BoF: Early Platform Drivers in Linux Kernel

Bartosz Golaszewski
ELCE 2018
Edinburgh

About us

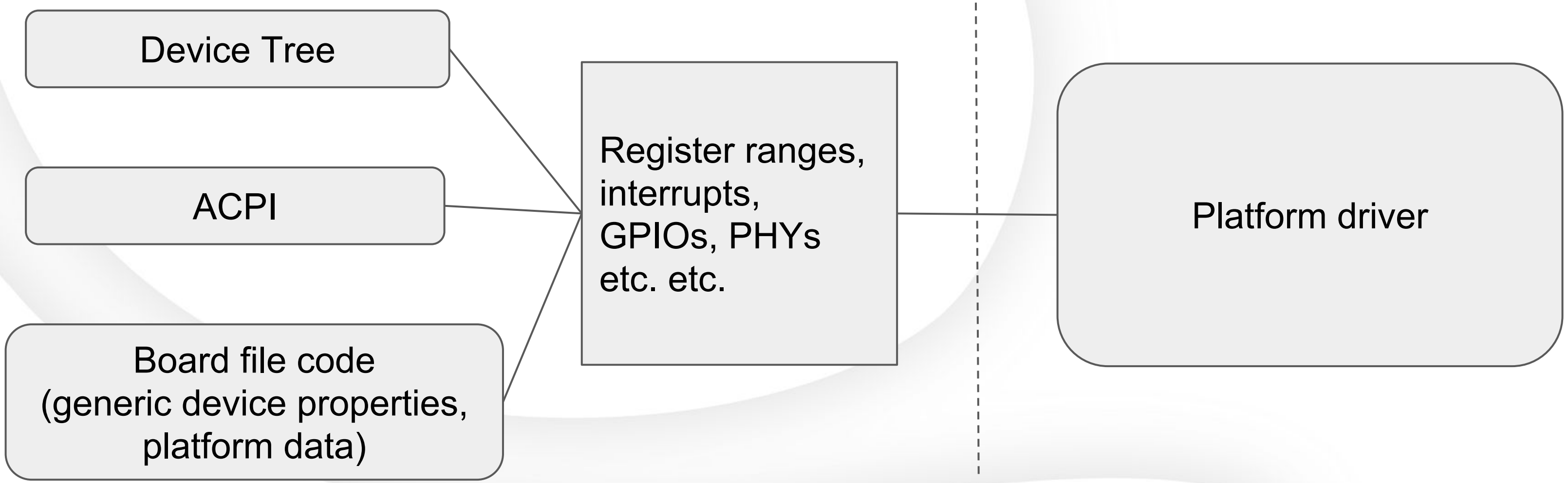
- Embedded Linux Engineering Firm
- ~30 senior engineers, coming from the semiconductor world
- HW and SW products: from concept to manufacturing
- Upstream Linux kernel development and maintenance
- Founding developers of kernelCI.org project

About me

- 9 years experience
- Kernel and user-space developer
- Maintainer of libgpiod

Platform drivers

- Platform drivers are great for separation of code and resources

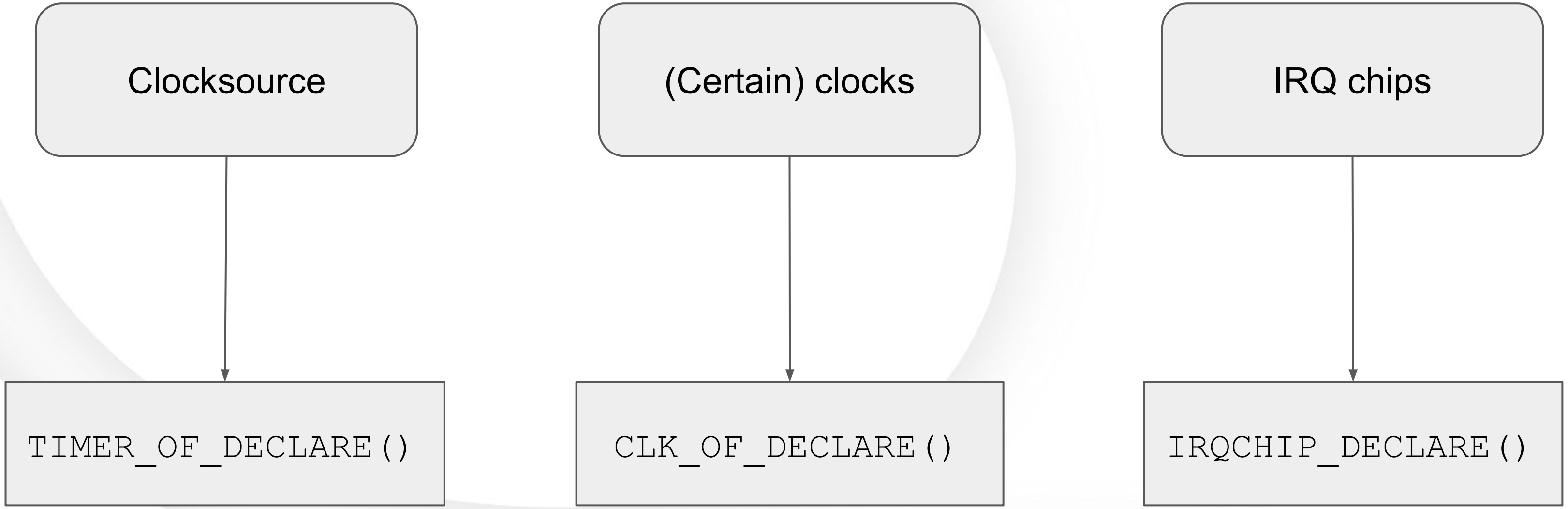

Device Tree
ACPI
Board file code (generic device properties, platform data)
Register ranges, interrupts, GPIOs, PHYs etc. etc.
Platform driver

Platform drivers

- Connected via a virtual platform bus
- Populated from:
  - Device tree:
    - `of_platform_default_populate_init()` as `arch_initcall_sync()`
  - Board files:
    - usually registered from within `init_machine()` callback from `arch_initcall(customize_machine)`
  - ACPI:
    - ????

Problem: some platform devices need to be available early in the boot process

Early devices



Early devices - `OF_DECLARE()`

- generalized `OF_DECLARE()` mechanism
- Not real devices (as in: no `struct device` is being created) :(
- No devres
- No device properties
- No resource setup

Example: TI DaVinci clock driver

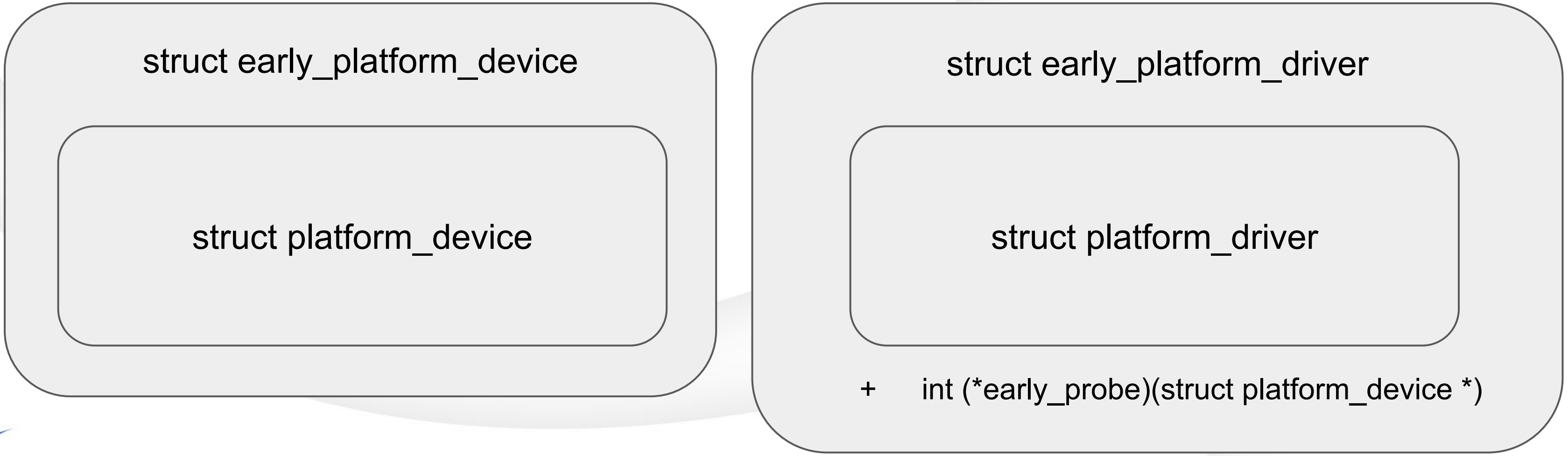
```
commit 043eaa70ad736380a631e820e32ad9176b020887
Author: David Lechner <david@lechnology.com>
Date:   Fri May 25 13:11:49 2018 -0500

    clk: davinci: psc: allow for dev == NULL

    On some davinci SoCs, we need to register the PSC clocks during early
    boot because they are needed for clocksource/clockevent. These changes
    allow for dev == NULL because in this case, we won't have a platform
    device for the clocks.

    Signed-off-by: David Lechner <david@lechnology.com>
    Reviewed-by: Sekhar Nori <nsekhar@ti.com>
    Signed-off-by: Michael Turquette <mturquette@baylibre.com>
    Link: lkml.kernel.org/r/20180525181150.17873-9-david@lechnology.com
```

Early platform drivers AD 2009

- Commit 13977091a988 (“Driver Core: early platform driver”)
- Based on early_param()
- Mostly specific to SuperH arch
- Cumbersome
- Not real platform drivers/devices
  - Not part of the linux kernel device model
  - Uses devres_head to link devices (!)

New idea for early platform drivers

- [PATCH 00/12] introduce support for early platform drivers
  - https://lkml.org/lkml/2018/5/11/488


struct early_platform_device
struct platform_device
struct early_platform_driver
struct platform_driver
+ int (*early_probe)(struct platform_device *)

New idea for early platform drivers

- `early_platform_start()` called by architecture code
- `early_platform_finalize()` called from `postcore_initcall()` seamlessly converts early platform drivers into regular platform drivers
- device resource management available
- device resources and properties
- device logging
- code unification
- deferred probe

Example: dummy early platform driver

New idea for early platform drivers - feedback

- *“I skimmed through this and it doesn't look horrible [...]”* - Rob Herring
- Good fit for a device driver that generally manages memory-mapped system resources that are part of the system glue and not really tied to a specific bus. - Mike Turquette
- *“Clockevents and interrupt controllers can have a module clock.*
  All three can be part of a PM Domain, which requires a struct device to be handled properly.” - Geert Uytterhoeven

New idea for early platform drivers - feedback

- *“They can't be modules. They can't be hotplugged. Can they be runtime-pm enabled?”* - Rob Herring
- *“Doing things earlier is not the only way to solve the problems. Perhaps we need to figure out how to start things later.”* - Rob Herring

New idea for early platform drivers - feedback

- *“You may want to split it because of dependencies. OF_DECLARE doesn't handle EPROBE_DEFER, while some critical parts may be needed early.”* - Geert Uytterhoeven
- *“The fixed probe order imposed by OF_DECLARE() limits this: if your OF_DECLARE() driver depends on something else, the latter must become an early device. If all subsystems would use real devices, EPROBE_DEFER would handle most of it automatically.”* - Geert Uytterhoeven

Questions:

- should we proceed with implementing support for early platform drivers?
- is the example implementation any good?