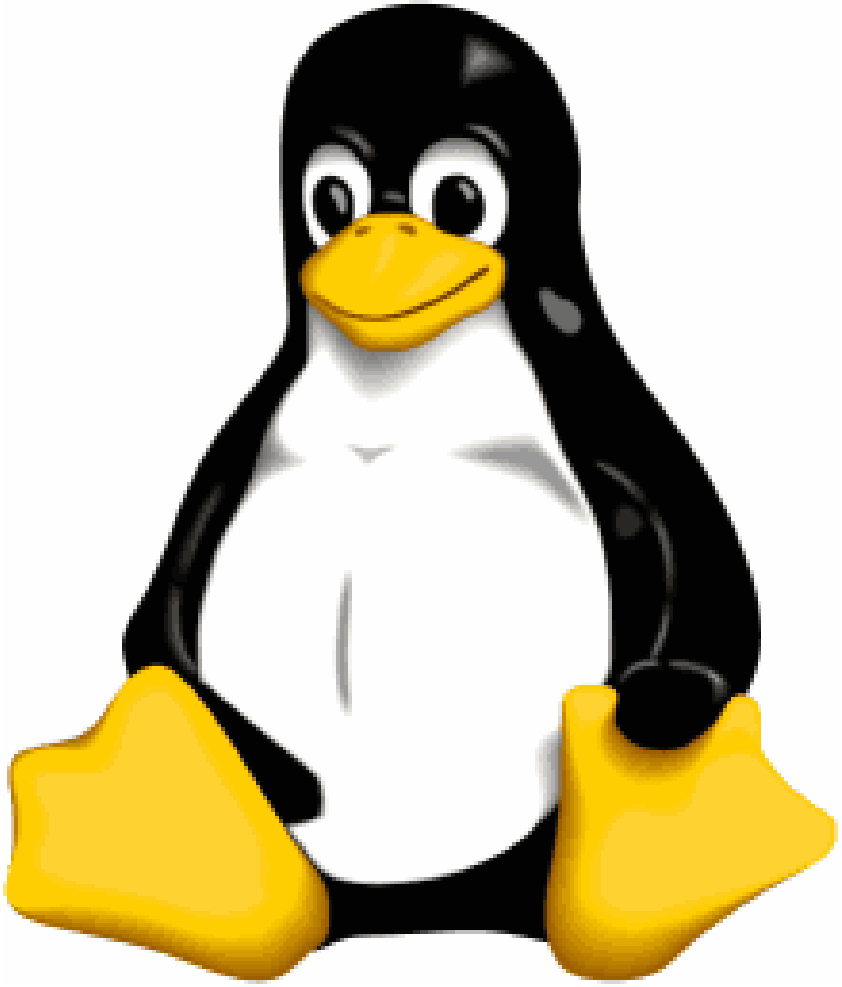
A Generic Clock Framework in the Kernel: Why We Need It & Why We Still Don't Have It

ELC Europe 2011

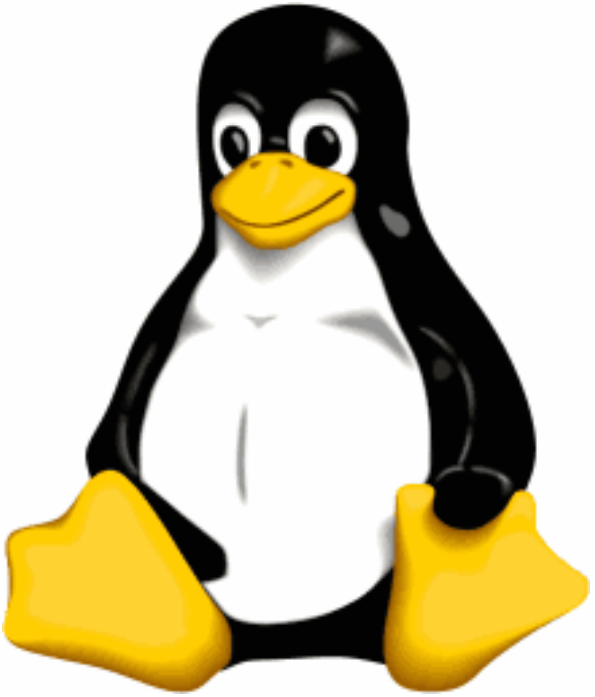
Sascha Hauer
<s.hauer@pengutronix.de>



Pengutronix.

Overview

- What are clocks?
- What do we have now
- Problems
- What are we working on



What are clocks?

- Powermanagement
- Exact Baud/Pixel clocks

Good old days

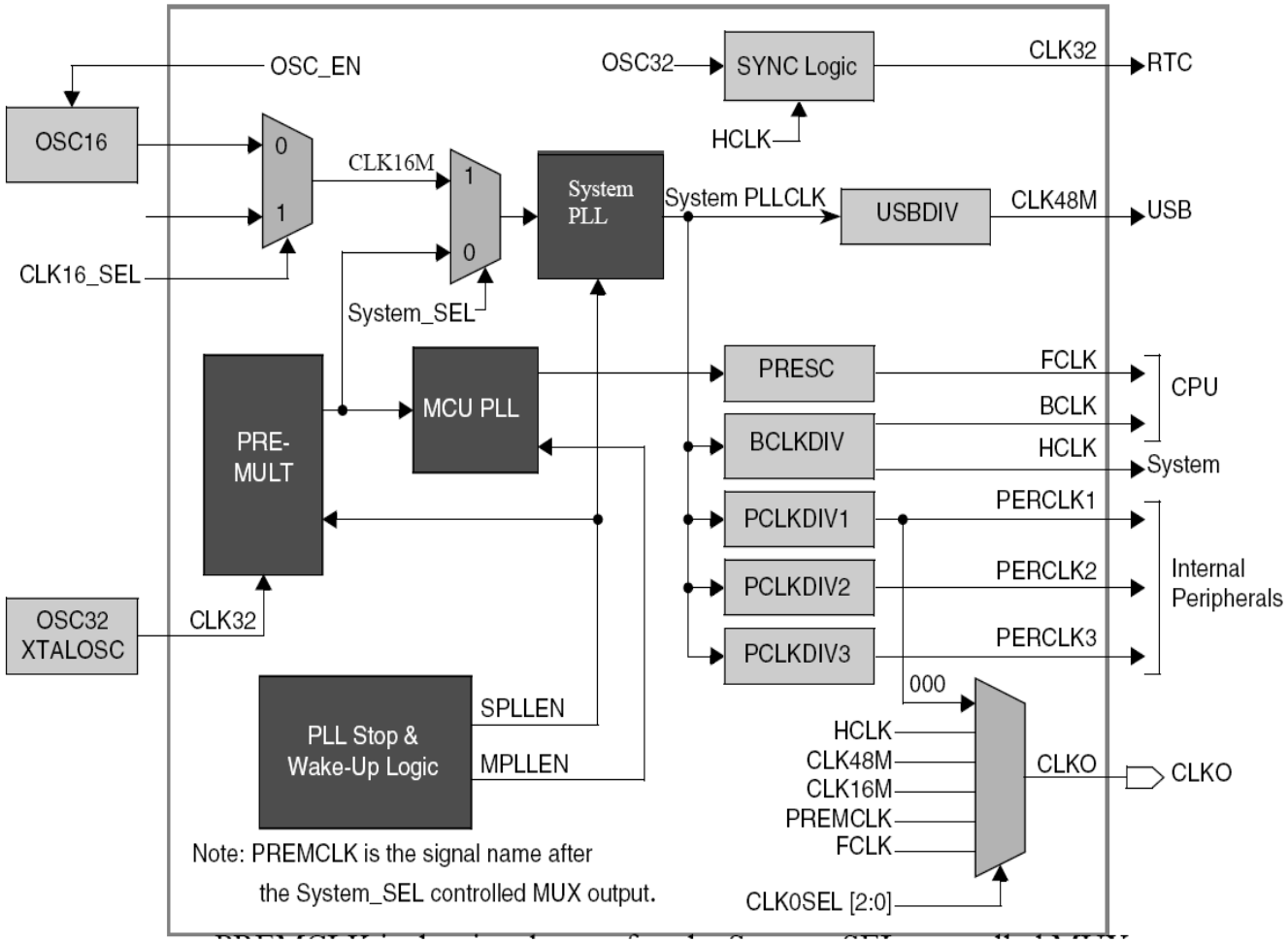
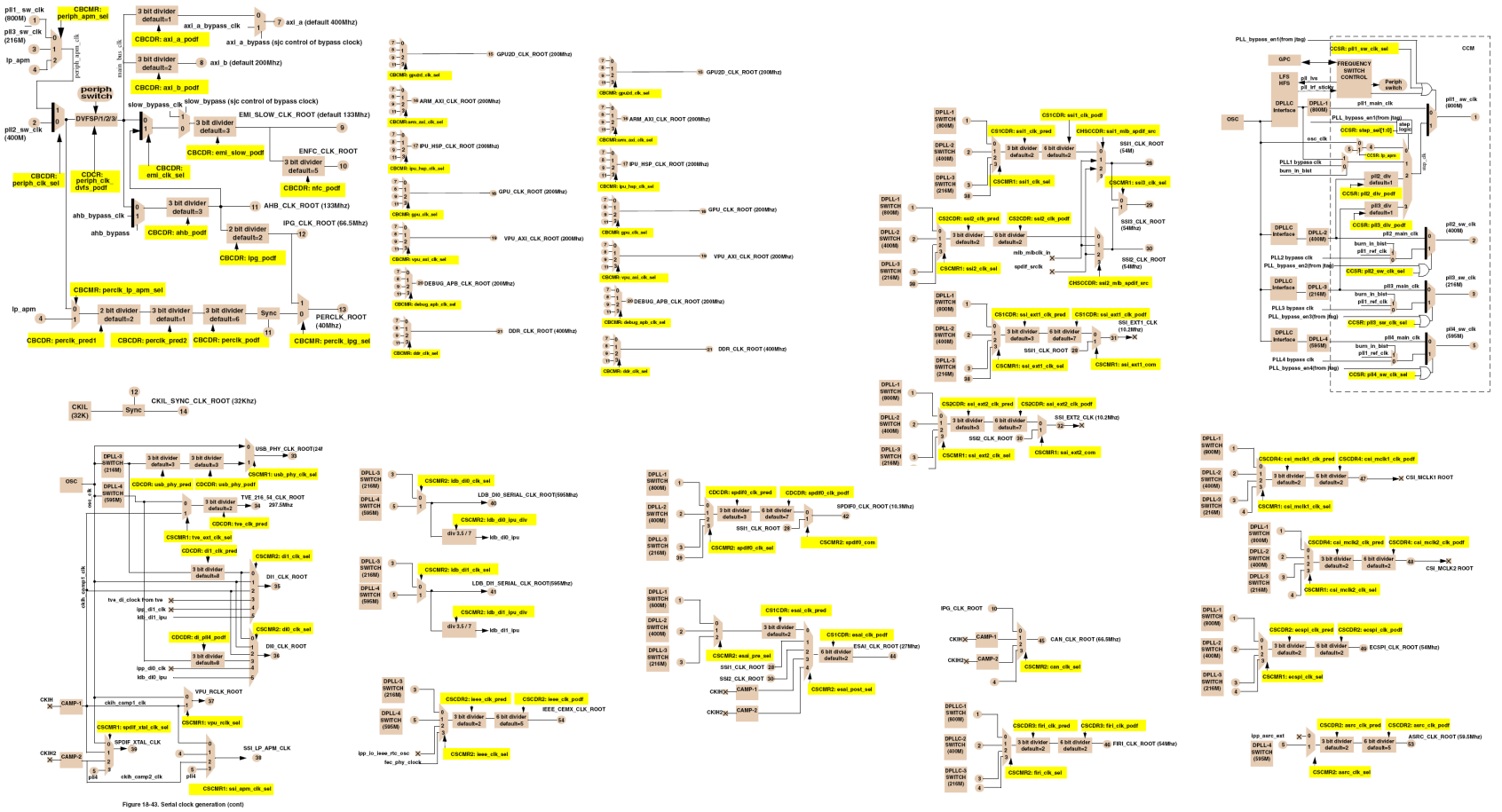


Figure 12-1. Clock Controller Module

Source: Freescale I.MX1 reference manual

Today



Source: Freescale I.MX53 reference manual

What do we have now? The API

Getting clocks:

- struct clk *clk_get(struct device *dev, const char *id);
- void clk_put(struct clk *clk);

Enabling/disabling:

- int clk_enable(struct clk *clk);
- void clk_disable(struct clk *clk);

Rate functions:

- unsigned long clk_get_rate(struct clk *clk);
- int clk_set_rate(struct clk *clk, unsigned long rate);
- long clk_round_rate(struct clk *clk, unsigned long rate);

Parent functions:

- int clk_set_parent(struct clk *clk, struct clk *parent);
- struct clk *clk_get_parent(struct clk *clk);

Problems

There are currently 37 implementations of the clock API in the tree!

This means:

- 37 times reinventing the wheel
- 37 ways to have buggy implementations
- 37 different struct clk
- ~60000 lines of code
- code and data mixed
- No way to build Kernels for multiple SoCs
- No way to implement off-SoC clocks

Our way out. History

Jeremy Kerr: A common struct clk

- Unify the different implementations of struct clk
- A single implementation for clock functions
- Create a common debugfs entry for clocks

Locking

- For maximum power saving clk_enable/clk_disable shall be usable in Interrupt context
- Most implementations use spinlocks
- Slow PLLs and clocks behind I2C busses cannot be controlled with spinlocks held
- The solution: clk_prepare/clk_unprepare

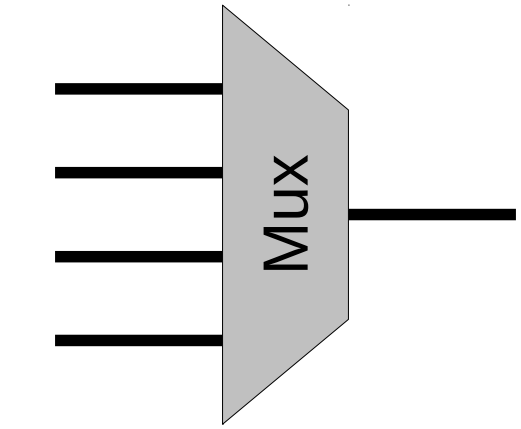
clk_prepare() / clk_unprepare()

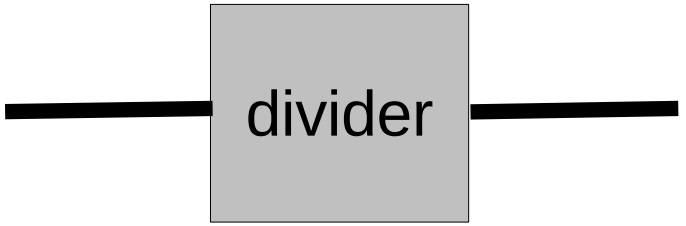
	Fast Clock	Slow Clock
clk_prepare()		enable()
clk_enable()	enable()	
clk_disable()	disable()	
clk_unprepare		disable()

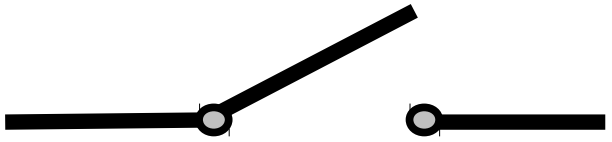
Locking

- Jeremys patches used a lock per clock
- Too complex

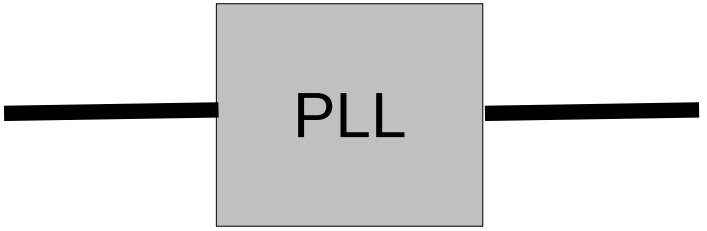
Building blocks



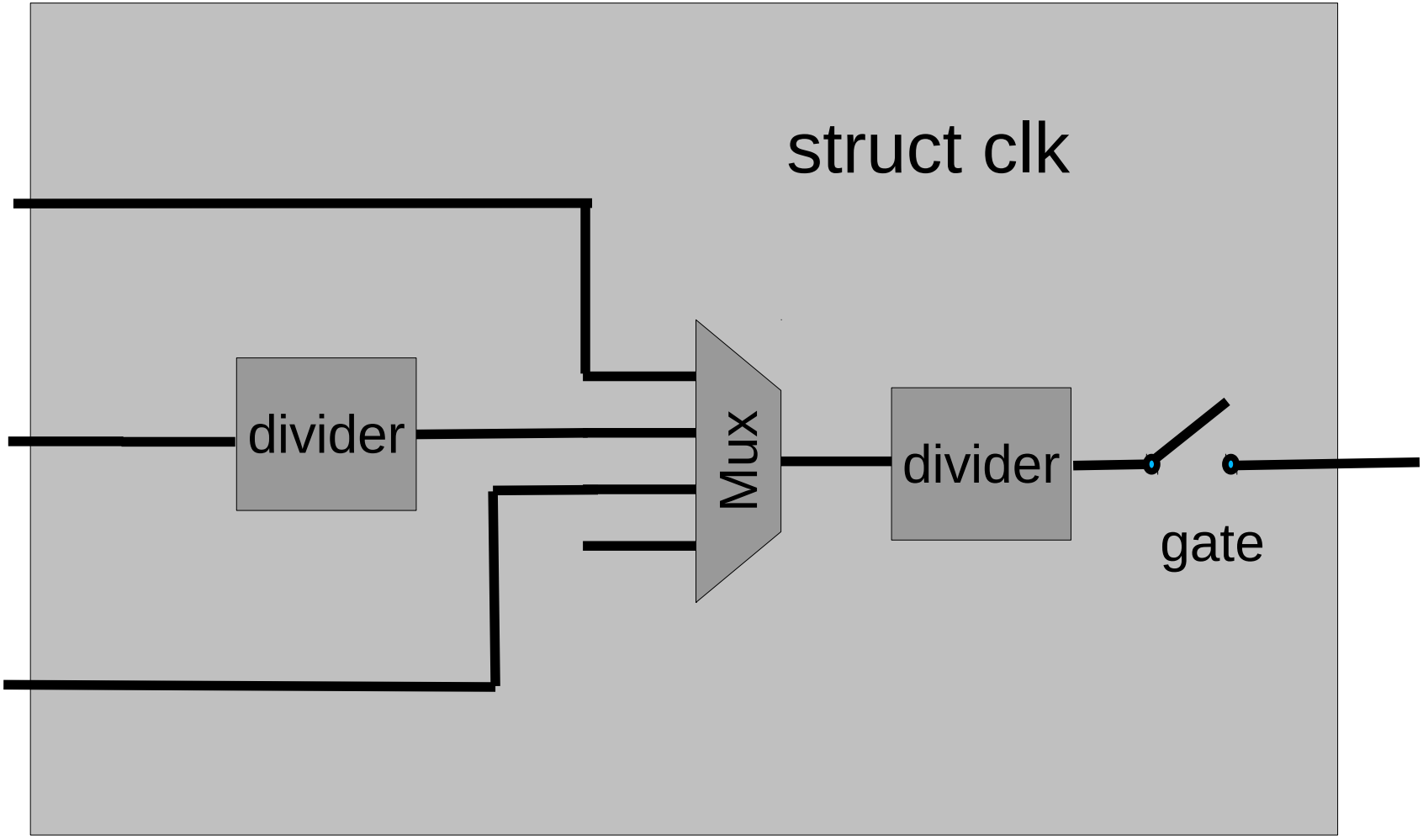




gate



How we implement this today



Pengutronix.

How we implement this today

- Complex structure
- Layout is unknown to the kernel
- Hard to represent as data
- Makes for huge and hard to change SoC specific files

Building blocks

- Each block is simple with one or few implementation
- Each block can be represented as few bytes of data
- code for building blocks can be shared across architectures
- Code and data can be seperated

Towards a generic clock framework

- The tree layout belongs to the core and nowhere else
- Struct clk shall be opaque to both users and implementers of a clock
- Clocks need to be registered

Initialization

- The clock tree layout has to be initialized and synchronized with the hardware

  and

- Boards want to configure dividers and muxes for their needs

The devicetree

- The devicetree is ideal to describe the exact layout (including registers) of the clocktree
- The devicetree makes it easy to associate clocks with their users
- The devicetree can be used to both describe the clock tree and the static configuration

The devicetree

Board.dts

```
/include/ “SoC.dtsi”

...

clocks {
    uart_divider:uart_divider {
        divider-value = <3>;
    };
};

...
```

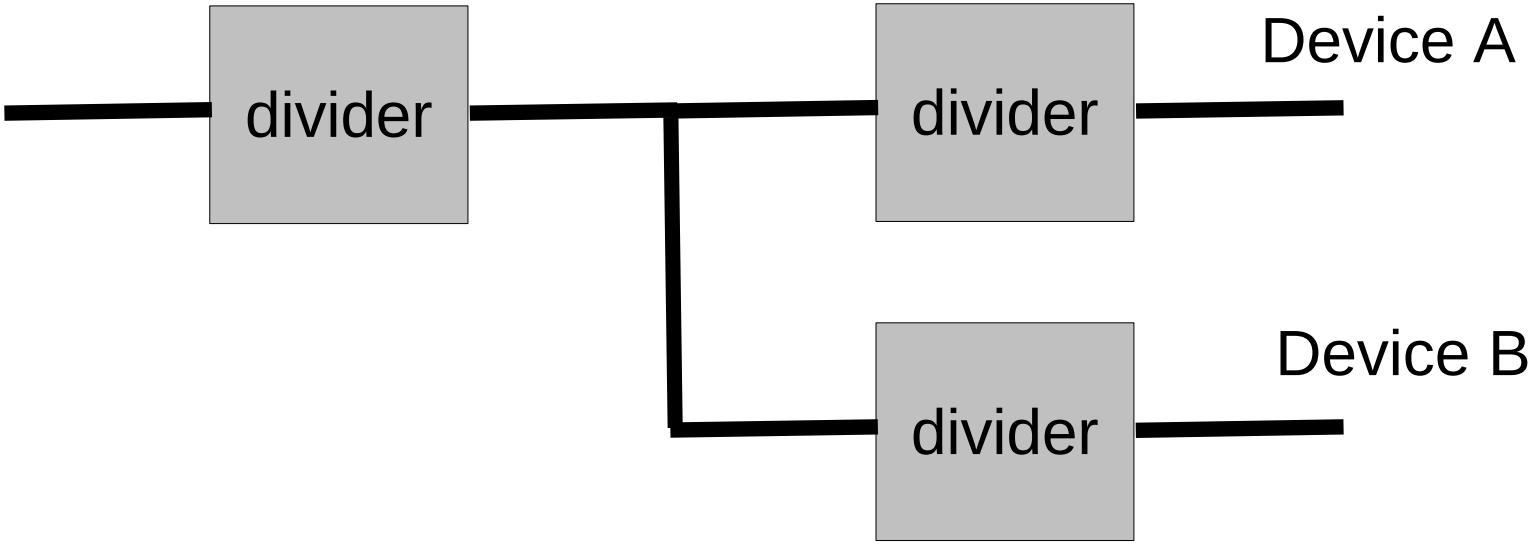
SoC.dtsi

```
...

clocks {
    uart_divider:uart_divider {
        reg = <0x30 0x0>;
        clk-parent = <&other_clk>;
        shift = <13>
        width = <4>
    };
};

...
```

Pengutronix.

The rate propagation problem

- How do we make sure unrelated clocks are not affected during clk_set_rate()



Clocks and DVFS, ...

- Yes, we can do....

  But

- Let's not mix this with clock stuff

Conclusion

- Locking problems solved
- Patches on Mailing list are in good shape and mostly agreed upon
- Still need to implement drivers for dividers, muxes and PLLs
- Devicetree bindings have to be fixed
- Rate propagation has to be worked upon
- Still need to mass convert the SoC Clock trees

Questions?