Getting your Open Source software ready for 0-day SoC bringup - *success stories & strategies*

Bhupesh Sharma

@bhupesh_sharma

$ whoami

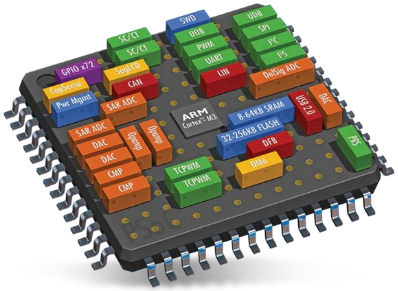
- Currently @ *ARM*
- Been hacking on boot loaders & kernel since past 18 years.
- Contribute to:
  - Linux,
  - EFI/u-boot bootloader & Secure FW
- User-space utilities like:
  - kexec-tools, and
  - Makedumpfile.
- Co-maintainer hat(s):
  - U-boot UFS, Qcom Ethernet & crash-utility tool.
- Area of Interest: *0-day Silicon bringup* using open-source software.

Outline

- System On Chip *(SoC)*
- 0-day Silicon Bringup *(what)*
- 0-day Silicon Bringup *(why)*
- Getting SW Ready for Silicon *(Before Silicon)*
- Complex SoC Boot Flows *(getting complex'er)*
- De-Risking OS development
- Open Source to the rescue *(how)*
- Optimizing Boot flow via Open Source
- Suggested Strategies/ Challenges
- Next Steps

- ***System on Chip.***

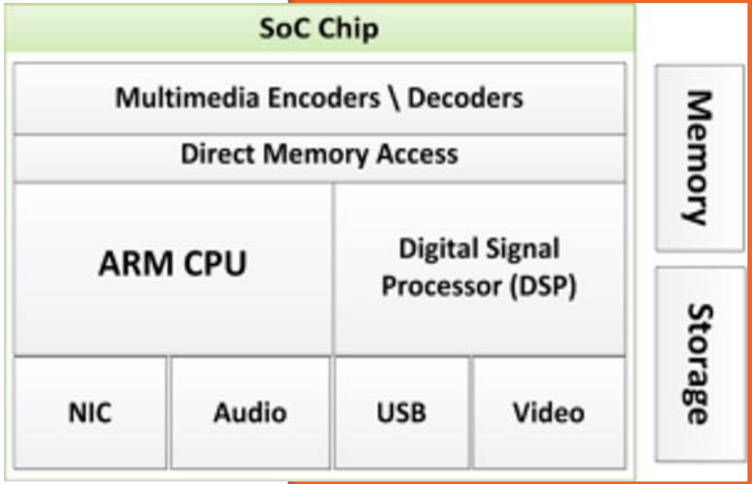
- ***0-day Silicon Bringup.***



SoC

System on Chip (SoC)

- ***Integrated circuit*** - contains all of system components in single piece of Silicon.
- ***multiple advantages*** - including cost and lower power consumption.
- ***multiple-iterations*** are required before final delivery to customer(s).

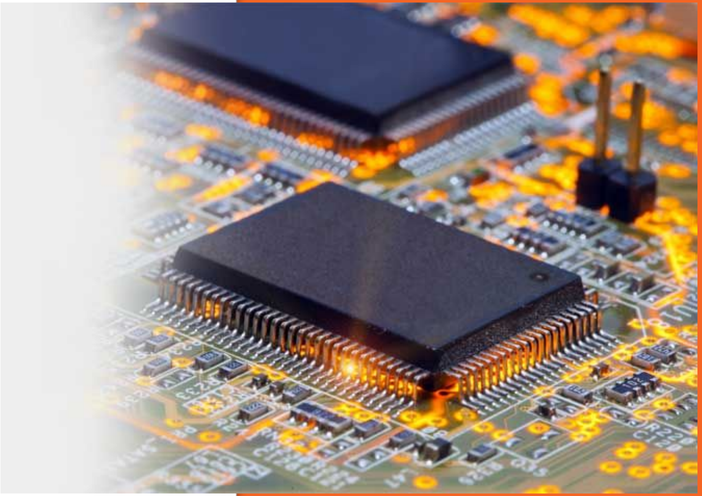


0-day Silicon Bringup (*what*)

- Cost of taping out a System on Chip (SoC) ~$ **2** to **3** million.

- ***Aim*** - timely tape-out with 0 Silicon Bugs.

- ***Reality*** - most chips, require multiple-iterations before final delivery to customer(s).



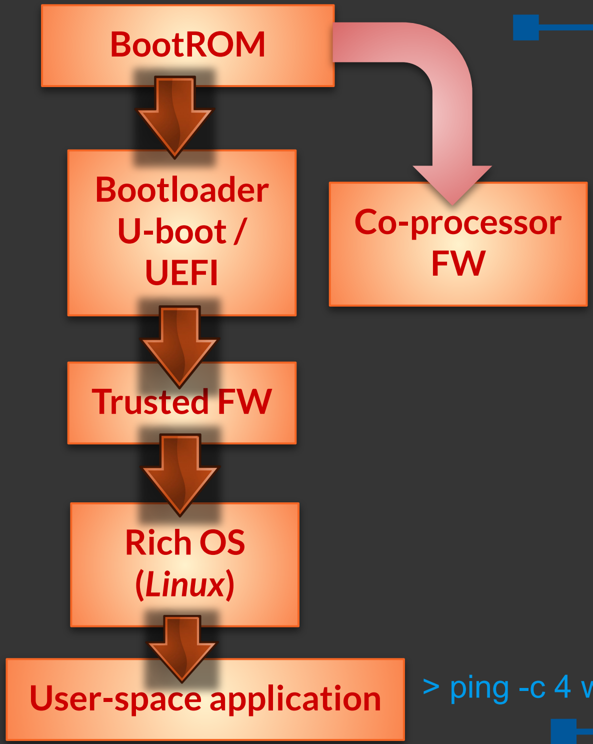
0-day Silicon Bringup (*why*)

Boot SW not ready for 0-day bringup.

▲ Probability of program failure

▲ Cost of program

0-day Silicon Bringup *(example)*



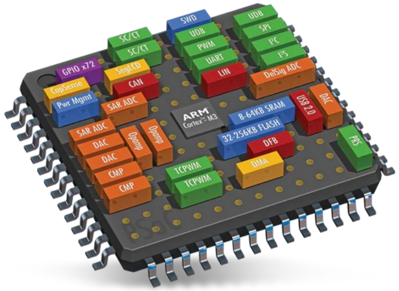
<u>0-day</u>: Silicon Arrives..

> ping -c 4 www.google.com

<u>0-day</u>: Ping Demo..

- ***Getting SW ready.***

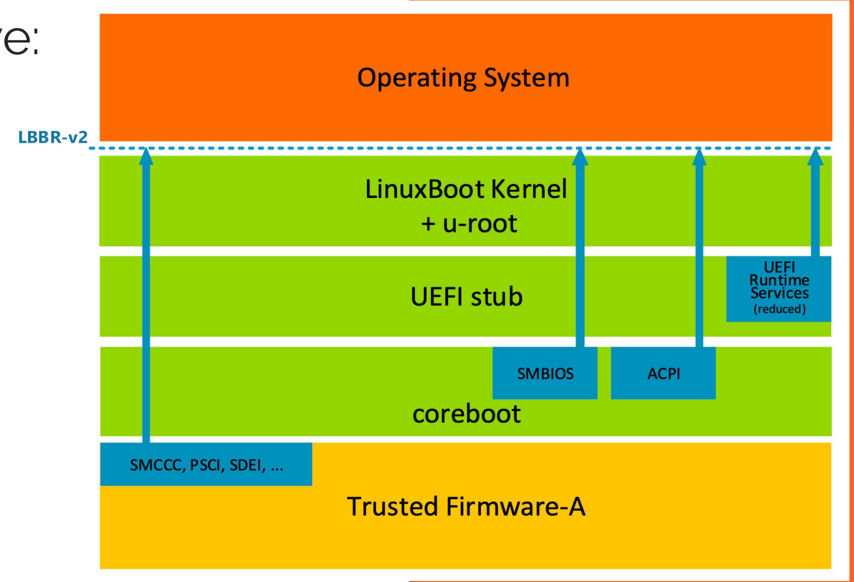
- ***Software choices for Complex SoC Boot Flows.***



Software Readiness for SoC

Getting SW ready for Silicon *(before silicon)*

- Silicon bringup strategies, involve:
  - Booting Source,
  - Boot + Secure FW,
  - Boot loader,
  - OS (Linux, AOSP, RTOS)
  - User space
    - Busybox, Yocto ..
  - Hypervisor (Type 1, Type 2)



Reference: 9elements joins ARM System Ready Program

[Reference](#): Arm® Confidential Compute Arch

Complex SoC Boot Flows

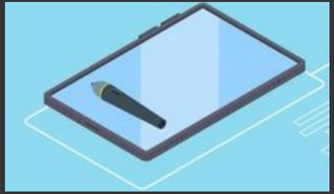
Runtime Security Subsystem (RSS)

System Control Processor (SCP)

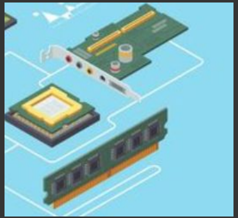
Manageability Control Processor (MCP)

Application Processor (AP)

HDD

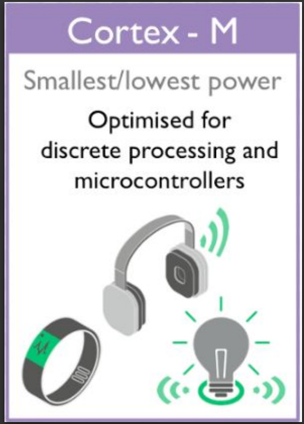


HDMI Display



PCIe cards / WIFI adapter

Compute Subsystem



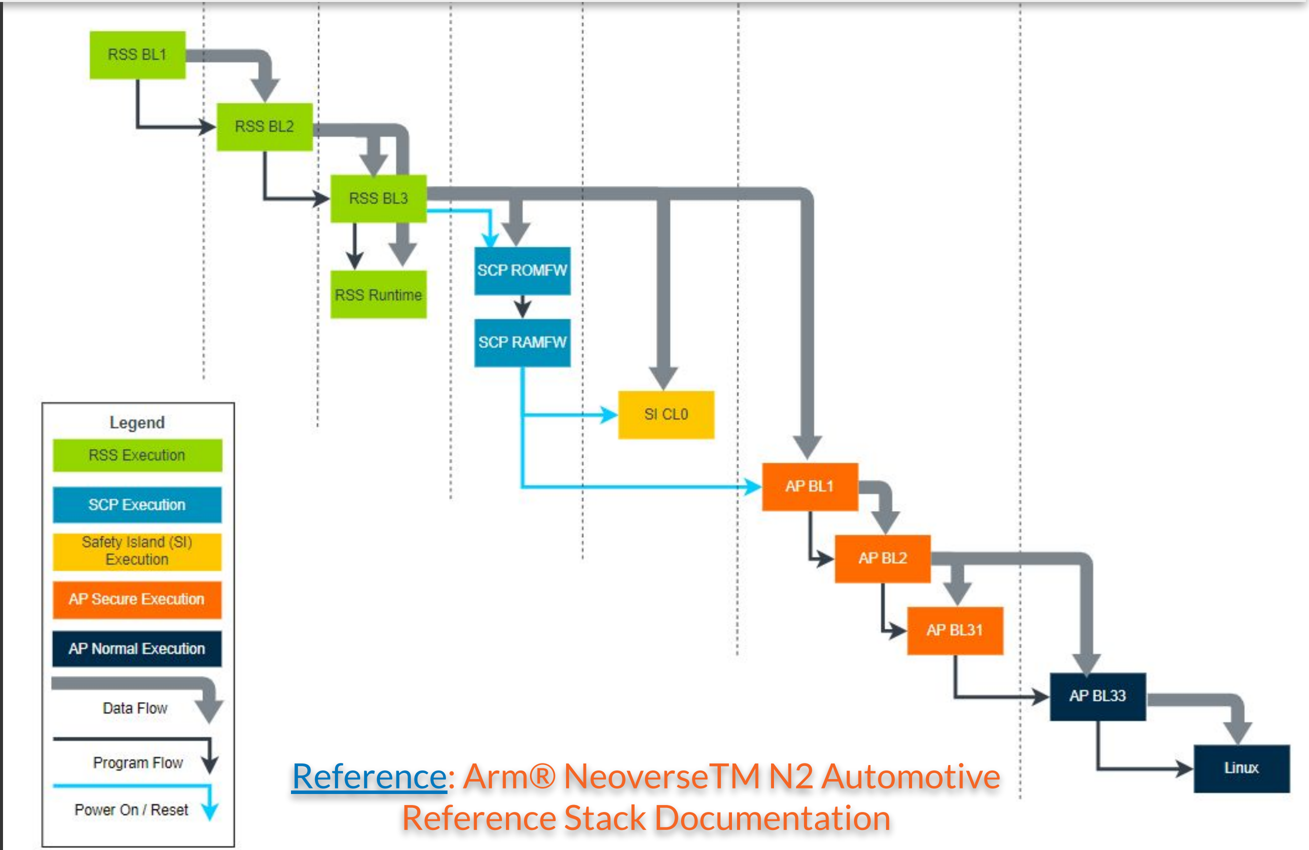
Peripheral I/O Subsystem

Complex SoC Boot Flows (getting complex'er)

- Root of trust / Security,
- Power Management,
- BMC,
- Bootloader,
- Operating system loader

Features 🔼

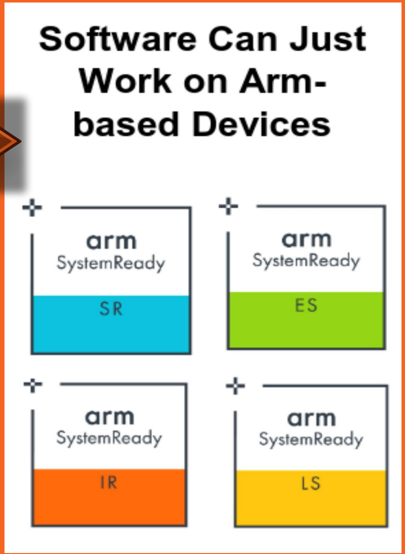
Complexity 🔼



Reference: Arm® NeoverseTM N2 Automotive Reference Stack Documentation

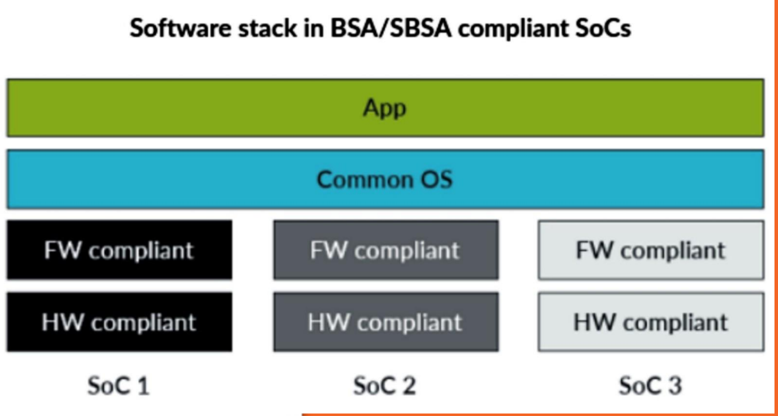
De-Risking & Accelerating OS development for SoCs

- Booting generic (OS) on any SoC:
  - is *significant* milestone, and
  - requires SoC to meet a set of *minimum* hardware & firmware standards.
- Arm SystemReady - software can **just work** on ARM SoCs.



De-Risking & Accelerating OS development for SoCs

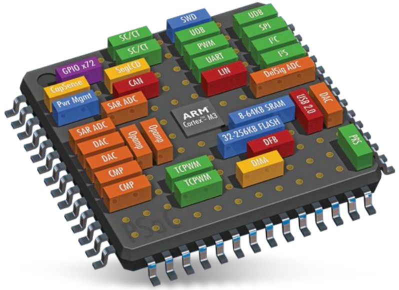
- Base System Architecture (***BSA***):
  - enables hardware compatibility,
  - defines minimum ***CPU & system requirements*** to boot & run an OS.
  - BSA compliance needs to be achieved in silicon hardware.



[Reference](#)

Enabling systems where software 'just works'

- *Open Source Firmware and Software.*
- *Demonstrating pre-tapeout OS-boot.*
- *Testing Pre-Tapeout SW.*
- *Contributing back to open-source.*

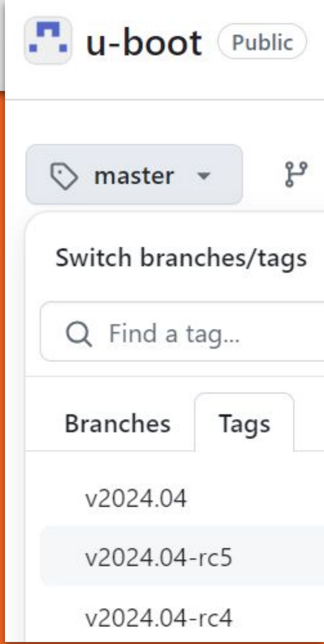


Pre-Tapeout Software Readiness

Open Source to the rescue (*how*)

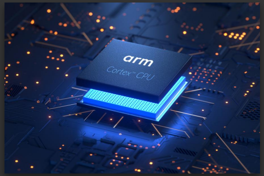
- No need to reinvent the wheel.

- Solid code review by maintainers & contributors.
- Release version management.

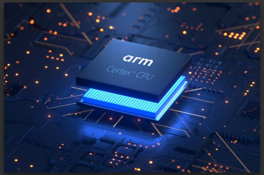


- Adherence to specifications - example UEFI EDK2
- Interoperability checks.
- Easy to setup CI tests.

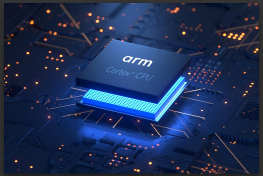
Open Source SW (*Primary Compute*)



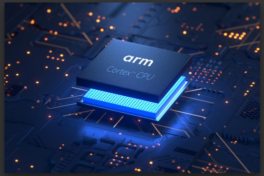
Runtime Security Subsystem (RSS)



System Control Processor (SCP)



Manageability Control Processor (MCP)

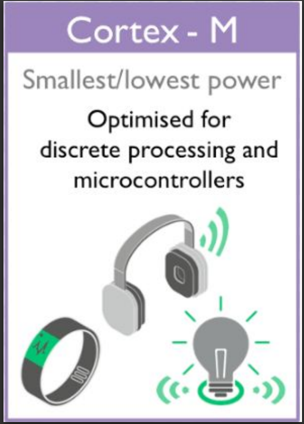


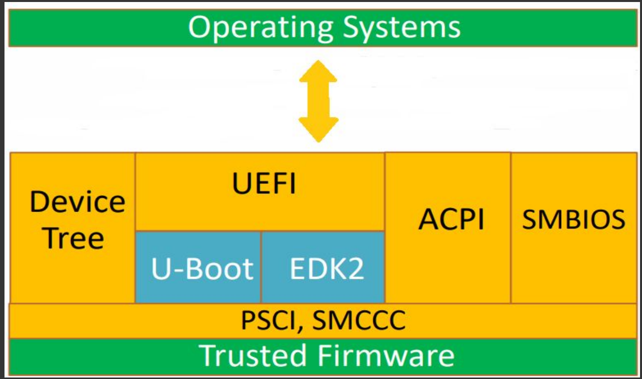
Application Processor (AP)

ARM-software / **SCP-firmware**

Reference: Processor Firmwares

Compute Subsystem



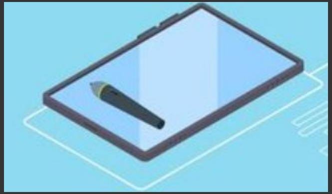


Application processor SW

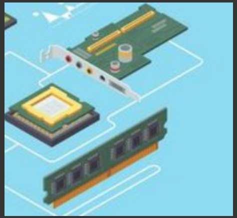
Open Source SW *(IO subsystem)*

- Open Source Firmware:
  - Linux Vendor Firmware Service (LVFS)
    - secure portal which allows hardware vendors to upload firmware updates
  - Linux firmware
    - contains firmware *binary blobs* necessary for functionality of hardware devices.
- Open Source Device Drivers

HDD



HDMI Display



PCIe cards / WIFI adapter

Peripheral I/O Subsystem

Optimizing Boot flow (some ideas)

- Boot-to-prompt / userspace (*KPI*):
  - critical for IoT and hand-held devices,
- Secondary compute cores sit idle while the boot firmware runs on the primary application core.
  - Boot timings can be optimized using available multiple-cores in the SoC,
- Handing over security, power management etc to dedicated co-processors.boot flow optimization,

source

Optimizing
Android boot time

Optimizing Linux
Boot time

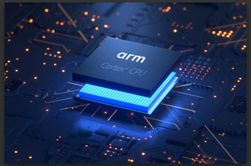
Demonstrating Pre-Tapeout SW

OSes

vmware ESXi | Windows | redhat | SUSE | ubuntu

CentOS | ORACLE Linux | fedora | NetBSD | FreeBSD

TrustedFirmware	• Open source for Secure World firmware
TianoCore / EDK2	• Open source for UEFI, ACPI, SMBIOS standard system firmware
U-Boot	• Open source for embedded systems firmware
LinuxBoot	• Open source for cloud providers Linux-based firmware
OpenBMC	• Open source BMC firmware

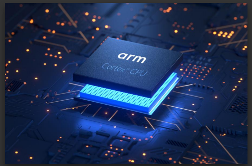


Application Processor (AP)

Primary Compute

ARM-software / **SCP-firmware**

Reference: Processor Firmwares



Runtime Security Subsystem (RSS)

Root of Trust

System Control Processor (SCP)

Power Control

Manageability Control Processor (MCP)

Baseboard Management

Demonstrating Pre-Tapeout SW

Reference: functionally accurate simulation models

ARM Fast models *(Free)*

Qemu

ARM FVP

TLM model(s)

Reference: cycle accurate RTL models

Synopsys Zebu

Cadence PXP / PDP

Reference: ARM Morello Board

Actual Board

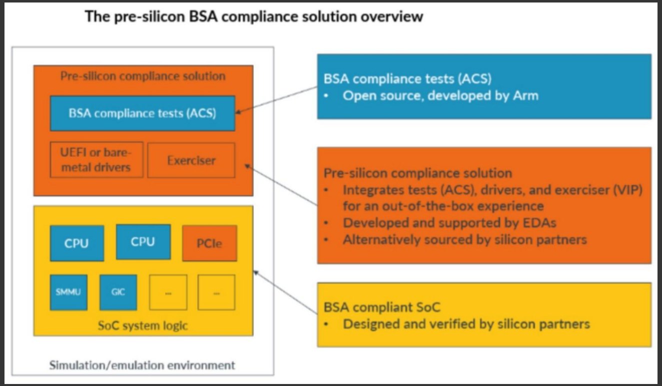
Pre-Silicon Evaluation platforms

Post-Si Evaluation platform

Testing Pre-Tapeout SW

UEFI

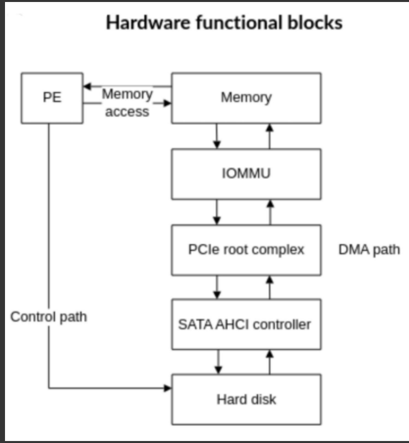
ACS - UEFI Application



shell> sbsa.efi

ACS - Linux Application

Linux



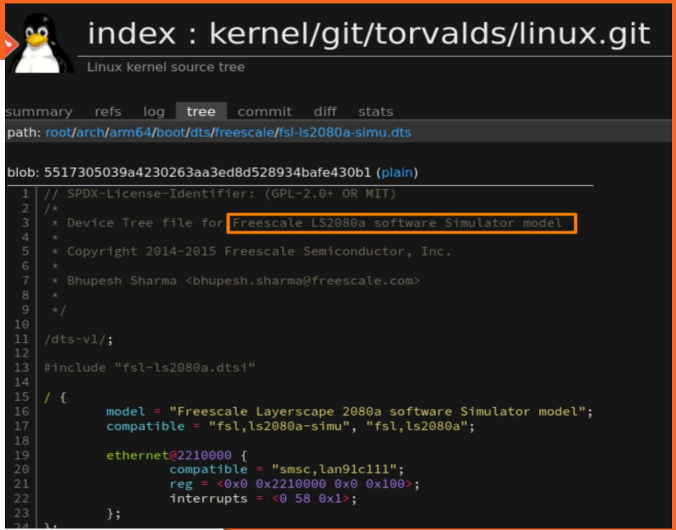
Reference

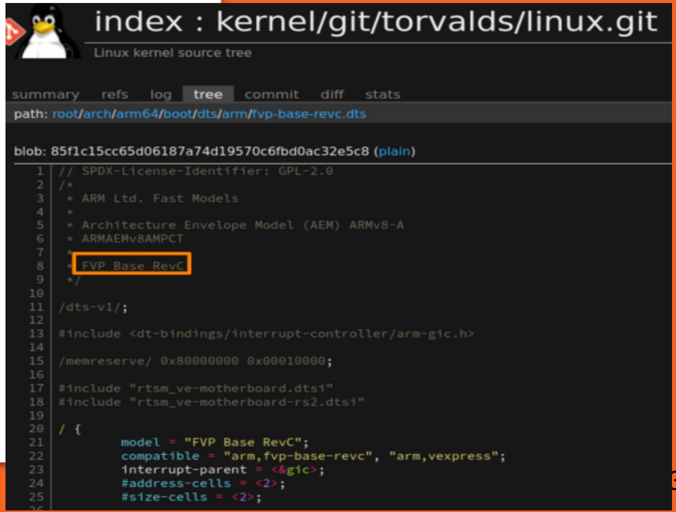
shell> insmod sbsa_acs.ko
shell> ./sbsa

Test environment	Modules
UEFI Shell	PE, GIC, Timers, Watchdog, Wakeup, PCIe, NIST, Peripherals, SMMU, PMU, MPAM, RAS, Memory, and ETE
Linux command line	PCIe, SMMU, PMU

Contributing Pre-Tapeout SW back to Open-Source

- Boot FW:RSS, SCP, MCP FW.
- Bootloaders: u-boot, UEFI
- Secure FW: Trusted Firmware Architecture
- Linux - dts, drivers, test-tools
- Test suites - ACS, etc





Quick Recap..

- *Maximizing reuse of* SW layers for Pre-silicon and Post-silicon bringup.
- try *standardized OS boot* and simple *user-space application(s)* on both Pre and Post-silicon platforms.
- *upstream* code developed on Pre-silicon platforms.

- ***Standardization***:
    - Specifications,
    - Interoperability checks.
- One size ***doesn't*** fit all:
    - End use-case,
    - Available RAM, flash resources.
- ***Open source*** first advocacy.

Challenges

- Use ***virtualized*** test platforms:
  - Qemu,
  - ARM fast models,
  - ARM FVP models.
- Use ***low-cost*** test platforms:
  - UEFI showcase on RPI4
- ***Open source*** first:
  - Use mailing lists & discussion forums.

Next Steps

Questions?

Slides can be found on [github]

@bhupesh_sharma

The talk is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.

References

- Boot Flow for RD-Fremont platforms.
- Boot Flow for Neoverse N2 automotive platform.
- UEFI plugfest talk.
- Repos:
  - https://trustedfirmware.org/
  - https://www.tianocore.org/
  - https://www.denx.de/wiki/U-Boot