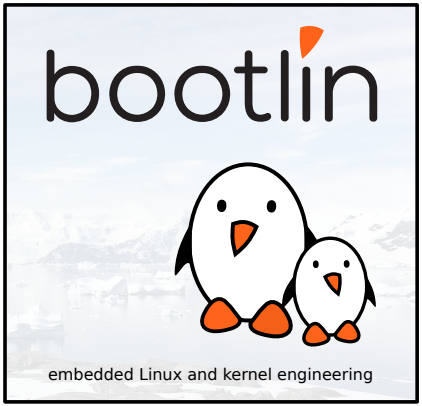
Supporting Hardware Codecs in a Linux system

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, Linux driver development, Yocto Project / OpenEmbedded and Buildroot **training**, with materials freely available under a Creative Commons license.
  - `https://bootlin.com`
- Contributions
  - **Co-maintainer of the sunXi SoCs** from Allwinner in **Linux** and **U-Boot**
  - **Co-maintainer of drm-misc** in **Linux**
  - Contributor to a couple of other open-source projects, **Buildroot**, **U-Boot**, **Barebox**
- Living in **Toulouse**, south west of France

Introduction to Video Decoding

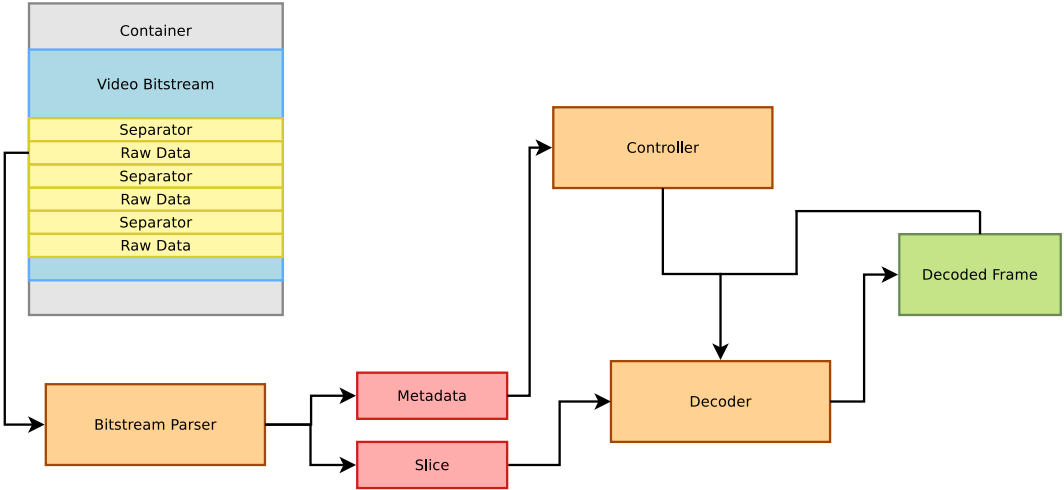
Video Decoding

- An encoded video is basically two things:
  - A container, which will organize the video and audio streams, and the metadata
  - A codec, which will encode a given audio or video stream
- We're only interested in the (video) codec here
- With a codec, the video stream is encoded into a bitstream

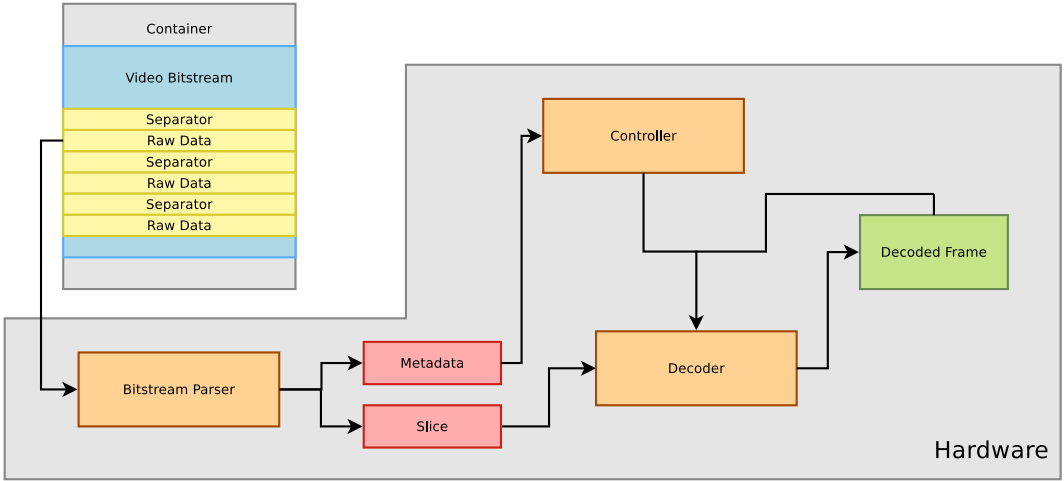
Bitstream

- *A bitstream is a sequence of bits.* – Wikipedia
- In the context of codecs, it refers to the compressed output of the encoder, the at-rest data
- It's mainly composed of a few things:
  - Separators between encoded data
  - Metadata holding the compression parameters
  - Slices, buffers holding the compressed data

Decoder


Container
Video Bitstream
Separator
Raw Data
Separator
Raw Data
Separator
Raw Data
Controller
Decoded Frame
Bitstream Parser
Metadata
Slice
Decoder

Stateful Codec

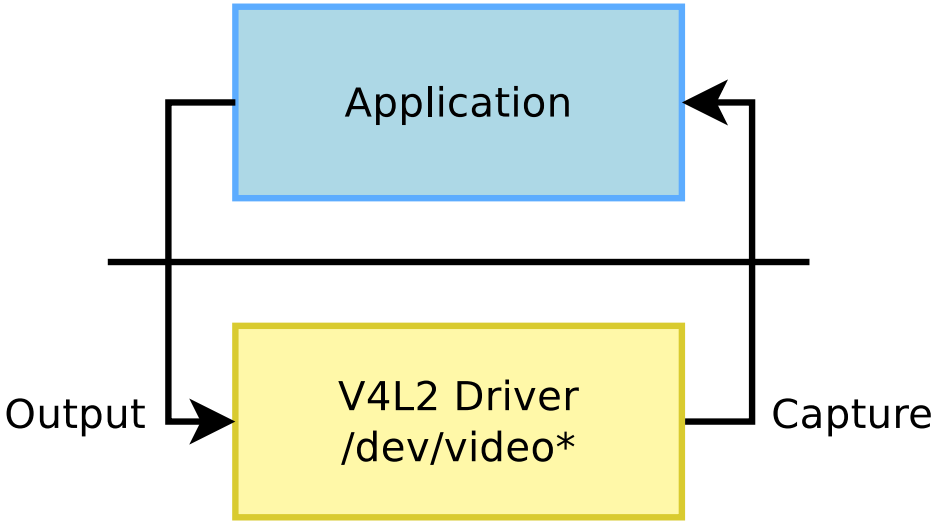


Video Decoding in Linux

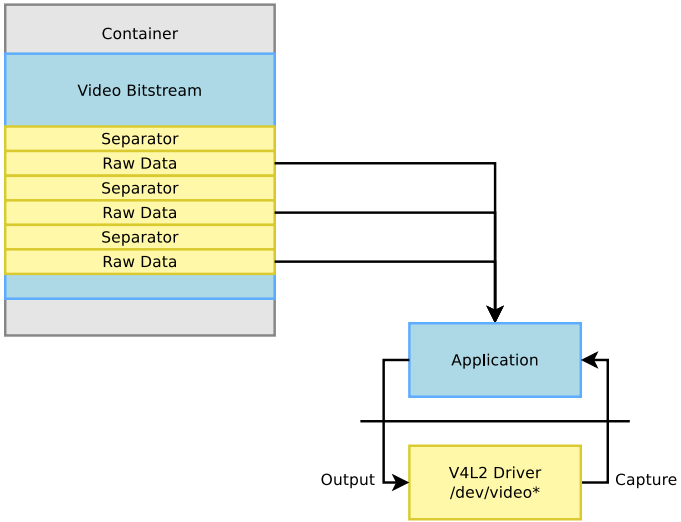
V4L2

- Introduced in 2002, in 2.5.46
- Supports a wide range of devices
  - Video Capture (camera, tuners)
  - Memory to memory devices (hardware codecs, scalers, deinterlacers)
  - Radio receivers and transceivers
  - SDR

V4L2: A Dumb M2M Pipeline

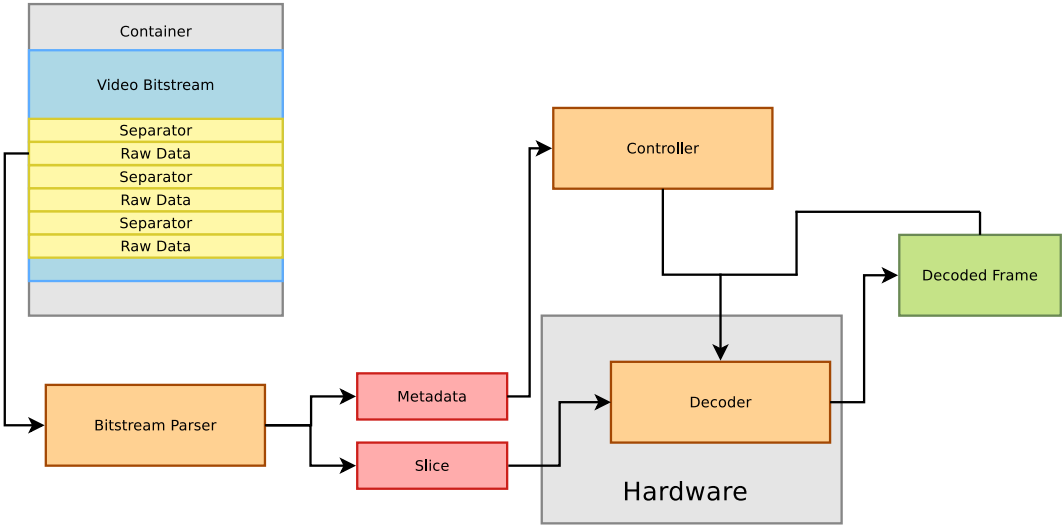


V4L2: Stateful Codec Support



And then, everything falls apart...

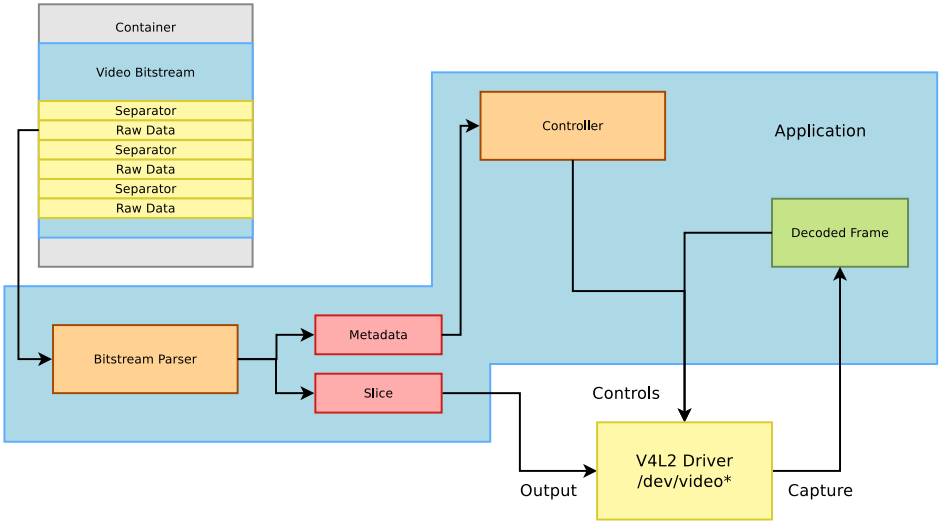
Stateless Codec



Request API

- An API that let's you tie controls with buffers
- First RFC sent in 2015 by Hans Verkuil
- Taken over in 2016 by Laurent Pinchart and Sakari Ailus
- Taken over in 2017 by Alexandre Courbot
- Taken over in 2018 by Hans Verkuil
- Merged in 4.20 (Yay!)

Stateless Codec Stack



Enters Allwinner

Allwinner

- Produces multimedia SoCs, targeted at tablets and STBs
- Found in a number of cheap SBCs
- Just like many multimedia SoCs, it has a hardware codec...
- ...a stateless codec that is

Allwinner BSP Stack

- Linux 3.4, or 3.10 these days
- Not using v4l2, but a stack split in two parts:
  - A kernel driver that manages the resources, and provides access to the codec registers
  - A userspace stack implementing the logic to decode and encode videos, that used to be closed source
- Obviously not compatible with mainline kernel

Reverse-Engineering Effort

- Reverse-engineering done by the *Cedrus* effort in order to have an opensource stack
- MPEG2, MPEG4, H264, H265 and VP8 Decoding and H264 Encoding
- Targeted at Allwinner's BSP kernel
- Kernel obviously not maintained, and seriously outdated now
- But very functional, supports many codecs, decoding and encoding, pretty much done
- libvdpau implementation to hook into popular media players

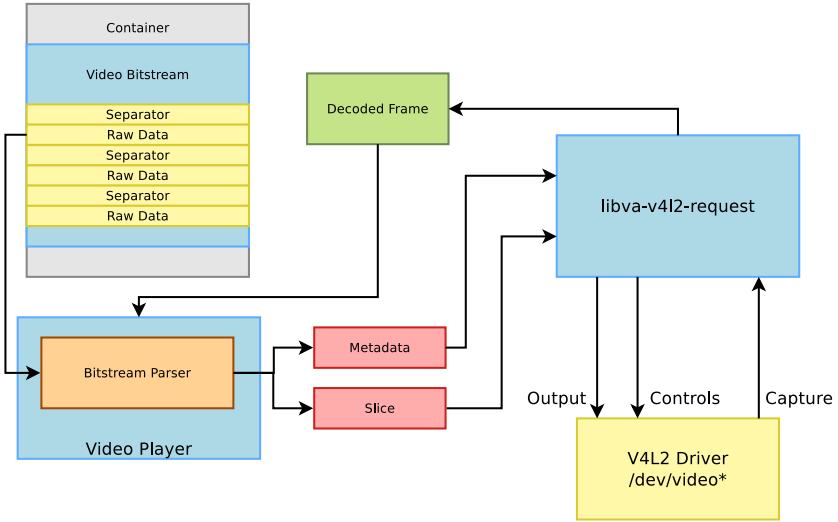
First attempt at a mainline driver

- Summer Internship in 2016, by Florent Revest
- Worked on an RFC, to do a mainline-based driver
- Quite successful:
  - MPEG2 implementation
  - libva implementation to integrate in the popular media players
- But was still a prototype:
  - Still had bugs
  - Slow, and could only play videos at any other resolution than the display
  - Who cares about MPEG2 anymore?

Kickstarter Campaign

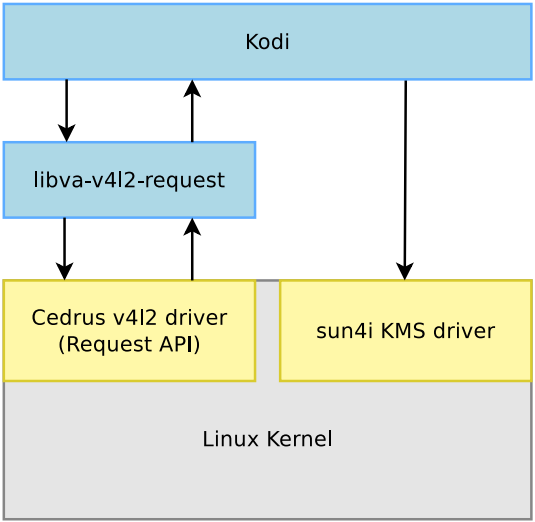
- Huge effort to bring it into a production-ready state
- Let's fund it through a Kickstarter campaign!
- The goals we achieved were to develop the driver for the most used SoCs, with H264 and H265 decoding support
- Funded a full-time 6-month internship, and a part-time engineer
- Built on top of the prototype

Cedrus Stack



Displaying is hard

In an ideal world...



Except that...

- The decoded frame is in a proprietary format...
  - ... that the display engine can understand
- The decoded frame is at the video resolution, not the window resolution...
  - ... and the display engine can upscale or downscale it at will
- But X11 doesn't let you do that easily

Possible solutions

- X doesn't have any knowledge about proprietary formats
- Let's convert it in software!
  - Reaaaaaally slow, even without scaling
  - Hardware video playback is supposed to be **more** efficient
- The X Video Extension is supposed to let you do that
  - But it's completely deprecated
  - Only one video would be playable at the same time (on our platform)
  - Possible glitches
- You could import the decoded frame in the GPU, and do the format conversion and scaling there
  - But the OpenGL blob has a lot of constraints
  - And memory bandwidth is a sparse resource
- Wayland!
  - You have to patch all Wayland compositors to deal with your proprietary format
  - And you leave out all the users that are stuck with X11

Current State

- The Request API is merged in 4.20
  - Libva implementation working on top of it, with MPEG2, H264 and H265 decoding support
  - Test tool to simulate some video decoding (v4l2-request-test)
- Our Cedrus driver is merged in 4.20 as well (but in staging)
  - Only MPEG2 for now
  - H264 and H265 patches have been sent

Questions? Suggestions? Comments?

Maxime Ripard
maxime@bootlin.com

`https://bootlin.com/pub/conferences/2018/elce/ripard-v4l2-codec`