Mainline Linux on recent Qualcomm SoCs: Fairphone 4

A look into the work of getting a modern Qualcomm SoC into mainline Linux.

Luca Weiss - FOSDEM 2023

Who am I?

- Luca Weiss (z3ntu)
- Mainlining phones since 2017
- postmarketOS core team member
- Android Platform Engineer at Fairphone

FAIRPHONE

Background

- Qualcomm has many SoCs
- Many are already supported (launched 2018 and later)
  - sm8550, sm8450, sm8350, sm8250, sm8150, sm7225, sm6375, sm6350, sm6125, sm6115, sm4250, sdm850, sdm845, sdm670, sdm632
- Still many more are not supported
  - High-end is quickly supported thanks to Linaro
  - Mid- and low-end not
- You can do it yourself!

Fairphone 4 (SM7225)

- Launched in September 2021 (1 year, 4 months ago)
- Stock kernel: msm-4.19
- Supported functionality as of February 2023 (not everything upstream)
  - Basics such as serial console, power & volume buttons, regulators, RTC
  - USB (nearly type-C functionality with role switching)
  - Internal storage & SD card
  - Display (with backlight control) & Touchscreen & GPU
  - WiFi
  - Remoteprocs but ***not quite*** modem and mobile data
  - Vibration motor, flash/torch LED, camera I2C bus
  - + lots of plumbing

What isn’t working yet?

- Some parts work-in-progress with some success
  - Speaker audio (actually kind of working)
  - Bluetooth (some parts worked)
- Other parts don’t work at all
  - Modem…
  - Microphones
  - Camera subsystem (CAMSS) & ToF sensor
  - Video encoding/decoding (VENUS)
  - NFC
  - Fuel gauge & charging
  - Displayport over USB-C

New SoC - first boot

- Need to figure out bootloader, esp. regarding dtbo
  - “fastboot erase dtbo” probably works
  - Serial console is helpful if you can
- First boot after some hours
  - Earlycon (serial) & display (simple-framebuffer)
  - ~180 lines of SoC .dtsi and ~40 lines of device .dts
  - No driver changes necessary!
- Check out https://mainlining.dev/ from Iskren Chernev
  - Very nice blog/guide for early porting!

New SoC - going further

- Clock driver
  - Take driver from downstream kernel and adjust
  - Add power domains (GDSCs)
- More clocks (rpmh)
- Add more to dts (smem, tcsr_mutex, apss_shared, aoss_qmp)
- Get USB up for better debugging
- Pinctrl driver
  - Also take from downstream
- Add regulators

Things that go wrong

- IOMMU
  - Bootloader initializes some mapping
  - Downstream keeps them and configures some
  - Mainline doesn’t keep bootloader mappings! (but you can dump them if you need)
- Devices like to reboot when something’s wrong
  - Clock isn’t on -> reboot
  - Writing to wrong register -> reboot
  - IOMMU wrong -> reboot
- Debug
  - Print source location to kernel log (framebuffer) and sleep
  - `printk(KERN_ERR “%s:%d\n”, __func__, __LINE__);`
  - Sprinkle it everywhere!
  - Or build as module

To remember

- Commit your progress often
  - When you get something to work -> commit!
  - Clean up afterwards, you can squash it later
- Send patches upstream!
  - Don’t let your progress rot in your git repo
  - Start upstreaming patches early
  - Patches take a while to trickle upstream
  - git send-email is not difficult

Thanks for listening!