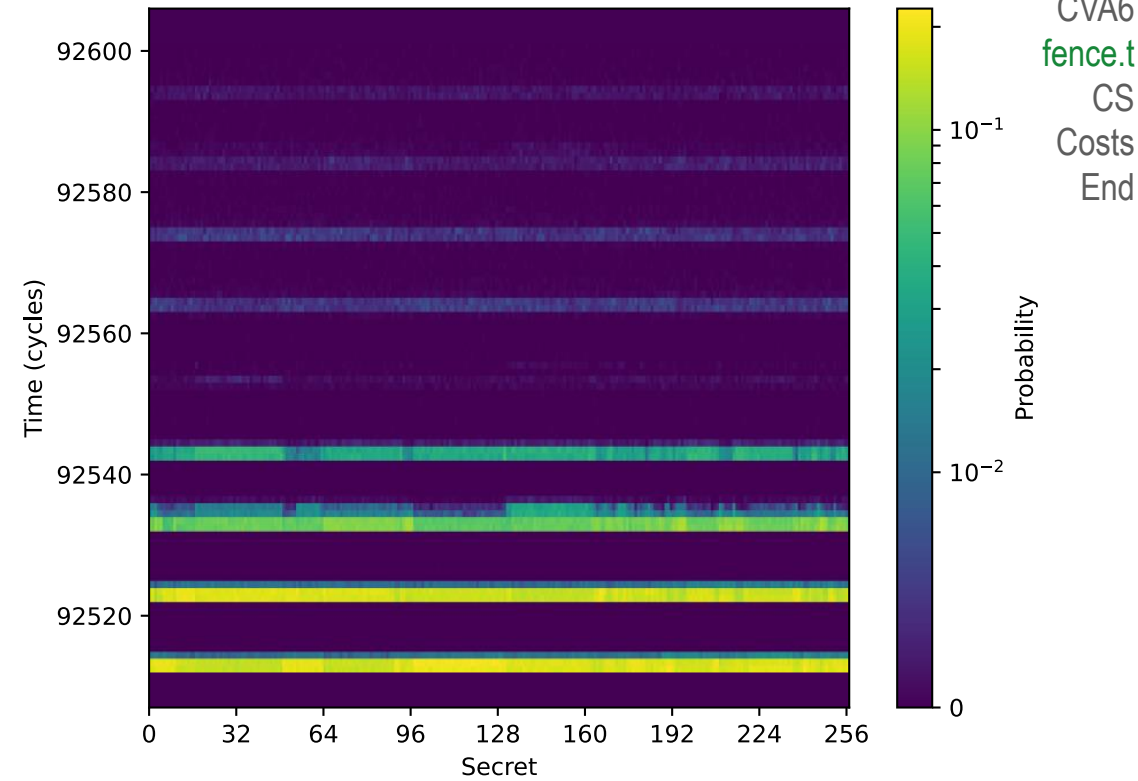


Unmitigated



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

fence.t (flush targeted components on context switch)

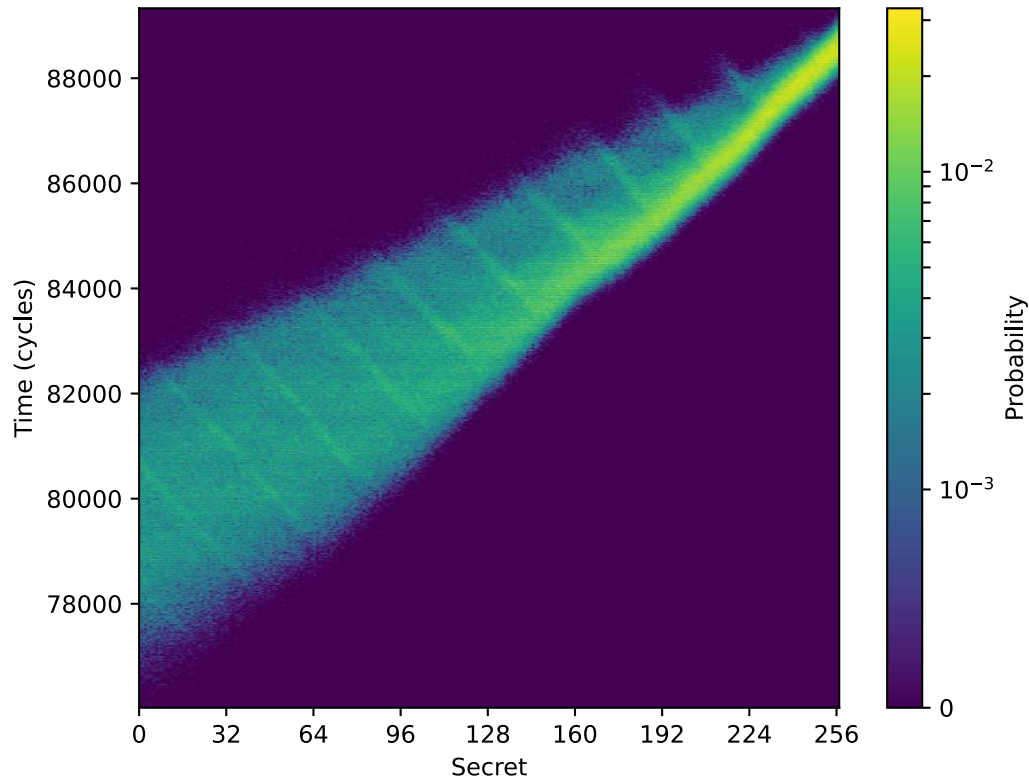


$N = 10^6$, $M = 7.7$ mb, $M_0 = 1.4$ mb

L1 Data Cache: Basic Flush

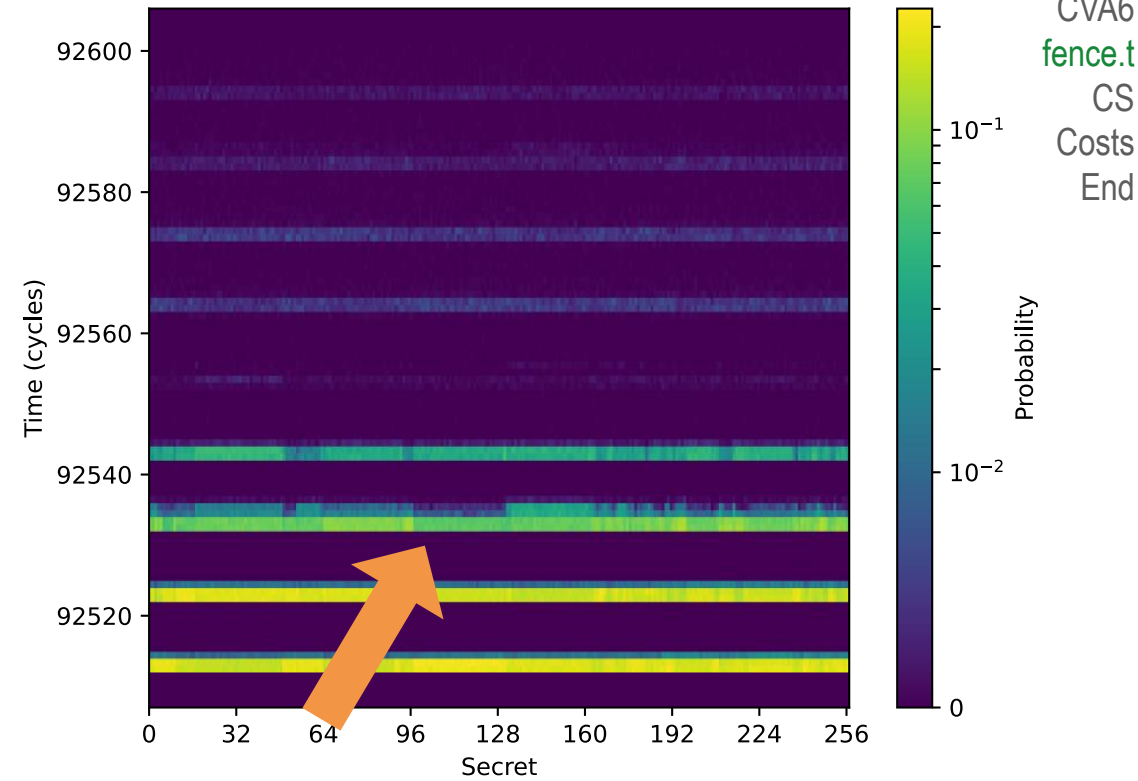


Unmitigated



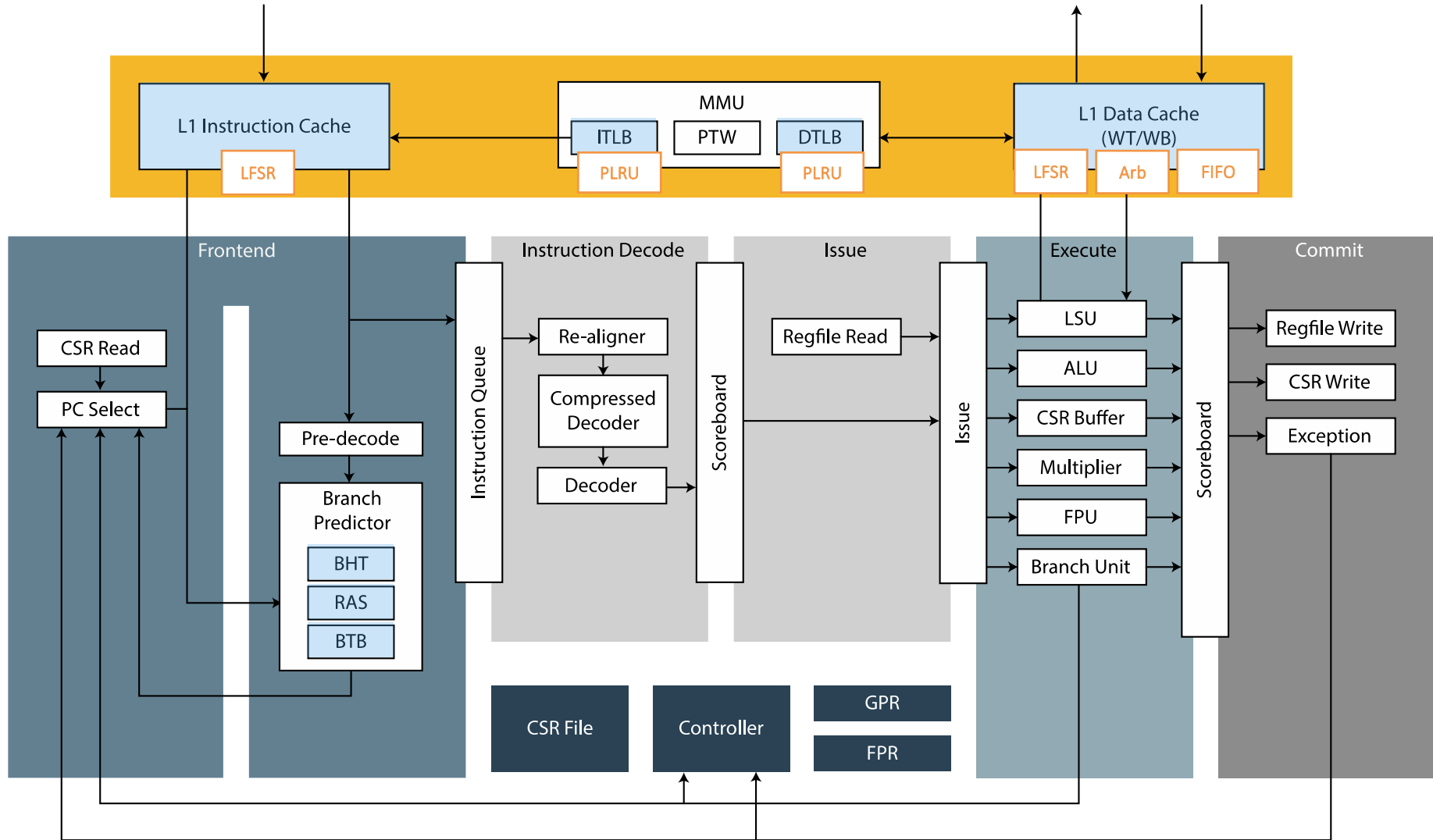
$N = 10^6$, $M = 1667.3$ mb, $M_0 = 0.5$ mb

fence.t (flush targeted components on context switch)



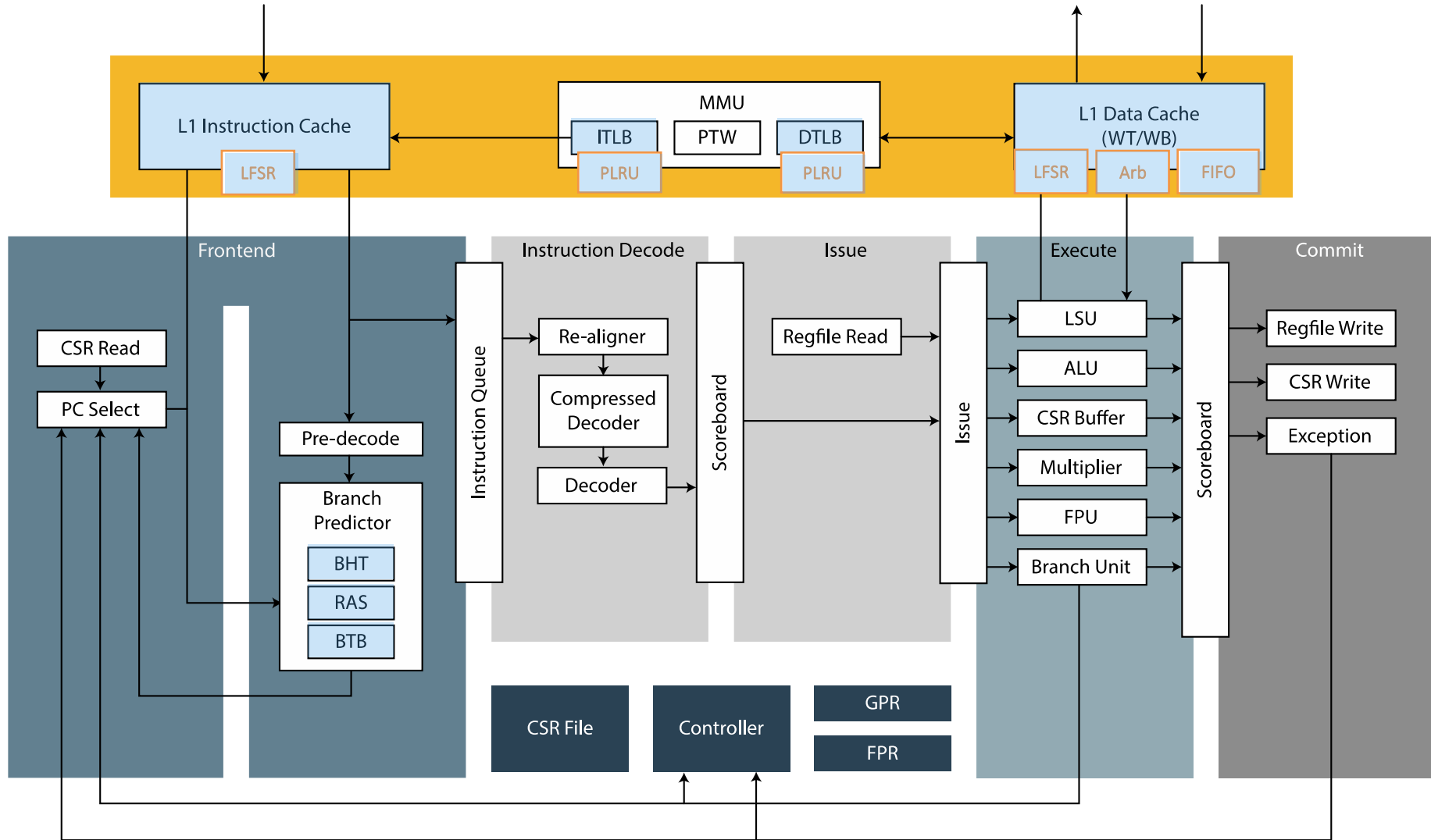
$N = 10^6$, $M = 7.7$ mb, $M_0 = 1.4$ mb

2nd Order Microarchitectural State



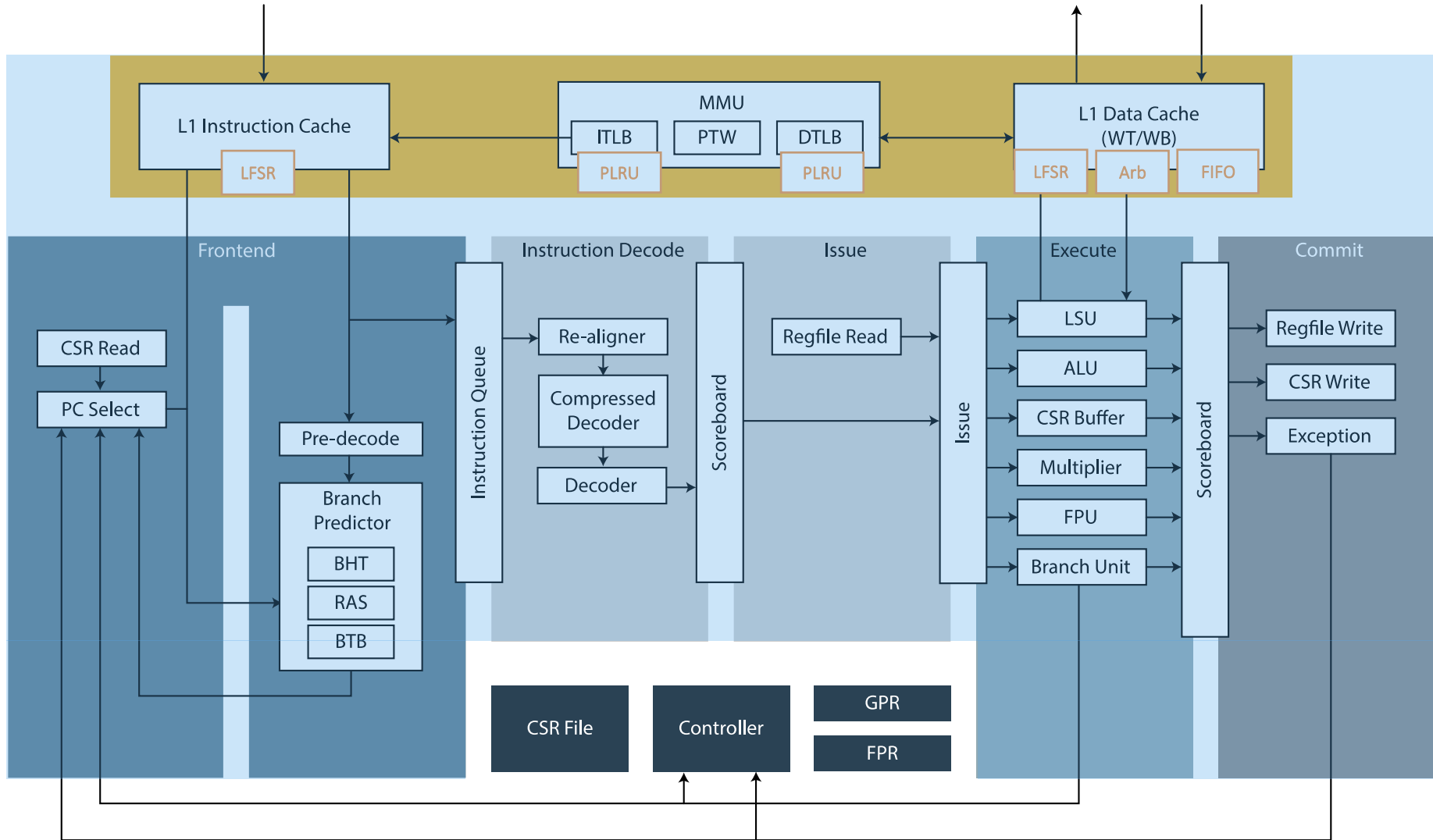
TCs
CVA6
fence.t
CS
Costs
End

fence.t Implementation: Extensive Selective Flush



TCs
CVA6
fence.t
CS
Costs
End

fence.t Implementation: Microreset



TCs
CVA6
fence.t
CS
Costs
End

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.

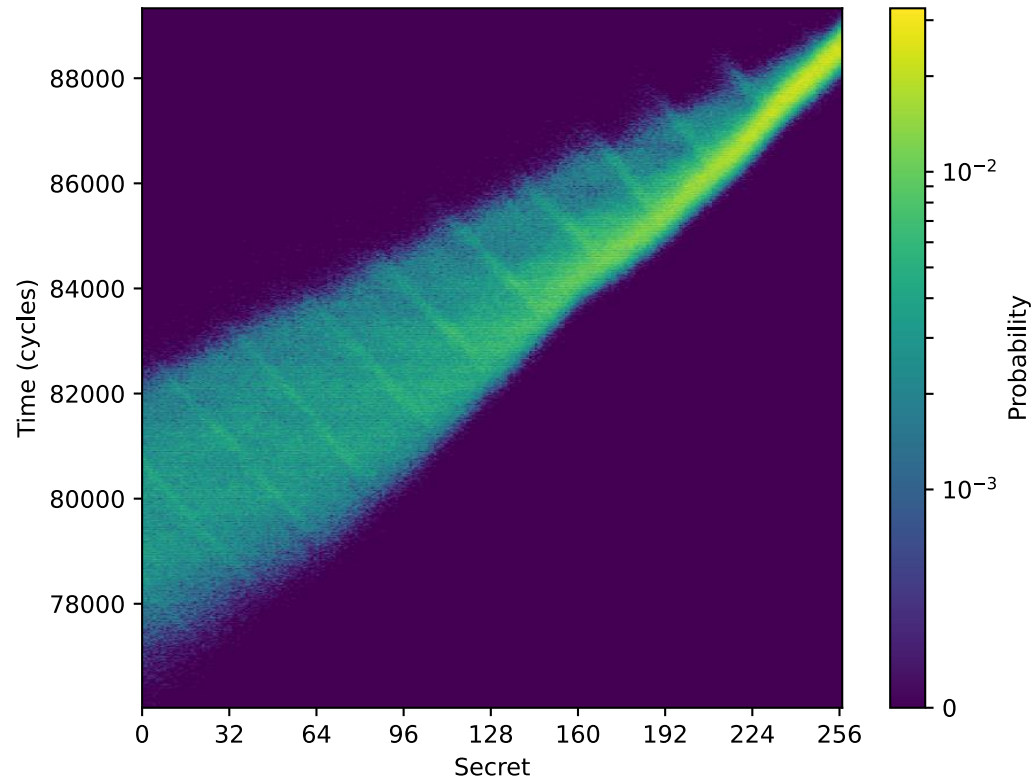


TCs
CVA6
fence.t
CS
Costs
End

L1 Data Cache Channel: Microreset

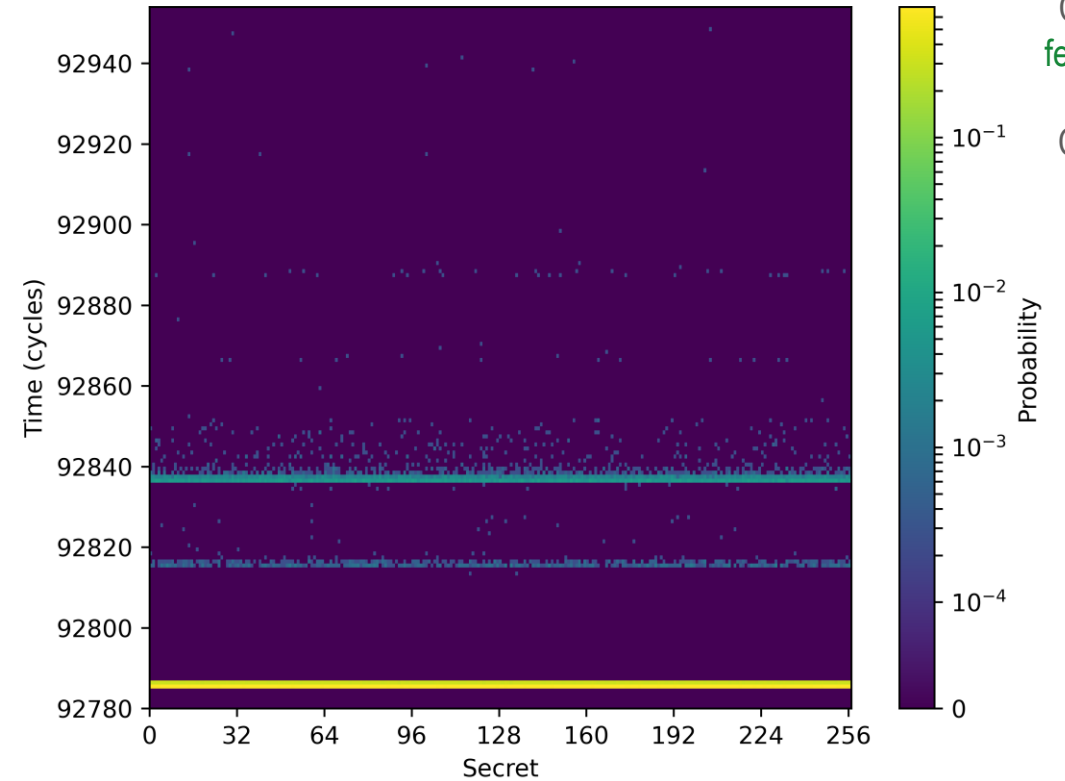


Unmitigated



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

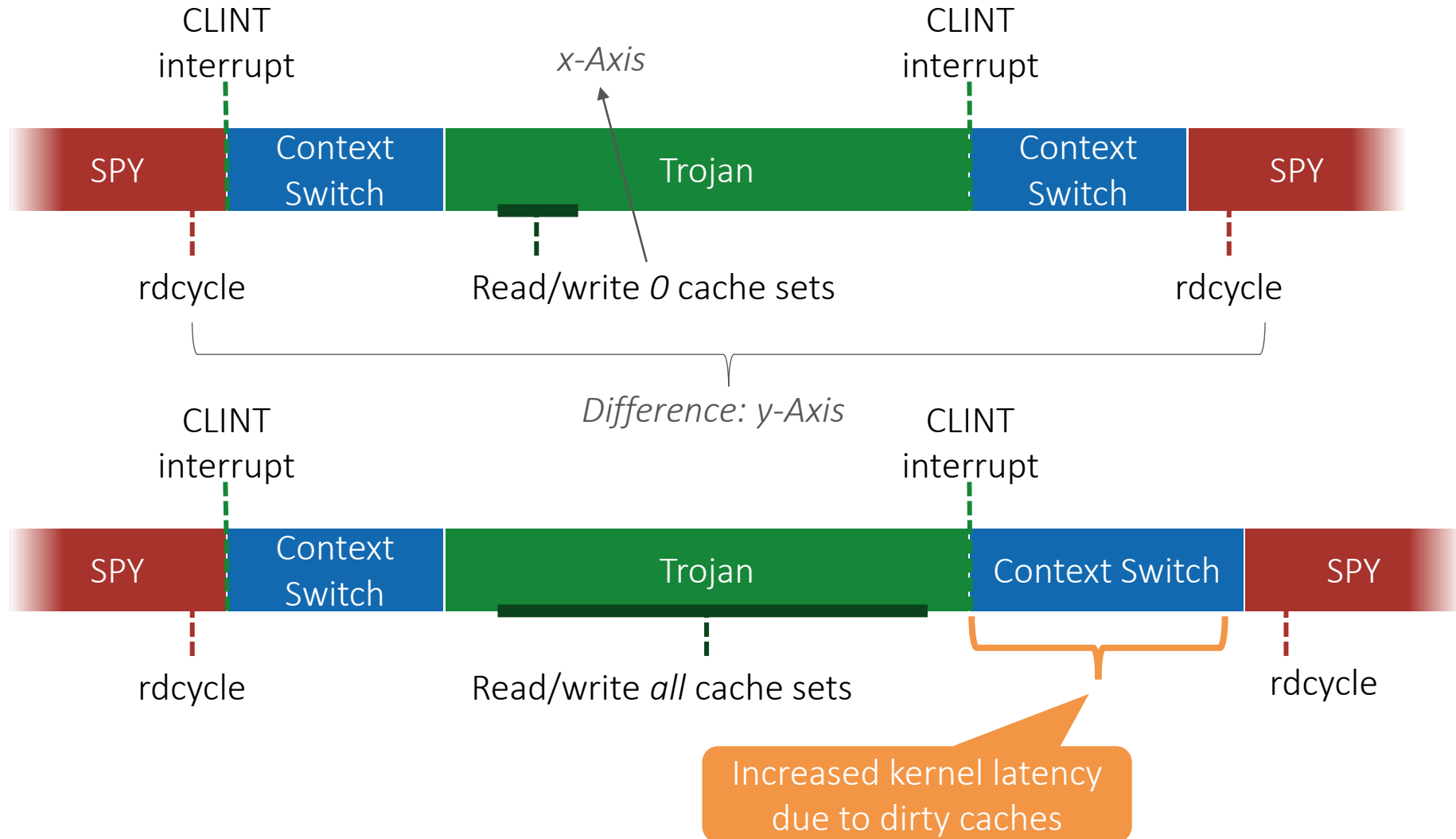
fence.t (Microreset)



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

TCs
CVA6
fence.t
CS
Costs
End

Context-Switch Latency Channel



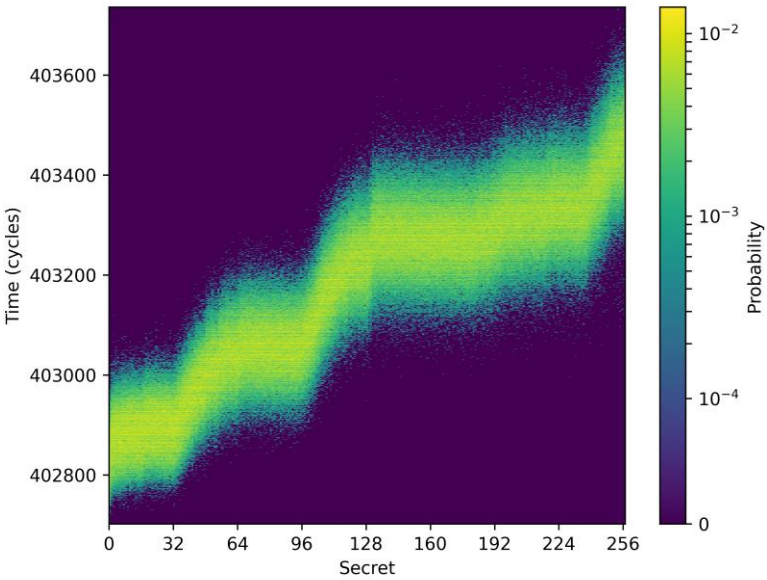
TCs
CVA6
fence.t
CS
Costs
End

Context-Switch Latency Channel



TCs
CVA6
fence.t
CS
Costs
End

Unmitigated

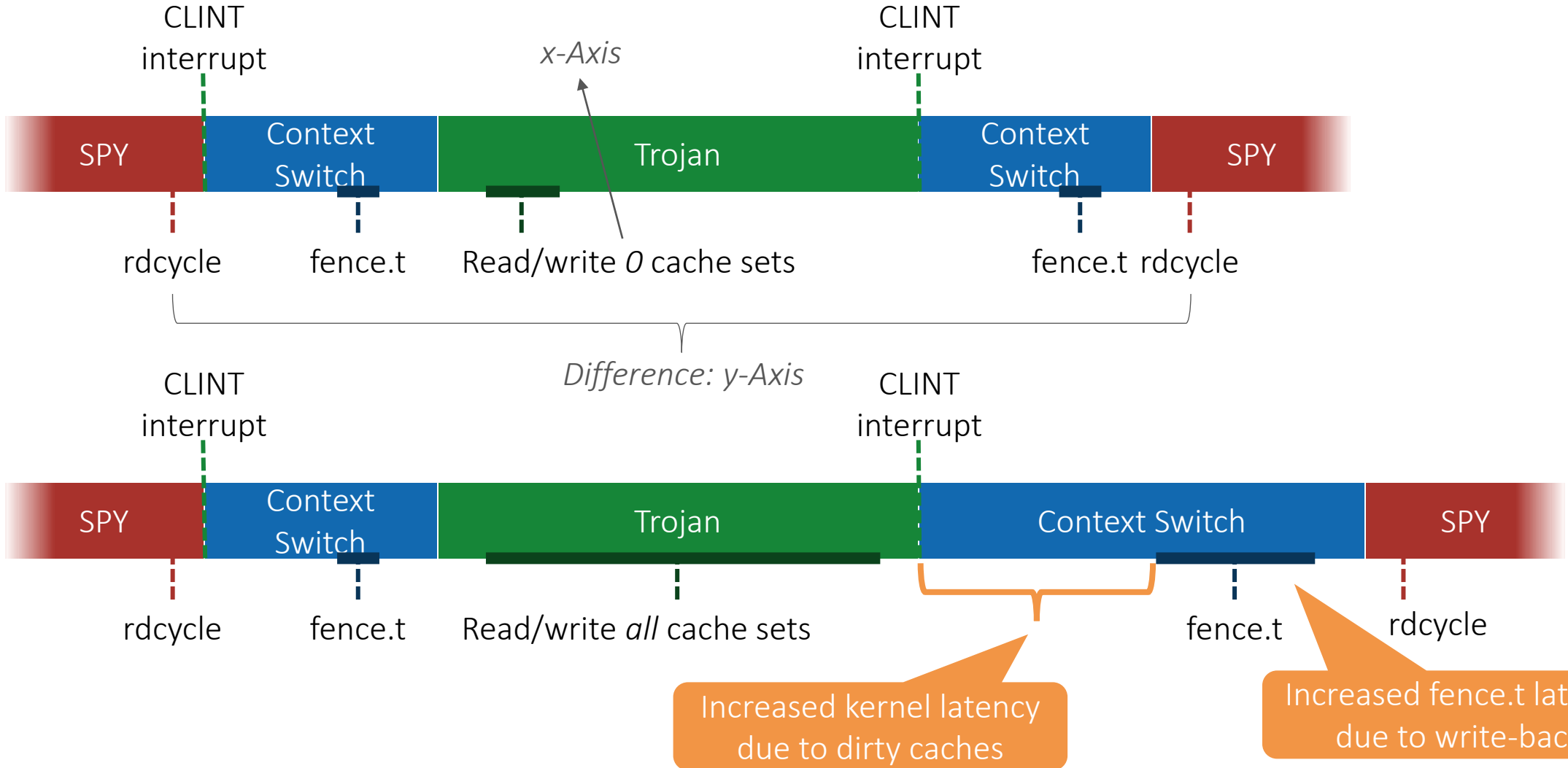


$M = 1297.9$ mb, $M_0 = 0.5$ mb

Context-Switch Latency Channel



TCs
CVA6
fence.t
CS
Costs
End



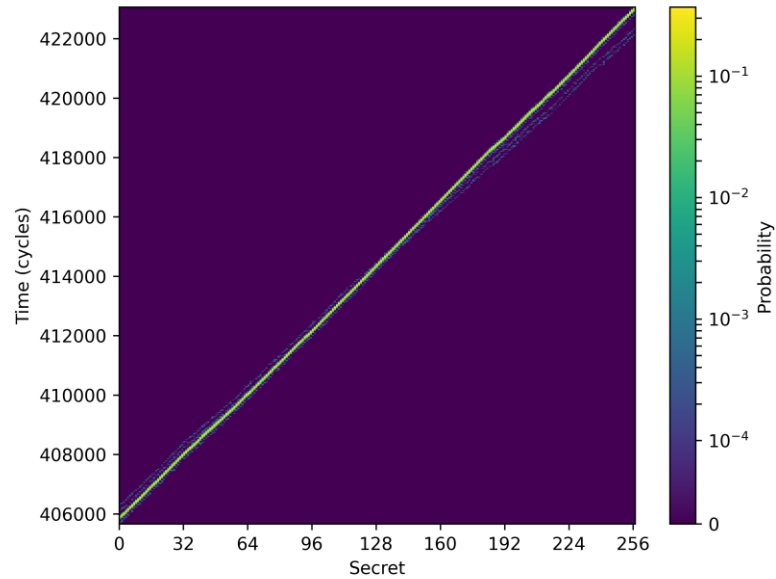
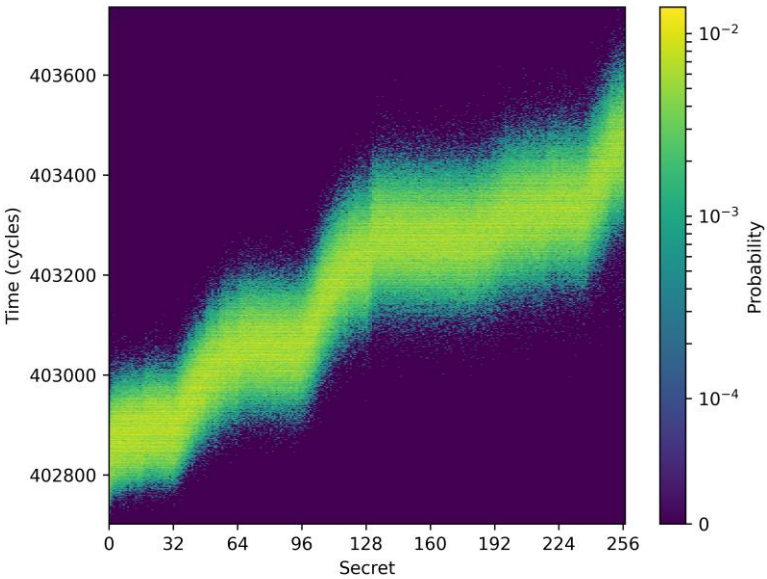
Context-Switch Latency Channel



TCs
CVA6
fence.t
CS
Costs
End

Unmitigated

fence.t



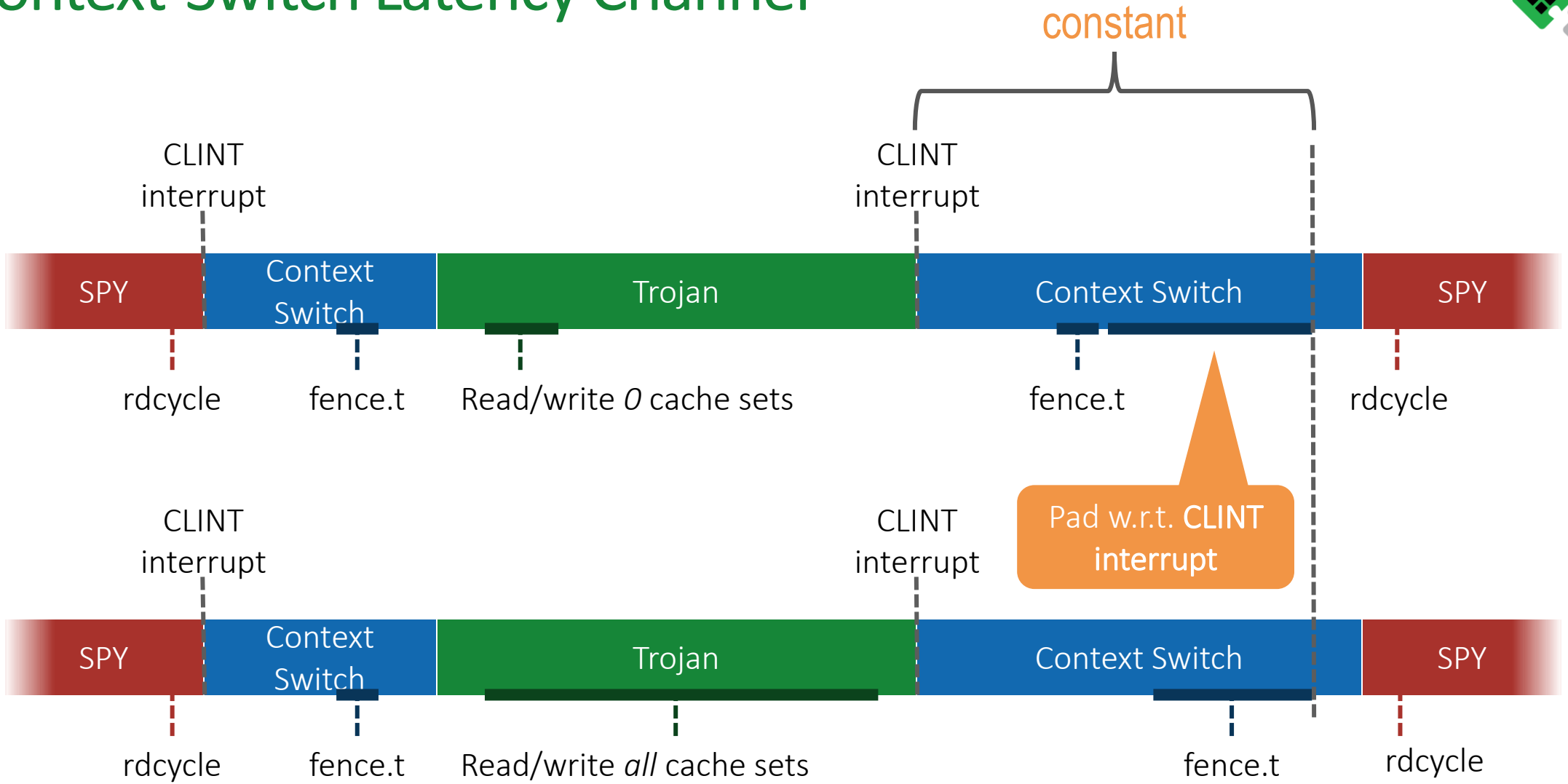
$M = 1297.9 \text{ mb}, M_0 = 0.5 \text{ mb}$

$M = 7257.2 \text{ mb}, M_0 = 0.4 \text{ mb}$

Context-Switch Latency Channel



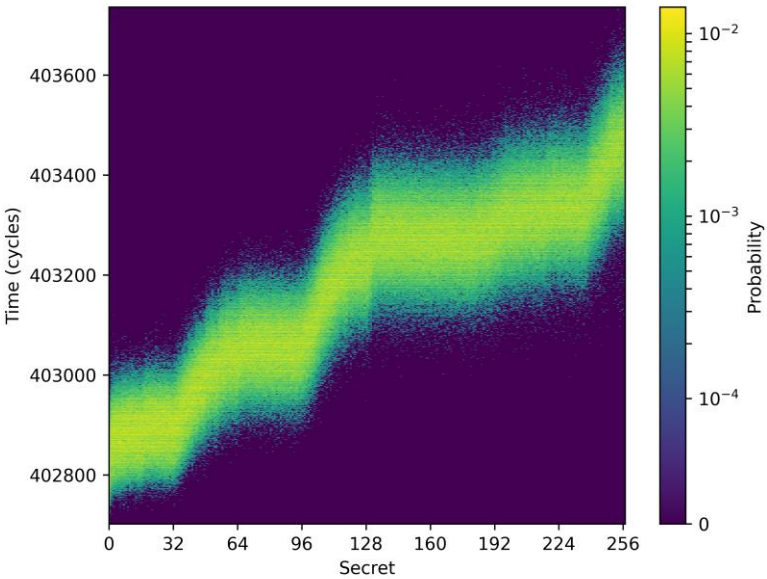
TCs
CVA6
fence.t
CS
Costs
End



Context-Switch Latency Channel

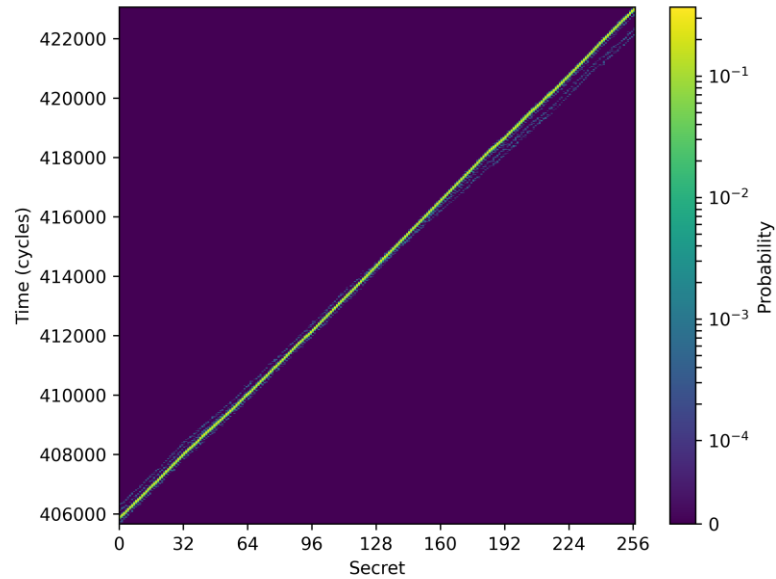


Unmitigated



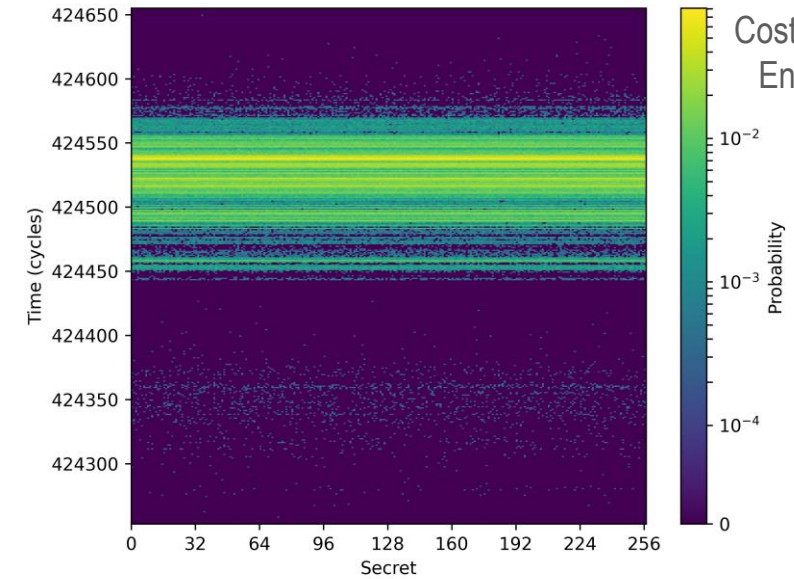
$M = 1297.9$ mb, $M_0 = 0.5$ mb

fence.t



$M = 7257.2$ mb, $M_0 = 0.4$ mb

fence.t + padding



$M = 1.4$ mb, $M_0 = 1.6$ mb

TCs
CVA6
fence.t
CS
Costs
End

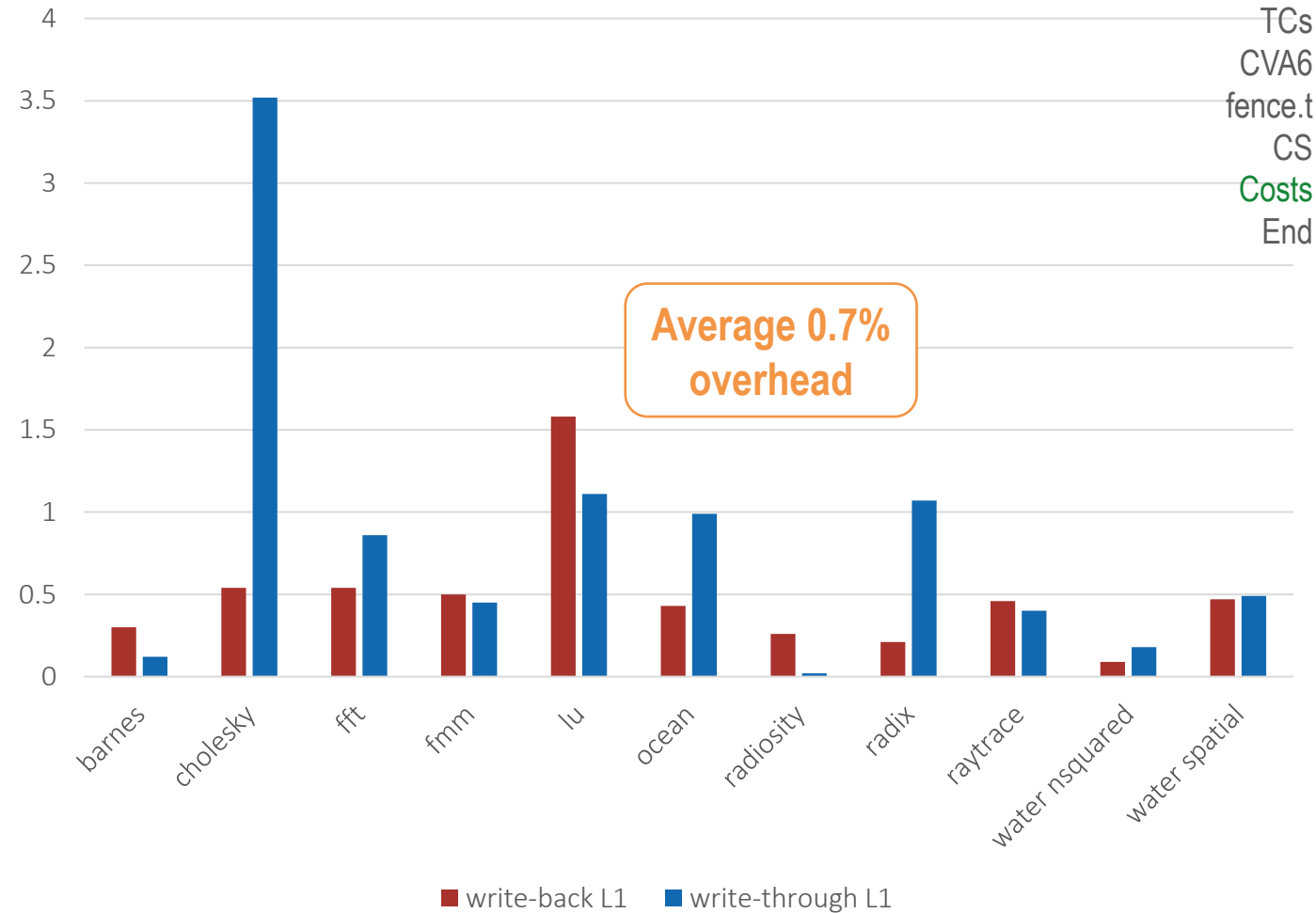
Costs

- 2 threads: 1 benchmark + 1 idle
- 1GHz system clock
- 10ms timeslice
- Context switch every 10M cycles
- Synthesis in GF22FDX @1GHz

Negligible hardware costs



Splash-2 Benchmark Overhead (%)



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



TCs
CVA6
fence.t
CS
Costs
End