State of the common struct clk

Mike Turquette
Linaro PM Working Group
What do we have today?

- Common definition of struct clk
- Common implementation of API in `include/linux/clk.h`
- Implementations of basic clock types that are common on many platforms
  - Fixed-rate
  - Gateable
  - Multiplexer
  - Adjustable Divider
What else do we have today?

- Clock rate change notifiers
- Out-of-order initialization and orphan clocks
- Standardized debugfs interface
- Support for statically allocated clocks and dynamically allocated clocks
- Flexible initialization options
What's blocking merge?

- struct clk globally defined
- Platform support
  - OMAP4 work in progress
    - Breaks OMAP2+ single image
  - i.MX5 and i.MX6 fully converted to V4 series
    - Breaks i.MX single image
- Convert your platform, please

- Your reviews and ACKs
  - Who do I send the next series To?
  - arm-soc or linux-next?

mturquette@linaro.org
Linaro PM Working Group
struct clk global definition

• Original series from Jeremy Kerr
  • struct clk defined in drivers/clk.c
  • struct clk_hw defined in include/linux/clk.h
  • Nice abstraction, but did not account for statically initialized clocks during early boot

• Series V3 & V4
  • struct clk defined in include/linux/clk.h
  • Platform folks were happy, porting was easier
  • NACK'd by TGLX since struct clk is too exposed
struct clk global definition, 2
attempt to find middle ground

• Expose struct clk in drivers/clk/clk-private.h
  • Static clock data cannot reside in arch/*
  • Those clocks must reside in drivers/clk/

• Statically initialized platform-specific clocks are problematic
  • The platform-specific clk ops must be accessible from drivers/clk/
  • This is painful for existing complex clock trees
  • Should all platform clock code and clock data live in drivers/clk/?
struct clk definition, 3 best of both worlds

- drivers/clk/clk-private.h is too limited
- Instead create include/linux/clk-private.h
  - With a very large comment at the top warning driver authors not to use that header
- Reinstates original struct clk_hw semantics while not ruling out statically initialized clocks
- Macros in clk-private.h should allay concerns from platform folks over messy forward declarations
API definition issues

• clk_get_rate & clk_get_parent
  • no locking, synchronisation or critical section mechanism
  • clk_block_rate_change / clk_allow_rate_change

• mutex vs spinlock race conditions

• clk_prepare semantics and use
  • clk_enable should be able to block

• clk_ops_can_block: necessary for complex clock locking
Other unanswered questions

• Can you set the rate of a disabled clock?
  • What behavior is expected in this case?

• Should clocks support constraints?
  • Track unique users of the clock and remember their requested rates

• DVFS
  • Should the clock framework be the control mechanism for initiating a DVFS transition?
  • Or should a new API be built on top of the clock framework?
Feedback?