2023 DESIGN AND VERIFICATION™ DVCON CONFERENCE AND EXHIBITION EUROPE 10 YEAR ANNIVERSARY

Reverse Hypervisor
Hypervisor for fast SoC Simulation

François-Frédéric Ozog, Shokubai

Mark Burton, Accelera CPSWG chair

accellera SYSTEMS INITIATIVE

Shift left for complex “market” applications

Design House

Silicon Provider

Board Maker

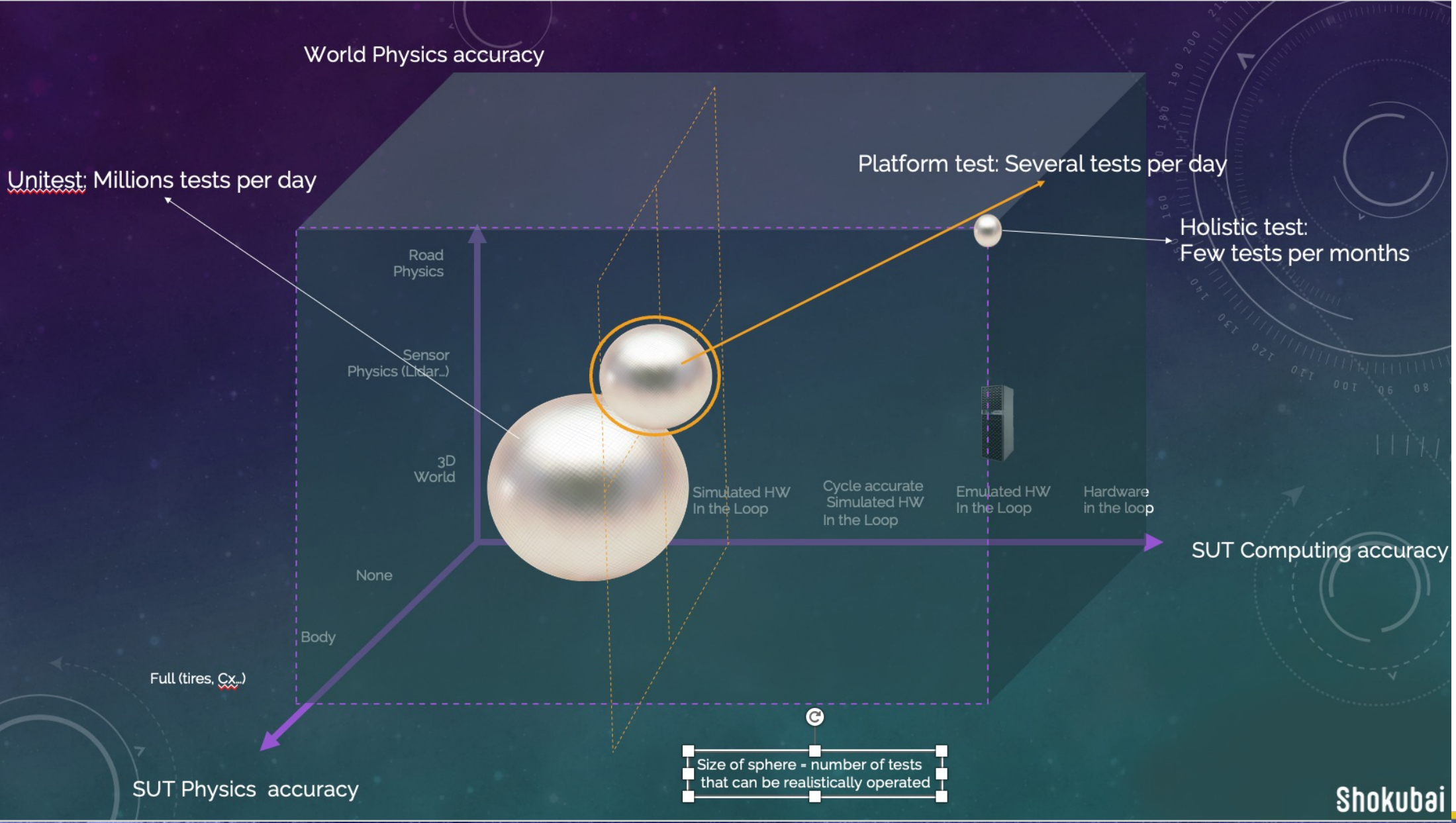
Tier 1

Auto/Plane/Satellite maker

Pre-silicon dev

- **Market apps**
- **Operating System**
- Hypervisor
- Boot Firmware
- Secure Firmware

Presentation focuses on LT while can do AT too



1M km autodrive
ISO21434
UN156

Virtual Platform for “Virtual Prototyping”

- Fast enough VCML processor component (see T4.3 tutorial)
  - From instruction up to “infinite realtime” quantum (1 min. = 1min.)
  - bigLITTLE: 8*3Ghz performance cores + 4*1Ghz low power cores
- **Simulate as little as possible**
- Use of processor facilities to trap what has to be simulated in terms of instructions, architectural state, registers

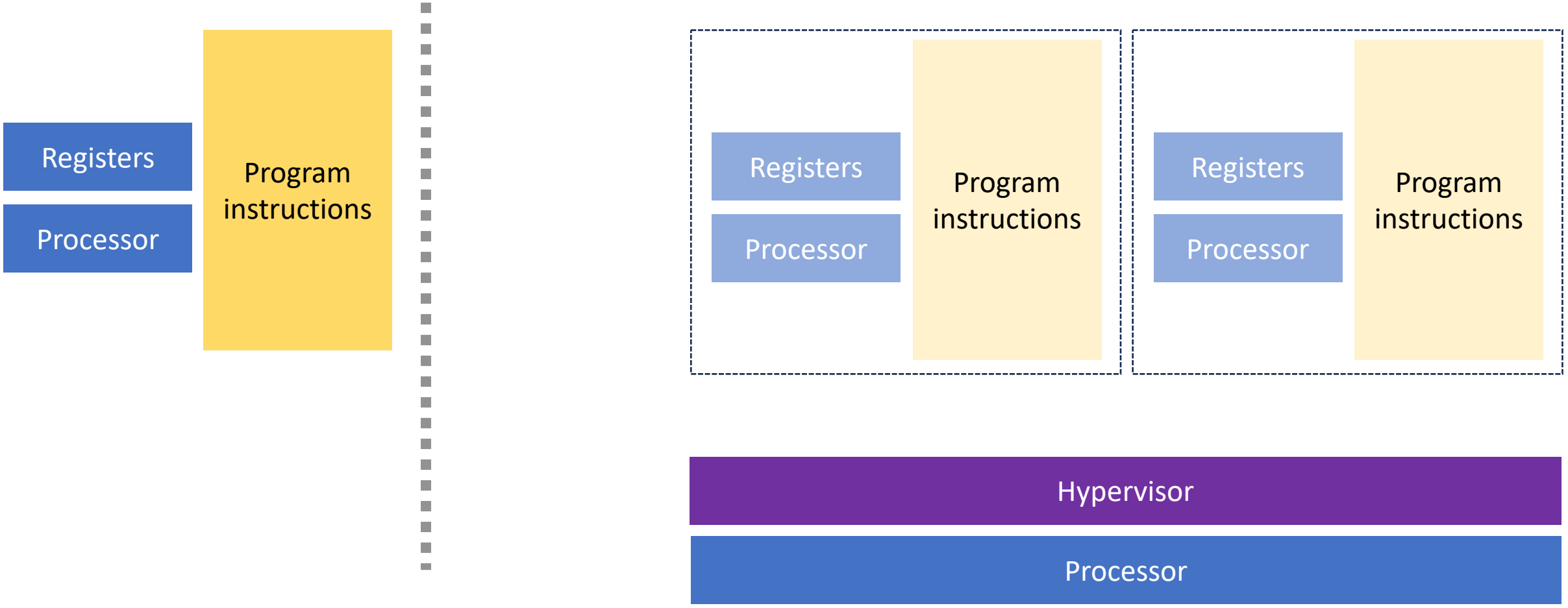
→ ***Standalone hypervisor based proof of concept***
(no constraints from ether SystemC or VCML)

Reality check: OS boot

- DUT: Solidrun Macchiatobin, quadcore 2Ghz Cortex-A72 Marvell
  - Emulate cores, devices (SD-card…) and chips (GIC, mailbox, thermal…)
- Test Case
  - Downloaded SD-CARD image (TFA, U-Boot, Linux) installed as SD-CARD backend
  - U-Boot has loaded Linux, ready to “booti” (after that almost noting to do)
- Measures
  - Roughly 50% faster on Apple M1 which is Arm core at 3.2GHz
  - 32K traps out of 105M instructions (0.03%) out of 400 “trap sites”
  - Infinite quantum boot: 11s (incl. timeouts for non present devices)
  - 1 instruction quantum boot (NOP “decoration”): 165s
- Expectations
  - 1,000 instructions quantum (leveraging PMU infrastructure): 12s

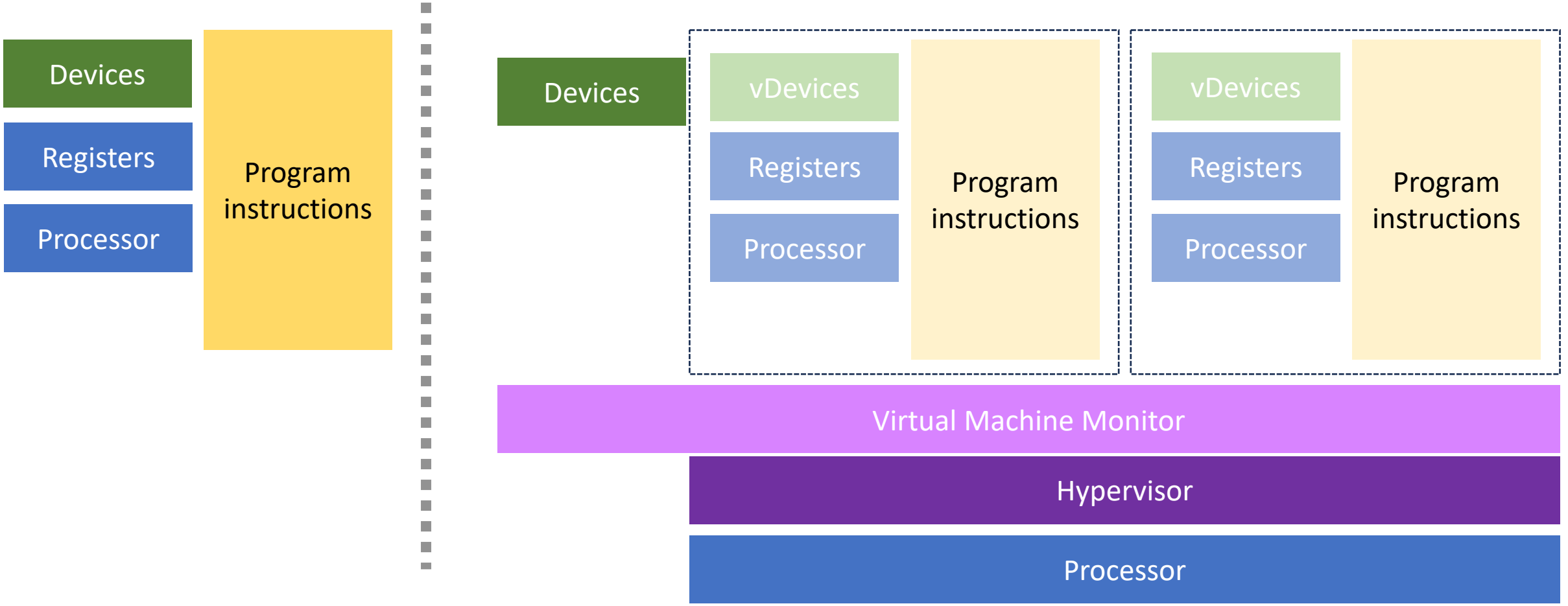
From Hypervisor…

(MacOS HyperVisor Framework)

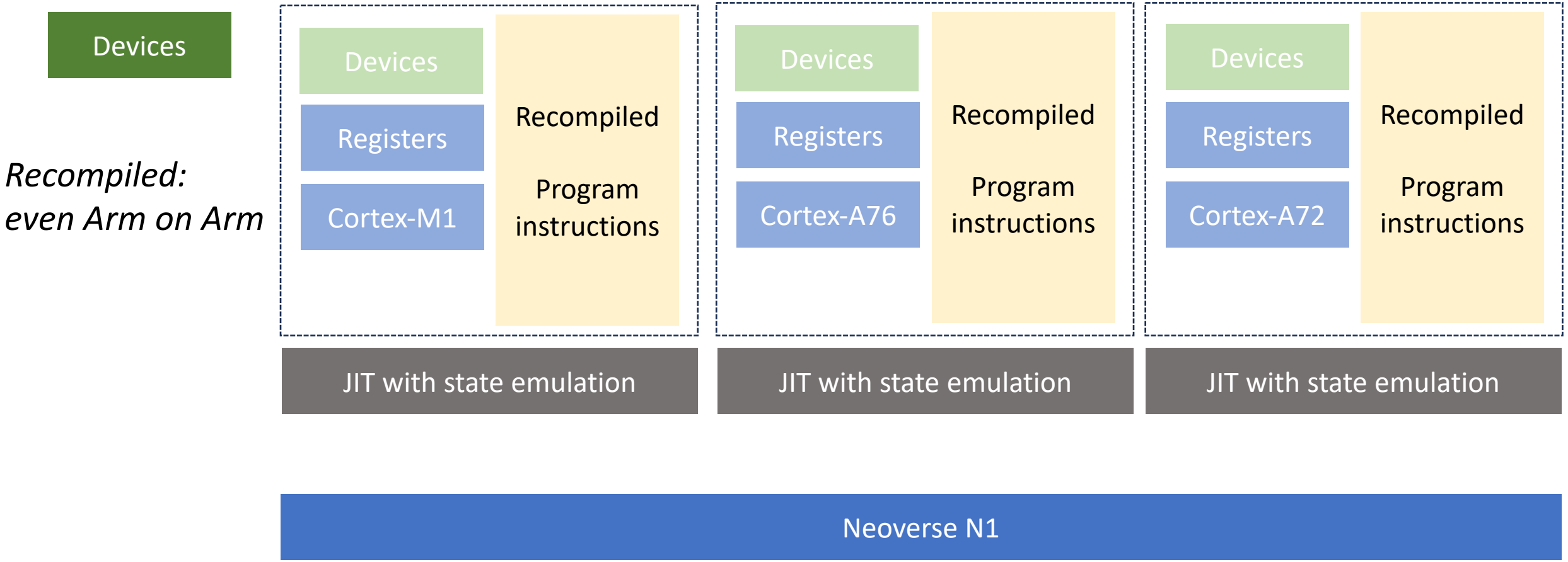


… to Virtualization …

(MacOS Virtualization Framework)



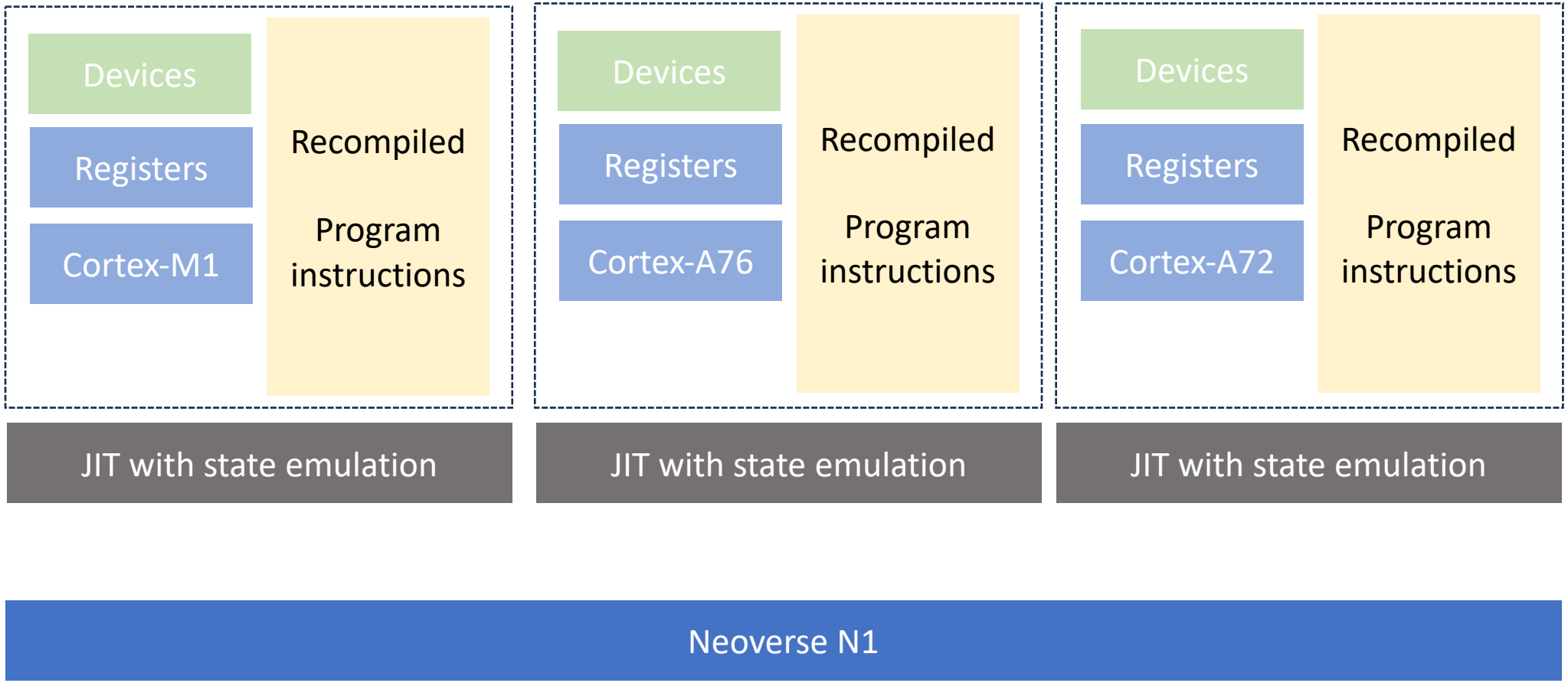
AVP64 VCML: Qemu with single threaded TCG



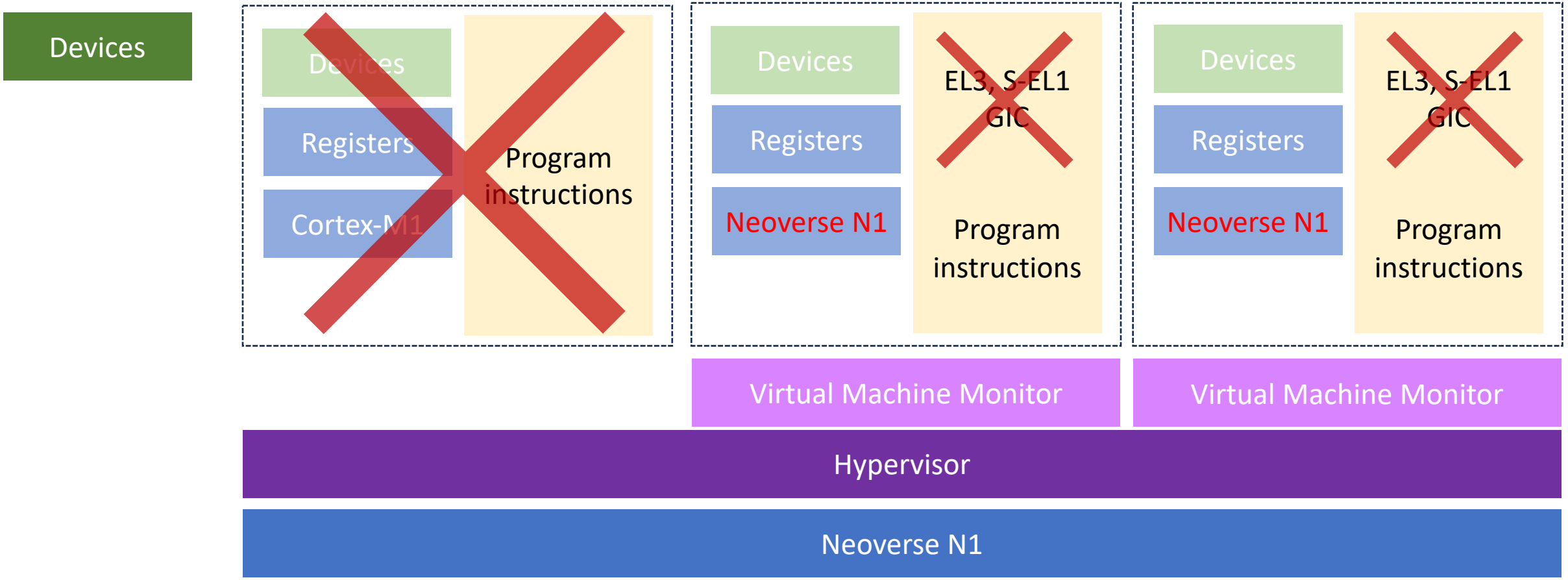
Qbox VCML: Qemu + multi-threaded TCG

Devices

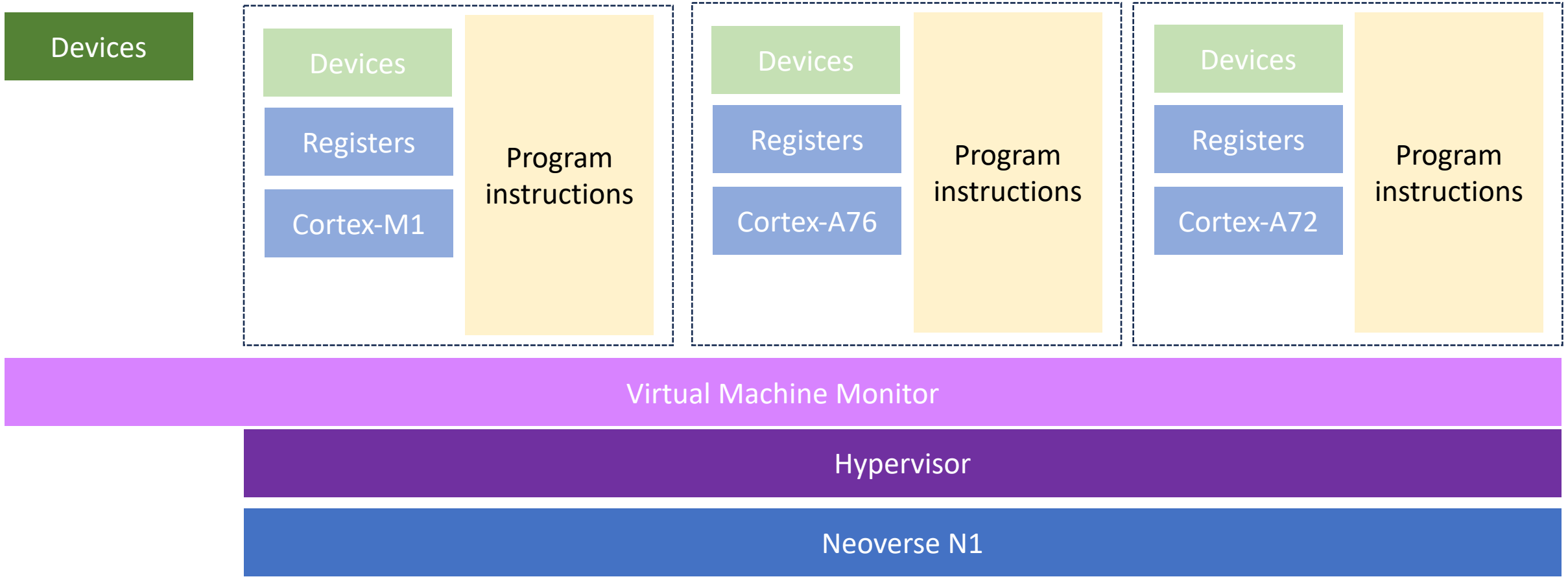
Recompiled:
even Arm on Arm



Qbox+KVM/HVF VCML: blazing fast but…



Emula4 VMCL: synthetic CPU on top of host



Apple to Apple benchmarks

- Emulate and compare to Qemu “virt” machine on same hardware
- Boot time check point: 8.0s for Qemu, 1.2s for Emula4
- Same 6GB/s virtio (Emula4 is still just a prototype…)
- Much better cpu-bound performance

	raid6: int64x8	Raid6: neonx4	xor: 32reg	xor: neon
Qemu	2GB/s	3GB/s	5GB/s	3GB/s
Emula4	14GB/s	36GB/s	44GB/s	54GB/s

Synthetic core capabilities

- Processor states
  - EL3 (Firmware), EL2 (Hypervisor), EL1 (OS), EL0 (apps)
  - Secure mode and non-secure mode: EL3, S-EL2, EL2, S-EL1, EL1, S-EL0, EL0
  - Secure memory (memory firewall)
- bigLITTLE with SCMI control of host cores
- Fast enough to validate 4K WideVine DRM
- MPAM isolation but not bandwidth measures/enforcement
- Arm v9 Realms on Arm v8 cores
- Custom instructions and/or registers
- Enhanced “standard” instructions

The journey to Emula4 “Interception” model

- Try 0: possible ?
  - EL3 instructions, CurrentEL, Exceptions
- Try 1: No patching: use debug registers
  - KVM: no go because of forced GIC and other constraints
  - HVF: provided the basis, ensure can build on other commercial hypervisors
  - Slow pre-analysis, complex metadata to execute on different hardware

The journey to Emula4 “Interception” model

- Try 2: hypervisor trap injection as debugger breakpoint injection
  - No stored image change
  - Smart traps injection as code uncompress/autorelocate itself/loads artifacts
  - Smart JITting to limit traps to VMM
  - …
- Next (not in any order, not complete)
  - Application patching through OS hooking from hypervisor
  - Mind experiment with rev.ng to recompile on the fly (SVE2 on Armv8)
  - LT or AT behavior when executing complex emulation
  - SystemC VCML packaging

Hypervisor wish list

- PMU for quantum implementation
- SCMI control of cores to ensure “exact” performance
- Fine grained TLB maintenance
- Fine grained trap control (high jacking HVF to get access to private control for internal assessment)
- Address spaces per vcpu (Secure Memory, Realms)
- Lightweight sync (IRQ injection, WFI, WFE)
- Coresight traces from VM
- Nested virtualization

Machine assembly

???IP-XACT??? (silicon provider)

```
-vobj "RAM#address=0x4000000||hostmem#size=4"
-vobj "SECRAM#address=0x4400000||hostmem#size=12"
-vobj "RAM#address=0x05000000||hostmem#size=2048"
-vobj "AP806@MARVELL#address=0xf0000000"
-vobj "CP110@MARVELL#address=0xf2000000"
   -vobj "GIC@AP806@MARVELL#name=main_gic;root=true"
-vobj "CP110@MARVELL#address=0xf4000000"
   -vobj "PL011#uartclk=main_clock;apb_pclk=main_clock;irq=spi:1@main_gic…"
   -vobj "PL011#uartclk=main_clock;apb_pclk=main_clock;irq=spi:2@main_gic…"
```

Device Tree(S)

Comments & questions

Please feel free to just state what you think

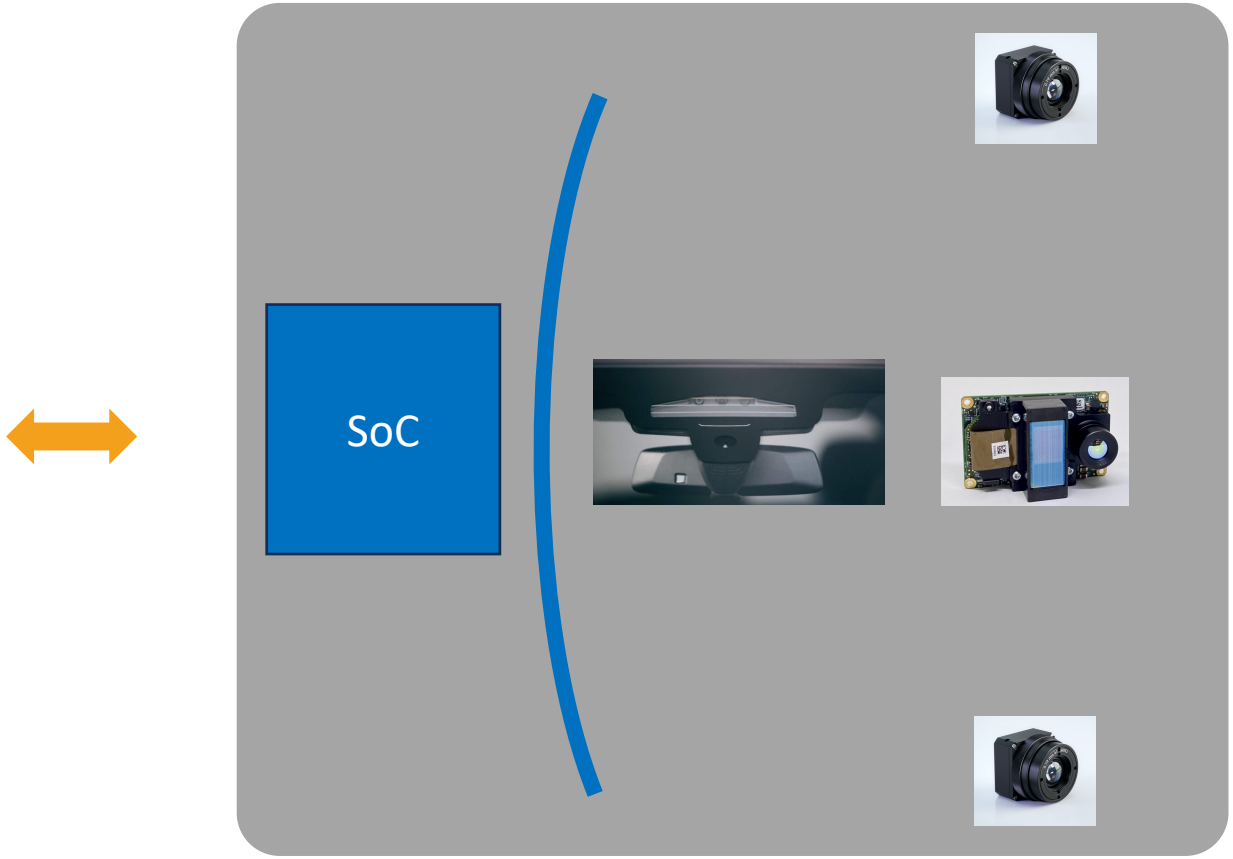
Questions are welcome too

Solidrun Macchiatobin, a Marvell 8040 board

- HW simulation
  - Chips
    - Complete/partial: A cores, Marvell Cache Coherency Unit, Marvell GIC…
    - “Responders”: Memory controller, temperature sensors, mailbox, SERDES, SCP…
  - Devices (SD-Card, eMMC, UARTs…)
- Boot
  - Binary bootable image from Solidrun website
  - Boot ROM and processor substitute: place image at the right place
- Software
  - Secure software running at EL3 and S-EL1
  - 3 times faster than real hardware on Apple M1

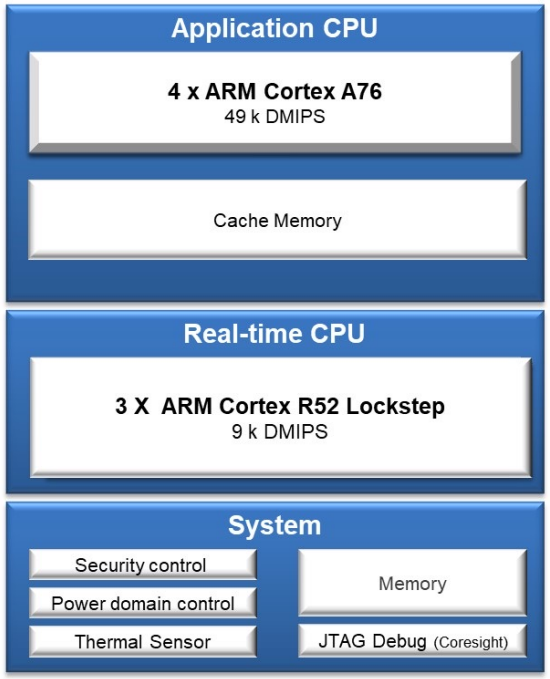
SDV ”sensor fusion” driving use case

Rest of SDV (ADAS, heads up display, storage…)



Stimuli: Replay or Digital worlds

Deploying simulation



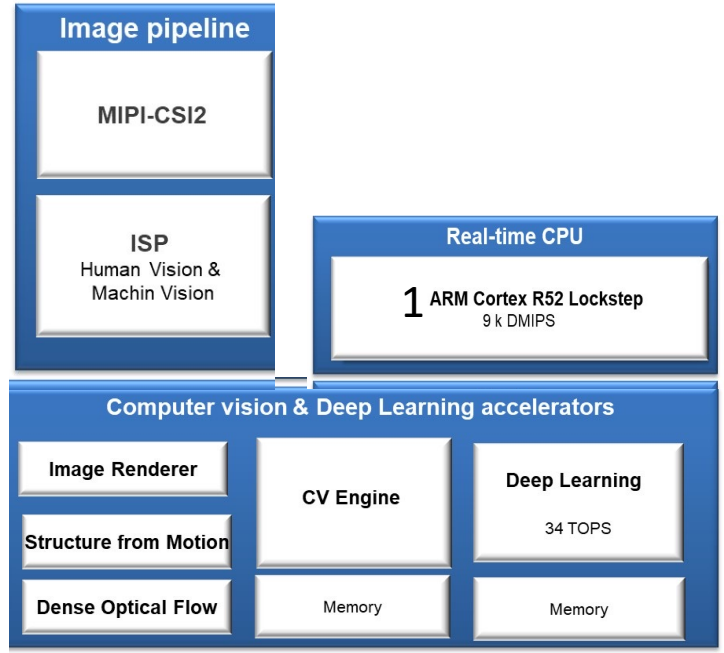
Rest of SDV (ADAS, heads up display, storage…)

4K camera

IR HD camera

LIDAR 2.0

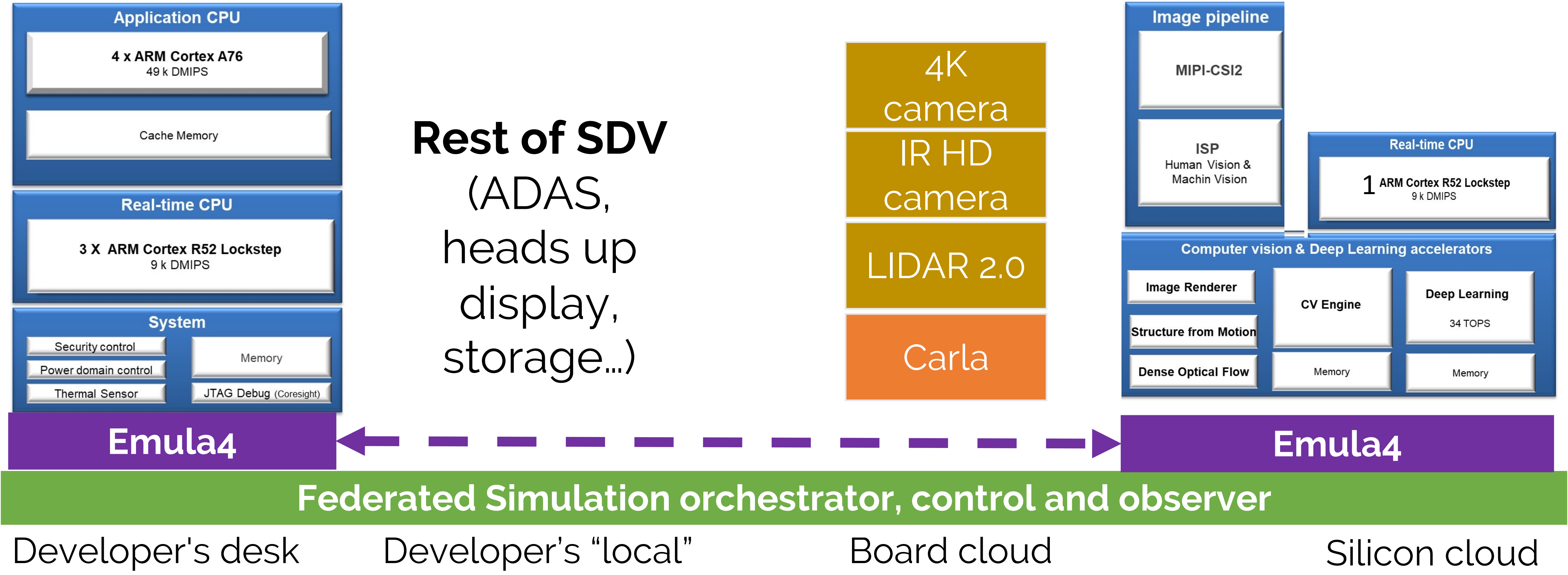
Carla



Federated Simulation orchestrator, control and observer

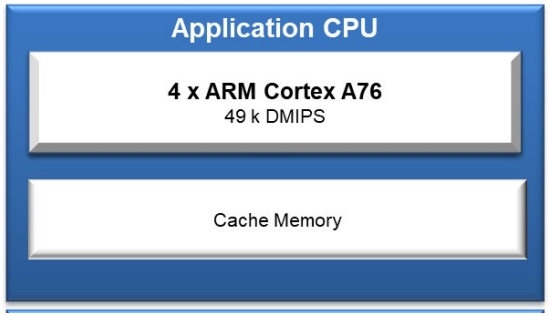
Developer's desk | Developer's "local" | Board cloud | Silicon cloud

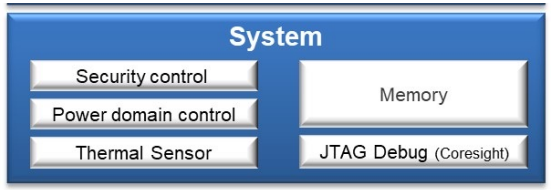
Emula4 cluster , FSS layers

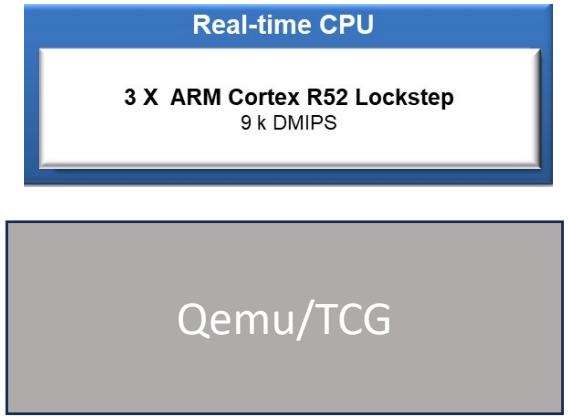


Emula4 focuses on A-cores & context

Qemu/TCG only used for instruction simulation
Emula4: Memory backends, chips (GIC..), devices







Emula4

Federated Simulation orchestrator, control and observer

Developer's desk