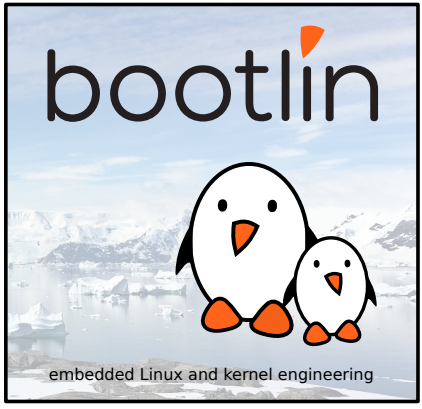
Runtime hotplug on non-discoverable busses with device tree overlays

Luca Ceresoli, Hervé Codina
luca.ceresoli@bootlin.com
herve.codina@bootlin.com



Luca Ceresoli

- Embedded Linux engineer at Bootlin
  - Embedded Linux **expertise**
  - **Development**, consulting and training
  - Strong open-source focus
- Linux kernel device driver developer
- Bootloaders, Buildroot and Yocto integration
- Open-source contributor
- Living in **Bergamo**, Italy

https://bootlin.com/company/staff/luca-ceresoli/

Hervé Codina

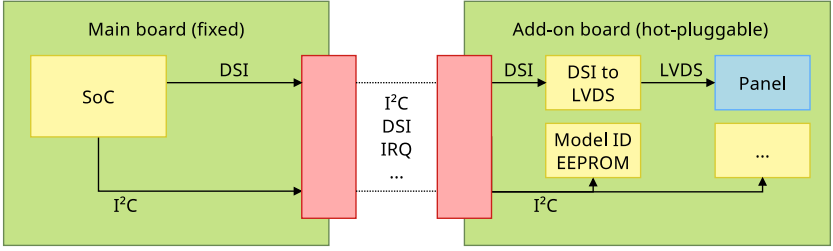
- Embedded Linux engineer at Bootlin
  - Embedded Linux **expertise**
  - **Development**, consulting and training
  - Developer of the **Microchip LAN966x PCI device** support based on device tree overlays over PCI
  - Strong open-source focus
- Open-source contributor
- Living in **Toulouse**, France

https://bootlin.com/company/staff/herve-codina/

Context

Goal

- A healthcare product, advanced hardware prototype, to be launched in 2025
- Classic ARM64 embedded system, using device tree
  - Works standalone with basic features
- Has a connector for an add-on to extend features
  - Proprietary connector
  - Hot-pluggable by user at any moment
  - Connector uses non-discoverable busses (I2C, MIPI DSI, interrupts)
  - Multiple add-on models supported
    - Add-ons have an EEPROM with model ID



Overall solution

- Device tree + Adding hw = calls for **device tree overlays**
- Wrote a connector driver
- Loads 2 overlays
  - 1st: minimum to describe EEPROM with addon model ID (common to all add-on models)
  - 2nd: describes everything else (model-specific)
- Works but with several issues still open
- v4: https://lore.kernel.org/all/20240917-hotplug-drm-bridge-v4-0-bc4dfee61be6@bootlin.com/

Somewhat related work

Support device tree overlays for PCI device board

- by Hervé Codina and Clément Léger (Bootlin)
- `[PATCH v5 0/8] Add support for the LAN966x PCI device using a DT overlay`
- Discussed at Linux Plumbers Conference 2023 (Rob Herring)

BeaglePlay MikroBus support

- by Ayush Singh (beagleboard.org)
- `[PATCH v5 0/7] misc: Add mikroBUS driver`

Generic "adapter" overlays for BBB capes, RPi hats, MikroBUS, Grove...

- by Andrew Davis (TI)
- `[PATCH RFC 0/3] Add generic Overlay for Grove Sunlight Sensor`

Prober for non-discoverable 2nd source hardware

- by Chen-Yu Tsai (chromium.org)
- `[PATCH v7 00/10] platform/chrome: Introduce DT hardware prober`

Device tree bindings

Device tree bindings

- The connector is a device on its own
- The overlay should add nodes, no properties (deadprops, leaks)
- Need to decouple base and addon boards
  - Phandles in overlay pointing to base tree are not allowed
- v4 approach agreed with Rob (except NVMEM cells, still pending)
- See patch 1

Device tree bindings — Main board DTS

```
/ {
    addon-connector {
        compatible = "ge,sunh-addon-connector";
        reset-gpios = <&gpio1 1 GPIO_ACTIVE_LOW>;
        plugged-gpios = <&gpio1 2 GPIO_ACTIVE_LOW>;

        i2c-dbat {
            i2c-parent = <&i2c2_ch1>;   // <-- Decouple adapter on base from devices on addon
            #address-cells = <1>;
            #size-cells = <0>;
        };
        // more i2c busses...

        dsi {
            ports {
                #address-cells = <1>;
                #size-cells = <0>;

                port@0 {        // <-- the base board side; port@1 is missing for now
                    reg = <0>;

                    hotplug_bridge_sink: endpoint {
                        remote-endpoint = <&dsi_to_hotplug_bridge>;
                    };
                };
            };
        };
    };
};
```

Device tree bindings — 1st overlay DTSO

```
/ {
    fragment@0 {
        target-path = ""; // <-- Driver will map this to the connector node
                          //     ("/addon-connector") at runtime (allows multiple connectors)
        __overlay__ {
            nvmem-cells = <&addon_id>;    // <-- FIXME adding properties to base node
            nvmem-cell-names = "id";

            i2c-dbat {
                eeprom@51 {
                    compatible = "atmel,24c64";
                    reg = <0x51>;
                    pagesize = <32>;

                    nvmem-layout {
                        compatible = "fixed-layout";
                        #address-cells = <1>;
                        #size-cells = <1>;

                        /* Data cells */
                        addon_id: addon-id@400 { // <-- phandles can reference labels *within the overlay*
                            reg = <0x400 0x1>;
                        };
                    };
                };
            };
        };
    };
};
```

Device tree bindings — 2nd overlay DTSO

```
/ {
    fragment@0 {
        target-path = "";

        __overlay__ {
            dsi {
                ports {
                    port@1 {
                        reg = <1>;
                        hotplug_bridge_source: endpoint {
                            remote-endpoint = <&sn65dsi84_from_bridge>;
                        };
                    };
                };
            };

            devices { // <-- ``no bus'' platform devices (normally in the root node)
                reg_addon_3v3_lcd: regulator-addon-3v3-lcd {
                    compatible = "regulator-fixed";
                    regulator-name = "3V3_LCD_ADDON";
                };

                backlight_addon: backlight-addon {
                    compatible = "pwm-backlight";
                    power-supply = <&reg_addon_3v3_lcd>;
                };
            };
        };
    };
};
```

NVMEM description

- In v4, the only remaining addition of properties to nodes in the base tree
- Different description needed
- Idea #1: move NVMEM cell properties to a subnode (`/connector/addon-id`)

```
/ {
    fragment@0 {
        target-path = "";

        addon-id {
            nvmem-cells = <&addon_id>;
            nvmem-cell-names = "id";
            // nothing else here
        };
        // *everything* on the add-on stays here
    };
};
```

  - Implementation: use `of_nvmem_device_get()` to get it from the subnode instead of the simpler `nvmem_cell_read_u8()`

NVMEM description — other ideas

- Idea #2: don't use NVMEM cells, open code in connector driver the logic to read from EEPROM
- Idea #3 (under investigation): overlay only adds one node with everything inside

```
/ {
    fragment@0 {
        target-path = "";

        addon { // <-- overlay only adds _one_ node (/addon-connector/addon)
            nvmem-cells = <&addon_id>;    // <-- FIXME adding properties to base node
            nvmem-cell-names = "id";

            // *everything* on the add-on is inside this node
        };
    };
};
```

  - But this would break the current I²C and DSI bindings

Implementation

Topics

- Connector driver overview
- GPIOs and interrupts
- Platform devices
- I^2C devices
- DRM devices
- Devlink issues

Connector driver

- Specific to this proprietary connector
  - `compatible = "ge,sunh-addon-connector"`
  - Can be generalized to similar use cases
- Basic workflow
  1. Detect connection (GPIO)
  2. Insert base overlay
     - Devices are populated
  3. NVMEM notifier: ID cell is available, add-on model is read
  4. Insert 2nd overlay
     - Devices are populated
  5. Profit!
  6. Detect removal (GPIO)
  7. Removes overlays in reverse order
     - Devices are depopulated
  8. Goto 1
- See patch 8

GPIOs and interrupts: tentative idea, not working

Base device tree

```
addon-connector {
    devices {
        addon-gpios {
            compatible = "hotplug-connector-gpios"; // <-- gpio-aggregator
            gpios = <&gpio4 1 IRQ_TYPE_EDGE_FALLING>, <&tca6424_1 9 GPIO_ACTIVE_HIGH>;
            interrupt-controller;  #interrupt-cells = <1>;
            gpio-controller;       #gpio-cells = <2>;

};  };  };
```

Overlay

```
__overlay__ {
    devices {
        addon_gpios: addon-gpios {}; // <-- add a label
    };
    some-chip@123 {
        interrupt-parent = <&addon_gpios>; // <-- phandle to in-overlay label
        interrupts = <0 IRQ_TYPE_EDGE_FALLING>;
};  };
```

- `add_changeset_property()` discards `phandle` properties (would leak anyway)
- Removing the check and letting `phandle` be applied still does not work

GPIOs and interrupts: tentative idea, not tested

Base device tree

```
addon-connector {
    devices {
        addon-gpios {
            compatible = "hotplug-connector-gpios"; // <-- new ad-hoc driver
            // Define the GPIOs to use from the main board controllers
            gpios = <&gpio4 1 IRQ_TYPE_EDGE_FALLING>, <&tca6424_1 9 GPIO_ACTIVE_HIGH>;
            // This is not the GPIO controller
};  };  };
```

Overlay

```
__overlay__ {
    devices {
        addon-gpios {
            addon_gpios: controller { // <-- the node to expose as a controller
                interrupt-controller;  #interrupt-cells = <1>;
                gpio-controller;       #gpio-cells = <2>;
    };  };  };
    some-chip@123 {
        interrupt-parent = <&addon_gpios>; // <-- phandle to in-overlay label
        interrupts = <0 IRQ_TYPE_EDGE_FALLING>;
};  };
```

- Keep the main-board-facing info in the `addon-gpios` node
- Move the addon-facing info to the `controller` subnode

Platform devices

- Two kinds:
  - Devices on a parallel I/O bus (not likely, not in this hardware)
  - Devices without any bus: GPIO/fixed regulators, backlights, panels...
- In DT, clients are in the `devices` node → not probed automatically
- Connector driver calls `of_platform_default_populate()` after overlay is loaded to populate clients
  - Passing pointer to `<connector>/devices` node
- Depopulated automatically by existing code
- See patch 8

I²C devices

- In DT, clients are in the `i2c-xyz` node, not in the adapter node
- Added code to `of_i2c_notify()` to populate clients upon DT node insertion
- Depopulated automatically by existing code
- Issues with this approach
  - Cannot have clients in base `addon-connector/i2c-xyz` node (no big deal)
  - Based on DT notifier: won't populate clients if adapter appears after overlay is loaded (unbind/bind, module reload...)
  - Can be solved by adding info in DT
- See patch 5

DRM

- Discussed on Wednesday
- “Hotplug DRM pipeline components on non-discoverable video busses”
- https://lpc.events/event/18/contributions/1750/

Devlink issues 1/2

- Firmware-inferred links are relaxed and dropped after boot

→ for components in overlays, this can happen before probing

→ devlink not available on removal

→ some suppliers are removed before consumers

  - Details and solution (workaround?) in patch 7
  - Bug exposed by FW device removal

- Some devlinks not created

→ some suppliers are removed before consumers

  - This is a bug even without FW device removal
  - LED using GPIO: fix proposed in a separate patch by Hervé, rejected
  - Backlight using LED: patch 6, similar to Hervé's

Devlink issues 2/2

- DT nodes from overlays default to `FWNODE_FLAG_NOT_DEVICE`

→ some devlinks won't be created for overlays

  - Overlay nodes are 2nd class citizens
  - Bug specific to runtime DT overlays
  - `[PATCH 0/2] devlink: Take care of FWNODE_FLAG_NOT_DEVICE in case of DT overlays`, rejected

- Devlink is known to be broken with overlays
  - Some attempts have been made to fix it
    - `[PATCH v3 0/2] fw_devlink overlay fix` by Saravana

Questions? Suggestions? Comments?

Luca Ceresoli, Hervé Codina
luca.ceresoli@bootlin.com
herve.codina@bootlin.com