Building Modern Industrial SoC Support in ARM Trusted Firmware

Pragmatic Approaches to Power and Clock Management

Dhruva Gole

Kamlesh Gurudasani

Introduction to the Speakers

Dhruva Gole

(Linux Kernel Engineer at Texas Instruments)

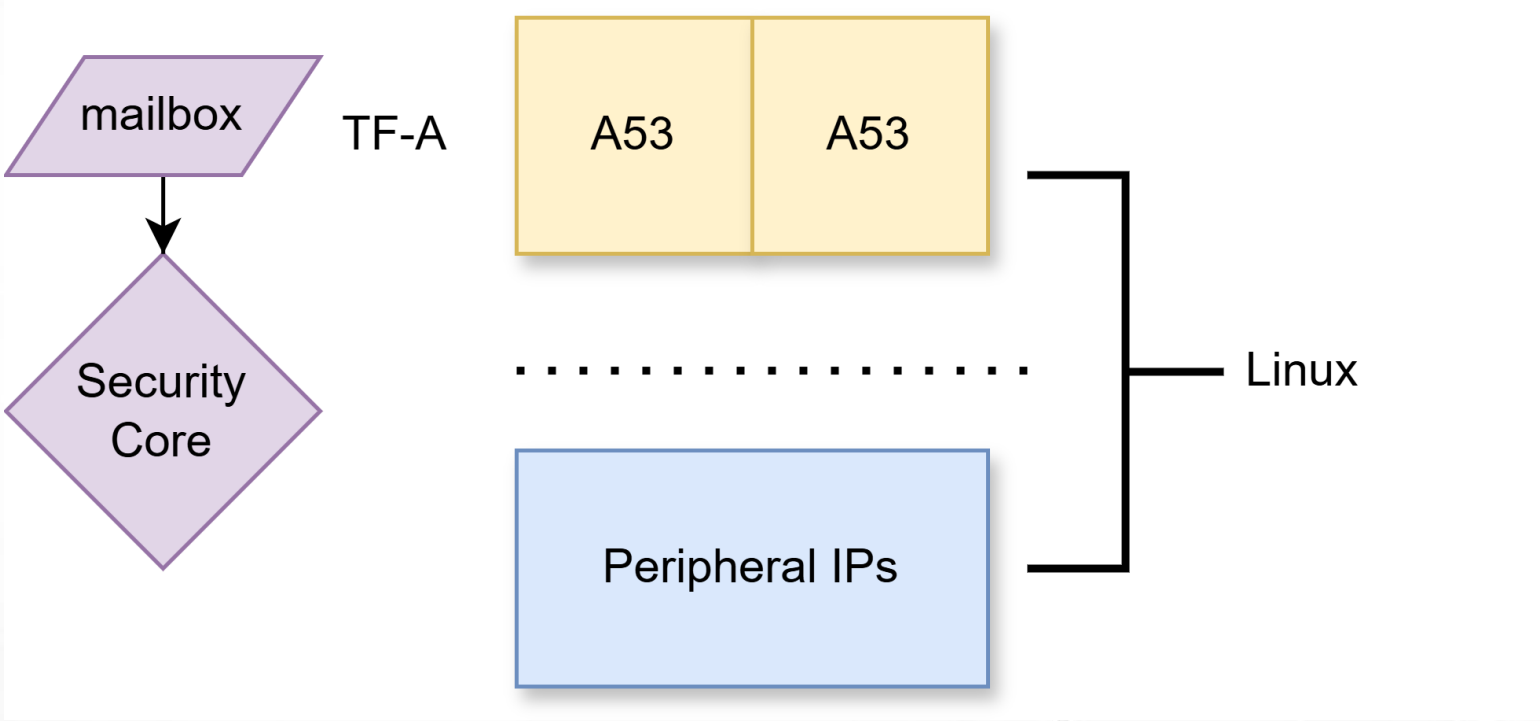
- Power Management on Sitara SOC’s
- U-Boot, TF-A, Linux device driver development

Kamlesh Gurudasani

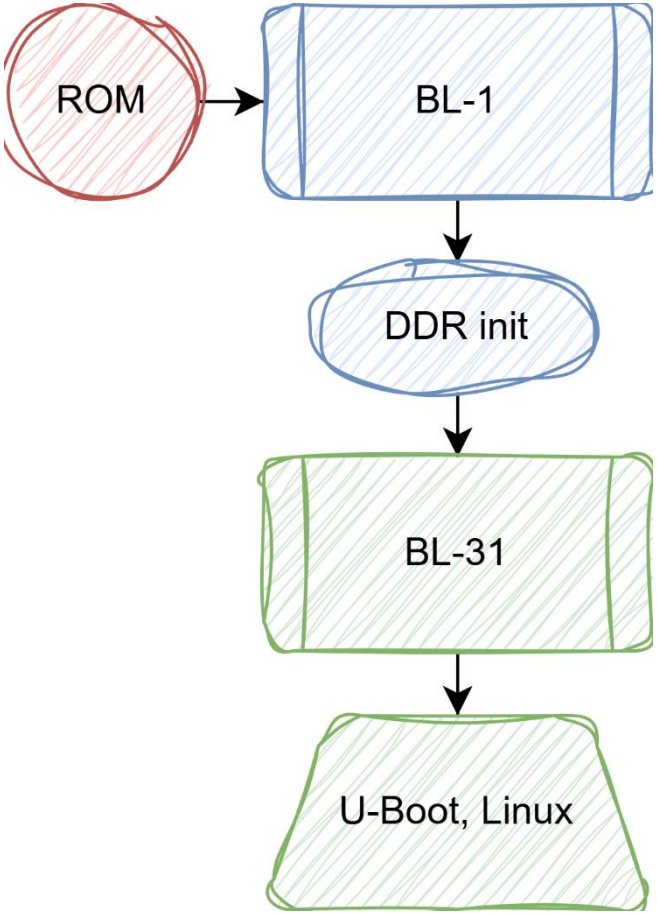
(Linux Kernel Engineer at Texas Instruments)

- Security and Power management on Sitara SOC’s
- U-boot, TF-A, OPTEE, Linux device drivers development

Overview of the SoC: AM62L



Brief boot flow



K3 vs K3low key differences

- K3 had a **co-processor** for power and clock management.
- Boot flow never made use of a BL1 on the cortex A53. DDR was initialized by co processors.
- Heavy reliance on **TI SCI protocol** for clock and PM

- K3Low ie. AM62L no longer has this. Power and clock mgmt. is done in TF-A.
- Due to absence of co-processor, we use the BL-1 now to init DRAM.
- Shift to using **SCMI** for TF-A based clock and PM

Reuse of K3 device drivers for PM and Clocks

- It showcases the capabilities of abstraction possible using SCMI.
- The underneath drivers that interact with SoC specific registers remained the same.
- Porting these drivers from K3 co processor to TF-A was minimal effort.
- The low level stack, has many features that allow abstraction of multiple SoCs via SoC specific data structures.
- It provides flexibility to the higher layers to be able to abstract out details of exact hardware power and clock domain hierarchy.
- Major effort was to create scmi handlers to convert power and clock requests into something the low level drivers understood.

Clock Parent support in TF-A

- The support for clock parent handling missing from TFA.

- A PR has been sent but there is a problem of handling different SCMI clock protocol version having different Structure making them usable from only one version at a time.
- The support for clock protocol version mitigation is available from SCMI Spec 3.2(Clock protocol version 2.0) but here we are talking about SCMI Specification version <3.2 (clock protocol version 1.0)
- Mostly u-boot needs to be bumped to use SCMI Specification version 3.2
- Work in progress

Representing power/clock data → how we are doing as device data structs

```
[AM62LX_PSC_LPSC_LPSC_MAINIP_COMMON] = {
»       .powerdomain = AM62LX_PSC_PD_PD_MAINIP,
»       .depends_psc_idx = AM62LX_PSC_INST_SAM61_WKUP_PSC_WRAP_WKUP_0,
»       .depends = AM62LX_PSC_LPSC_LPSC_MAIN_GP_ISO0_N,
»       .lpsc_dev.dev_array = {
»       »       AM62LX_DEV_CPT2_AGGR0,
»       »       AM62LX_DEV_MSRAM_96K0,
»       »       AM62LX_DEV_ROM0,
»       »       AM62LX_DEV_TRNG_DRBG_EIP76D_WRAP0,
»       },
»       .flags = LPSC_MODULE_EXISTS|LPSC_DEPENDS,
},
[AM62LX_PSC_LPSC_LPSC_MAINIP_DSS] = {
»       .powerdomain = AM62LX_PSC_PD_PD_MAINIP,
»       .depends_psc_idx = AM62LX_PSC_INST_SAM61_WKUP_PSC_WRAP_WKUP_0,
»       .depends = AM62LX_PSC_LPSC_LPSC_MAINIP_COMMON,
»       .lpsc_dev.dev_array = {
»       »       AM62LX_DEV_DSS0,
»       »       DEV_ID_NONE,
»       »       0,
»       »       0,
»       },
»       .flags = LPSC_MODULE_EXISTS|LPSC_DEPENDS,
},
[AM62LX_PSC_LPSC_LPSC_MAINIP_DSI] = {
»       .powerdomain = AM62LX_PSC_PD_PD_MAINIP,
»       .depends_psc_idx = AM62LX_PSC_INST_SAM61_WKUP_PSC_WRAP_WKUP_0,
»       .depends = AM62LX_PSC_LPSC_LPSC_MAINIP_DSS,
»       .lpsc_dev.dev_array = {
»       »       AM62LX_DEV_DSS_DSI0,
»       »       DEV_ID_NONE,
»       »       0,
»       »       0,
»       },
»       .flags = LPSC_MODULE_EXISTS|LPSC_DEPENDS,
},
```

Alternative representation idea: Device Trees

- Standard approach – used in Linux/ U-boot since many years to describe hardware.
- Easy to visualize – complex hierarchies can be easily represented in DT
- Easy to maintain even for non-TI maintainers.

Challenges:

- **Missing OF APIs** for extracting information from DTs.
- There is no clock framework in ATF for creating the hierarchy.

Plan for upstreaming component-wise

- **Phase 1: Refactor** existing vs new device into k3 and k3low, bring out the common drivers in drivers/ti
- **Phase 2:** Setup and initialization code like boot notification from co processor, MMU config, etc.
- **Phase 3:** SCMI protocol, power and clock drivers, which will be used by SCMI agents to turn on peripherals.
- **Phase 4:** PSCI, dual core boot support.
- **Parallelly:** BL-1 support.
- **Phase 5:** Low Power Mode sequences.

Summary

- AM62L is the first of it’s kind SoC to perform complex clock and power management from TF-A.
- Due to this fact, a major blocker faced was the missing clock parent support in TF-A via SCMI.
- TF-A doesn’t have a proper framework like genPD in the Linux kernel, which pushes a lot of complexity like ref counting and power domain representation into the platform specific drivers.
- Missing standardization of how to represent SoC specific data like clock and power domain information causes every vendor to reimplement using their own ideas.

Q&A

- Contact Information:
  - Dhruva Gole d-gole@ti.com
  - Kamlesh Gurudasani Kamlesh@ti.com

- Also on IRC @ libera.chat #linux-ti

Learn more about TI products

- https://www.ti.com/linux
- https://www.ti.com/processors
- https://www.ti.com/edgeai

Why choose TI MCUs and processors?

Scalability

Our products offer scalable performance that can adapt and grow as the needs of your customers evolve.

Efficiency

We design products that extend battery life, maximize performance for every watt expended, and unlock the highest levels of system efficiency.

Affordability

We strive to make innovation accessible to all by creating cost-effective products that feature state-of-the-art technology and package designs.

Availability

Our investment in internal manufacturing capacity provides greater assurance of supply, supporting your growth for decades to come.