THE LINUX FOUNDATION OPEN SOURCE SUMMIT EUROPE

Embedded Linux Conference Europe

Baylibre

What The Clock !

Linux Clock subsystem internals

THE LINUX FOUNDATION

Who ?

- 14y Kernel & Firmware Hacker
  - Entirely Ported Linux on custom ARM SoCs
  - Worked with SoC design team
- 5y BayLibre Engineer
  - Writes support for Amlogic Mainline Linux & U-Boot
- 3y1/2 Amlogic Clock driver Contributor/Co-Maintainer

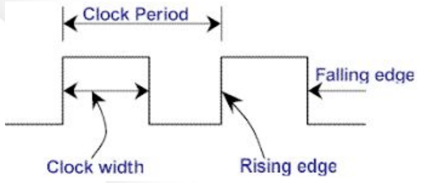
What The Clock !

- Hardware
- Software
  - Clock in Linux
  - Clock framework is a library
  - Clock framework and drivers
  - Clock framework and device tree
- Clock framework limitations

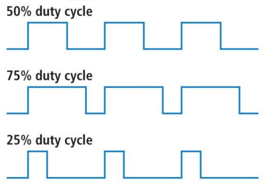
Hardware

Hardware

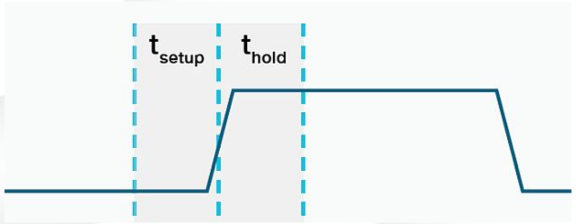
- Clock signal has a width, period => frequency



- Clock signal has a duty cycle

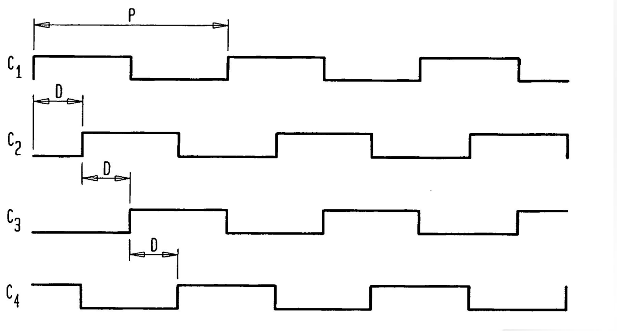


- Clock signal has setup & hold times

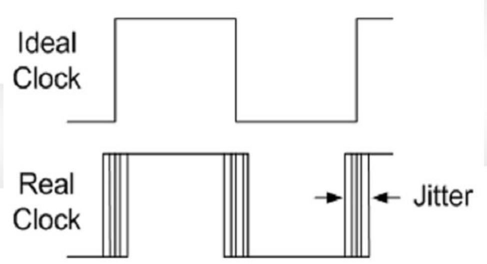


Hardware

- When multiple clocks, they can have different phases



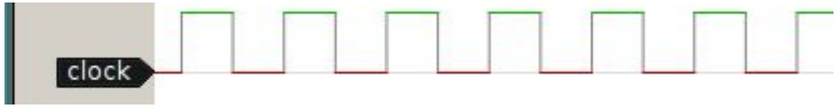
- And Jitter



Hardware

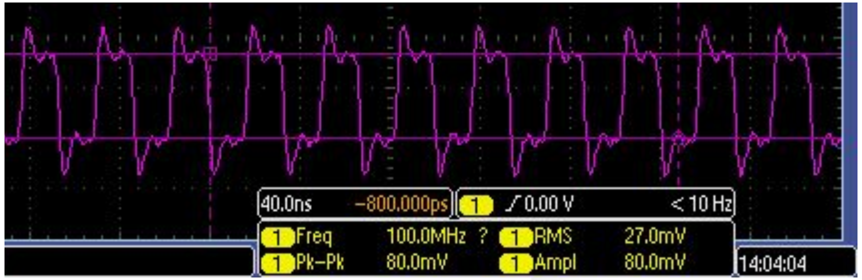
EXPECTATION...

Clock seen in simulation or logic analyzer



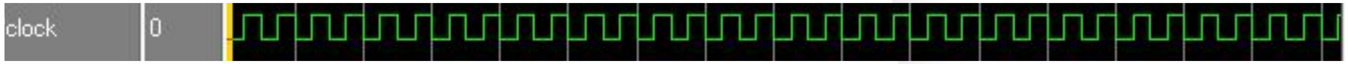
REALITY...

Clock seen on an oscilloscope



Hardware

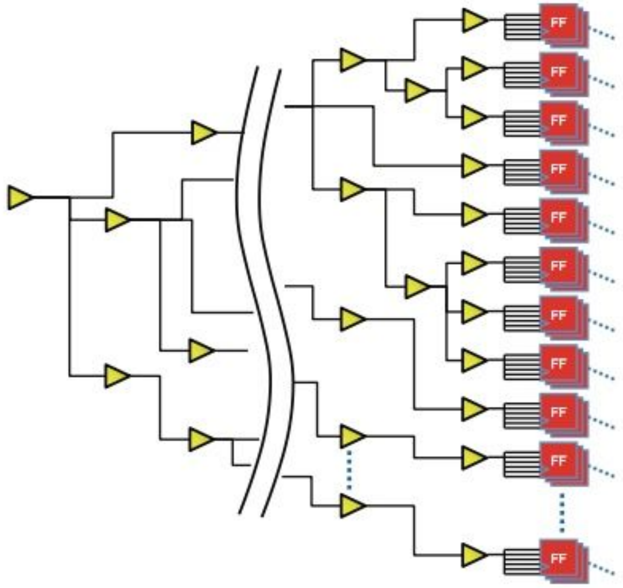
In an electronic system, the clock is the heartbeat



Everything is synchronous toward a clock

System often takes an external clock as source

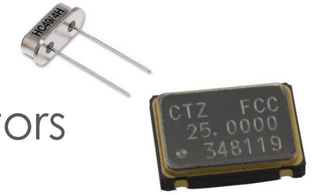
And generate a tree of clocks for all functions



Hardware

In order to generate and propagate clock into the system

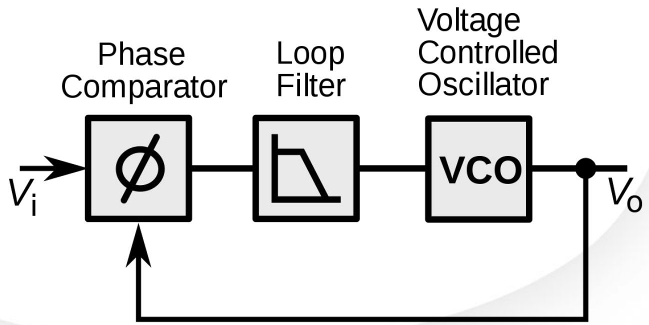
- Cristal
- Oscillators
- PLLs
- Dividers
- Gates
- Muxes
- Clock synchronization
- ...



Hardware

In order to generate and propagate clock into the system

- PLLs

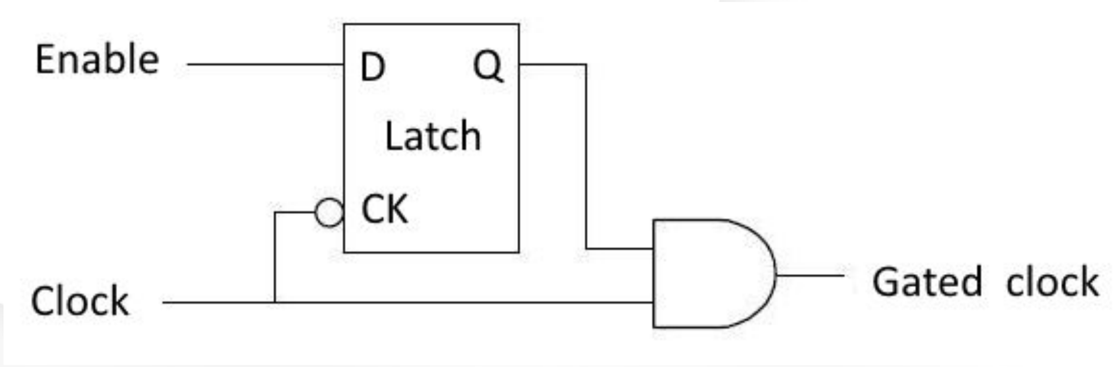


https://en.wikipedia.org/wiki/Phase-locked_loop

Hardware

In order to generate and propagate clock into the system

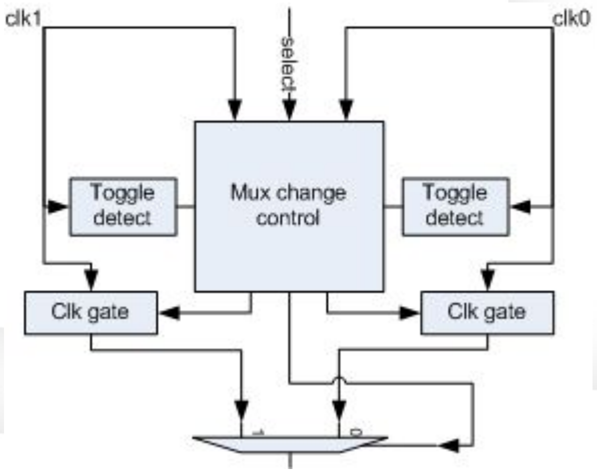
- Gate



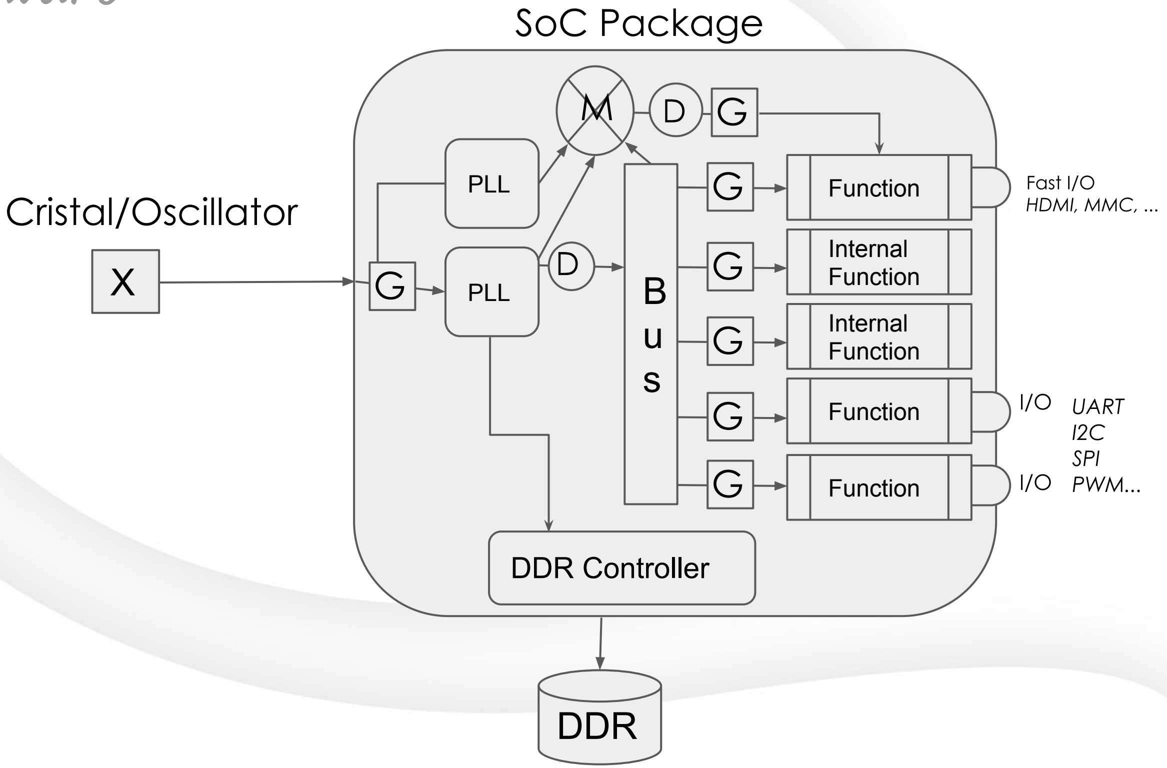
Hardware

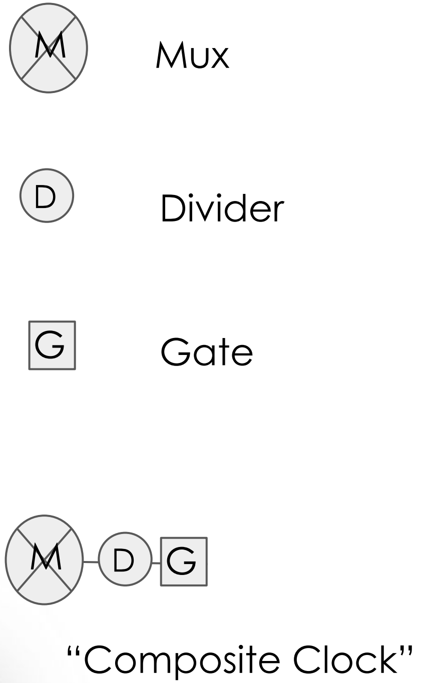
In order to generate and propagate clock into the system

- Digital Glitch-free Mux



Hardware


SoC Package
Cristal/Oscillator
X
G
PLL
PLL
D
M
D
G
B u s
G
G
G
G
G
Function
Internal Function
Internal Function
Function
Function
Fast I/O
HDMI, MMC, ...
I/O
I/O
UART
I2C
SPI
PWM...
DDR Controller
DDR


M
Mux
D
Divider
G
Gate
M
D
G
"Composite Clock"

Software

Clock in linux

Historically, Linux drivers only managed clocks by their frequency for:

- CPU speed
- External Bus speed (I2C, SPI, UART, …)
- Video pixel frequency

But each driver managed this on their side.

Clock in linux

Started with `arch/arm/mach-integrator/clock.h` (Jun 18, 2004) :

```
struct clk {

        struct list_head    node;
        unsigned long       rate;
        struct module       *owner;
        const char          *name;
        const struct icst525_params *params;
        void                *data;
        void                (*setvco)(struct clk *, struct icst525_vco vco);
};
int clk_register(struct clk *clk);
void clk_unregister(struct clk *clk);
```

Clock in linux

And became linux/include/linux/clk.h (Jan 7, 2006):

```
/*
 * struct clk - an machine class defined object / cookie.
 */
struct clk;
struct clk *clk_get(struct device *dev, const char *id);
int clk_enable(struct clk *clk);
void clk_disable(struct clk *clk);
unsigned long clk_get_rate(struct clk *clk);
void clk_put(struct clk *clk);
long clk_round_rate(struct clk *clk, unsigned long rate);
int clk_set_rate(struct clk *clk, unsigned long rate);
int clk_set_parent(struct clk *clk, struct clk *parent);
struct clk *clk_get_parent(struct clk *clk);
```

Clock in linux

Covered most of the clock management needs,
BUT, each platform needed to fill these function accordingly.

```
arch/arm/mach-aaec2000/clock.c
arch/arm/mach-integrator/clock.c
arch/arm/mach-omap1/clock.c
...
```

AND The platform's driver used them according to the implementation.

```
arch/arm/mach-omap1/serial.c
drivers/i2c/busses/i2c-s3c2410.c
...
```

Clock in linux

each platform needed to fill these function accordingly

.... Not equaly :

`arch/arm/mach-integrator/clock.c :`

```
int clk_enable(struct clk *clk)

{
        return 0;
}
EXPORT_SYMBOL(clk_enable);


void clk_disable(struct clk *clk)
{
}
EXPORT_SYMBOL(clk_disable);
```

Clock in linux

Some platform did a complete implementation (omap),

And even added some more platform specific functions (omap):

```
int clk_use(struct clk *clk)
void clk_unuse(struct clk *clk)
int clk_get_usecount(struct clk *clk)
void clk_deny_idle(struct clk *clk)
void clk_allow_idle(struct clk *clk)
```

Clock in linux

So there was often a clash for multi-platform drivers
like Generic IPs (network, i2c, ...):

- Wrong API usage/behavior
- Usage of platform specific extensions, or custom implementation
- Adding of fake clock to satisfy driver (*Yeah I did it* ©)
- Duplication of clock logic
  - Rate calculation
  - Rate propagation
  - Optimal Parenting

Clock framework is a library

To solve the inconsistency of clk.h implementation
Mike Turquette introduced the Common Clock Framework (March 2012):

The common clock framework defines a common struct clk useful across most platforms as well as an implementation of the clk api that drivers can use safely for managing clocks.

The net result is **consolidation of many different struct clk definitions** and platform-specific clock **framework implementations.**

This patch introduces the common struct clk, struct clk_ops and an **implementation of the well-known clock api** in include/clk/clk.h.

Platforms may define their own hardware-specific clock structure and their own clock operation callbacks, so long as it wraps an instance of struct clk_hw.

Clock framework is a library

To solve the inconsistency of clk.h implementation
Mike Turquette introduced the Common Clock Framework (March 2012):

`TL:DR`

Drivers are responsible for populating the framework with clock tree topology and plugging in the ops physically program the hardware.

Clock framework is a library

With the library, clock controller provides clk_ops for each clock with:

```
struct clk_ops {
        int                 (*prepare)(struct clk_hw *hw);
        void                (*unprepare)(struct clk_hw *hw);
        int                 (*enable)(struct clk_hw *hw);
        void                (*disable)(struct clk_hw *hw);
        int                 (*is_enabled)(struct clk_hw *hw);
        unsigned long       (*recalc_rate)(struct clk_hw *hw,
                                    unsigned long parent_rate);
        long                (*round_rate)(struct clk_hw *hw, unsigned long,
                                    unsigned long *);
        int                 (*set_parent)(struct clk_hw *hw, u8 index);
        u8                  (*get_parent)(struct clk_hw *hw);
        int                 (*set_rate)(struct clk_hw *hw, unsigned long);
        void                (*init)(struct clk_hw *hw);
};
```

Clock framework is a library

Only necessary *ops* were passed to `clk_register()`.

- Gates: `enable/disable/is_enabled`
- Dividers: `recalc_rate/round_rate/set_rate`
- Muxes: `set_parent/get_parent`
- PLLS: `enable/disable/is_enabled/recalc_rate/round_rate/set_rate`

And `prepare/unprepare/init` were mandatory.

Clock framework is a library

With all these ops provided, the framework:

- Builds a clock tree with the parents list of each lock
  - The current parent is cached
- Calculates a rate *per-clk* by walking the tree
  - The current rate is cached
- On rate setting/calculation
  - The tree is walked recursively to closely match the request
  - When possible rate is the closest, re-parenting is done
- Enable/Disable propagates from leaf clock to root clocks
  - Each clock has an internal clock enable/request counter

Clock framework is a library

The Common Clock Framework has evolved over time, adding:

- Clock notifier
- DT support
- Clock accuracy support (in parts per billion)
- Clock phase support (in degrees)
- Clock duty cycle support (in numerator/denominator ratio)
- Clock exclusivity (keep clock rate/… exclusive to a consumer)
- `set_rate` variants (range, min, max)

Clock framework and device tree

In pre-DT times:

- device <-> clock mapping was fixed
- "/arch/*/mach-*" code statically linked devices and clocks.
- clocks were associated to the "device" structure.

Link between clock output to clock input between controllers and drivers was blurry, *often not described at all*.

Clock framework and device tree

DT *provides a way to link a clock output to a clock input.*

The Common Clock Framework works across the system:

- Can link clocks between clock controllers
- Can link clocks between devices
- Can link clock between devices and clock controllers

All this was impossible/very complex before DT.

Clock framework and device tree

With Device Tree, it's possible to:

- Declare multiple clock providers
  - Controllers
  - Simple clocks (cristal/oscillators)
  - Clocks provided by devices
  - Special clock (PWM clocks)
- Link clocks between devices
- Set clock parenting/rate constraints from DT

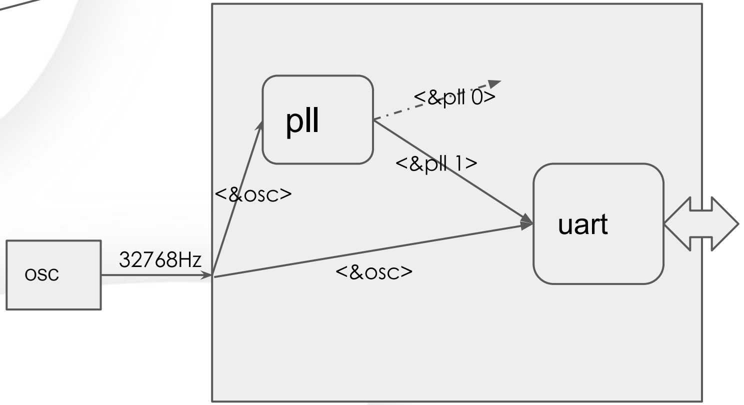
Clock framework and device tree

Example:

```
/* external oscillator */
osc: oscillator {
    compatible = "fixed-clock";
    #clock-cells = <0>;
    clock-frequency  = <32678>;
    clock-output-names = "osc";
};

/* phase-locked-loop device, generates a higher frequency clock
 * from the external oscillator reference */
pll: pll@4c000 {
    compatible = "vendor,some-pll-interface"
    #clock-cells = <1>;
    clocks = <&osc>;
    clock-names = "ref";
    reg = <0x4c000 0x1000>;
    clock-output-names = "pll", "pll-switched";
};
```

```
/* UART, using the low frequency oscillator for the baud clock,
 * and the high frequency switched PLL output for register
 * clocking */
uart@a000 {
    compatible = "vendor,some-uart";
    reg = <0xa000 0x1000>;
    interrupts = <33>;
    clocks = <&osc>, <&pll 1>;
    clock-names = "baud", "register";
};
```



Clock framework limitations

- Clock controllers implementation is heterogenous
- Clock tree walking is recursive
- Doesn’t handle some now important properties:
  - Jitter, PLL filters, ...
- Firmware handled/needed clocks is badly handled
  - No way to properly describe them
- Clock handoff mechanism from firmware to device is missing
- Dynamic clock path prioritization is missing
  - Often HW engineers design specific clock paths for use-cases
  - For example: HDMI 2.0 4k60 clock needs a very clean clock path

Thanks !

Questions ?