Video4Linux: Current Status and Future Work

Hans Verkuil
Cisco Systems Norway

What Is V4L2?

- Video Capture and Output (compressed and uncompressed, analog and digital)
- VBI Capture and Output
- Memory-to-Memory Devices (codecs, scalers, etc.)
- Radio/RDS Receiver and Transmitter Support

It is not:

- Digital Video Broadcasting

Video HW Architecture

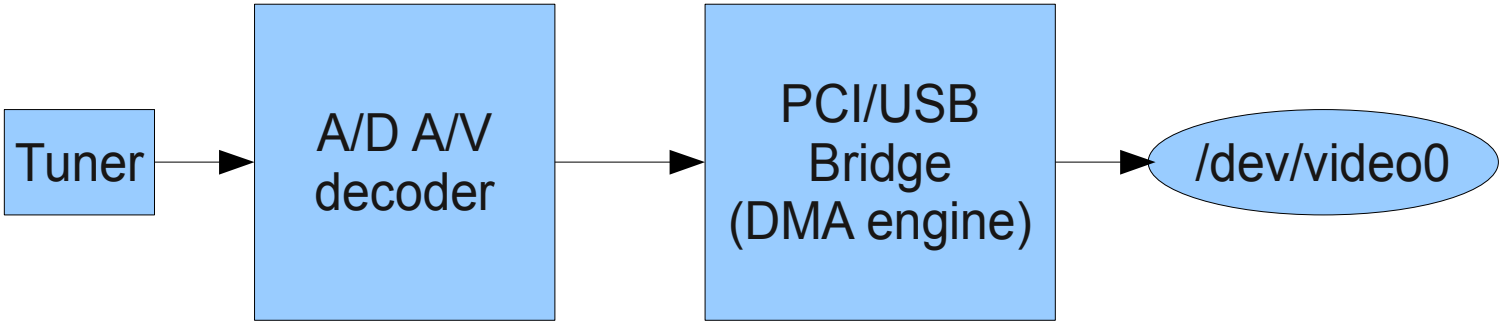
- Constellation of devices:
  - DMA engine
  - Sensor
  - Video receiver
  - Video transmitter
  - Tuner
  - Demodulators
  - IR receivers/transmitters
  - Muxers
  - Audio amplifiers
  - Audio mixers
  - De-ghosting
  - Comb filters
  - Scalers
  - Image processors
  - Camera flash
  - RDS encoders/decoders
  - Modulators

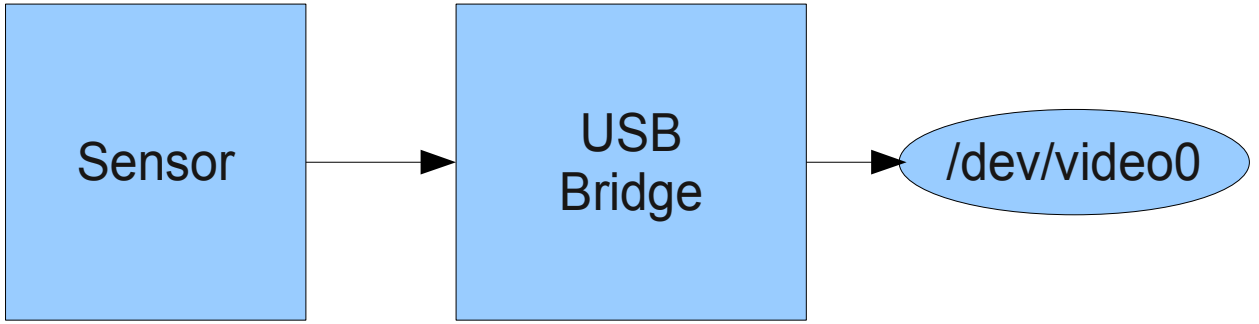
Statistics

- Until kernel 2.6 the v4l subsystem was small: 1-2% of all drivers.
- 2.6 added dvb and many new v4l drivers and it has grown from 5% to >9%.
- Second-largest subsystem after net.

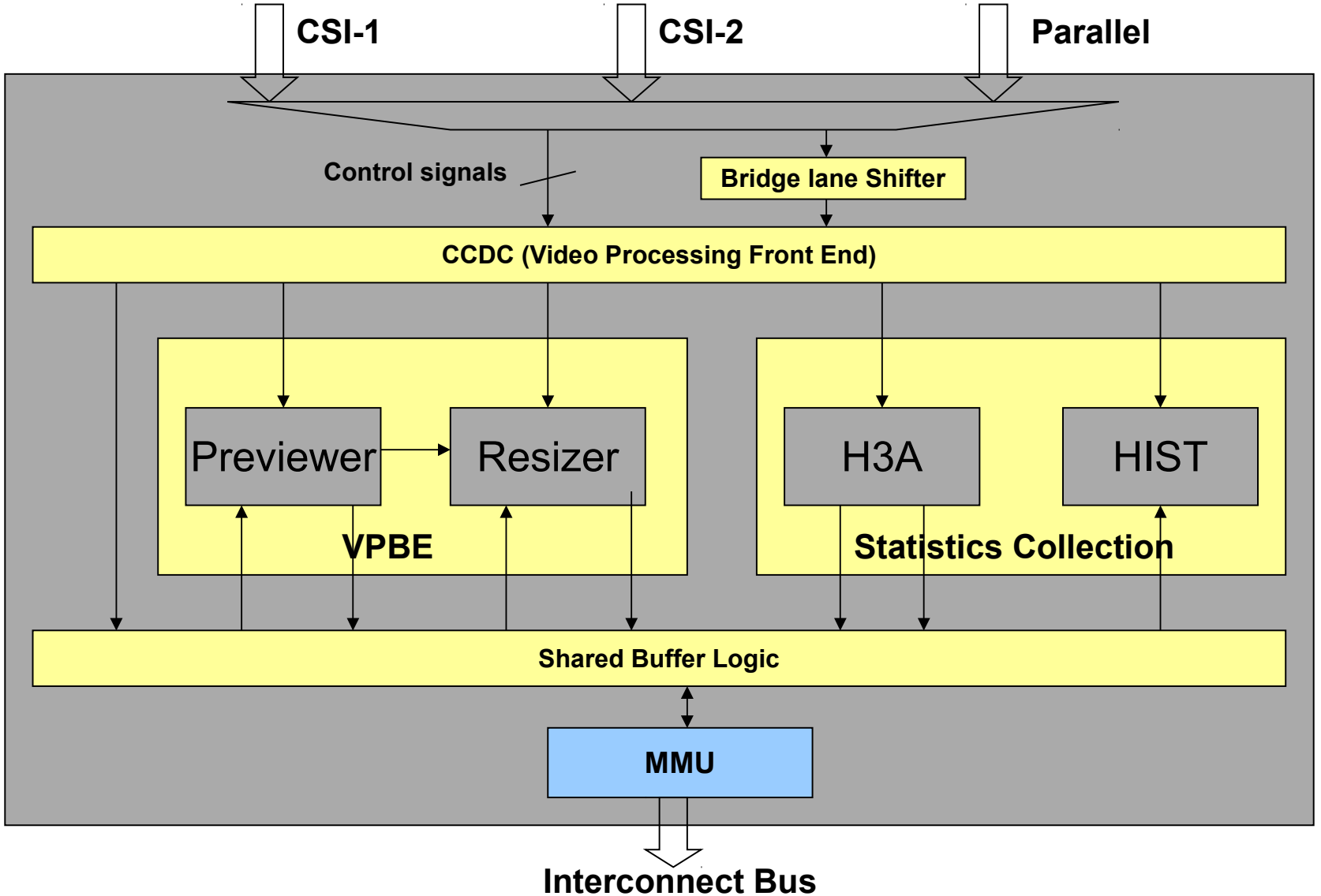
Kernel	media tree	% of all drivers
2.0	287953	1.00%
2.4.0	829547	1.50%
2.6.0	4634875	5.80%
2.6.10	4232200	4.60%
2.6.20	8419446	7.80%
2.6.30	14866804	8.90%
3.0	20512195	9.10%
3.9-rc1	25281975	9.75%

Typical Consumer Hardware



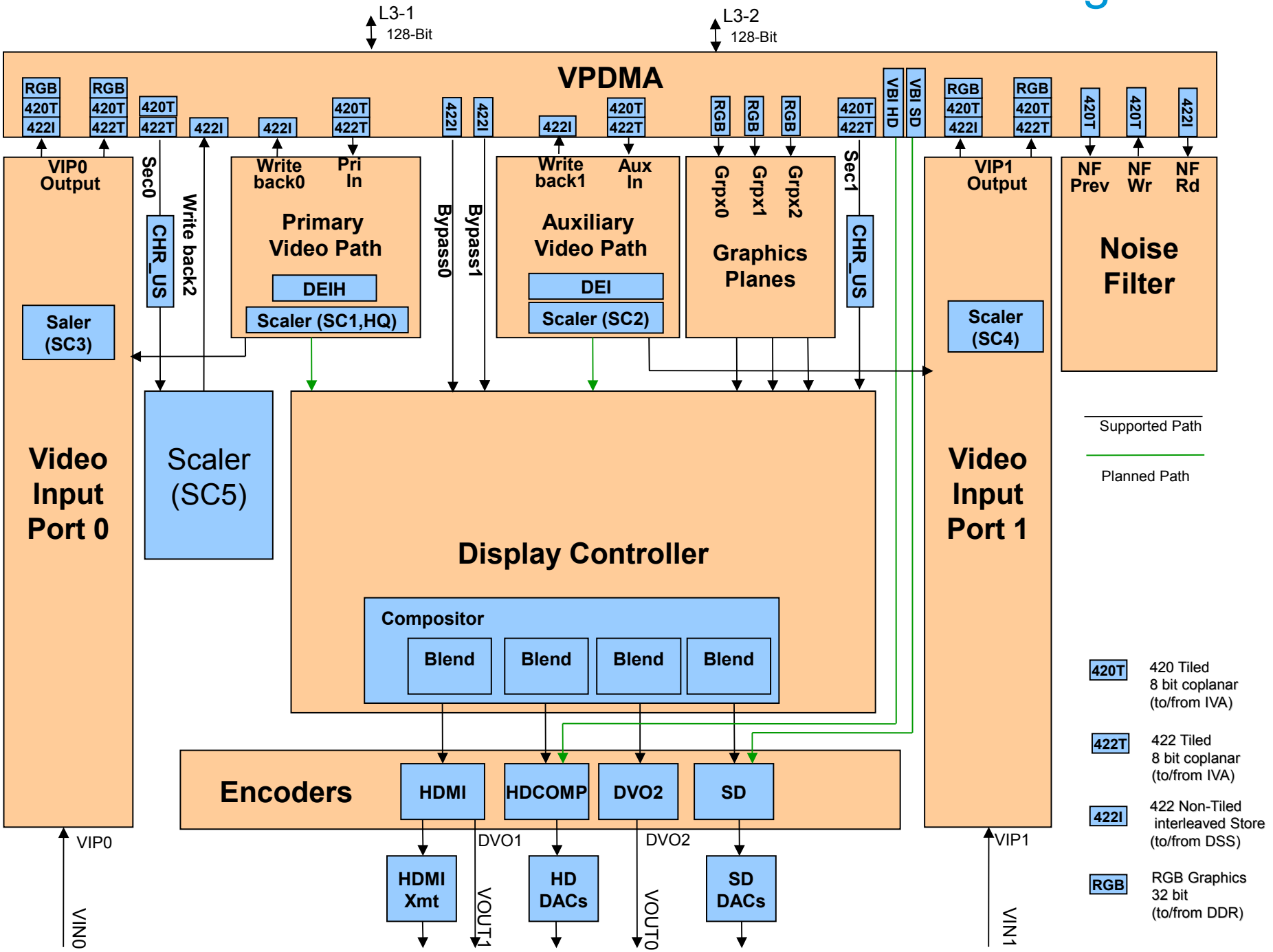


SoC HW: TI OMAP3 ISP



Courtesy of Texas Instruments

TI DM8147: HDVPSS – Broad Market Block Diagram



Courtesy of Texas Instruments

SoC Video Devices

- Very complex devices.
- Multiple video and graphics streams.
- Flexible video stream routing.
- Applications require much more control.
- Digital cameras, mobile phones, media players, TVs, surveillance applications, video conferencing, in-flight entertainment, etc., etc.
- Since the V4L2 API did not support these advanced devices, SoC manufacturers made their own custom drivers.

Core Framework

- Created a struct v4l2_device for basic device-global data.
- Created a struct v4l2_subdev to communicate with (usually i2c) sub-devices. Register them with v4l2_device. When v4l2_device is removed, unregister the sub-devices automatically.
- The 'sub-device' concept is an abstract concept: it does not care on what (if any) bus the sub-device is located.
- Ensures a unified API towards sub-devices to make it easy to swap one chip for another.
- Can also be used to expose internals of the video subsystem of a SoC.
- Created a struct v4l2_fh to keep per-filehandle data. Used to implement core support for priority and event handling.

Control Framework

- Too much hard work for drivers.
- A lot of code duplication.
- A lot of buggy code.
- Inconsistencies between drivers.
- All controls go through the new framework.
- A driver only needs to supply a s_ctrl function in most cases.
- Future enhancement: expose controls to debugfs.
- Ability to have bridge drivers inherit controls from subdevs.
- Merged in 2.6.36. Added the control event + other enhancements in 3.1.

Events API

- Standard API for V4L2 events.
- Can be used with select() since it arrives as an exception.
- Per-filehandle event queue and event subscription.
- Merged in 2.6.35.
- Improved event handling in 3.1: changes to per-filehandle, per-event type queues.
- Added an event that is sent whenever a control changes value. Merged in 3.1.

HDTV Timings API

- The DV_TIMINGS ioctls make it possible to specify exact timings (front & backporch, sync widths, pixelclock, etc.), to enumerate available timings and detect the timings of the incoming video signal.
- Merged in 3.5.
- Support for DVI/HDMI/DisplayPort/VGA connectors: odds 'n ends like EDID handling, hotplug detect, rx sense detection.
- Merged in 3.7 with some enhancements in 3.8.

Media Controller

- Modern v4l drivers often also support framebuffer, alsa, i2c, lirc and/or dvb devices, hard to keep track of by applications. Need some central authority to tell apps what is what.
- Many SoCs can reroute the internal videostreams. E.g. capture from a sensor and do memory-to-memory resizing, or send the sensor output directly to the resizer.
- Apps writing for SoCs want much more control about the various components of the device. A way is needed to provide that control without compromising the normal API which tends to hide complexity from the user.

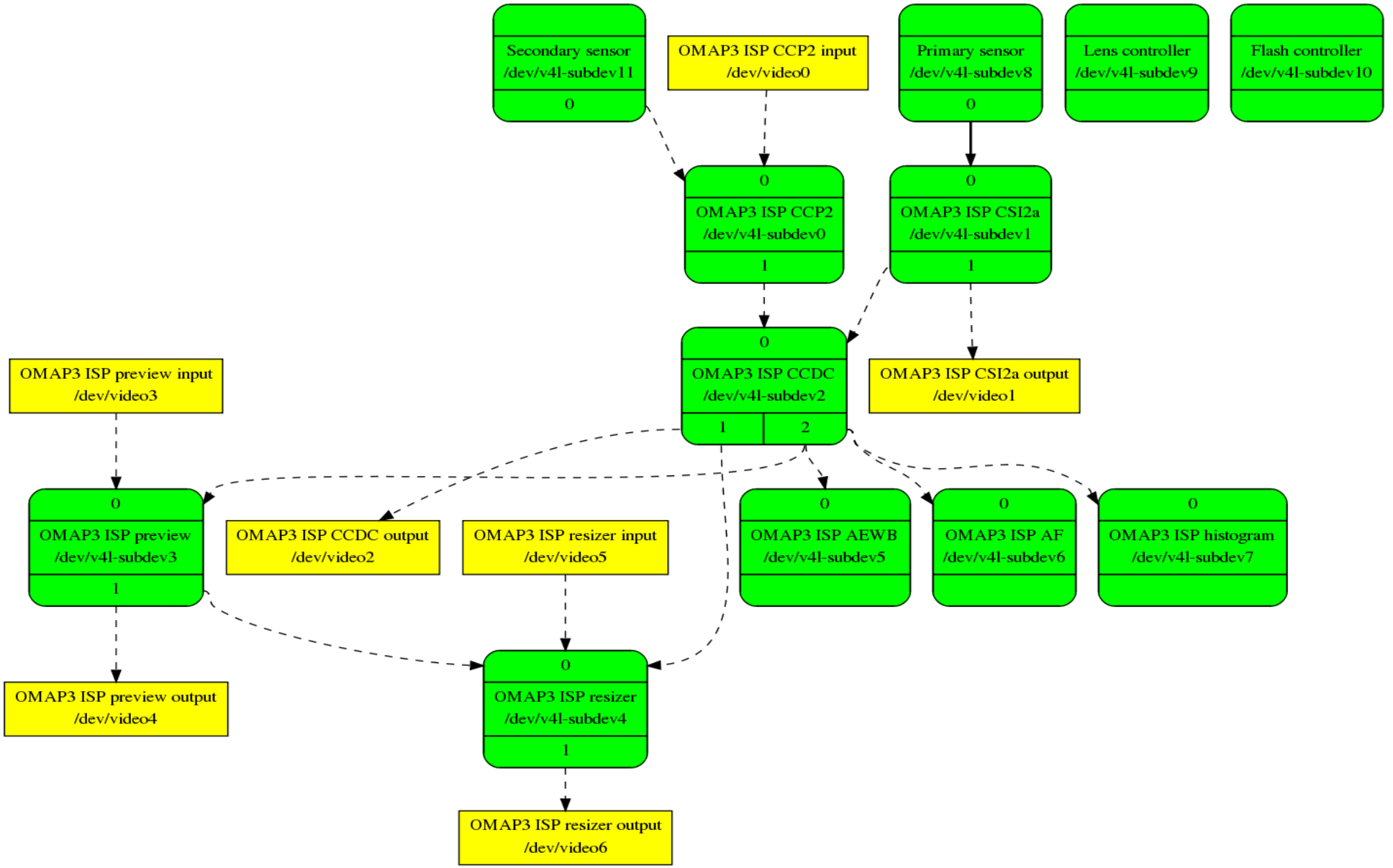
Media Controller

- Create a /dev/mediaX device (X >= 0) that can be used to enumerate the mesh-topology of the device.
- Nodes in the mesh are sub-devices and device nodes. The general name for a mesh node is entity.
- The mc will also enumerate the possible and current links between entities.
- The mc allows you to change the links.
- Some sub-devices will have their own device node for advanced control (/dev/v4l-subdevX).

Media Controller

- For embedded devices the mc controls the internal data flow of the device.
- For embedded devices the /dev/v4l-subdevX device nodes allow direct control of the advanced and hardware-specific features of sub-devices.
- For each particular SoC a userspace library is required to use the hardware optimally. This allows us to keep the kernel driver simple. This will be based on libv4l2 and be implemented as plugins. This is still work in progress.
- Merged in 2.6.39.

TI OMAP3 ISP: Media API



Courtesy of Laurent Pinchart

videobuf2

- Existing videobuf framework was very bad code.
- A new videobuf2 framework was created that solves the existing problems with videobuf2. In particular buffer operations are now separate from memory operations.
- Merged in 2.6.39.
- The vb2 framework also made multiplanar support and memory-to-memory devices possible.

Codec & Flash support

- Codec: support for H.264/MPEG4/DIVX/etc. elementary streams. Merged in 3.1.
- Support for Flash controllers. Merged in 3.1.
- Added support for JPEG compression (replacing the ugly VIDIOC_JPEGCOMP ioctl). Merged in 3.4.

Selection & Radio support

- Improve crop and compose support through the new selection API. Merged in 3.2.
- Added support for multiple frequency bands for radio receivers/transmitters. Merged in 3.6.
- RDS/TMC decoder library: mostly done, expected to be released in v4l-utils version 0.10.0.

Memory Handling

- Video hardware often requires large amounts of physically contiguous memory.
- Hard to allocate in the kernel due to memory fragmentation.
- Many vendors made their own incompatible 'solutions', based on reserving memory at boot and creating an API to allocate from that pool of memory.
- Disadvantage: vendor-specific solutions, and the memory can't be used for anything else.
- Solution: Contiguous Memory Allocator from Samsung, merged in 3.5. Memory can be marked as available for DMA buffers or movable pages. So pages can be moved elsewhere (or discarded) when needed for DMA buffers.

Buffer Sharing

- We want zero-copy video pipelines.
- This requires easy transfer of buffers from one video device node to another, or to gpu textures or framebuffers, without requiring a memcpy.
- Linaro created the dmabuf API (merged in 3.3). Each buffer is represented by a file descriptor.
- V4L2 can import and export such buffers through the videobuf2 framework. Merged in 3.8. Exporting is currently only supported for physically contiguous buffers.
- An application that wants to use these buffers will have to ensure that both sides can understand the format of the buffer.

Device Tree

- The ARM kernel code is relying more and more on the device tree for discovering available hardware.
- Support for the device tree in V4L2 will appear in kernel 3.10.
- The order of initialization of modules is undefined when using the device tree: e.g. a video receiver driver may be loaded before or after the SoC video driver is loaded.
- Video hardware is usually a constellation of devices, so we have to wait until all components are loaded before we can proceed: support for this is in progress and is expected in 3.11.

In Progress

- CEC (Consumer Electronics Control) support. Work on this is stalled at the moment.
- Update all existing drivers to use the latest frameworks and to pass the compliance tests from the v4l2-compliance tool. When done, we can drop ugly legacy code.
- Improve handling of tuner ownership: some tuners support analog TV, FM radio and DVB. Switching between one mode or the other is flaky in many drivers. We need support in the core to handle this safely.
- Continue streamlining and cleaning up code.
- Improve cooperation between v4l2/drm. I2C video transmitter drivers can work only with v4l2 or with DRM, not both.
- Related topic: Common Display Framework.

In Progress

- SDR (Software-Defined Radio) support.
- Motion Detection API.

Slide from Japan Linux Symposium, Oct 2009:

Future: results from mini-summit

- New timings API for HDTV and other high-definition sources and sinks.
- Standard event passing API.
- New core control framework.
- Buffer pool management API.
- Support for multi-planar framebuffers.
- Media controller.

Many Thanks To:

- Sumit Semwal & Linaro: DMABUF API.
- Sakari Ailus & Nokia: Events API, sensors, flash, omap3.
- Laurent Pinchart & Nokia: Media Controller, omap3, sensors.
- Pawel Osciak, Sylwester Nawrocki, Marek Szyprowski, Kamil Debski & Samsung: videobuf2, CMA, M2M (Codec) APIs, Samsung SoCs.
- Mauro Carvalho Chehab & Red Hat: media subsystem maintainer.
- Tandberg/Cisco Systems Norway: supporting me for the last 5 years.
- Linux Foundation: organizing media summits and great conferences.

Questions?

e-mail:

hansverk@cisco.com
hverkuil@xs4all.nl

linux-media mailinglist:

http://www.linuxtv.org/lists.php