Instruction Accurate FW Simulation on QEMU

Presenter: Bruce Cheng, Beijing Starblaze Technology

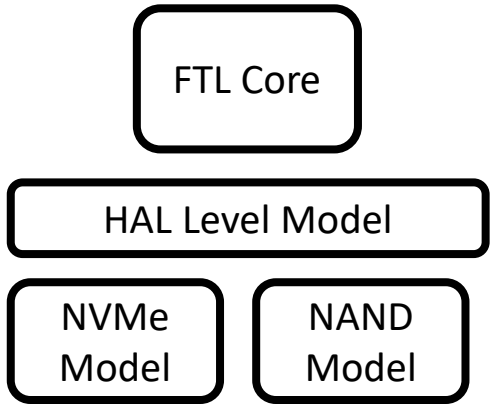
Firmware Development Challenge

- SSD Controller chip is hardware + firmware
  - Firmware determines the major features of SSD Controller.
  - To get best performance and power consumption, firmware needs to be fine tuned on well-optimized hardware.

- Both hardware and firmware are customized
  - Most hardware components are designed from scratch and need to be carefully optimized according to firmware usage.
  - SSD firmware is very customized and optimized to fit the hardware.

SSD Controller chip design needs very close firmware and hardware co-design!
Simulation technology is the key point!!!

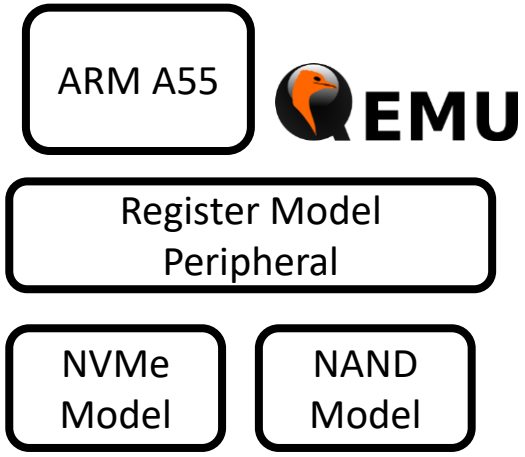
Various Level of Simulation



FTL Simulation

Emulation

FPGA

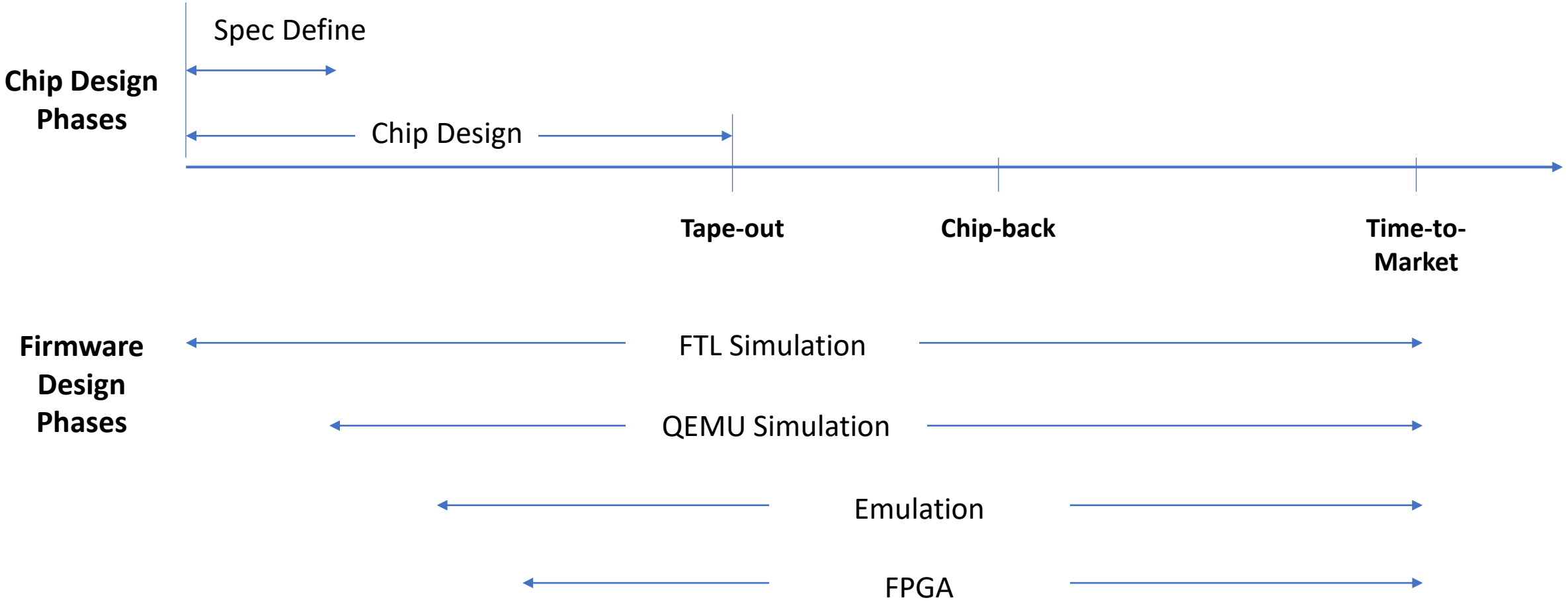


QEMU Simulation

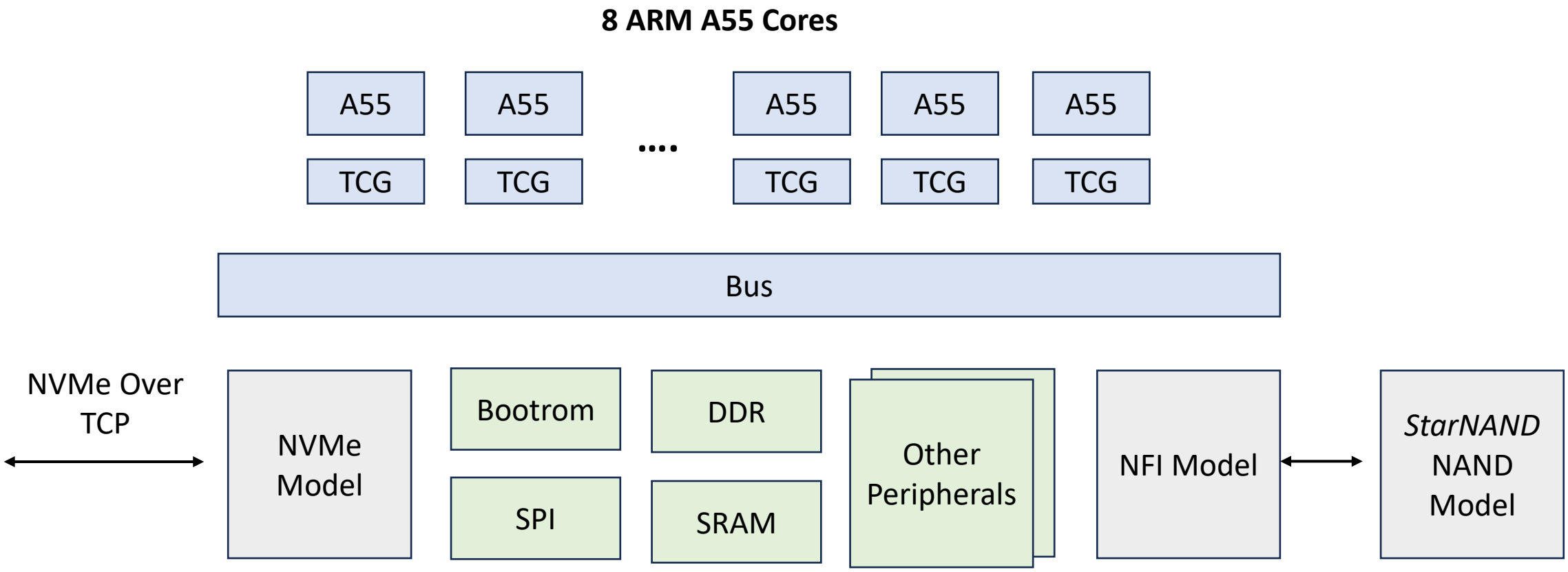
Simulation Pro and Cons

`	Pros	Cons
FTL Simulation	FAST simulation by natively compile code in x86, fast bring-up and SoC independently	Only give hal API level accuracy, good for FTL algorithm and architecture simulation
Emulation	Nearly 100% accurate simulation using RTL code on emulator and accurate interface models. Good for tape-out performance/function simulation	Very slow simulation speed, high effort for bring-up. Late ready
FPGA	Good accuracy using most logical part RTL code and FPGA substitution for interface. Good simulation speed. Good for function verification	Verify high effort bring-up, interface not accurate.
QEMU Simulation	Fast simulation and fast bring-up. Good accurate simulation using register level models. Run native SSD firmware binary directly. **A good balance between FTL simulation and Emulation/FPGA**	Accuracy and performance between host simulation and emulation.

Firmware Simulation Phases



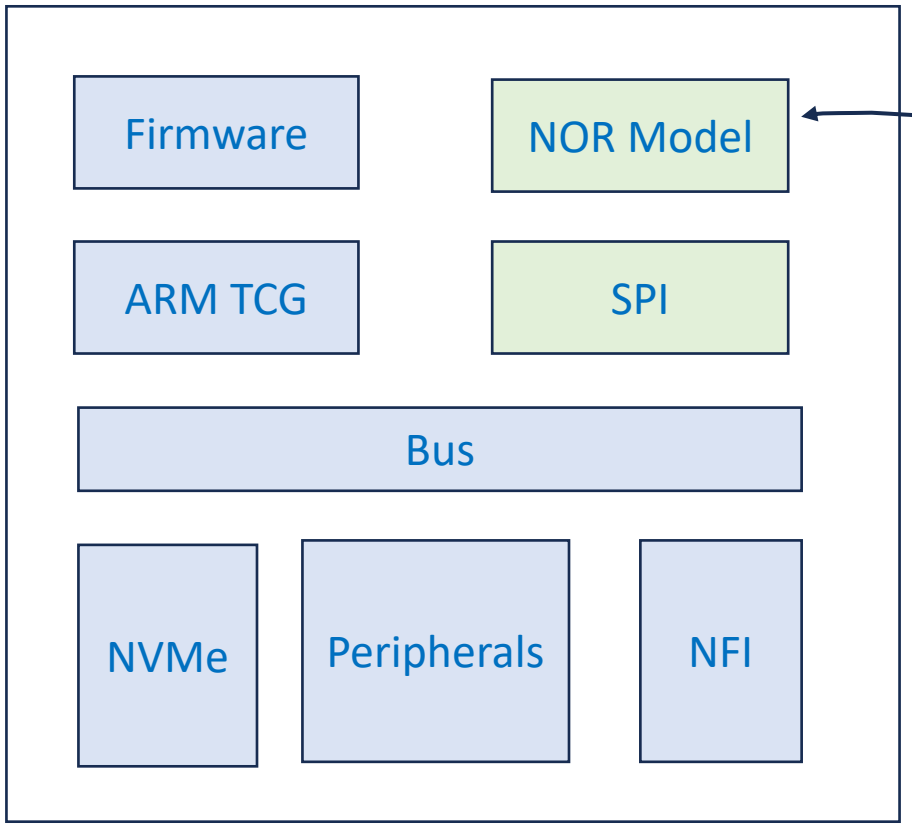
QEMU Simulator Architecture



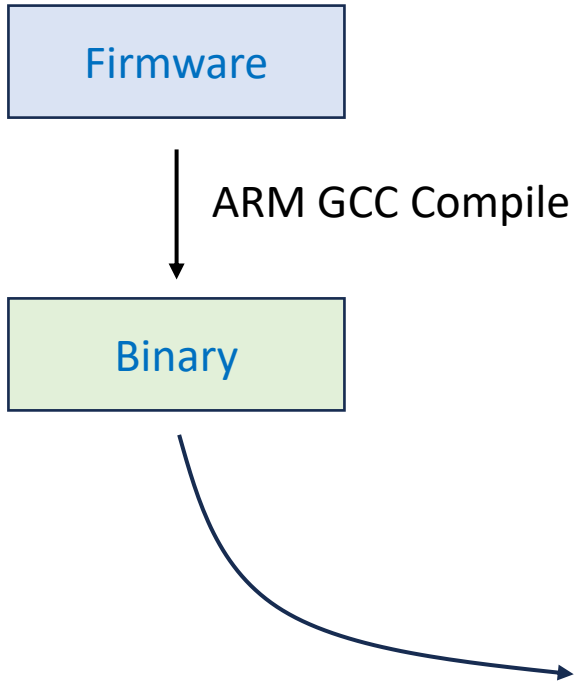
QEMU Customization

1. QEMU code reduction:
    1. Delete all other processor type code
    2. Delete all other un-used peripherals, e.g. GPU, NIC, Sound Card and etc
    3. Fix some ARM A55 code simulation bug (QEMU natively support A53)
2. Design our Soc Peripherals: DMA, UART, SPI, Bootrom and etc
3. Design NVMe Model and virtual host interface by using NVMe Over TCP protocol
4. Design NFI Model and connect to our *StarNAND* NAND simulation model (shared NAND model for QEMU, FTL Simulation and Emulation)

Firmware Code Binary Share



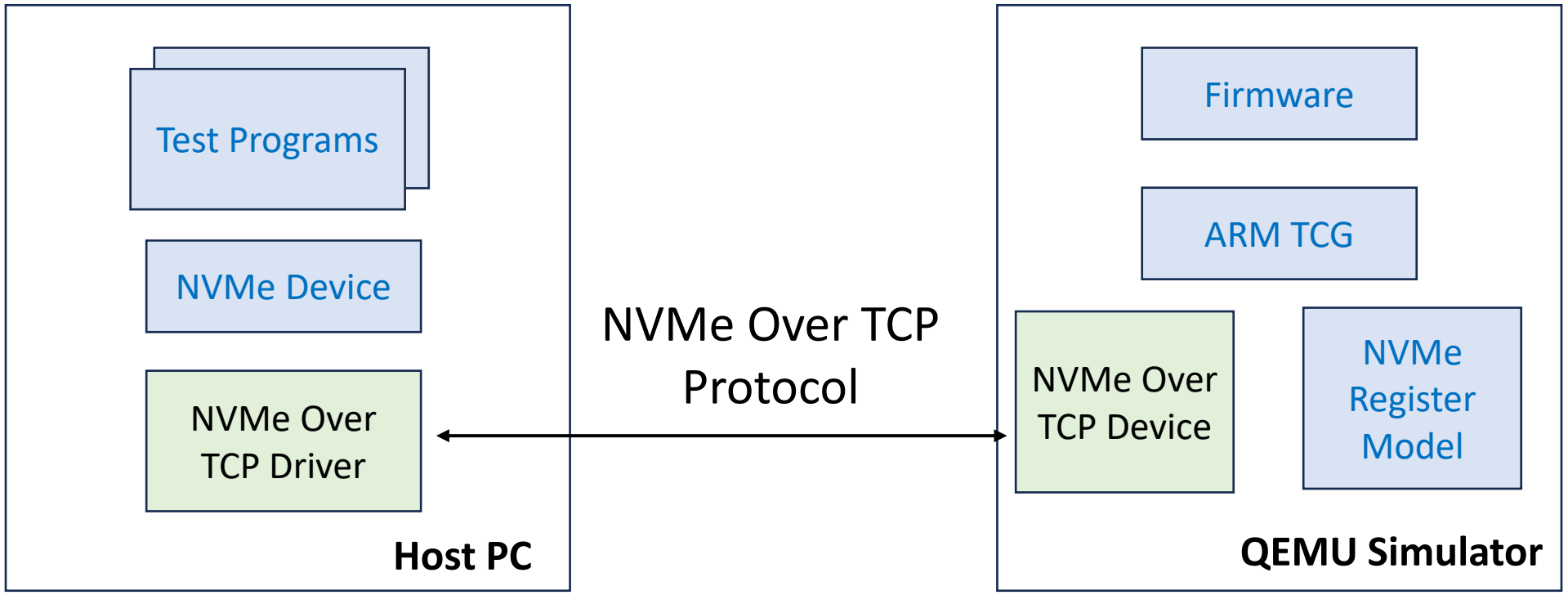
QEMU Simulator



QEMU Simulation and SSD Product use exactly the same Firmware Binary code, **this makes code related feature simulation possible, such as online firmware update, complete bootrom->spi boot flow and etc**

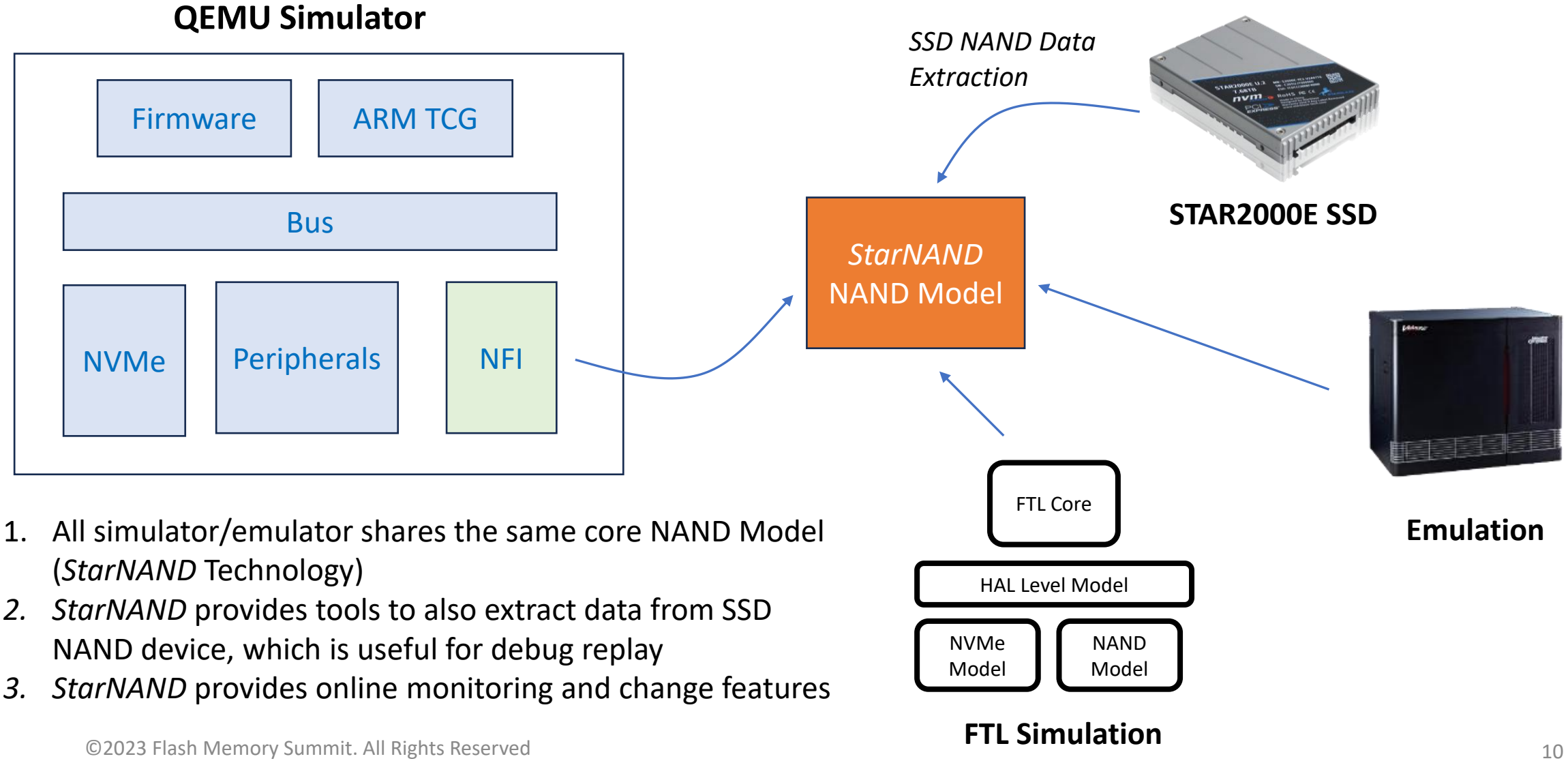
STAR2000E SSD

NVMe Frontend Simulation



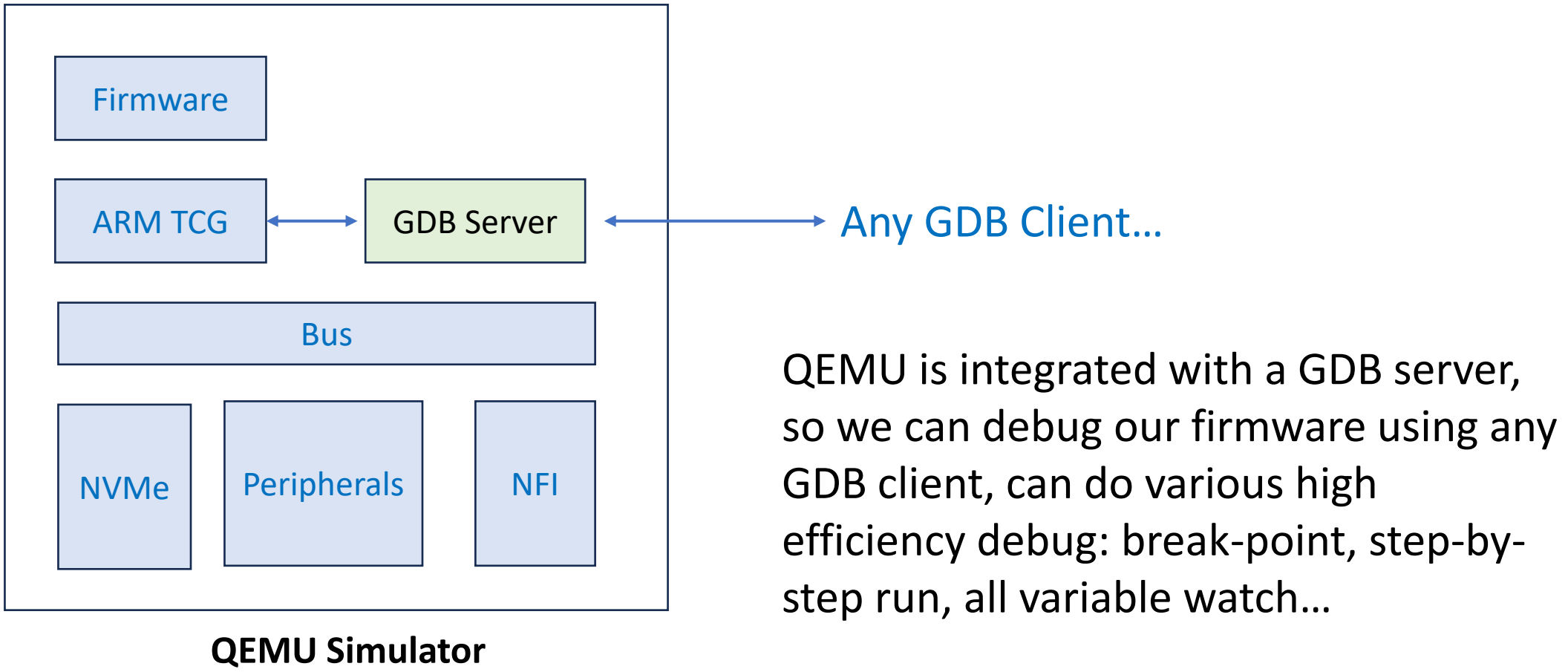
Our QEMU implement and expose our simulator as a NVMe Over TCP device, so we can operate our simulator as a standard NVMe device, which is good for both I/O and NVMe protocol test. **This makes possible that we re-use our SSD test programs directly**

Shared NAND Model



1. All simulator/emulator shares the same core NAND Model (*StarNAND* Technology)
2. *StarNAND* provides tools to also extract data from SSD NAND device, which is useful for debug replay
3. *StarNAND* provides online monitoring and change features

Debug with GDB



QEMU is integrated with a GDB server, so we can debug our firmware using any GDB client, can do various high efficiency debug: break-point, step-by-step run, all variable watch…

Starblaze Software-driven Design Flow