FAIRPHONE

From phone hardware to mobile Linux

FOSDEM 2024, Luca Weiss

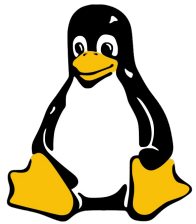
About me

Luca Weiss

Android Platform Engineer @ Fairphone

Open-Source maintainer and contributor

- postmarketOS
- Linux kernel
- OpenRazer



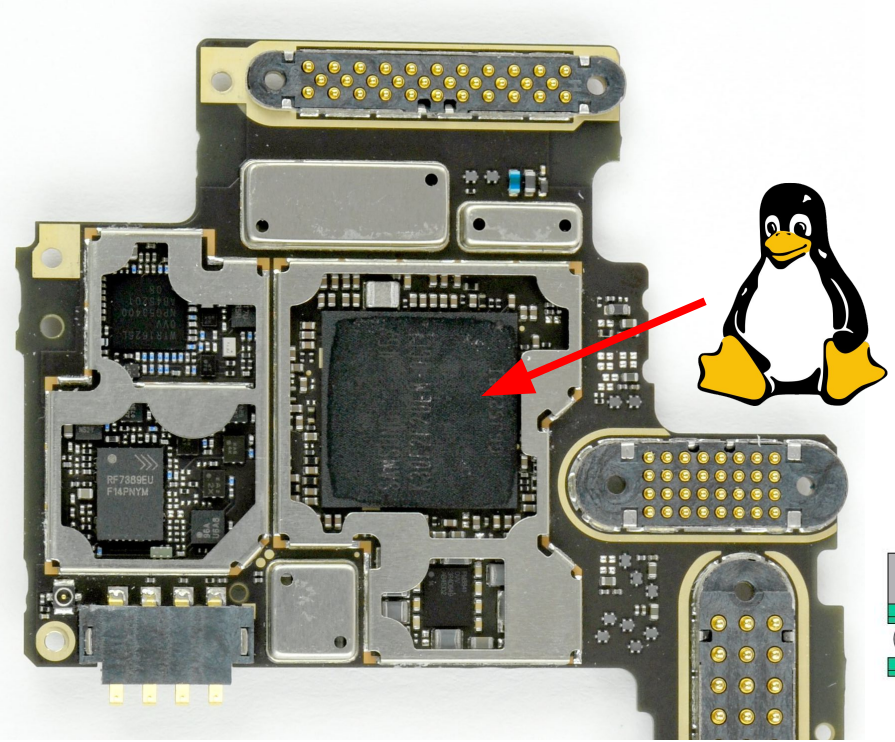
postmarketOS

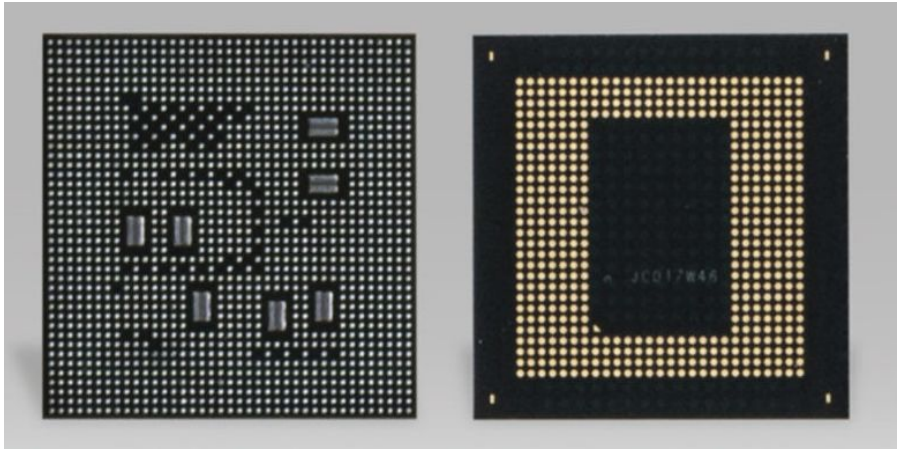
fosstodon.org/@z3ntu

What is this presentation about?

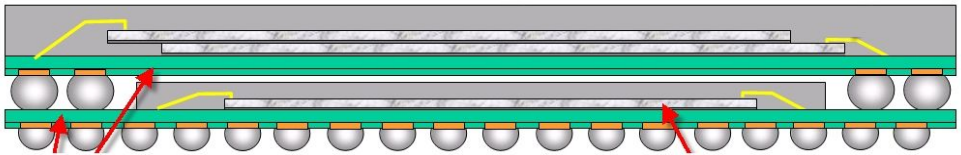
- Understand software concepts by understanding hardware
- Going from PCB level using schematics to Linux and devicetree
- How Linux communicates with the different hardware to make it work
- You might not have schematics for *your* device but concepts are the same

On the Printed Circuit Board (PCB)





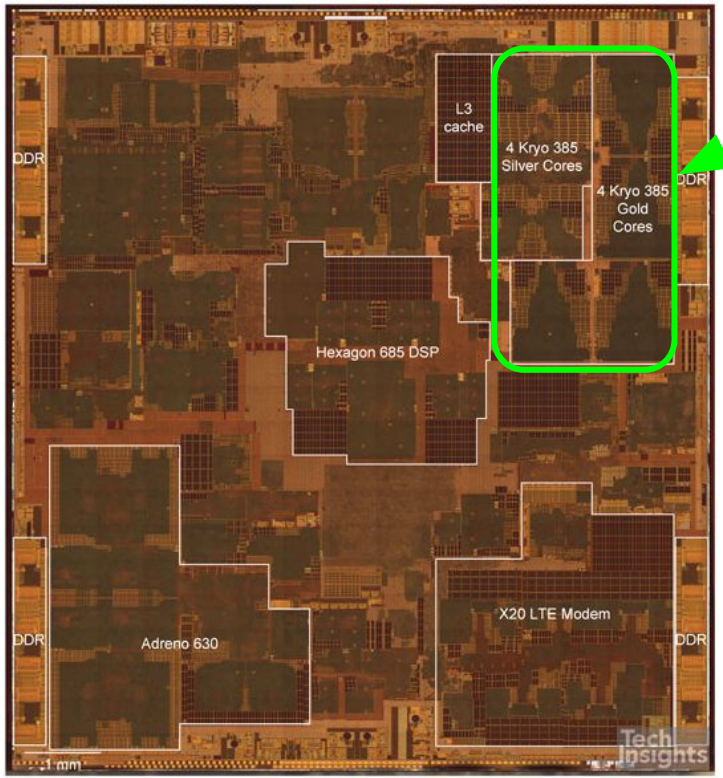
Ball Grid Array (BGA)

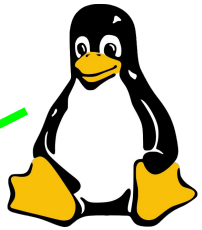


Package on a package (PoP)

SoC and RAM/internal storage are stacked on top of each other (keyword: uMCP - UFS-based multi-chip package)

Inside the System on a chip (SoC)



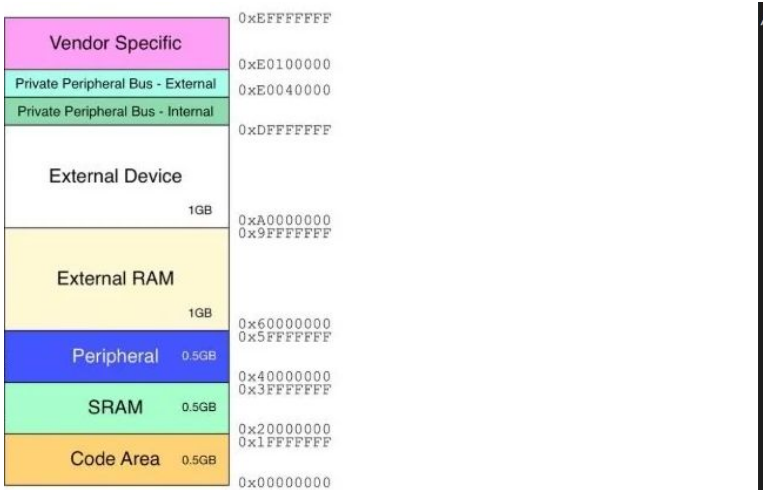


Many co-processors with their own code!

```
fairphone-fp5:~$ ls -1 /lib/firmware/qcom/qcm6490/fairphone5/ | grep mbn
a660_zap.mbn
adsp.mbn
cdsp.mbn
ipa_fws.mbn
modem.mbn
venus.mbn
wpss.mbn
```

```
fairphone-fp5:~$ file /lib/firmware/qcom/qcm6490/fairphone5/adsp.mbn
/lib/firmware/qcom/qcm6490/fairphone5/adsp.mbn: ELF 32-bit LSB executable, QUALCOMM DSP6,
 version 1 (SYSV), statically linked, no section header
```

How do we address anything? MMIO!




```
fairphone-fp5:~$ cat /proc/iomem # edited for simplicity!
00100000-002effff : 100000.clock-controller clock-controller@100000
00408000-00408fff : 408000.mailbox mailbox@408000
[...]
00900000-0095ffff : 900000.dma-controller dma-controller@900000
00984000-00987fff : 984000.i2c i2c@984000
00988000-0098bfff : 988000.i2c i2c@988000
00990000-00993fff : 990000.i2c i2c@990000
00994000-00997fff : 994000.serial serial@994000
0099c000-0099ffff : 99c000.serial serial@99c000
009c0000-009c1fff : 9c0000.geniqup geniqup@9c0000
[...]
18592000-18592fff : 18591000.cpufreq cpufreq@18591000
18593000-18593fff : 18591000.cpufreq cpufreq@18591000
80000000-27ffffff : System RAM # 8 GiB
```

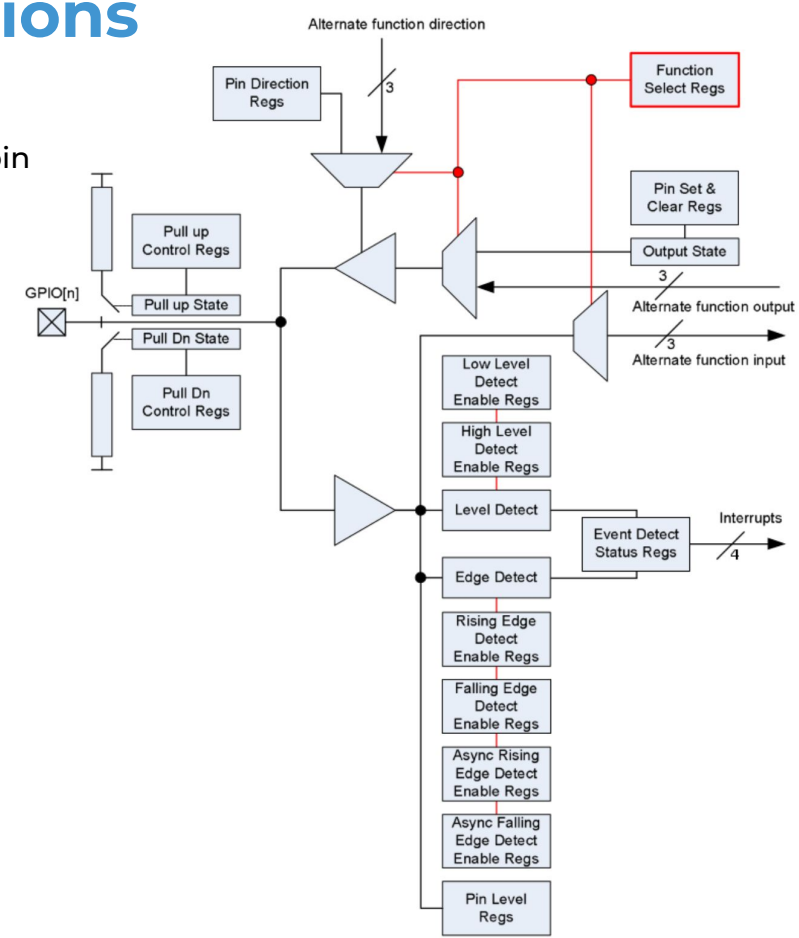
```
/ {
        interrupt-parent = <&intc>;

        #address-cells = <2>;
        #size-cells = <2>;

        soc: soc@0 {
                #address-cells = <2>;
                #size-cells = <2>;
                ranges = <0 0 0 0 0x10 0>;
                dma-ranges = <0 0 0 0 0x10 0>;
                compatible = "simple-bus";

                gcc: clock-controller@100000 {
                        compatible = "qcom,gcc-sc7280";
                        reg = <0 0x00100000 0 0x1f0000>;
                        // [...]
                };

                ipcc: mailbox@408000 {
                        compatible = "qcom,sc7280-ipcc", "qcom,ipcc";
                        reg = <0 0x00408000 0 0x1000>;
                        // [...]
                };

                gpi_dma0: dma-controller@900000 {
                        compatible = "qcom,sc7280-gpi-dma", "qcom,sm6350-gpi-dma";
                        reg = <0 0x00900000 0 0x60000>;
                        // [...]
                };

                qupv3_id_0: geniqup@9c0000 {
                        compatible = "qcom,geni-se-qup";
                        reg = <0 0x009c0000 0 0x2000>;
                        // [...]
```

Too few pins - too many functions

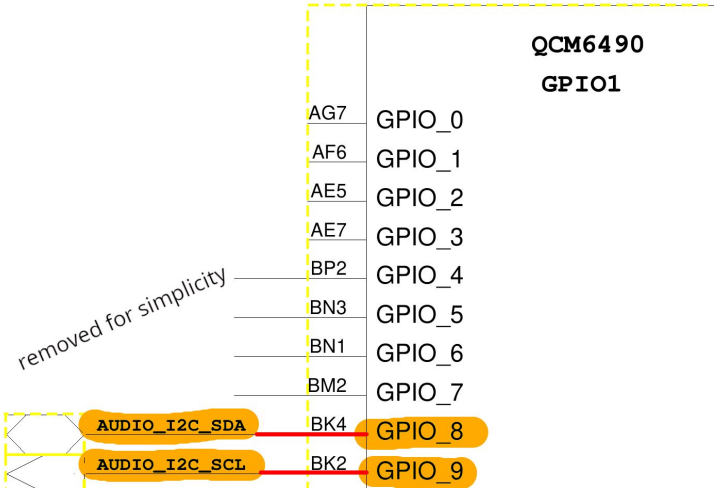
- Many GPIOs have multiple functionalities behind the same pin
- Enable flexibility in using SoC for different use cases
  - e.g. need more I²C busses or more SPI busses
  - Hardware designer chooses which function use
- Most pins have alternate functionality behind one or more functions
- Bit-banging (controlling GPIO high/low in software) is CPU intensive and prone to bad timing
  - Dedicated hardware solves this

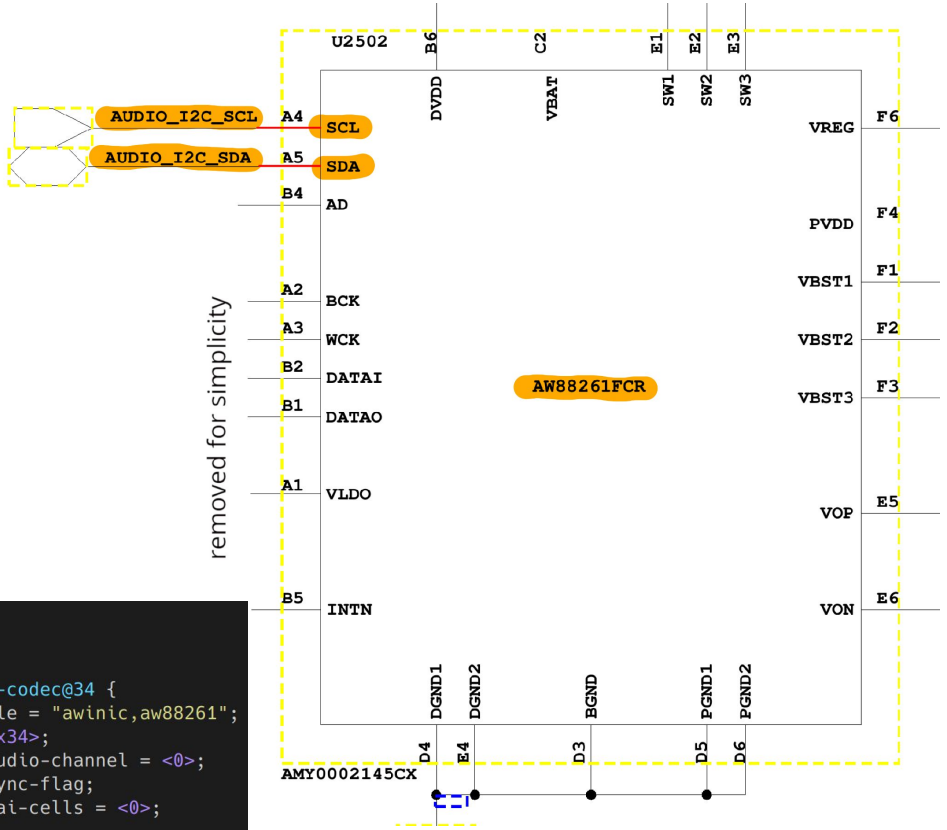
```
&tlmm {
        hall_sensor_default: hall-sensor-default-state {
                pins = "gpio155";
                function = "gpio";
                drive-strength = <2>;
                bias-pull-up;
        };
```

```
        qup_i2c6_data_clk: qup-i2c6-data-clk-state {
                pins = "gpio24", "gpio25";
                function = "qup06";
        };
```



Speaker with external amplifiers: Control path - I²C connection



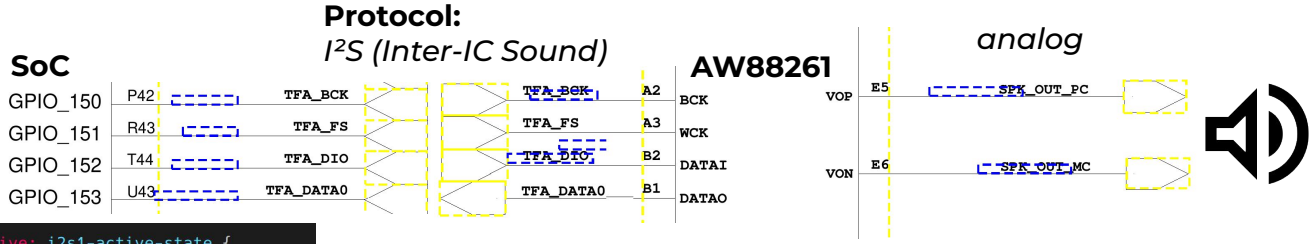



```
static const struct msm_pingroup sc7280_groups[] = {
        // [...]
        [8] = PINGROUP(8, qup02, _, qdss, _, _, _, _, _, _),
        [9] = PINGROUP(9, qup02, _, qdss, _, _, _, _, _, _),
```

```
qup_i2c2_data_clk: qup-i2c2-data-clk-state {
        pins = "gpio8", "gpio9";
        function = "qup02";
};
```

```
&i2c2 {
        status = "okay";

        aw88261_l: audio-codec@34 {
                compatible = "awinic,aw88261";
                reg = <0x34>;
                awinic,audio-channel = <0>;
                awinic,sync-flag;
                #sound-dai-cells = <0>;
        };
};
```

Speaker with external amplifiers: Data path - I²S connection



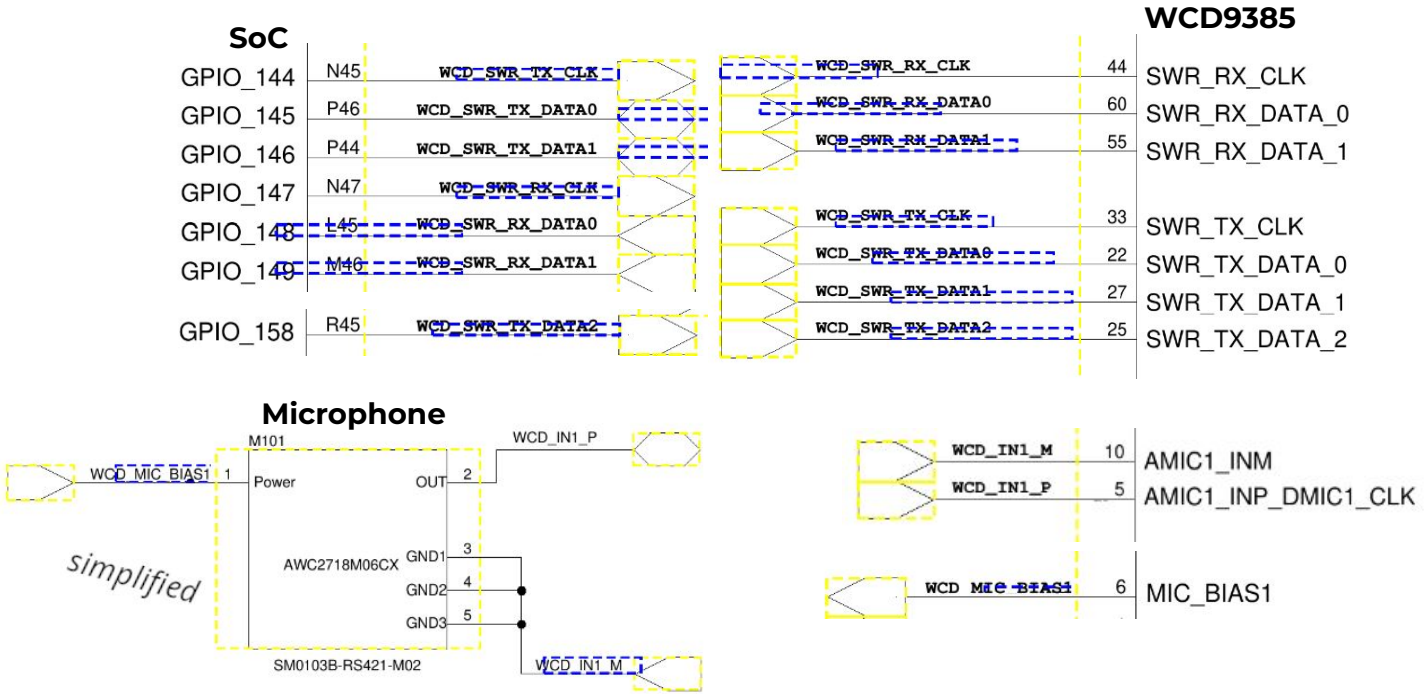

```
i2s1_active: i2s1-active-state {
        clk {
                pins = "gpio6";
                function = "i2s1_clk";
                drive-strength = <8>;
                bias-disable;
                output-high;
        };

        ws {
                pins = "gpio7";
                function = "i2s1_ws";
                drive-strength = <8>;
                bias-disable;
                output-high;
        };

        data {
                pins = "gpio8", "gpio9";
                function = "i2s1_data";
                drive-strength = <8>;
                bias-disable;
                output-high;
        };
};
```

```
&q6afedai {
    dai@127 {
        reg = <QUINARY_MI2S_RX>;
        qcom,sd-lines = <0>;
    };
};
```

```
i2s-dai-link {
        link-name = "I2S Playback";
        cpu {
                sound-dai = <&q6afedai QUINARY_MI2S_RX>;
        };

        platform {
                sound-dai = <&q6routing>;
        };

        codec {
                sound-dai = <&aw88261_l>, <&aw88261_r>;
        };
};
```

CPU via I²S to AW88261: Speaker (stereo: top & bottom)

Microphone:
Data & control path: Soundwire



CPU via SoundWire to WCD9385: Microphones (AMIC1, AMIC3, AMIC4), USB-C audio (HPH + AMIC2)

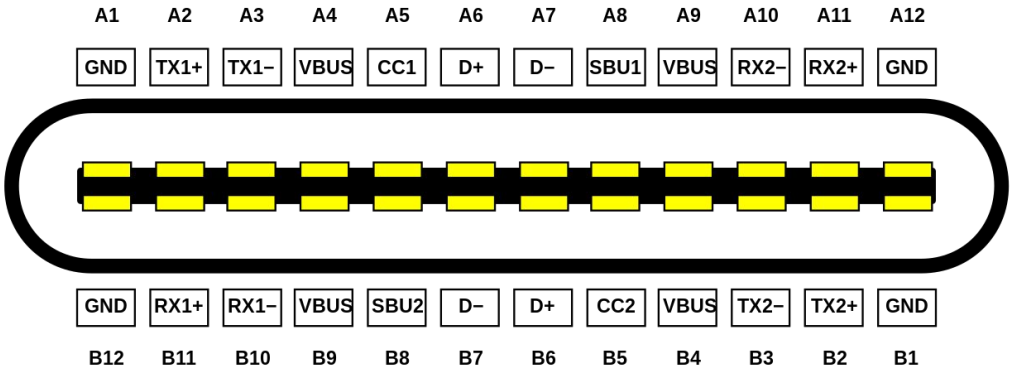
Microphone: Configuration in devicetree

```
&swr0 {
        status = "okay";

        wcd_rx: codec@0,4 {
                compatible = "sdw20217010d00";
                reg = <0 4>;
                qcom,rx-port-mapping = <1 2 3 4 5>;
        };
};

&swr1 {
        status = "okay";

        wcd_tx: codec@0,3 {
                compatible = "sdw20217010d00";
                reg = <0 3>;
                qcom,tx-port-mapping = <1 2 3 4>;
        };
};
```

```
wcd9385: audio-codec-1 {
        compatible = "qcom,wcd9385-codec";

        pinctrl-0 = <&wcd_default>;
        pinctrl-names = "default";

        reset-gpios = <&tlmm 83 GPIO_ACTIVE_LOW>;

        qcom,rx-device = <&wcd_rx>;
        qcom,tx-device = <&wcd_tx>;
        // [...]
};
```

```
wcd-capture-dai-link {
        link-name = "WCD Capture";

        cpu {
                sound-dai = <&q6afedai TX_CODEC_DMA_TX_3>;
        };

        platform {
                sound-dai = <&q6routing>;
        };

        codec {
                sound-dai = <&wcd9385 1>, <&swr1 0>, <&lpass_tx_macro 0>;
        };
};
```

USB Type-C



USB-C: the connector for everything

On Fairphone 5:

- USB2.0 (“High Speed”)
- USB3.0 (“SuperSpeed”)
- Analog audio (Audio adapter accessory mode)
- Display out (DisplayPort Alternate Mode)

To achieve this (incl. orientation switching) more components are needed:

- USB Type-C Analog Audio Switch (e.g. OCP96011)
- USB Type-C Redriver (e.g. PTN36502)

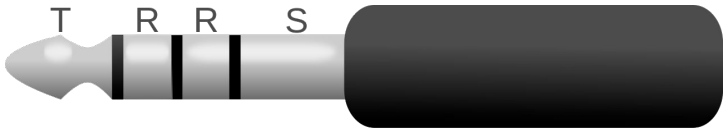
USB-C Audio adapter accessory mode

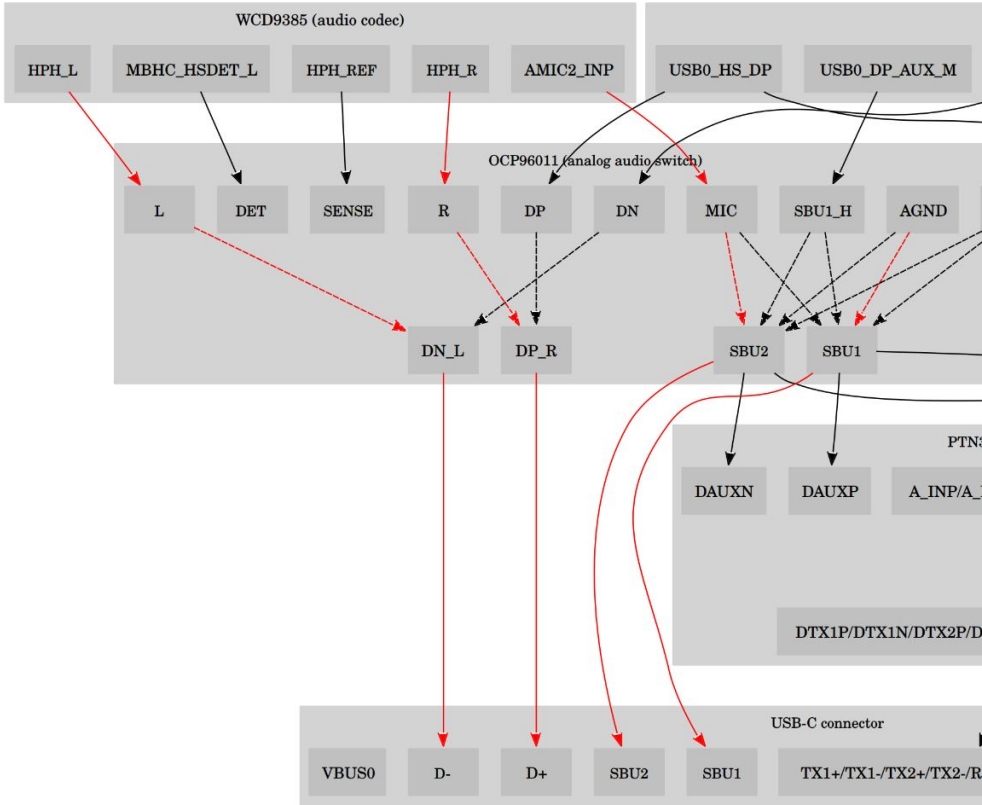
Analog signals use USB 2.0 & SBU pairs:

- D+ ⇒ Right channel
- D- ⇒ Left channel
- SBU1 ⇒ Ground/Microphone
- SBU2 ⇒ Microphone/Ground

(CTIA/OMTP pinout of TRRS 3.5mm)

Linux needs to configure routing for signals to flow



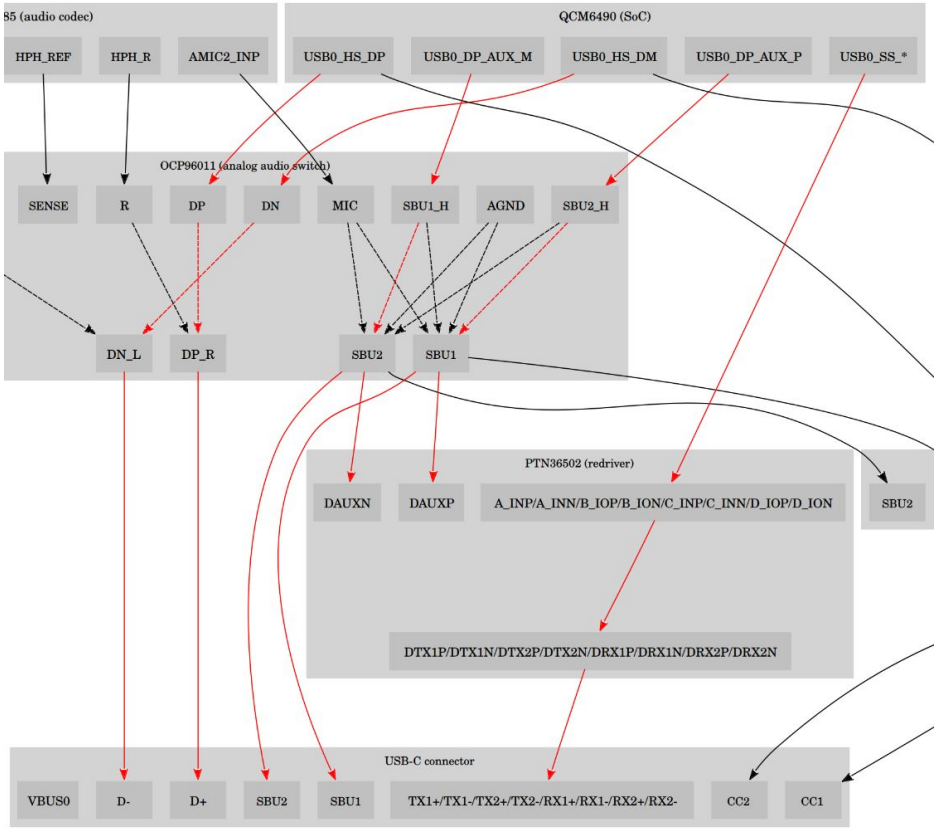


USB-C DisplayPort Alternate Mode

Some devices support DisplayPort over USB-C

- 1/2/4 “USB3.0 pairs” can be used for DisplayPort lanes
  - => DP+USB3.0 or DP+USB2.0
- DisplayPort AUX channels over sideband (SBU) pins

Linux needs to configure routing for signals to flow
DP AUX channel needs manual switching



Devicetree reminders

- Devicetree represents **hardware**
  - Write bindings & commit messages accordingly
- All power supplies and GPIOs should be represented in bindings
- Devicetree is operating system independent
  - U-Boot / FreeBSD / etc. should be able to use them

Thanks!

Questions?

@z3ntu@fosstodon.org

https://lucaweiss.eu