QEMU AND SOC SIMULATION

FRANÇOIS-FRÉDÉRIC OZOG

Shokubai

A non-exclusive, irrevocable, royalty-free copyright permission is granted by Shokubai.tech to use this material in developing all future revisions and editions of the resulting draft and approved Accellera Systems Initiative SystemC standard, and in derivative works based on the standard.

QEMU: USE CASES

- Develop for Arm 32 bits devices on Intel X86
- Use Windows x86 on Apple Mac Pro M1
- Cloud PCs
- Simulation

QEMU SOFTWARE PROJECT ASSESSMENT

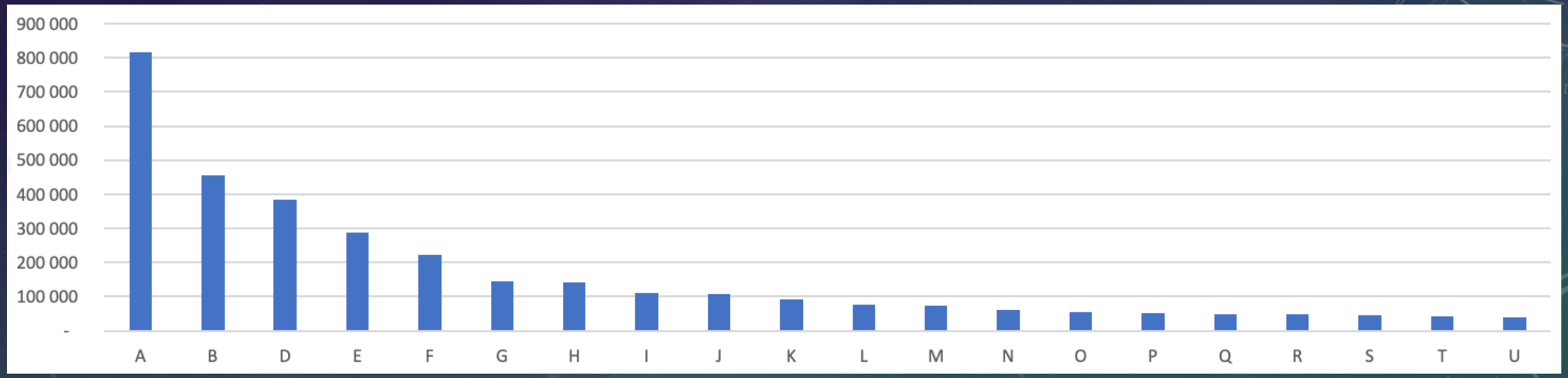
- Total Physical Source Lines of Code (SLOC) = **2,287,077**
  - Aarch64 80K, Intel 50K, IBM S390 20K, RiscV 20K
  - Devices 600K
  - Infrastructure 200K, test 200K
- Development Effort Estimate, Person-Years = 673.40
  - Reality: over 1,300 contributors
- Schedule Estimate, Years = 6.36
- Total Estimated Cost to Develop = **$ 90,967,540**
  average salary = $56,286/year, overhead = 2.40

generated using David A. Wheeler's 'SLOCCount'

Shokubai

QEMU PROJECT FACTS SHEET

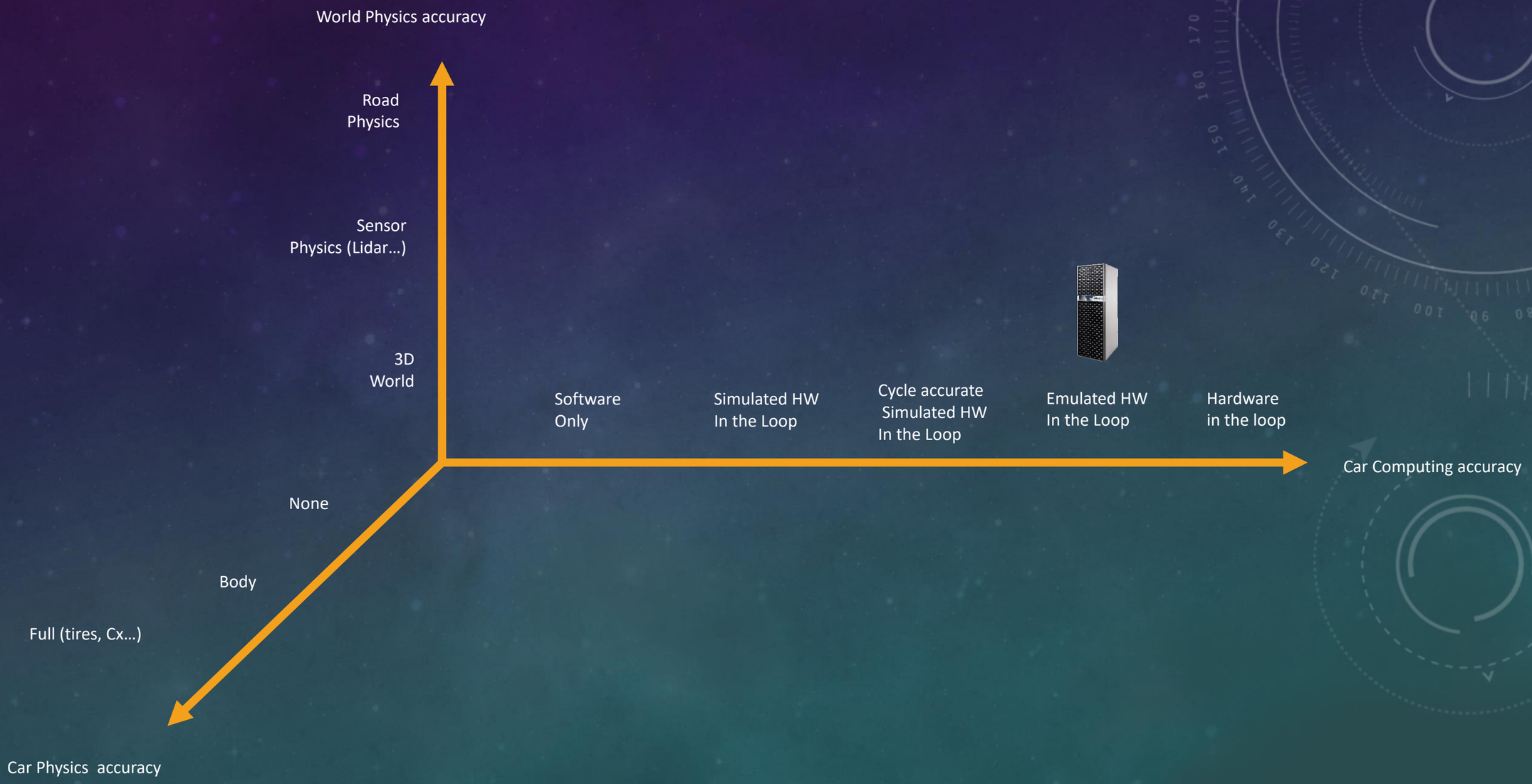
- Over 1,300 contributors
- Top 20: each added above 20K lines of code



OPEN SOURCE AND GOVERNANCE

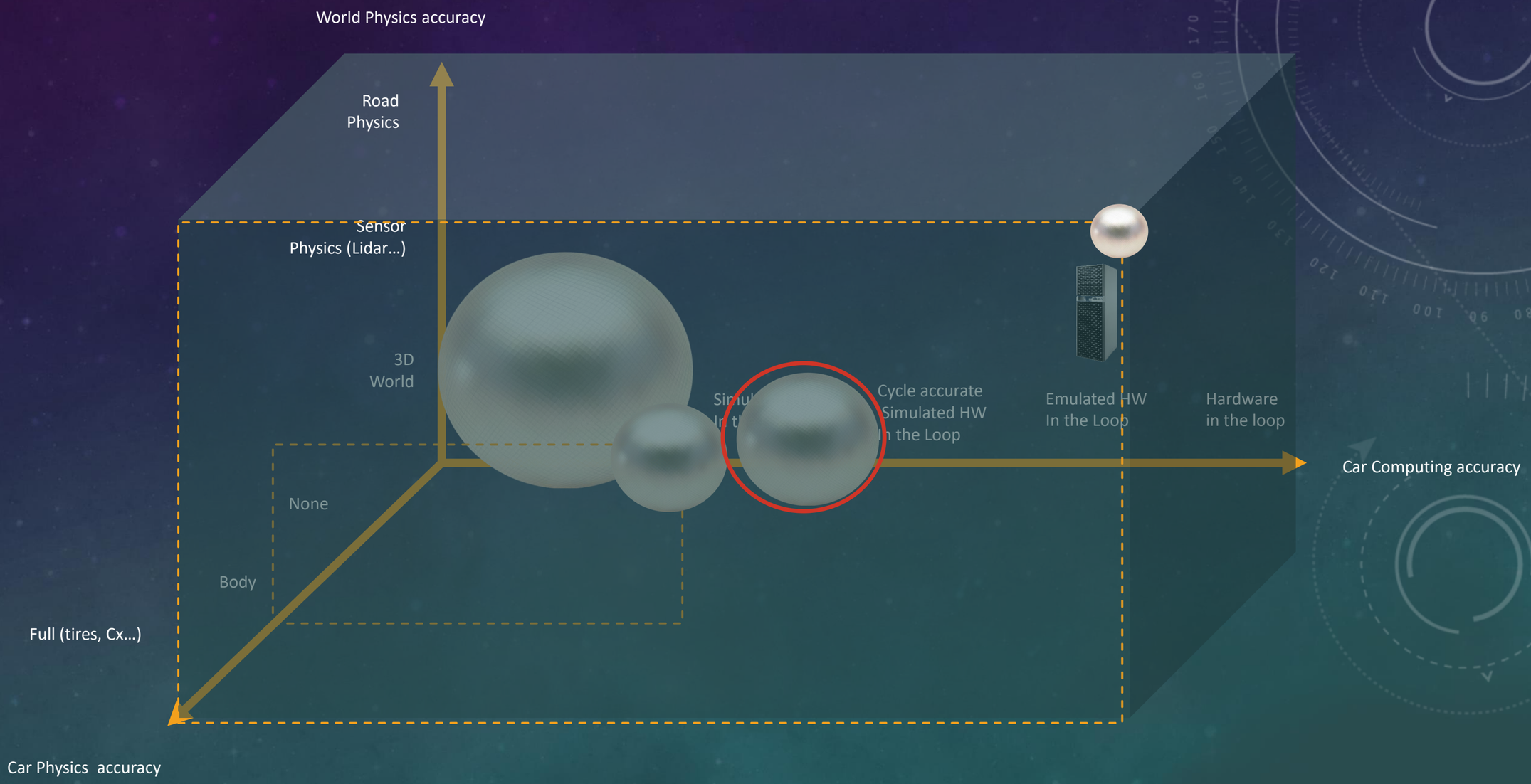
- Open Source closed governance
  - FreeRTOS, Microsoft Azure RTOS
- Strategic governance
  - SystemC
  - DPDK
  - OpenAMP
- Industry initiatives
  - SOAFEE
    - OEM and Tier1s, Hypervisor vendors, SoC vendors to address hypervisor portability, starting with device assignment
- Qemu/KVM
  - Evolving towards more “strategic” (latest kvm forum interactions)

VEHICLE VALIDATION THROUGH SIMULATION



Shokubai

VEHICLE VALIDATION THROUGH SIMULATION



Size of sphere - number of tests that can be realistically operated

Shokubai

SYSTEM UNITESTING VS SIMULATION

- System Unitesting
  - Existence of a simulation framework
  - Abstract time(can be controlled by simulator framework)
  - Runs most of the stack but some shim layers to deal with simulator framework
    - For instance time control
    - Supplemental testing code can be large
- Simulation
  - No simulation framework per say (digital car twin in digital world)
  - Runs the exact stack
  - Physical time (meaning no enforced timing 1s = 1s; not meaning time constraints)

SIMULATION GOALS

- Goals
  - what is the right SoC parameters (# performance cores, # low power cores, accelerators…)
  - Hybrid Zonal architecture analysis
  - See how the system behaves
- Ideas based on DVCon
  - “Elaboration” phase at a higher level (LIDAR pieces in the digital world)

VECU LEVEL 4 SIMULATION TECHNOLOGIES

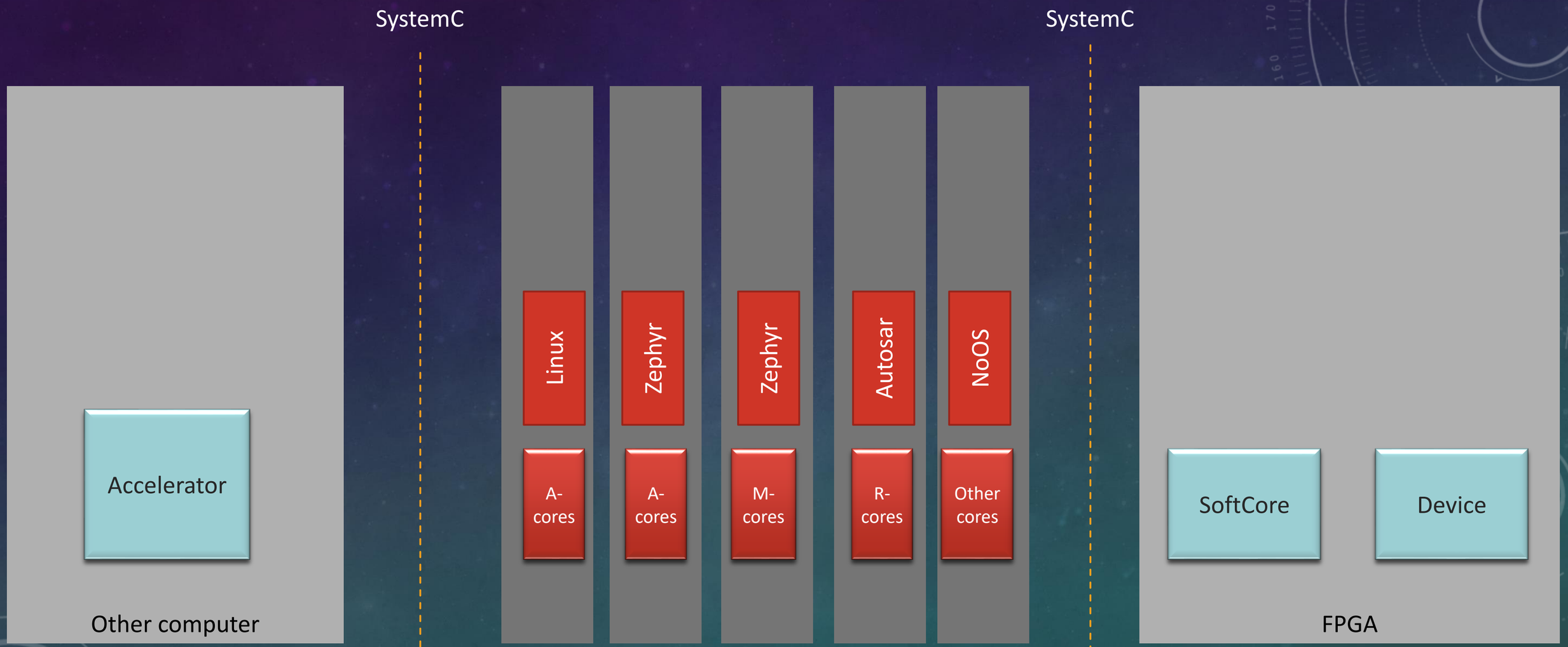
- Arm Fast Models, 100% accurate but really slow!
- Arm Virtual Hardware, in the cloud, derived from Corellium
  - Expects to have Arm AI accelerators integrated
  - Unclear how to simulate accelerators, heterogeneous platforms
- Corellium, high performance, centered on mobile devices emulation
- Qemu with SystemC/TLM, broad range of capabilities
- Siemens Veloce Hycon
  - Can leverage Qemu with SystemC TLM, Arm Fast Models
- …

For automotive, need interfaces with digital worlds such as CARLA

SIMULATION: HYBRIDATION QEMU+PROCESSOR SUPPORT

	Processor virt	Enhanced Processor Virt	Qemu
Processor architecture	Limited to CPU	Limited to CPU	Emulate any architecture
Processor generation/feature (MPAM for instance)	Limited to CPU	Emulate any feature (can be high cost)	Emulate any feature (can be high cost)
Devices	Flexible	Flexible	Flexible
Performance	High	High	Low to High
Heterogeneous simulation (multiple domains, processor types)	No	Architecturally built-in Qemu, SystemC…	Possible (Xilinx proprietary)

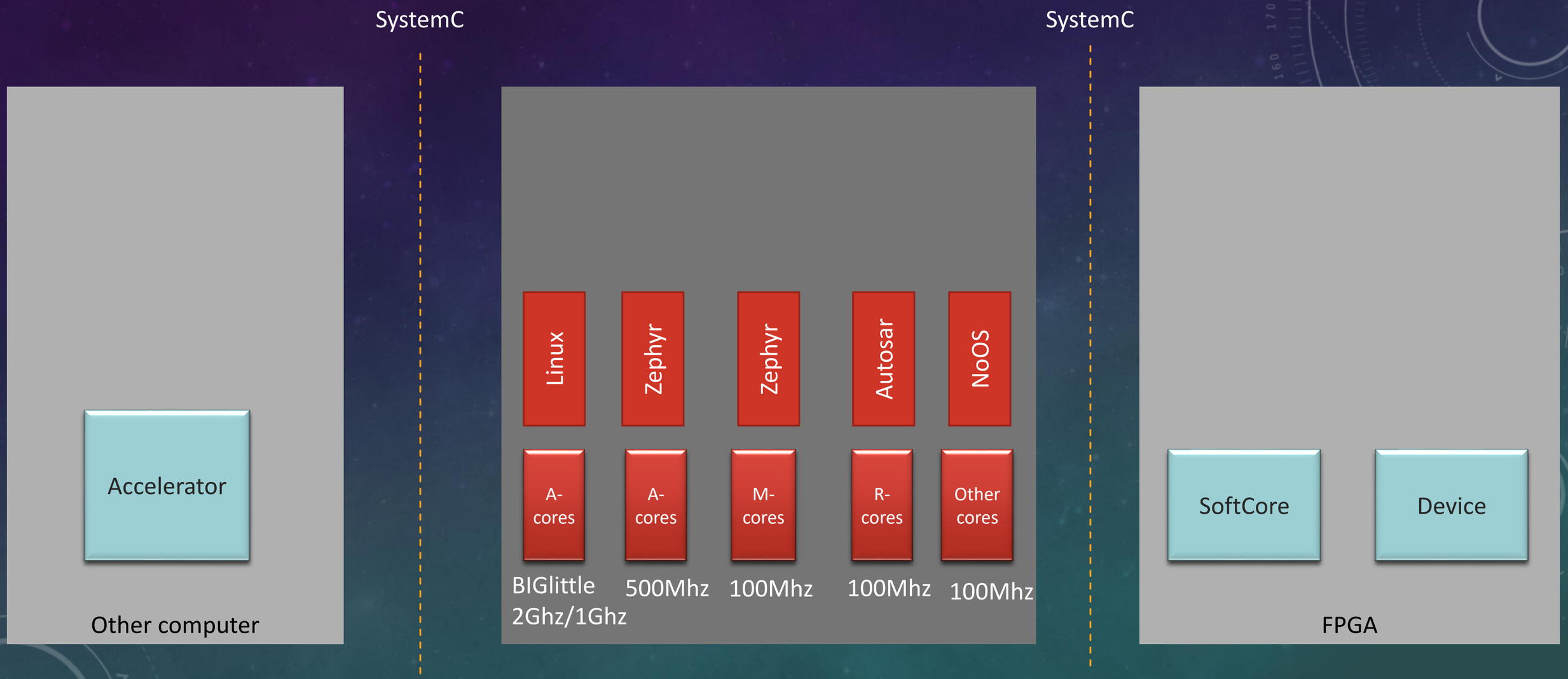
QEMU + SYSTEMC FOR SIMULATION



Concretely, not possible to empulate processor feature AND speed for Cortex-A side
Not out of the box (Linaro Heterogenous Platform Project may address that in the future)

Shokubai

ACCELERATED HETEROGENEOUS QEMU + SYSTEMC



EXPLORATIONS

- Linux KVM implements too much in-kernel, not exposing enough control
  - Nested virtualization on-hold
  - Lacks fine grained VMM control of instructions
- MacOS HVF, implements very little, not exposing enough control
  (based on VMM implementation (7.4Ksloc) capable of booting single processor Linux)
  - Lacks fine grained VMM control of instructions
  - Unknown nested virtualization support
  - Impossible to change in-kernel behavior support
  - 1300 lines of code to implement a GICv3,v4
    - but cannot support direct IRQ injection in VM as requires in-kernel support
  - Lacks control of SMMU and MMU

NEXT STEPS

- Implement HVF like API on top of KVM on Arm (no x86 work)
  - KVM in kernel extensions
    - fine grained “exits”
    - BIGlittle phase 1 (A72 2Ghz, A72 1GHz)
    - S-EL1 emulation
  - KVM “raw mode” (no upstream work)
    - BIGlittle phase 2: CPU (A72, A57)
    - S-EL1 in userland
    - S-EL3
  - KVM user enhancements phase 2
    - Heterogeneous phase 2 CPU (A72, R5, M7)
    - KVM enhancements: KVM enhancements: secure world support
    - KVM enhancements: partial MPAM feature implementation
    - KVM enhancement: simulated GICv4 on a physical GICv3 (no legacy)
- Planning
  - Around three months of workload over the next 18 months

THANK YOU

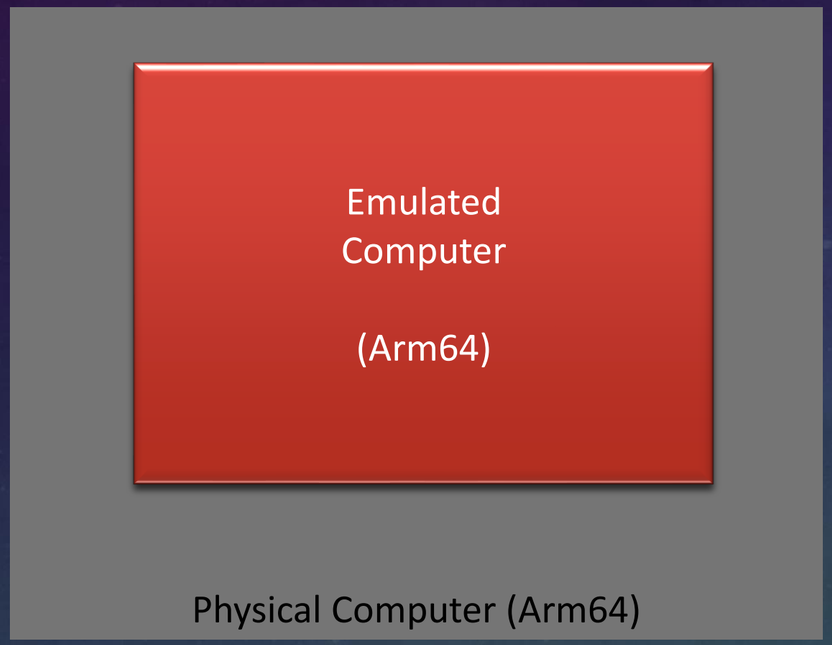
WWW.SHOKUBAI.TECH

Shokubai

QEMU: EMULATION

- Processing
  - Processor model: just-In-Time cross-compiler (TCG)
    - Speed is simulated by software with huge perf tax
  - Emulated MMU, IOMMU, interrupt controller
  - Emulated modes: Arm secure mode, Intel SMM…
- Devices
  - Emulated devices (x86 disk controller, PL011 UART, TPM…)
  - Para virtualized devices (Virtio)
- “Context”
  - Firmware (normal, secure, others)

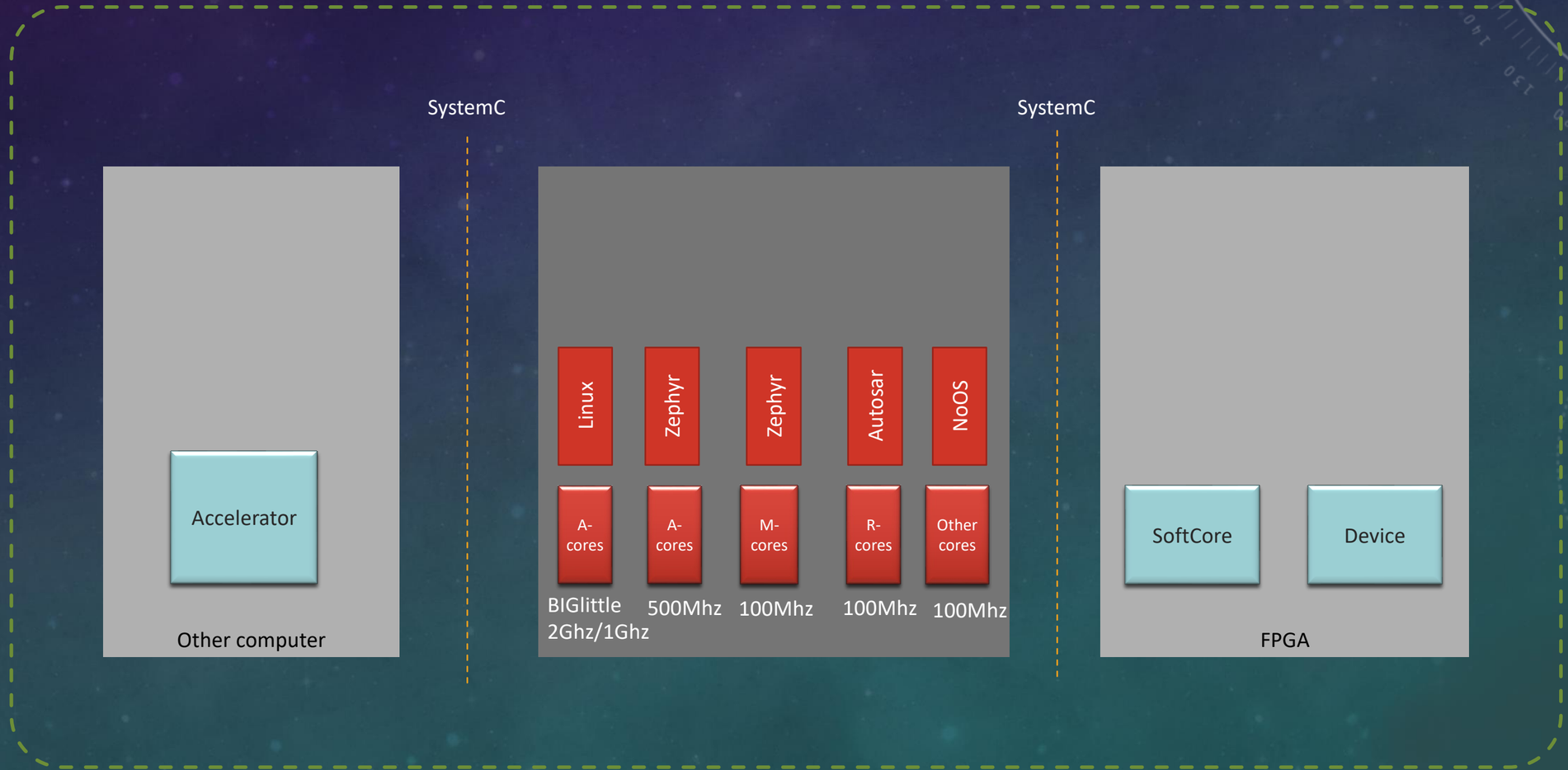
QEMU ACCELERATION BUILDING ON VIRTUALIZATION



- Processing
  - Methods: KVM on Linux, HVF on MacOS…
  - Processor model is the same as host
    - Speed can be hardware controlled with SCMI at no tax
  - Accelerated MMU, IOMMU, interrupt controller
  - Just normal processing mode
- “Context”
  - Firmware (normal only)

DIGITAL”THING” TWIN IN DIGITAL “WORLD” TWIN

CARLA, Nvidia Omniverse…



Shokubai