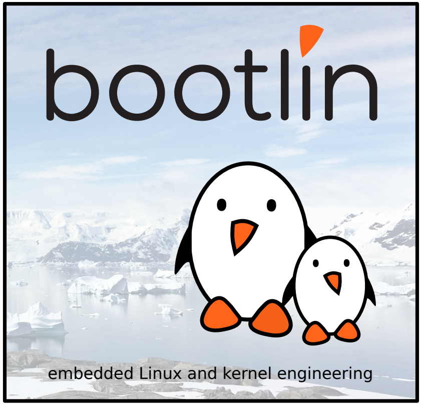
Device Tree overlays and U-boot extension board management

Köry Maincent
kory.maincent@bootlin.com



Köry Maincent

- Embedded Linux engineer at Bootlin
  - Embedded Linux, U-Boot, Linux kernel, Yocto, Buildroot **expertise**
  - **Development**, consulting and training
  - Strong open-source focus
- Open-source contributor
  - Contributed **extension board management support to U-Boot**
  - Contributed Ubuntu support to ELBE
- Living in **Toulouse**, France

Agenda

- Introduction to Device Tree overlays
  - Principle
  - Syntax
  - Support in Linux / U-Boot
- Beagleboard.org use of overlays for CAPE extensions
- The extension board manager in U-Boot

Device Tree

- Data structure that describes the hardware components and topology of the embedded platforms
- Used on a majority of CPU architectures
- See *Device Tree 101*, `https://bootlin.com/pub/conferences/2021/webinar/petazzoni-device-tree-101/petazzoni-device-tree-101.pdf`

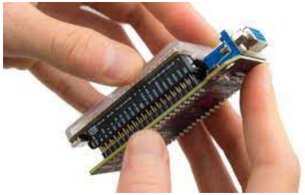
Device Tree

- Data structure that describes the hardware components and topology of the embedded platforms
- Used on a majority of CPU architectures
- See *Device Tree 101*, https://bootlin.com/pub/conferences/2021/webinar/petazzoni-device-tree-101/petazzoni-device-tree-101.pdf

```
/dts-v1/;
/ {
        compatible = "corp,foo";

        /* shared resources */
        res: res {
        };

        /* On chip peripherals */
        ocp: ocp {
                /* peripherals that are always instantiated */
                peripheral1 {
                        compatible = "foo,periph";
                };
        };
};
```

Device Tree Overlay

- Goal:
  - Modify a loaded Device Tree
  - Add, remove, disable or adjust a node of the existing Device Tree
- In practice:
  - Load the overlays corresponding to each extension boards plugged
  - Load the overlay corresponding to the API programed in a FPGA region



Device Tree Overlay syntax

- Old syntax
- New syntax (since devicetree-compiler version 1.5)

Device Tree Overlay syntax

- Old syntax

```
/dts-v1/;
/plugin/;       /* allow undefined label references and record them */
/ {
        /* various properties for loader use; i.e. part id etc. */
        fragment@0 {
                target = <&ocp>;
                __overlay__ {
                        /* bar peripheral */
                        bar {
                                compatible = "bar,periph";
                                /* various properties and child nodes */
                        };
                };
        };
};
```

- New syntax (since devicetree-compiler version 1.5)

Device Tree Overlay syntax

- Old syntax
- New syntax (since devicetree-compiler version 1.5)

```
/dts-v1/;
/plugin/;

&ocp {
        /* bar peripheral */
        bar {
                compatible = "bar,periph";
                /* various properties and child nodes */
        };
};
```

You can use the syntax `&{/ocp}` to specify the targeted node path

Device Tree Overlay syntax

- Old syntax
- New syntax (since devicetree-compiler version 1.5)
- Compilation
  - Build like normal devicetree
  - Use `.dtbo` naming (DeviceTree Blob Overlay)

```
$ dtc -O dtb -o bar-overlay.dtbo bar-overlay.dts
```

Device Tree Overlay syntax

- Old syntax
- New syntax (since devicetree-compiler version 1.5)
- Result: `$ fdtdump bar.dtbo`

```
/dts-v1/;
// magic:               0xd00dfeed
// totalsize:           0xf2 (242)
// off_dt_struct:       0x38
// off_dt_strings:      0xdc
// off_mem_rsvmap:      0x28
// version:             17
// last_comp_version:   16
// boot_cpuid_phys:     0x0
// size_dt_strings:     0x16
// size_dt_struct:      0xa4

/ {
    fragment@0 {
        target = <0xffffffff>;
        __overlay__ {
            bar {
                compatible = "bar,periph";
            };
        };
    };
    __fixups__ {
        ocp = "/fragment@0:target:0";
    };
};
```

Linux kernel support for DT overlays

- Internal kernel API to apply/remove overlays in `<linux/of.h>`
  - `of_overlay_fdt_apply`
  - `of_overlay_remove`
  - `of_overlay_remove_all`
  - + notifiers
- No user-space API provided in the upstream Linux kernel → no way to apply DT overlays from Linux.
- Various user-space APIs proposed over time, but never merged.
- Some downstream platform-specific kernels do have some custom user-space API
- Some subsystems are also not really ready to get extra DT description contributed at runtime

U-Boot support for DT overlays

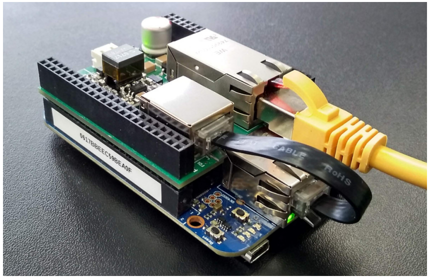
- Easier to handle overlays in U-Boot → they are applied before the DT is passed to the kernel
- `fdt` command has a `apply` subcommand to apply a DT overlay

```
=> fdt
fdt - flattened device tree utility commands

Usage:
....
fdt apply <addr>                    - Apply overlay to the DT
```

- Contributed by Bootlin to upstream U-Boot in 2016!

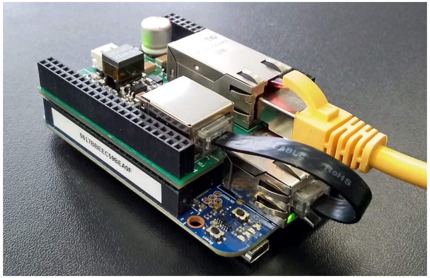
The case of BeagleBoard.org and the CAPEs

- Several possibles CAPEs plugged into one board
- CAPEs compatible with different base boards (BBB, BB-AI)



The case of BeagleBoard.org and the CAPEs

- Several possibles CAPEs plugged into one board
  - Manage the different CAPEs easily by selecting the right overlay
  - Each CAPE has a EEPROM filled with identification data, accessible over I2C
  - BeagleBoard.org made a U-Boot patch to read the EEPROM and load the overlays that matches
- CAPEs compatible with different base boards (BBB, BB-AI)



The case of BeagleBoard.org and the CAPEs

- Several possibles CAPEs plugged into one board
- CAPEs compatible with different base boards (BBB, BB-AI)
  - Different hardware between the different base boards
  - Have to deal with the node naming

BeagleBone_Black_buses.dtsi

```
// I2Cs
bone_i2c_1: &i2c1 {
};

bone_i2c_2: &i2c2 {
    // Already in use for cape EEPROM reading
};
```

BeagleBone_AI_buses.dtsi

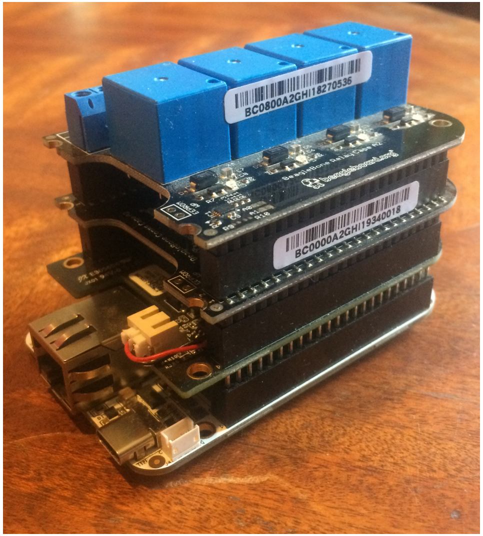
```
// I2Cs
bone_i2c_1: &i2c5 {
};

bone_i2c_2: &i2c4 {
    // Already in use for cape EEPROM reading
};
```

The extension board management in U-Boot

- Implement a **generic extension board manager**
- Replacement for the ad-hoc and platform-specific U-Boot scripts used in BeagleBoard.org forks of U-Boot
- Contributed by Bootlin, merged upstream, available at the v2021.07 release
- List of the commands implemented:
  - `extension scan` to detect available extension boards
  - `extension list` to list the detected extension boards
  - `extension apply` to apply the Device Tree overlay(s) corresponding to one extension board or to all expansion boards
- The `extension scan` command calls a board-specific function to enumerate the detected extension boards and fill-in information used by the generic extension board manager.

Demo

Show time !!



References

- References:
  - https://www.kernel.org/doc/html/latest/devicetree/overlay-notes.html
  - https://elinux.org/Beagleboard:BeagleBone_cape_interface_spec
  - https://lists.denx.de/pipermail/u-boot/2021-May/448794.html

Questions? Suggestions? Comments?

Köry Maincent
kory.maincent@bootlin.com