linaro™

Arm Solutions at Lightspeed

# Devicetree BoF

Linux Plumbers Conference 2024, Vienna

Krzysztof Kozlowski, Linaro
krzk@kernel.org, @krzk@social.kernel.org

# Agenda

1. Some ongoing problems to discuss
   a. Common board-id property
   b. "Configuration" parameters for SoC components
2. dtschema
3. Using Linux kernel DTS in U-Boot (aka OF_UPSTREAM)
4. More ideas
   a. Accepting SoC and/or board DTS purely for other systems
   b. Versioning of the same board DTS
   c. Devicetree and firmware-abstracted hardware
   d. Devicetree bindings for virtual systems and their devices
   e. Reference counting DT properties (Luca Ceresoli, Hervé Codina, today @17:45)

# Common board-id property

- Qualcomm SoC-based Android bootloaders rely heavily on properties:
  - qcom,msm-id - chipset revision
    - Funny thing: multiple IDs can be used per one SoC
  - qcom,board-id - platform or board identification
  - qcom,pmic-id - because they are very creative
  - These are array of integers to identify hardware and allow bootloader to choose appropriate DTB
- Why not compatible?
  - Provided arguments: Comparing strings is too difficult and EEPROM has limited size
    - Too many boards with slight differences to handle via FIT compatible matching
- There is RFC from Elliot Berman (Qualcomm) making the property generic:
  - dt-bindings: hwinfo: Introduce board-id https://lore.kernel.org/all/20240521-board-ids-v3-0-e6c71d05f4d2@quicinc.com/
  - But we need more than one user. Does generic property solve any other vendor's problem?

# Common board-id property - example

- Qualcomm's proposal is to have generic board-id node with per-vendor custom properties like:

```
/ {
    board-id {
        some-hw-id = <value>;
        other-hw-id = <val1>, <val2>;
    };
};
```

# Common board-id property - example (2)

- ... and then several Qualcomm properties

```
board-id {
    qcom,soc-version = <QCOM_ID_SM8650 QCOM_SOC_REVISION(1)>,
                       <QCOM_ID_SM8650 QCOM_SOC_REVISION(2)>;
    qcom,platform-type = <QCOM_BOARD_ID_MTP 0>, <QCOM_BOARD_ID_MTP 1>;
};
```

```
board-id {
    qcom,soc = <QCOM_ID_SM8650>;
    qcom,platform-version = <QCOM_BOARD_ID(MTP, 0, 1, 0)>,
                            <QCOM_BOARD_ID(MTP, 0, 1, 1)>;
    qcom,boot-device = <QCOM_BOARD_BOOT_UFS>;
};
```

# "Configuration" parameters for SoC components

- "Configuration" parameters for SoC components, like I2C timings or thermal characteristics, based on fused values
- The board with given SoC comes with one DTS, but the SoCs have different packages and bins or the board have different characteristics like I2C bus speed
  a. If board chooses some lower or higher clock frequency, other values like timings might need to be affected
- RFC from Krishna Yarlagadda (Nvidia)
  Introduce Tegra register config settings
  https://lore.kernel.org/linux-devicetree/20240701151231.29425-1-kyarlagadda@nvidia.com/

# "Configuration" parameters - Nvidia

- Quoting cover letter:
  *”NVIDIA Tegra SoCs have various I/O controllers and these controllers require specific register configurations based on:*
  *- Functional mode (eg. speed)*
  *- Interface properties (eg. signal timings)*
  *- Manufacturing characteristics (eg. process/package)*
  *- Thermal characteristics*
  *- Board characteristics”*

# "Configuration" parameters - example

```
configsettings {
    configi2c1: config-i2c3160000 {
        i2c-fast-cfg {
            nvidia,i2c-clk-divisor-fs-mode = <0x3c>;
            nvidia,i2c-sclk-high-period = <0x02>;
            nvidia,i2c-sclk-low-period = <0x02>;
            nvidia,i2c-bus-free-time = <0x02>;
            nvidia,i2c-stop-setup-time = <0x02>;
        };

        i2c-standard-cfg {
            nvidia,i2c-clk-divisor-fs-mode = <0x4f>;
            nvidia,i2c-sclk-high-period = <0x07>;
            nvidia,i2c-sclk-low-period = <0x08>;
            nvidia,i2c-bus-free-time = <0x08>;
            nvidia,i2c-stop-setup-time = <0x08>;
        };
    };
};
```

# dtschema - discussion

- What is missing? What could be improved?
  - Any volunteers to actually code it in dtschema?
- More DT schema example files?
  - Several YAML files serving as reference how to implement some bindings?
  - Or maybe better in-kernel docs with examples for common patterns? E.g.
    - How to: GPIO controller with gpio-hogs:

```
  "-hog(-[0-9]+)?$":
    type: object
    required:
      - gpio-hog
```

# Using Linux kernel DTS in U-Boot

- U-Boot since v2024.7 directly imports Linux kernel DTS and uses it for some of the platforms.
  - See: OF_UPSTREAM and commit https://source.denx.de/u-boot/u-boot/-/commit/e3a9829c87422417986432a8007786cd6f6e1c8e
- Several U-Boot platforms were converted to use OF_UPSTREAM, either entirely or partially
  - Exynos, i.MX, Meson, Renesas, Rockchip, Qualcomm and more
- What could it mean?
  - No ABI breaks in the Linux kernel allowed?
  - ABI breaks allowed, but should depend on some sort of analysis on U-Boot impact or Ack from U-Boot maintainers/custodians?
  - Anyway, be mindful about impact of incompatible DT bindings changes on other projects

# Accepting DTS purely for other systems

- Linux kernel is (or we want it to be) the source of DTS, so it might get DTS purely for other projects (e.g. OpenBSD)
- Such DTS was never tested with Linux and might not work, some drivers might be missing
  - Bindings are there, but no drivers
- It is fine from the maintainers point of view, but having it in the kernel creates impression that it is being supported
- Users might actually have such device, try that DTS and send bug reports
  - Real example: Qualcomm X1E80100-based Samsung Galaxy Book4 Edge laptop DTS for OpenBSD
- Is this a problem?
  - Hide it behind CONFIG_EXPERT?
  - Or CONFIG_UNTESTED_DTS?

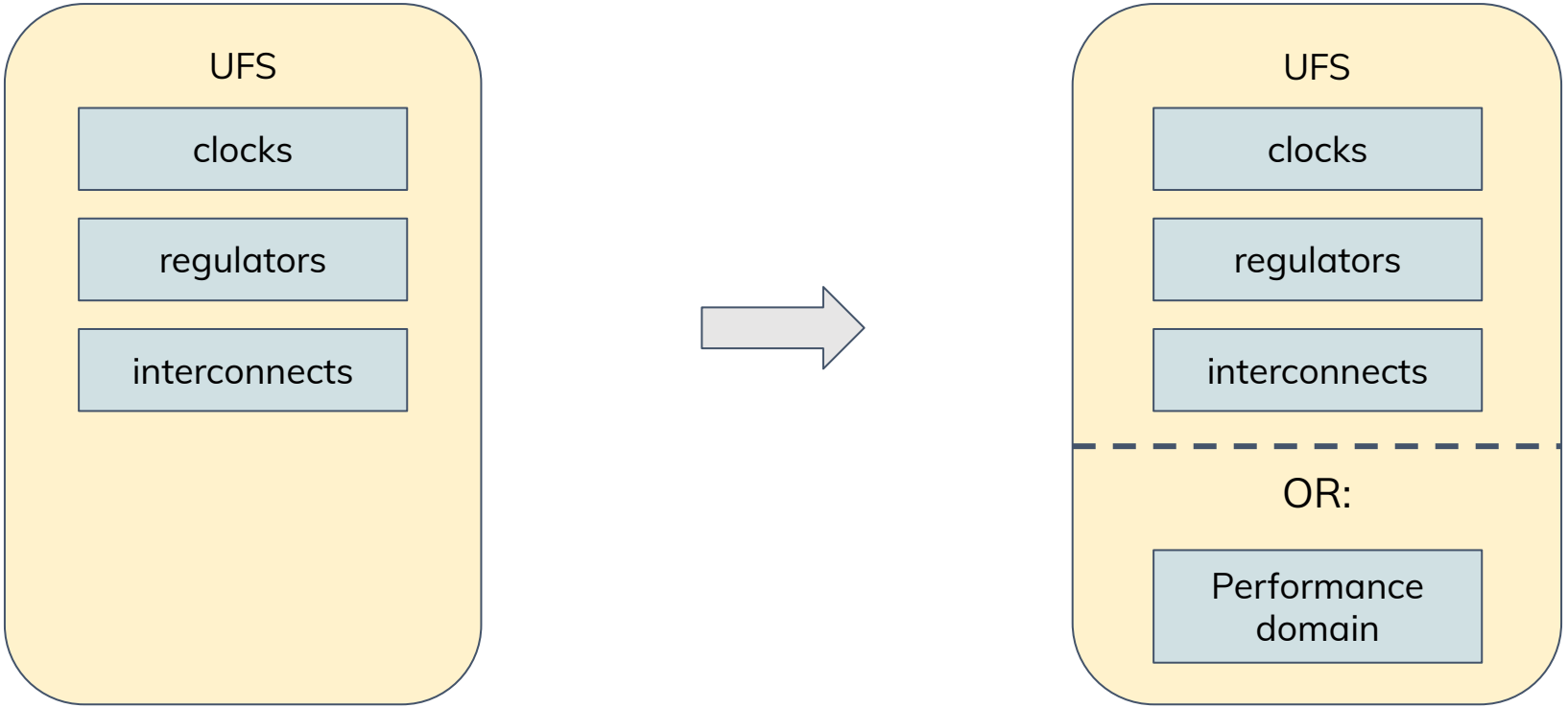
# Versioning of the same board DTS

- Anyone ever had a need to version same board DTS?
  - For example with A/B testing for customers with slight differences
  - Use different compatibles?
    - But hardware is the same
  - New top-level property?

# Devicetree and firmware-abstracted hardware

- Linux kernel has less and less direct access to hardware on modern SoC
- Typically performance and/or energy-saving aspects, like clocks, regulators, power-domains
  a. Hardware can be controlled by dedicated chip with its own firmware
  b. Hypervisor
- Some existing platforms might evolve or receive an updated firmware
- Firmware will expose different interface, e.g. SCMI, for managing exactly the same resources

# Devicetree and firmware-abstracted hardware



# Devicetree and firmware-abstracted hardware

- Instead of clocks -> performance-domains (dvfs/performance-domain.yaml), so is this actually a problem?
- Firmware interface is not discoverable, so standard DT ABI rules apply
  a. Just like hardware, given firmware must not keep changing in incompatible way?
  b. Review comments will be “*DTS describes the firmware*” instead of “*DTS describes the hardware*”?

# Devicetree bindings for virtual systems

- More and more bindings for fully virtualized environments
- Why? Probably because ACPI is heavy and ugly, and Devicetree is neat :)
  - Also comments from David Woodhouse:
    *“We don't want to add extra complexity and overhead on both host and guest side to make things discoverable in a *less* efficient way.”*
- Pushback from DT maintainers:
  - Devicetree came because of non-discoverable hardware
  - Virtualized environment means you have software on both sides, so probably you control them both
  - If you control both parts of software - hypervisor and guest - then come with discoverable protocol and no DT work is needed
- But maybe we should replace ACPI everywhere with Devicetree?

# Reference counting DT properties

- Aka “Runtime hotplug on non-discoverable busses with device tree overlays”
  - Let’s move the discussion there

# Any other topics?

# Thank you