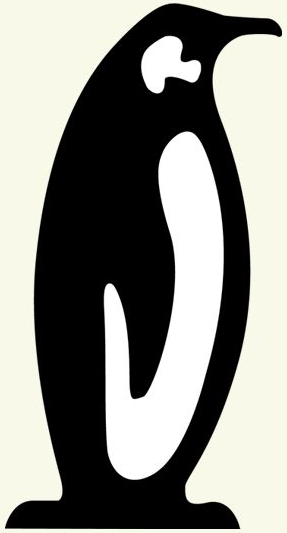


Linux Plumbers Conference

Richmond, Virginia | November 13-15, 2023

Linux
Plumbers
Conference | Richmond, VA | Nov. 13-15, 2023

Linaro
Developer Services

Powering up “discoverable bus-attached” devices on DT-based platforms

... or how not all buses are created equal.

“Discoverable buses that aren’t quite discoverable...”

Linus Walleij

Who am I ?

- Linaro engineer working on Qualcomm upstream support
- Involved in upstreaming a couple of recently announced Qualcomm platforms
- Maintainer of NXP's i.MX clocks
- Enjoy understanding HW inner bits from a kernel hacking perspective
- First time attendee at LPC

Outline

- Introduction
- X13s: Wi-Fi & Bluetooth use case
- Power sequencing Subsystem
- USB Onboard-hub approach
- Conclusion

Introduction

Why talk about this?

What's a discoverable bus?

What's a non-discoverable bus?

What's a "DT-based" platform?

What's the problem?

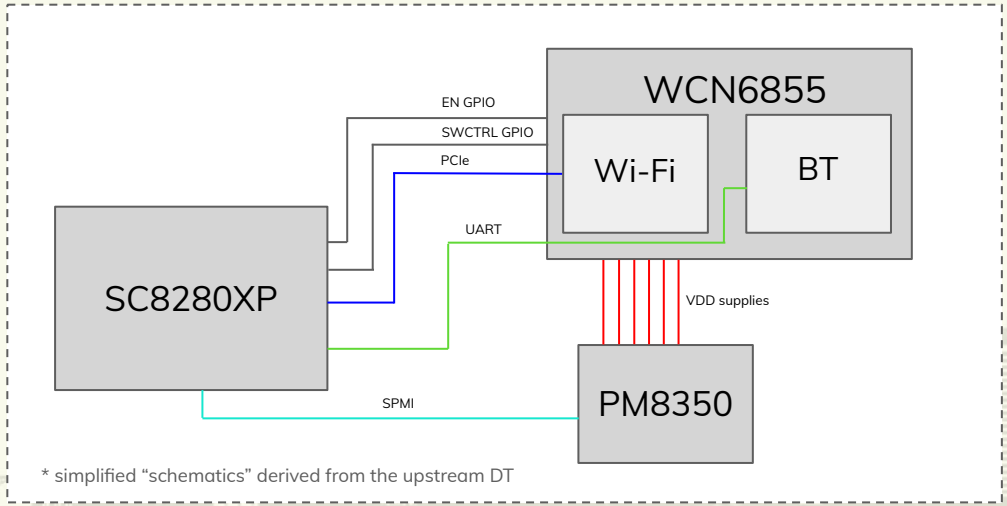
Linux
Plumbers
Conference | Richmond, VA | Nov. 13-15, 2023

Linaro
Developer Services

X13s: Wi-Fi & Bluetooth use case

X13s: PCIe Wi-Fi & UART Bluetooth

- Qualcomm FastConnect 6900 Series - Wi-Fi & Bluetooth (WCN6855)
- Qualcomm Power Management IC (PM8350)
- Qualcomm Snapdragon 8cx Gen 3 Compute Platform (SC8280XP)



X13s: PM8350 - regulators devicetree node

```
regulators-0 {
      compatible = "qcom,pm8350-rpmh-regulators";
      qcom,pmic-id = "b";

      ...
      vreg_s12b: smps12 {
                  regulator-name = "vreg_s12b";
                  regulator-min-microvolt = <984000>;
                  regulator-max-microvolt = <984000>;
                  regulator-initial-mode = <RPMH_REGULATOR_MODE_HPM>;
      };

      ...
};
```

X13s: Enable Bluetooth in devicetree

```
&uart2 {
        status = "okay";

        bluetooth {
                compatible = "qcom,wcn6855-bt";

                vddio-supply = <&vreg_s10b>;
                vddbtcxmx-supply = <&vreg_s12b>;
                vddrfacmn-supply = <&vreg_s12b>;
                vddrfa0p8-supply = <&vreg_s12b>;
                vddrfa1p2-supply = <&vreg_s11b>;
                vddrfa1p7-supply = <&vreg_s1c>;

                enable-gpios = <&tlmm 133 GPIO_ACTIVE_HIGH>;
                swctrl-gpios = <&tlmm 132 GPIO_ACTIVE_HIGH>;
                ,,,
        };
};
```

X13s: Bluetooth device probing

```
$ dmesg |grep Bluetooth
[    1.951305] Bluetooth: hci0: setting up wcn6855
[    2.022866] Bluetooth: hci0: Frame reassembly failed (-84)
[    2.078940] Bluetooth: hci0: QCA Product ID   :0x00000013
[    2.078945] Bluetooth: hci0: QCA SOC Version  :0x400c0210
[    2.078946] Bluetooth: hci0: QCA ROM Version  :0x00000201
[    2.078947] Bluetooth: hci0: QCA Patch Version:0x000038e6
[    2.087204] Bluetooth: hci0: QCA controller version 0x02100201
[    2.087207] Bluetooth: hci0: QCA Downloading qca/hpbtfw21.tlv
[    2.585998] Bluetooth: hci0: QCA Downloading qca/hpnv21.bin
[    2.737875] Bluetooth: hci0: QCA setup on UART is completed
$
```

X13s: Bluetooth - device “discovery”

- “Discovery” initiated by the Geni SE controller
- Loops through all child nodes of the controller devicetree node
- Registers each available node as a device
- Device can be powered off entirely

```
geni_se_probe()
   -> devm_of_platform_populate()
       -> of_platform_bus_create()
          -> of_platform_device_create_pdata()
             -> of_device_add()
                -> device_add()
```

X13s: Enable PCIe instance in devicetree

```
&pcie4 {
        vddpe-3v3-supply = <&vreg_wlan>;

        status = “okay”;
};

&pcie4_phy {
        vdda-phy-supply = <&vreg_l6d>;
        vdda-pll-supply = <&vreg_l4d>;

        status = "okay";
};
```

X13s: Wi-fi device probing

```
$ dmesg |grep ath11k
$
```

X13s: Enable PCIe instance in devicetree

```
&pcie4 {
        ...
        pcie@0 {
                device_type = "pci";
                reg = <0x0 0x0 0x0 0x0 0x0>;

                bus-range = <0x01 0xff>;

                wifi@0 {
                        compatible = "pci17cb,1103";
                        reg = <0x10000 0x0 0x0 0x0 0x0>;
                };
        };
};
```

X13s: Enable PCIe instance in devicetree

```
        wifi@0 {
              compatible = "pci17cb,1103";
              reg = <0x10000 0x0 0x0 0x0 0x0>;

              vddio-supply = <&vreg_s10b>;
              vddbtcxmx-supply = <&vreg_s12b>;
              vddrfacmn-supply = <&vreg_s12b>;
              vddrfa0p8-supply = <&vreg_s12b>;
              vddrfa1p2-supply = <&vreg_s11b>;
              vddrfa1p7-supply = <&vreg_s1c>;

              enable-gpios = <&tlmm 133 GPIO_ACTIVE_HIGH>;
              swctrl-gpios = <&tlmm 132 GPIO_ACTIVE_HIGH>;

        };
```

X13s: Wi-fi device probing

```
$ dmesg |grep ath11k
$
```

X13s: PM8350 - regulators devicetree node

```
regulators-0 {
      compatible = "qcom,pm8350-rpmh-regulators";
      qcom,pmic-id = "b";

      ...
      vreg_s12b: smps12 {
            regulator-name = "vreg_s12b";
            regulator-min-microvolt = <984000>;
            regulator-max-microvolt = <984000>;
            regulator-initial-mode = <RPMH_REGULATOR_MODE_HPM>;
            regulator-always-on;
      };

      ...
};
```

X13s: Wi-fi device probing

```
$ dmesg |grep ath11k
[    1.800111] ath11k_pci 0006:01:00.0: BAR 0: assigned [mem ...
[    1.800184] ath11k_pci 0006:01:00.0: enabling device (0000 -> 0002)
[    1.814757] ath11k_pci 0006:01:00.0: MSI vectors: 32
[    1.814781] ath11k_pci 0006:01:00.0: wcn6855 hw2.1
[    2.859800] ath11k_pci 0006:01:00.0: chip_id 0x2 chip_family 0xb ...
[    2.859813] ath11k_pci 0006:01:00.0: fw_version 0x110b196e ...
[    3.216708] ath11k_pci 0006:01:00.0 wlP6p1s0: renamed from wlan0
$
```

X13s: Wi-fi - device discovery

- Discovery initiated by the PCI host controller
- Loops through all slots by reading over the bus
- Registers only devices accessible
- Device needs to be powered on

```
qcom_pcie_probe()
  -> dw_pcie_host_init()
     -> pci_host_probe()
        -> pci_scan_child_bus_extend()
           -> pci_scan_slot()
              -> pci_scan_single_device()
                 -> pci_device_add()
                    -> device_add()
```

X13s: Bluetooth vs Wi-fi - device discovery

Platform device

"Discovery" is based on devicetree node, no bus scanning

Can be completely powered off during discovery

Devicetree compatible string used for driver matching

Pci device

Discovery is based on bus scanning, devicetree ignored

Needs to be powered on for discovery

Devicetree compatible string used for device matching

X13s: Bluetooth power up sequence

```
static int qca_power_on(struct hci_dev *hdev)
{
        ...
        ret = regulator_bulk_enable(power->num_vregs, power->vreg_bulk);
        ...
        ret = clk_prepare_enable(qcadev->susclk);
        ...
        msleep(50);
        gpiod_set_value_cansleep(qcadev->bt_en, 1);
        msleep(50);
        sw_ctrl_state = gpiod_get_value_cansleep(qcadev->sw_ctrl);
        ...
}
```

Linux
Plumbers
Conference | Richmond, VA | Nov. 13-15, 2023

Linaro
Developer Services

Power Sequencing Subsystem

Power Sequencing Subsystem

- "... handles complex power sequences, typically useful for subsystems that make use of discoverable buses ..."
- Initially proposed as dedicated subsystem in 2014
- Merged in v4.0 as part of MMC subsystem
- Respun as a dedicated subsystem in 2021
  - NACKed, mainly due to bindings
  - discoverable buses generic implementation need to control the pwrseq device before discovery
  - pwrseq as property of the bus controller might not be HW accurate

Power Sequencing - provider

```
struct pwrseq_ops {
	int (*pre_power_on)(struct pwrseq *pwrseq);
	int (*power_on)(struct pwrseq *pwrseq);
	void (*power_off)(struct pwrseq *pwrseq);
	void (*reset)(struct pwrseq *pwrseq);
};

struct pwrseq *pwrseq_create(struct device *dev, struct module *owner, const
                             struct pwrseq_ops *ops)

struct pwrseq_provider *__of_pwrseq_provider_register(struct device *dev,
                              struct module *owner,
                              struct pwrseq * (*of_xlate)(void *data,
                              struct of_phandle_args *args)
```

Power Sequencing - consumer

```
struct pwrseq *__must_check devm_pwrseq_get(struct device *dev,
                                            const char *id)

static inline int pwrseq_pre_power_on(struct pwrseq *pwrseq)

static inline int pwrseq_power_on(struct pwrseq *pwrseq)

static inline void pwrseq_power_off(struct pwrseq *pwrseq)

static inline void pwrseq_reset(struct pwrseq *pwrseq)
```

X13s: Enable Bluetooth in devicetree

```
&uart2 {
        status = "okay";

        bluetooth {
                compatible = "qcom,wcn6855-bt";

                vddio-supply = <&vreg_s10b>;
                vddbtcxmx-supply = <&vreg_s12b>;
                vddrfacmn-supply = <&vreg_s12b>;
                vddrfa0p8-supply = <&vreg_s12b>;
                vddrfa1p2-supply = <&vreg_s11b>;
                vddrfa1p7-supply = <&vreg_s1c>;

                enable-gpios = <&tlmm 133 GPIO_ACTIVE_HIGH>;
                swctrl-gpios = <&tlmm 132 GPIO_ACTIVE_HIGH>;
                ,,,
        };
};
```

X13s: Enable Bluetooth in devicetree

```
&uart2 {
        status = "okay";

        bluetooth {
                compatible = "qcom,wcn6855-bt";

                bt-pwrseq = <&pwrseq 0>;
                ,,,
        };
};
```

USB onboard hub approach

The USB onboard hub approach

- Merged in v6.0
- Solves the powering up via separate platform device
  - platform device “discovered” based on devicetree node - in charge with powering up the hub
  - usb device discovered based on USB bus scan - takes care of the rest
  - sysfs link between platform device and usb device
- Platform driver implements power related API - USB driver uses it
  - both drivers implemented in the same file

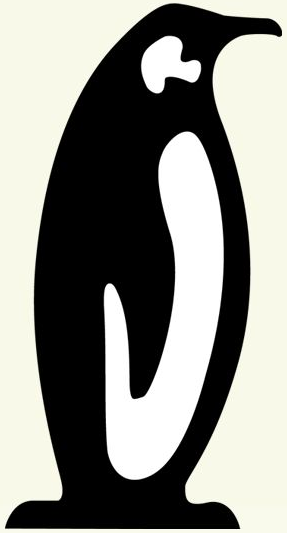
Conclusion

- Discoverable buses don't use DT node for discovery
  - Devices don't get discovered if they are powered off
  - Such buses lack support for powering up devices before discovery
  - Resources kept always enabled
- Dedicated Power sequencing subsystem useful, but optional
  - Takes the control of resources out of the consumer's hands
  - Resources needed for powering up the device can be shared between multiple devices
  - Sometimes there is no dedicated pwrseq device
- USB onboard hub approach can be useful, but hacky
  - Extra platform device and driver needed
  - Shares power related API with bus specific driver

Thank you

Resources

- https://lore.kernel.org/all/20211006035407.1147909-1-dmitry.baryshkov@linaro.org/
- https://lwn.net/Articles/602855/
- https://www.uwsg.indiana.edu/hypermail/linux/kernel/1406.2/03144.html
- https://lore.kernel.org/all/20230110172954.v2.1.I75494ebee7027a50235ce4b1e930fa73a578fbe2@changeid/



Linux Plumbers Conference

Richmond, Virginia | November 13-15, 2023