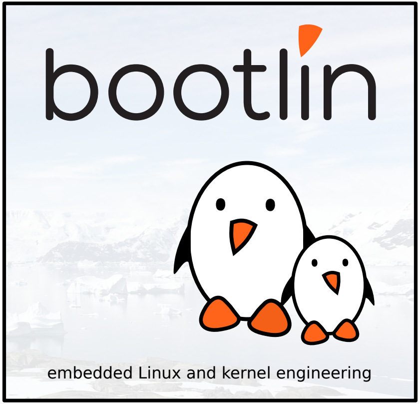
An introduction to the Linux DRM subsystem

Maxime Ripard
maxime@bootlin.com



Maxime Ripard

- Embedded Linux engineer and trainer at Bootlin
  - Embedded Linux **development**: kernel and driver development, system integration, boot time and power consumption optimization, consulting, etc.
  - Embedded Linux **training**, Linux driver development training and Android system development training, with materials freely available under a Creative Commons license.
  - `http://bootlin.com`
- Contributions
  - **Co-maintainer for the sunXi SoCs** from Allwinner
  - Contributor to a couple of other open-source projects, **Buildroot**, **U-Boot**, **Barebox**
- Living in **Toulouse**, south west of France

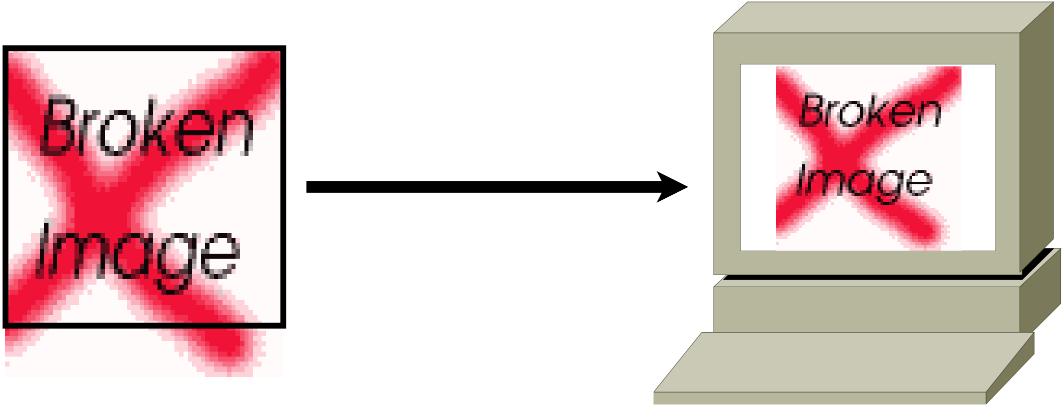
Introduction

A long long time ago, in a galaxy (not so) far, far away

Framebuffers

- Display hardware was dead simple...
- .. and so was the API to drive it.
- Introducing... fbdev!
- Allows for three things:
  - Mode-Setting
  - Accessing the (only) buffer
  - Optional 2d acceleration: draw, copy, etc.
  - And access to the device registers...

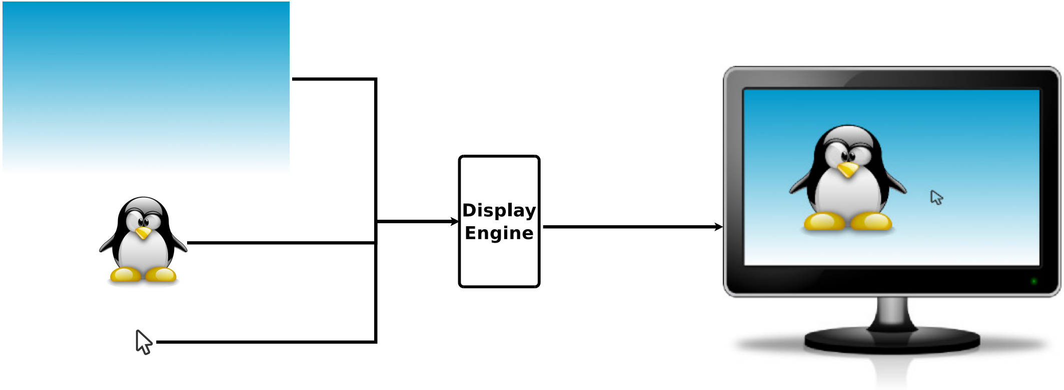
That 90's show



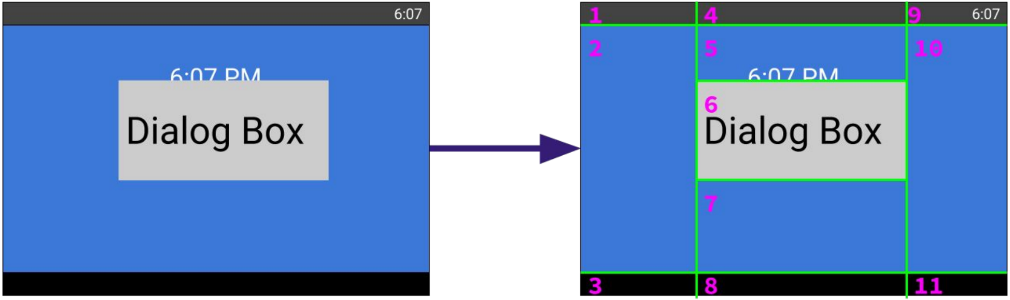
Back to the future

- Two different trends
  - Embedded devices starting to show up, with their low power needs $\Rightarrow$ Display engines need to accelerate more things
  - Desktop displays getting more and more fancy in order to play Quake in 4k VR $\Rightarrow$ Bigger and bigger GPUs
- Led to two different outcomes:
  - Interface to drive GPU devices through the kernel: DRM
  - Hacks piling on in order to fit embedded use-cases: omapdss, pxafb

Composition Evolved



The droid you're looking at



Sean Paul and Zach Reizner - Google - XDC2016

Can I talk to the manager?

- DRM was initially introduced to deal with the GPU's need
  - Initialize the card, load its firmware, etc.
  - Share the GPU command queue between multiple applications
  - Manage the memory (allocation, access)
  - But not modesetting!
- All the modesetting was in the userspace, especially in X
- Race between rendering and modesetting
- Only one graphical application that needed to remain there all the time
- (Lack of) Abstraction!
- Introduction of the Kernel Mode-Setting (KMS) to move the modesetting back into the kernel

Kill it with fire!

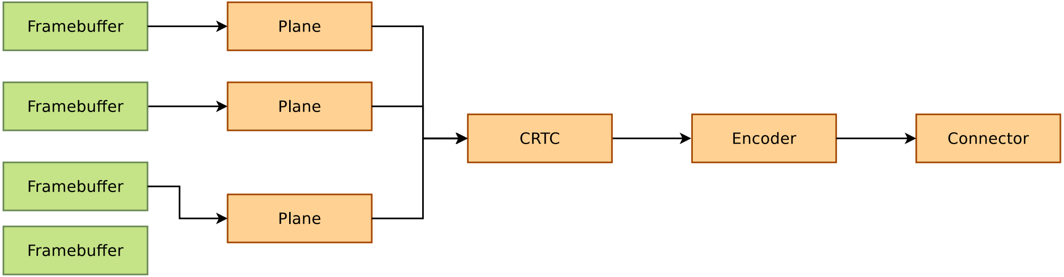
- Now, fbdev could be implemented on top of KMS...
- ... Or removed entirely
- Call for deprecation in 2012 (Hi Laurent!)
- Last fbdev driver merged in 2014
- First ARM DRM driver: exynos in 2011
- Followed: arm, armada, atmel-hclcdc, fsl-dcu, hisilicon, imx, mediatek, meson, msm, mxsfb, omapdrm, pl111, rcar-du, rockchip, shmobile, sti, stm, sun4i, tegra, tve200, etc...

Two nodes to go please

- Initially, DRM was created for devices that were both displaying and rendering (your traditionnal PC graphics card).
- On embedded devices, it's never really been like that
  - the GPU is discrete and comes from a third party
  - the display engine is usually designed by the SoC vendor
- DRM and KMS APIs requiring the same level of privilege, with one master, and were both exposed on the same device file
- Creation of render nodes
- Also useful for things like GPGPU, off-screen rendering, more flexible access control

DRM/KMS

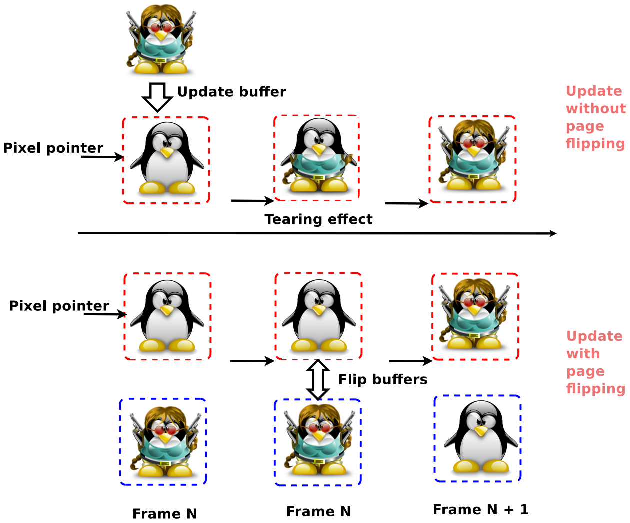
The pixels must flow



KMS components

- Planes
  - Image source
  - Associated with one (or more!) framebuffers
  - Holds a resized / cropped version of that framebuffer

Page flipping



KMS components

- Planes
  - Image source
  - Associated with one (or more!) framebuffers
  - Holds a resized / cropped version of that framebuffer
- CRTCs
  - Take the planes, and does the composition
  - Contains the display mode and parameters

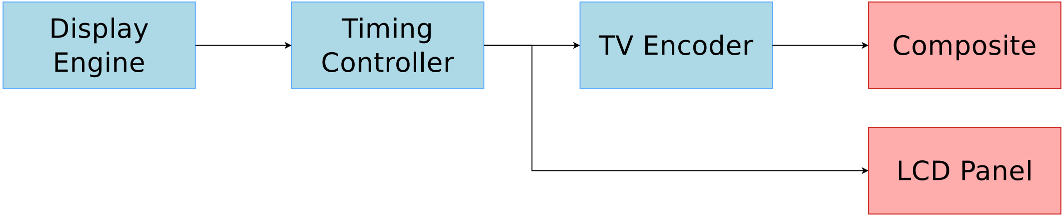
KMS components

- Planes
  - Image source
  - Associated with one (or more!) framebuffers
  - Holds a resized / cropped version of that framebuffer
- CRTCs
  - Take the planes, and does the composition
  - Contains the display mode and parameters
- Encoders
  - Take the raw data from the CRTC and convert it to a particular format

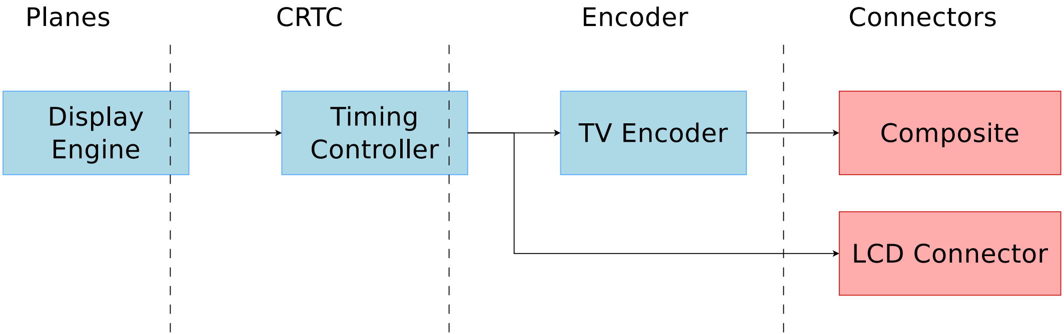
KMS components

- Planes
  - Image source
  - Associated with one (or more!) framebuffers
  - Holds a resized / cropped version of that framebuffer
- CRTCs
  - Take the planes, and does the composition
  - Contains the display mode and parameters
- Encoders
  - Take the raw data from the CRTC and convert it to a particular format
- Connectors
  - Outputs the encoded data to an external display
  - Handles hotplug events
  - Reads EDIDs

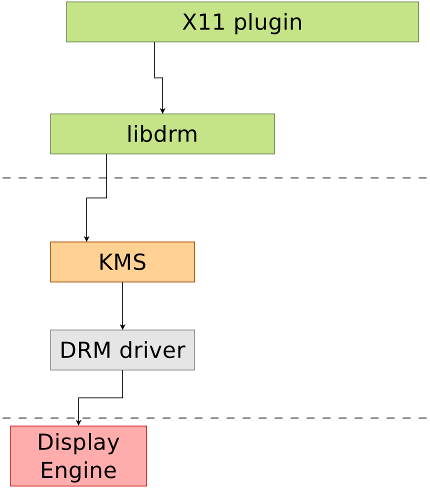
Allwinner display pipeline



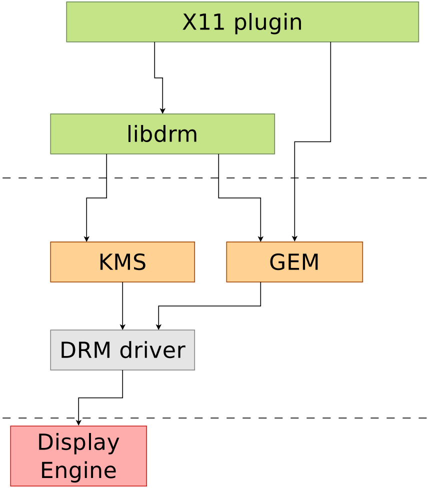
DRM vs SoC pipeline



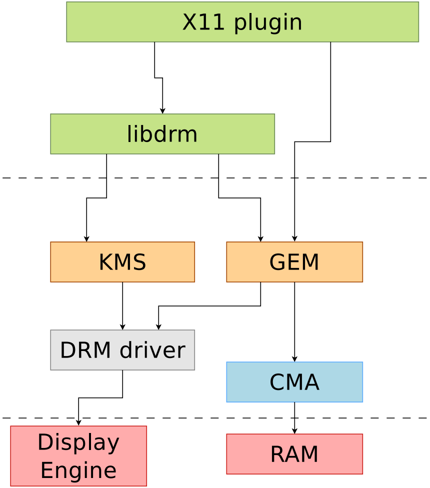
DRM Stack: KMS



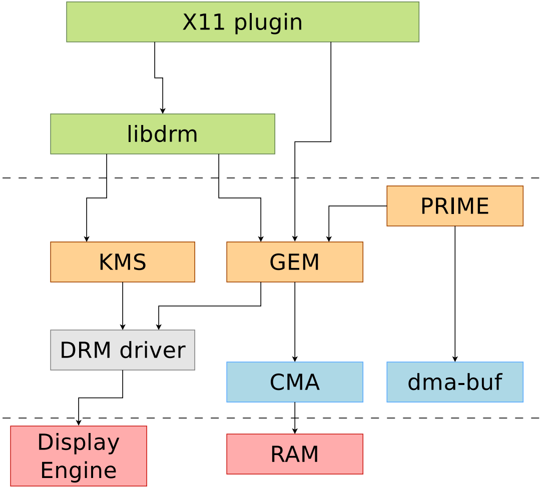
DRM Stack: GEM



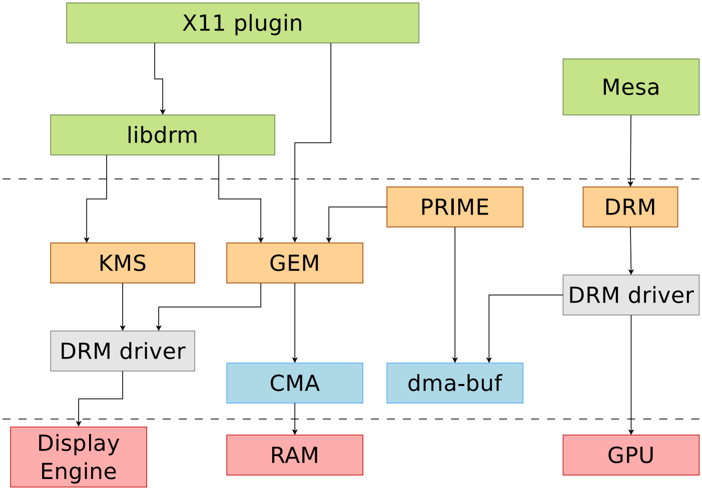
DRM Stack: CMA



DRM Stack: PRIME



DRM Stack: GPUs

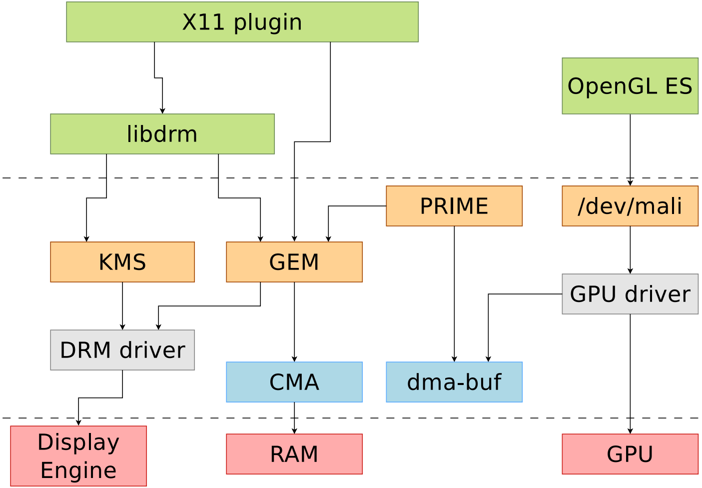


Vendor solutions...

Solutions

- The GPU found in most Allwinner SoCs is the Mali-400 from ARM (with a variable number of cores)
- There are two options to support that GPU:
  - Lima
    - Reversed engineered proof-of-concept
    - Triggered the reverse engineering effort of the GPUs (freedreno, etnaviv, etc.)
    - Development (close to?) stopped three years ago, and then resumed a couple of monthes ago
  - ARM-Provided support
    - Featureful
    - Two parts: GPL kernel driver and proprietary OpenGL ES implementation

DRM Stack: GPU



Development

- Everything is provided by ARM on their website (if you're lucky)
- On the userspace side, you just need to put the library they provided on your system
- On the driver side, you need to create a platform glue that will deal with:
  - Memory mapping
  - Interrupts
  - Clocks
  - Reset lines
  - Power Domains
  - Basically everything needed for the GPU to operate properly on your SoC

X11 integration

- We need a DDX (Device Dependent X) driver
- `xf86-video-modesetting` is working on top of KMS and GBM (MESA-defined user-space API to allocate buffers)
- ARM developped `xf86-video-armsoc` for SoC using a 3rd party GPU (Mali, PowerVR, Vivante, etc.)
- Relies on KMS for the display configuration, driver-specific ioctl for buffer allocations and vendor-provided OpenGL ES implementation
- Just have to write a small glue to use your driver allocator, and give some hints to X about what your hardware support (hw cursor, vblank, etc.)

Questions? Suggestions? Comments?

Maxime Ripard

maxime@bootlin.com

http://bootlin.com/pub/conferences/2017/kr/ripard-drm