C-state latency measurement infrastructure

Linux Plumbers Conference 2019

Artem Bityutskiy,
Intel Finland

Agenda

- **Introduction**
- What and how measured
- Why useful
- Visual
- Current implementation
- Challenges

Introduction

- Measure **C-state** latency
- Wake Up Latency Tracer (WULT)
- Plan:
  - Open source (Github, GPL)
  - Kernel drivers → upstream

Goal

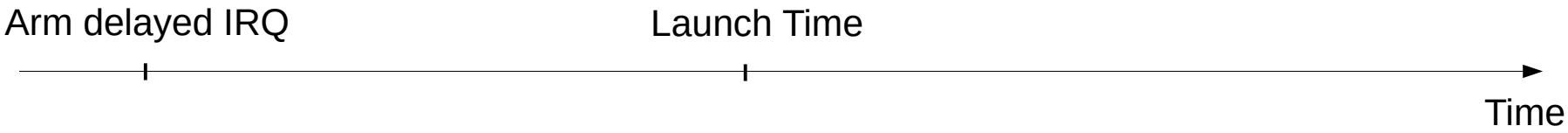
Get early feedback on WULT kernel drivers

- How do we transform them into something upstreamable

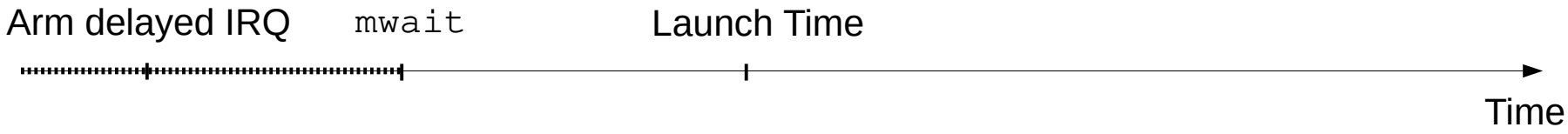
Agenda

- Introduction
- **What and how measured**
- Why useful
- Visual
- Current implementation
- Challenges

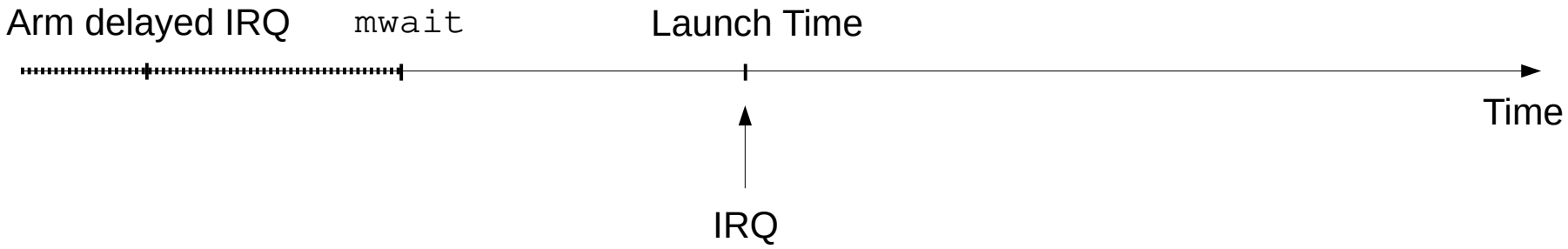
1. Arm delayed IRQ



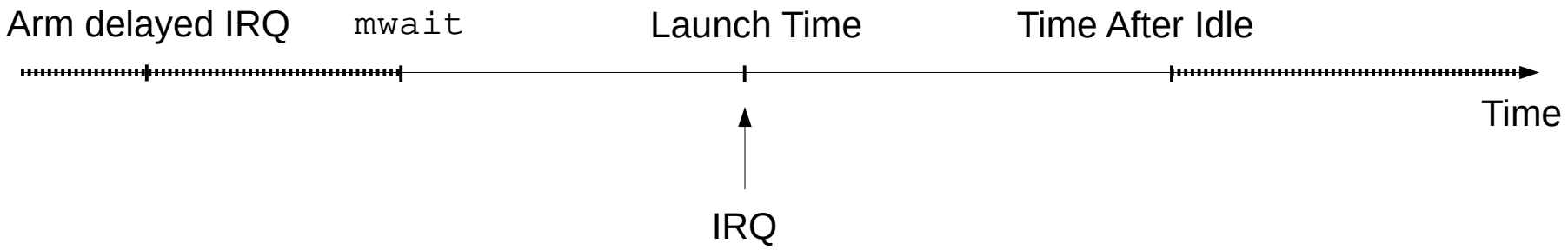
2. Request C-state



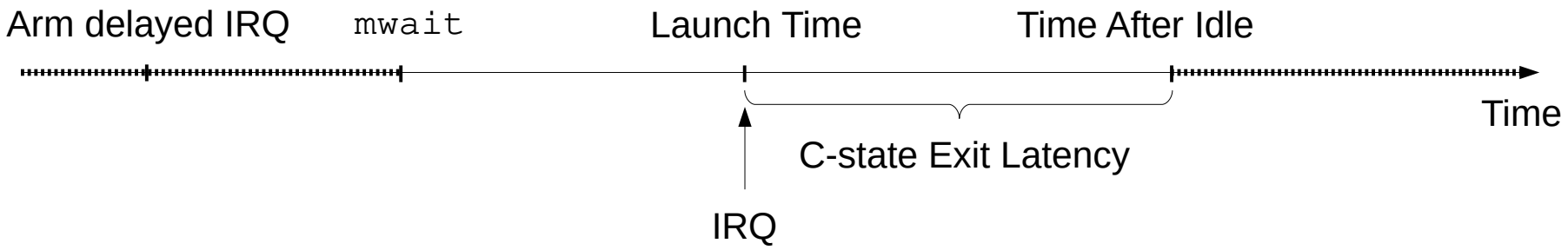
3. IRQ happens



4. Exiting C-state



4. C-state exit latency



C-state Exit Latency = Time After Idle – Launch Time

Agenda

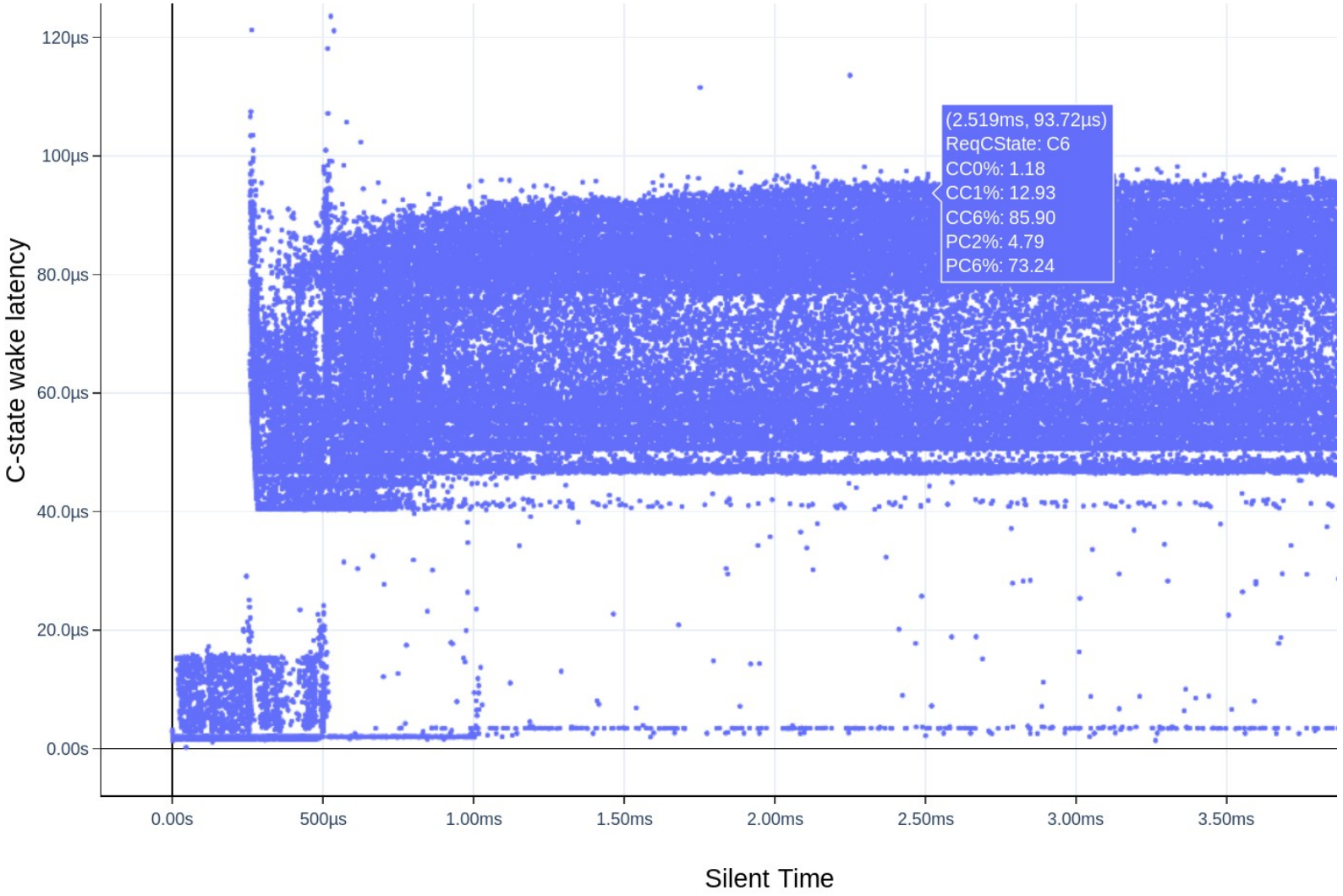
- Introduction
- What and how measured
- **Why useful**
- Visual
- Current implementation
- Challenges

Why useful?

- Improve Linux PM QoS subsystem
  - Improve intel_idle driver
- Analyze the HW platform
  - No need for expensive lab equipment

Agenda

- Introduction
- What and how measured
- Why useful
- **Visual**
- Current implementation
- Challenges


C-state wake latency
120µs
100µs
80.0µs
60.0µs
40.0µs
20.0µs
0.00s
0.00s
500µs
1.00ms
1.50ms
2.00ms
2.50ms
3.00ms
3.50ms
Silent Time
(2.519ms, 93.72µs)
ReqCState: C6
CC0%: 1.18
CC1%: 12.93
CC6%: 85.90
PC2%: 4.79
PC6%: 73.24

Agenda

- Introduction
- What and how measured
- Why useful
- Visual
- Current implementation
  - **Delayed interrupts**
  - High level design
- Challenges

Delayed interrupts

- Intel I210 PCIe NIC
- On-board 8ns resolution clock
- Host can read the clock
- Host can arm delayed interrupt

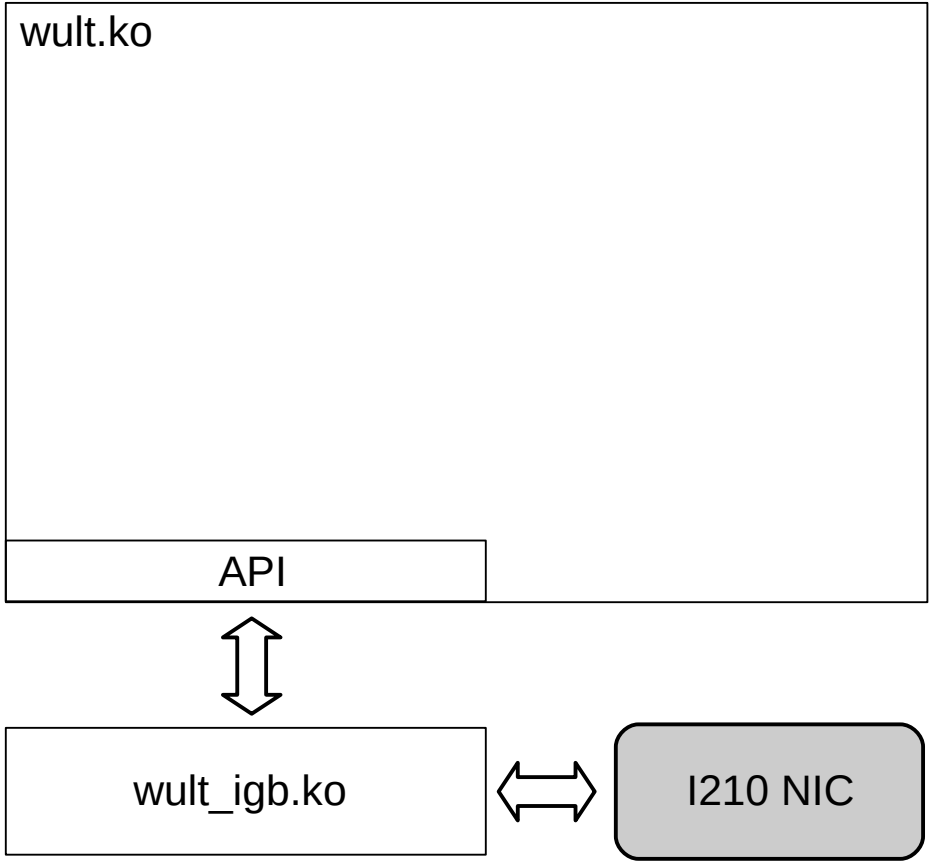
Why not timers?

- Intel: P-unit is aware of timers
  - Pre-wakes and removes latency
- May not be the case for other platforms
- External IRQ source is required

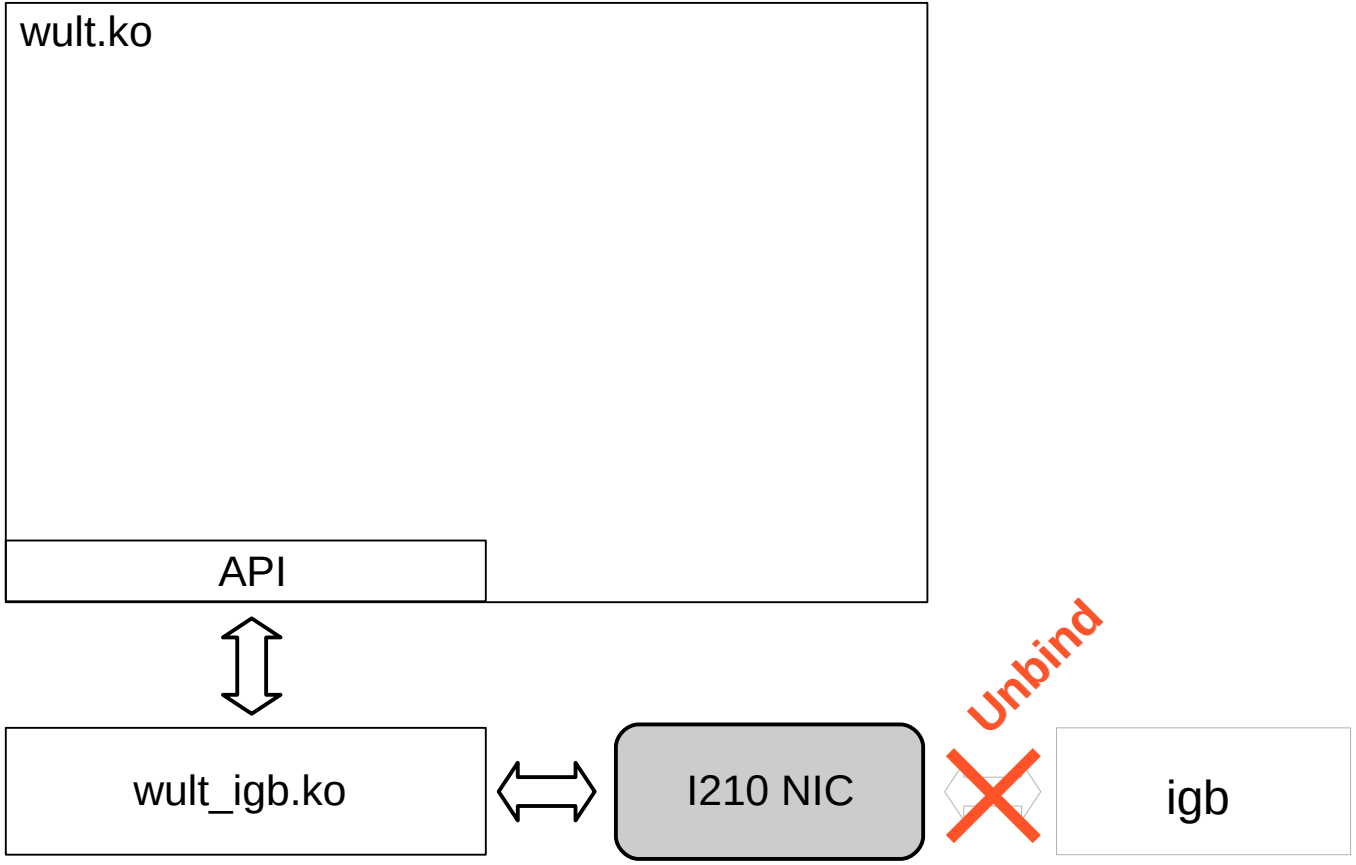
Agenda

- Introduction
- What and how measured
- Why useful
- Visual
- Current implementation
  - Delayed interrupts
  - **High level design**
- Challenges

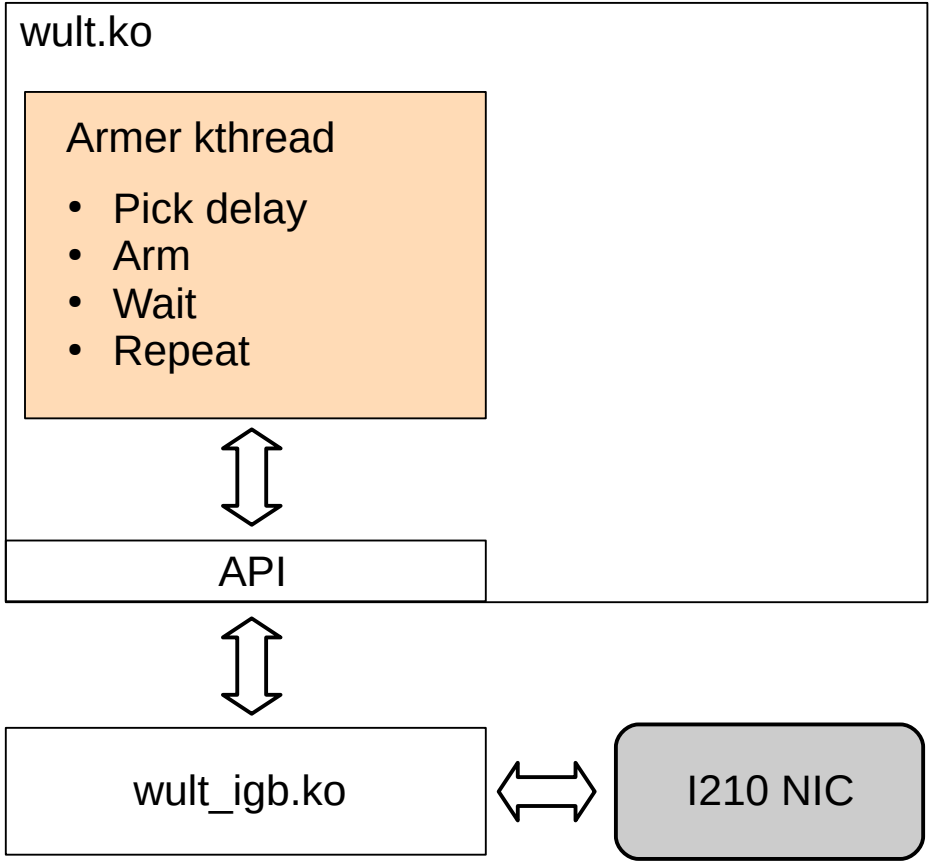
Main kernel components



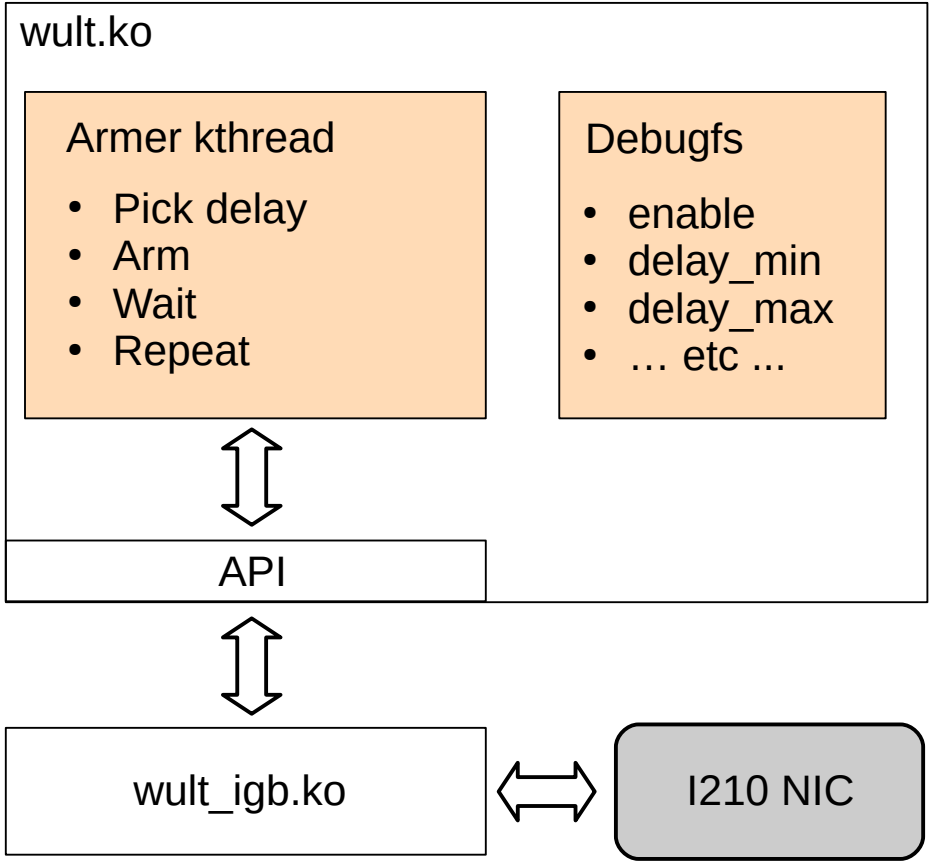
Standard driver



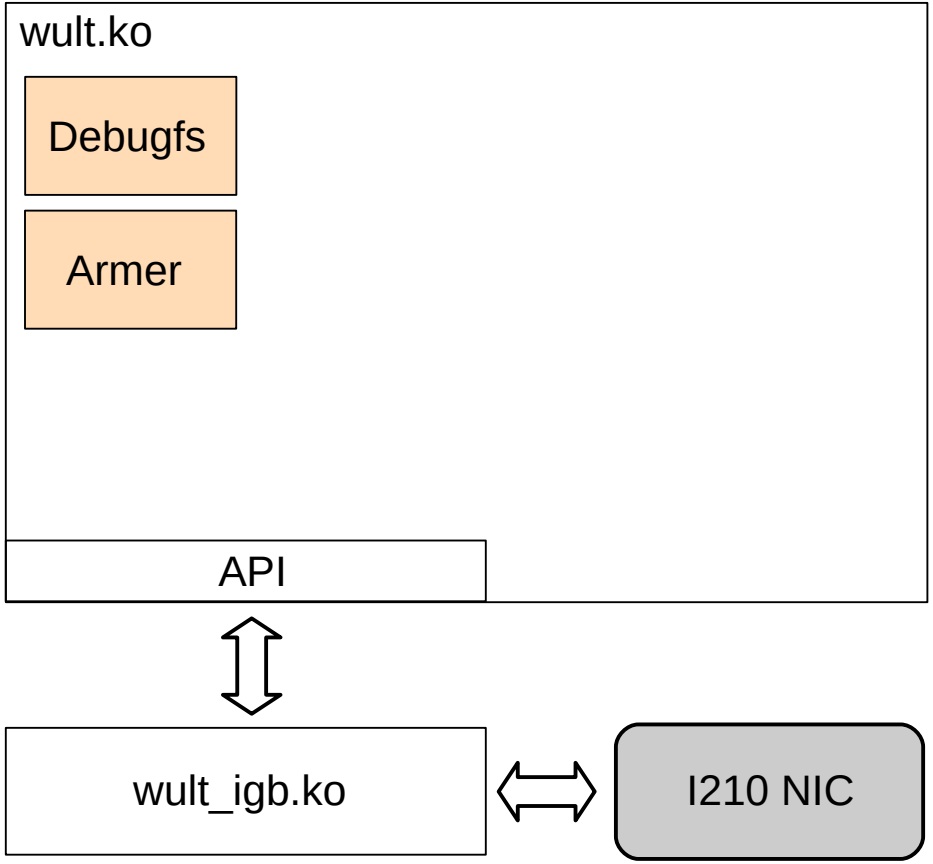
Armer


wult.ko
Armer kthread
• Pick delay
• Arm
• Wait
• Repeat
API
wult_igb.ko
I210 NIC

Debugfs

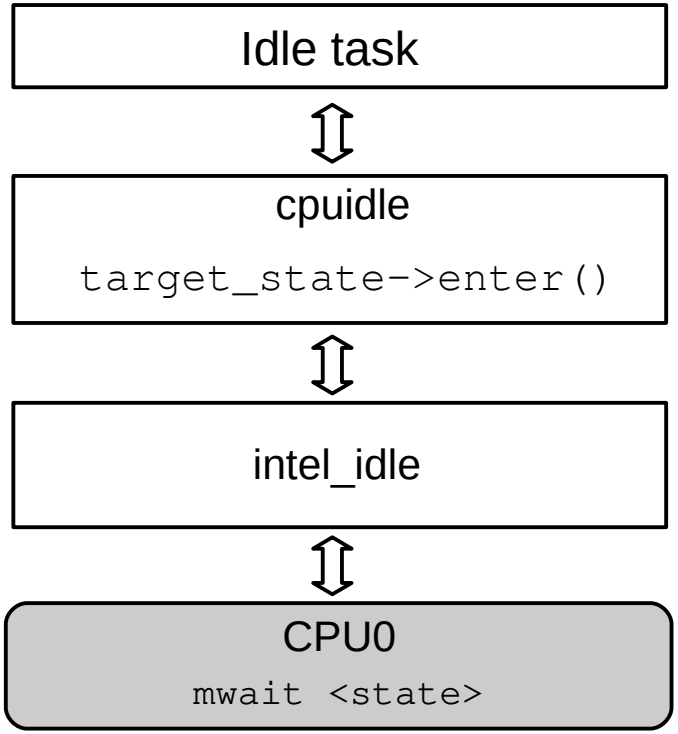

wult.ko
Armer kthread
• Pick delay
• Arm
• Wait
• Repeat
Debugfs
• enable
• delay_min
• delay_max
• … etc ...
API
wult_igb.ko
I210 NIC

Zoom out

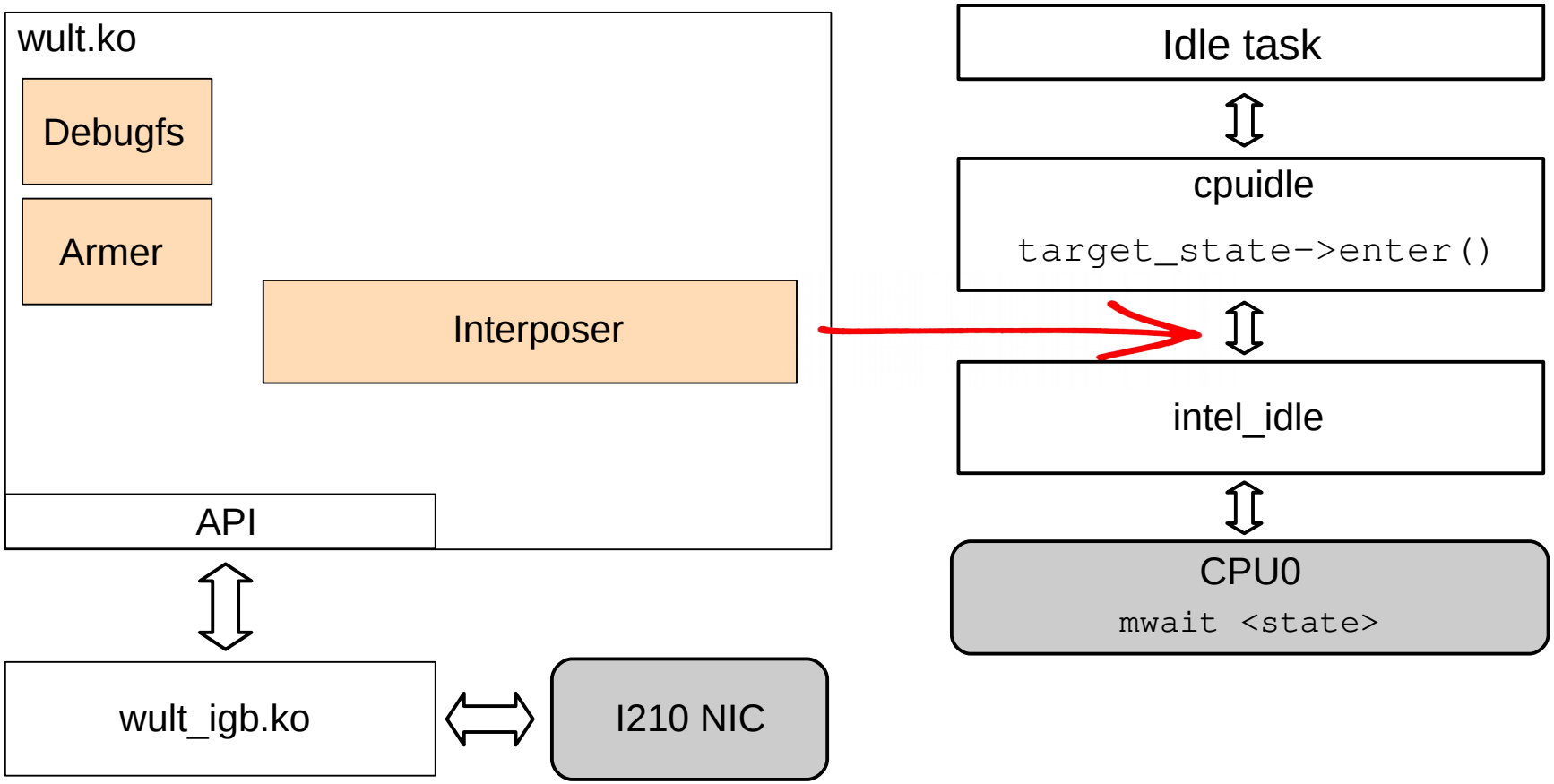


Idle flow

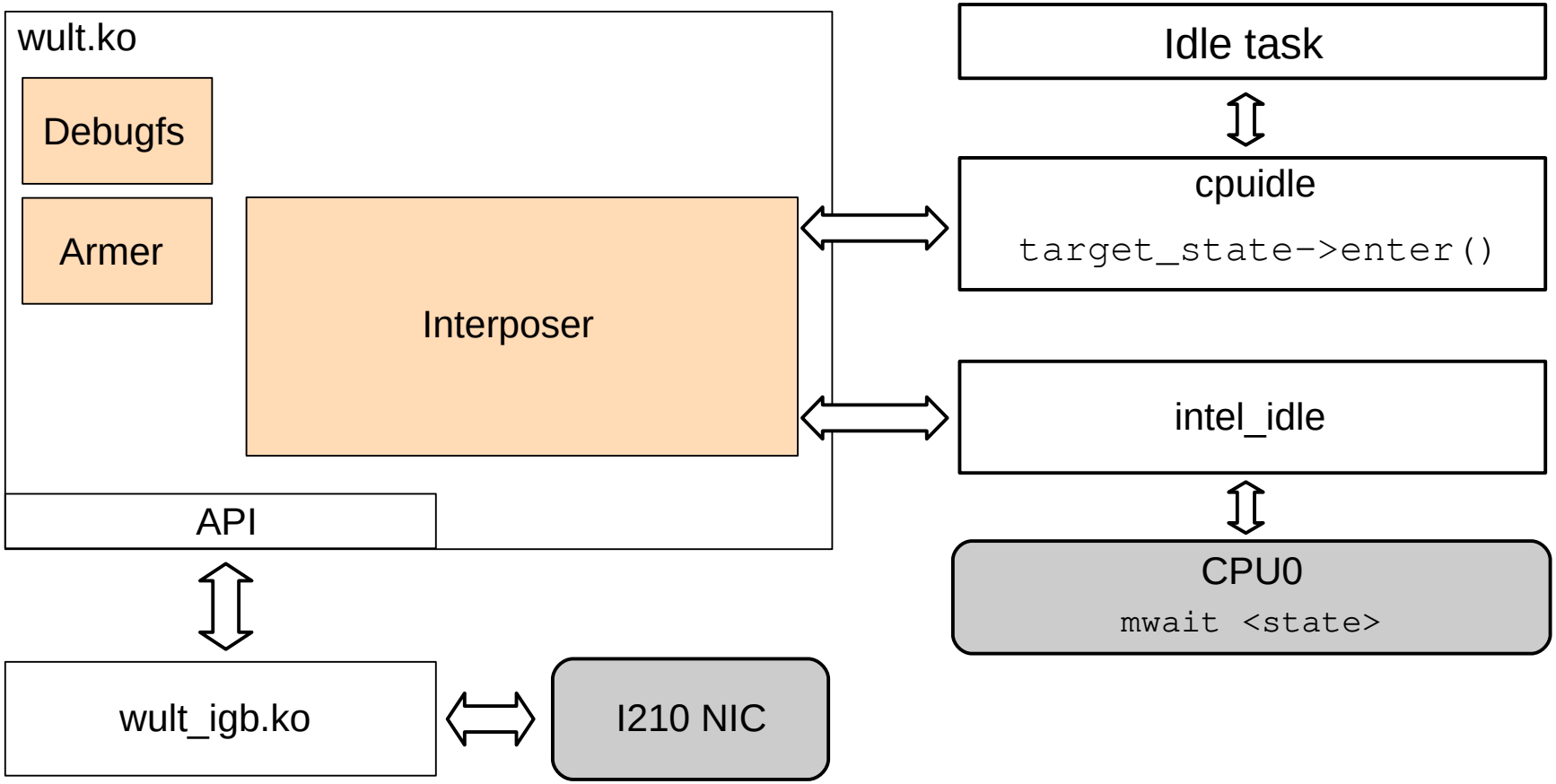
wult.ko
Debugfs
Armer
API
wult_igb.ko
I210 NIC


Idle task
cpuidle
target_state->enter()
intel_idle
CPU0
mwait <state>

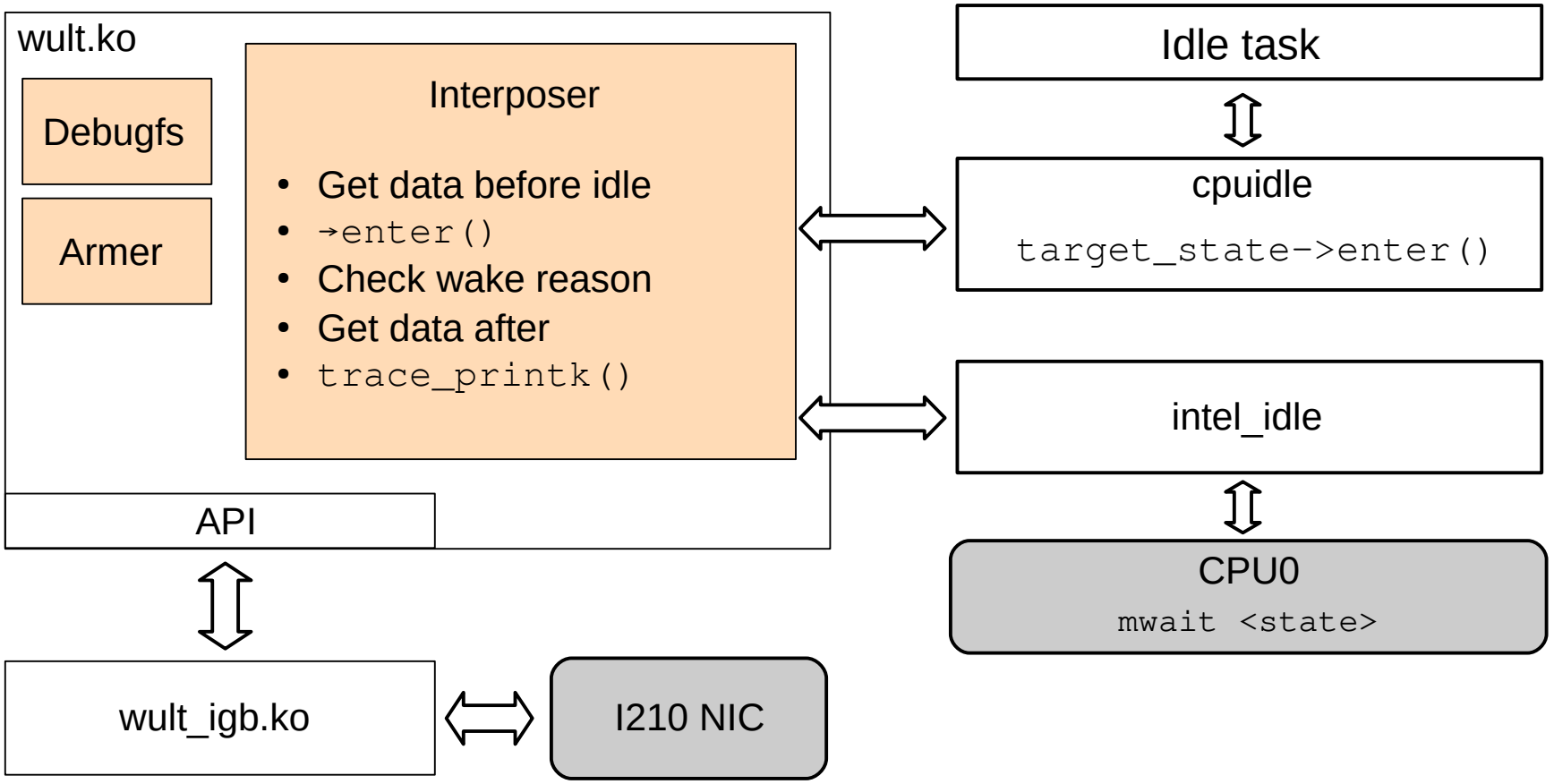
Interposer


wult.ko
Debugfs
Armer
Interposer
API
wult_igb.ko
I210 NIC
Idle task
cpuidle
target_state->enter()
intel_idle
CPU0
mwait <state>

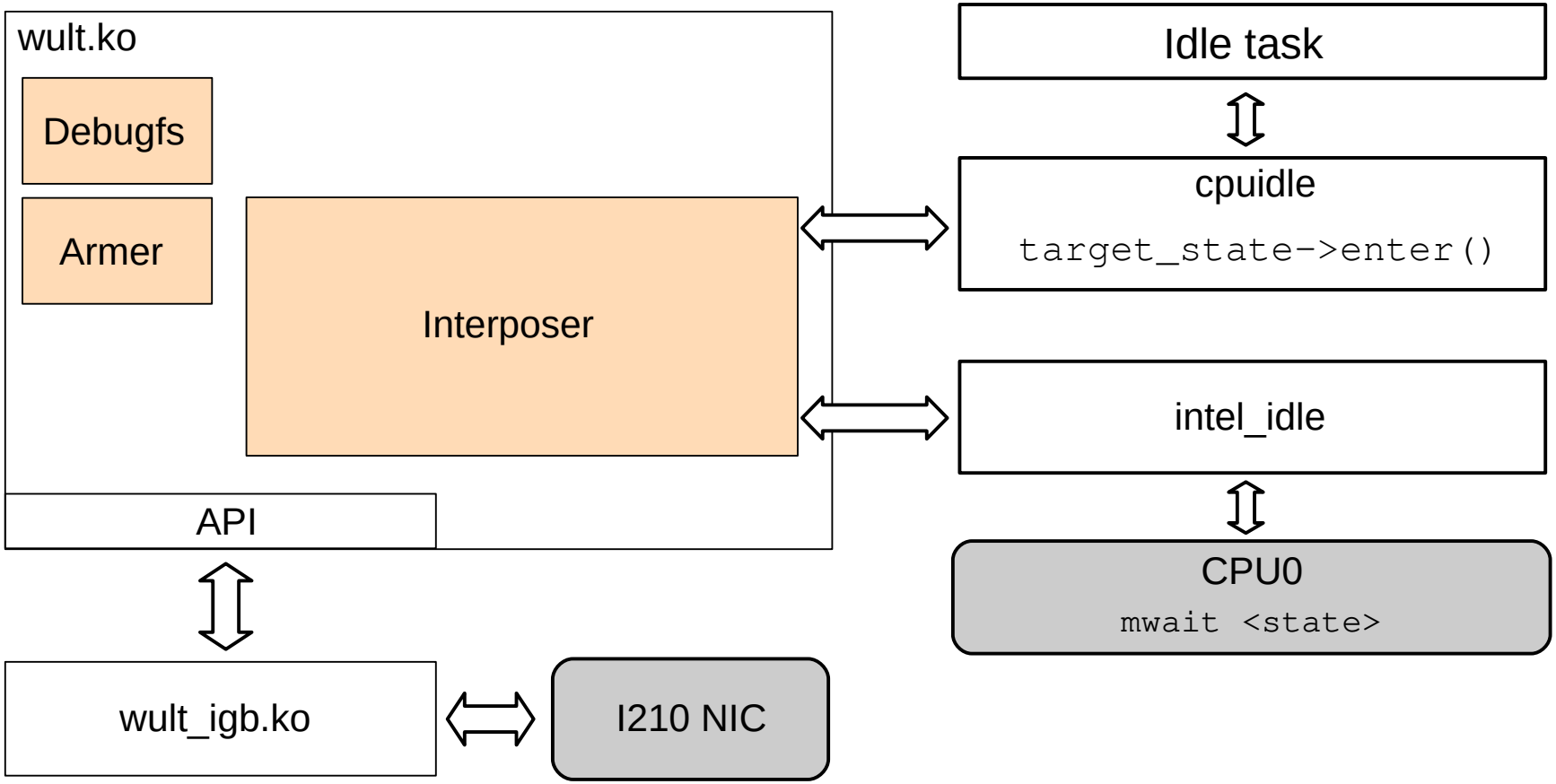
Interposed idle flow



Interposer details



Comments?


wult.ko
Debugfs
Armer
Interposer
API
Idle task
cpuidle
target_state->enter()
intel_idle
CPU0
mwait <state>
wult_igb.ko
I210 NIC

Agenda

- Introduction
- What and how measured
- Why useful
- Visual
- Current implementation
- Challenges
  - **The placement**
  - C-state statistics
  - User-space interfaces

The placement

- Where in the kernel tree this stuff belongs to?
  - Proposal: `drivers/idle/wult`

Agenda

- Introduction
- What and how measured
- Why useful
- Visual
- Current implementation
- Challenges
  - The placement
  - **Measurement data**
  - User-space interfaces

Measurement data

- Data sent to userspace includes:
  - Launch Time
  - Time before and after idle
  - Delta TSC / APERF / MPERF
  - C-state counters delta:
    - CC1, CC1E, CC3, CC6, CC7
    - PC2, PC3, PC6, PC7, PC8, PC9, PC10

Challenges

- Architecture-dependent
- C-states are platform/model dependent
  - E.g., servers have only few
- Today: use 0 for non-existing C-states
- Optimal: more C-state awareness in Kernel
  - Turbostat needs this too

Agenda

- Introduction
- What and how measured
- Why useful
- Visual
- Current implementation
- Challenges
  - The placement
  - Measurement data
  - **User-space interfaces**

Debugfs

- Today: debugfs for configuring
- Upstream
  - Sysfs ?
  - `/sys/class/wult` ?

trace_printk

- Today: `trace_printk(): /sys/kernel/debug/tracing/trace`
- Upstream
  - Plan: use dynamic tracepoints
  - Tom Zanussi’s patches
  - Hook to existing tracepoints
    - In `cpuidle.c` around ‘->`enter()`’

Thank you!

Backup

1. NIC sends MSI message at Launch Time
2. MSI reaches PCIe root complex
3. Core starts wakes up at Time After Idle

C-state Exit Latency = 1 + 2 + 3 = Time After Idle – Launch Time