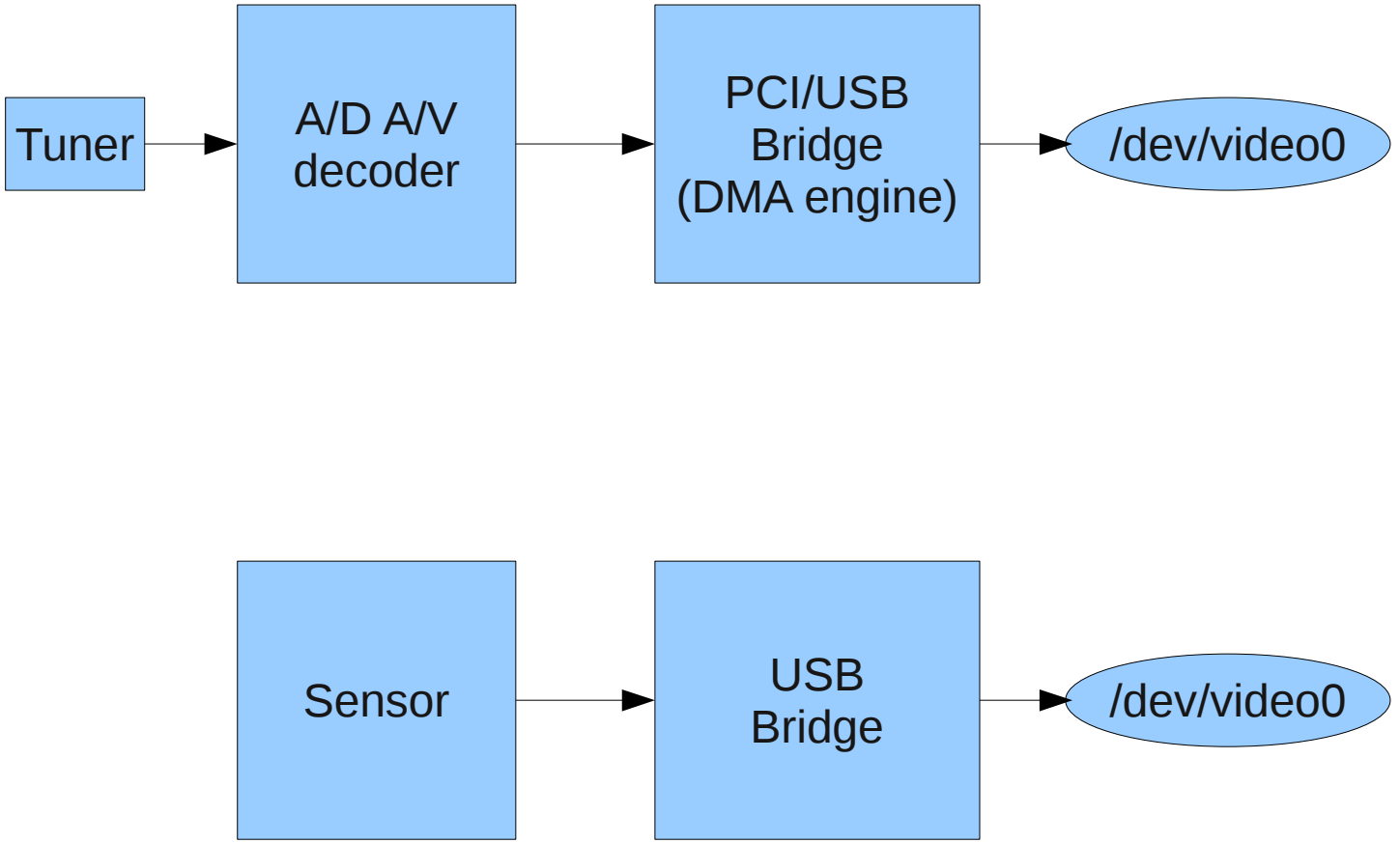
Video4Linux: Current Status and Future Work

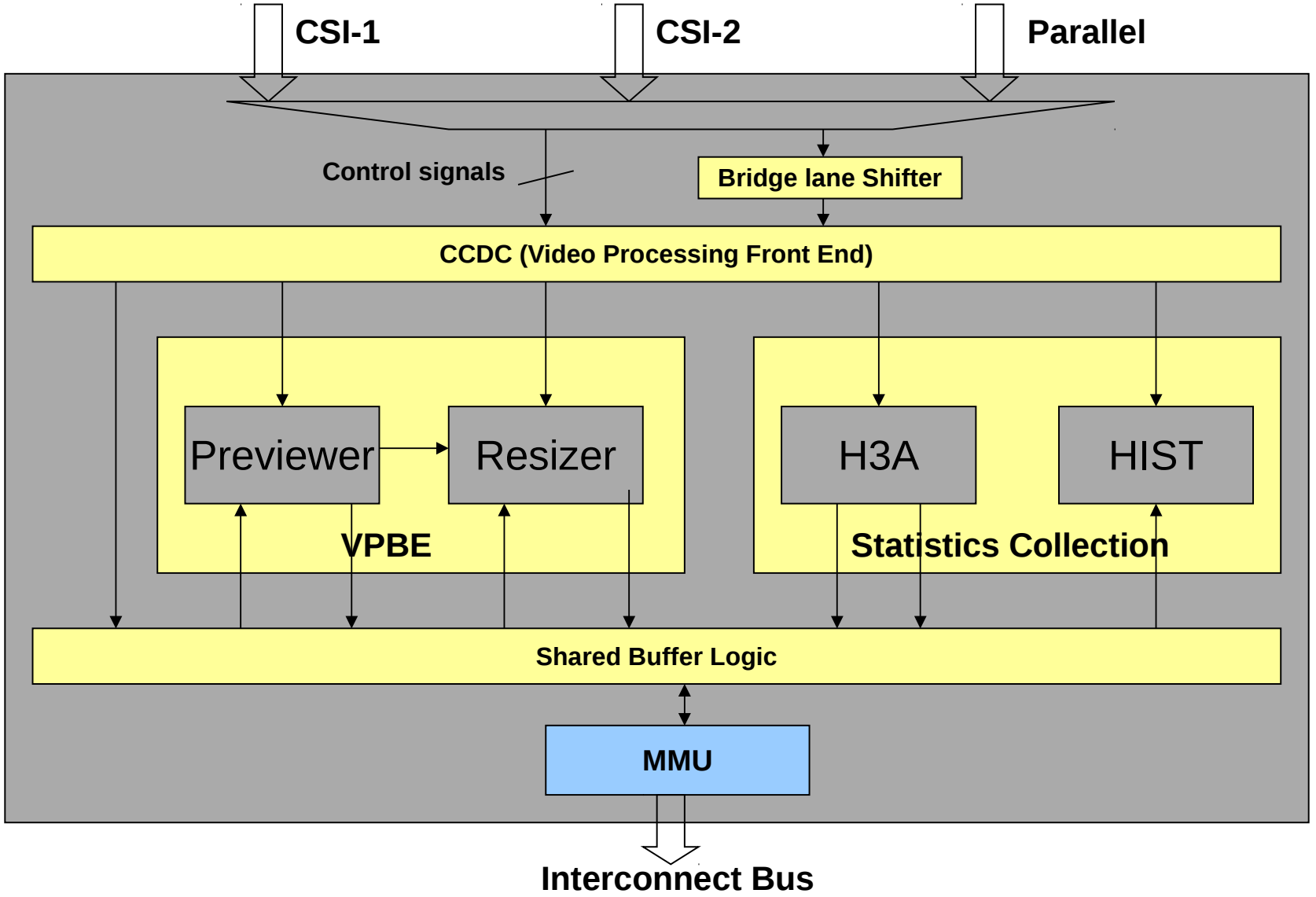
Hans Verkuil

Cisco Systems Norway

Typical Consumer Hardware

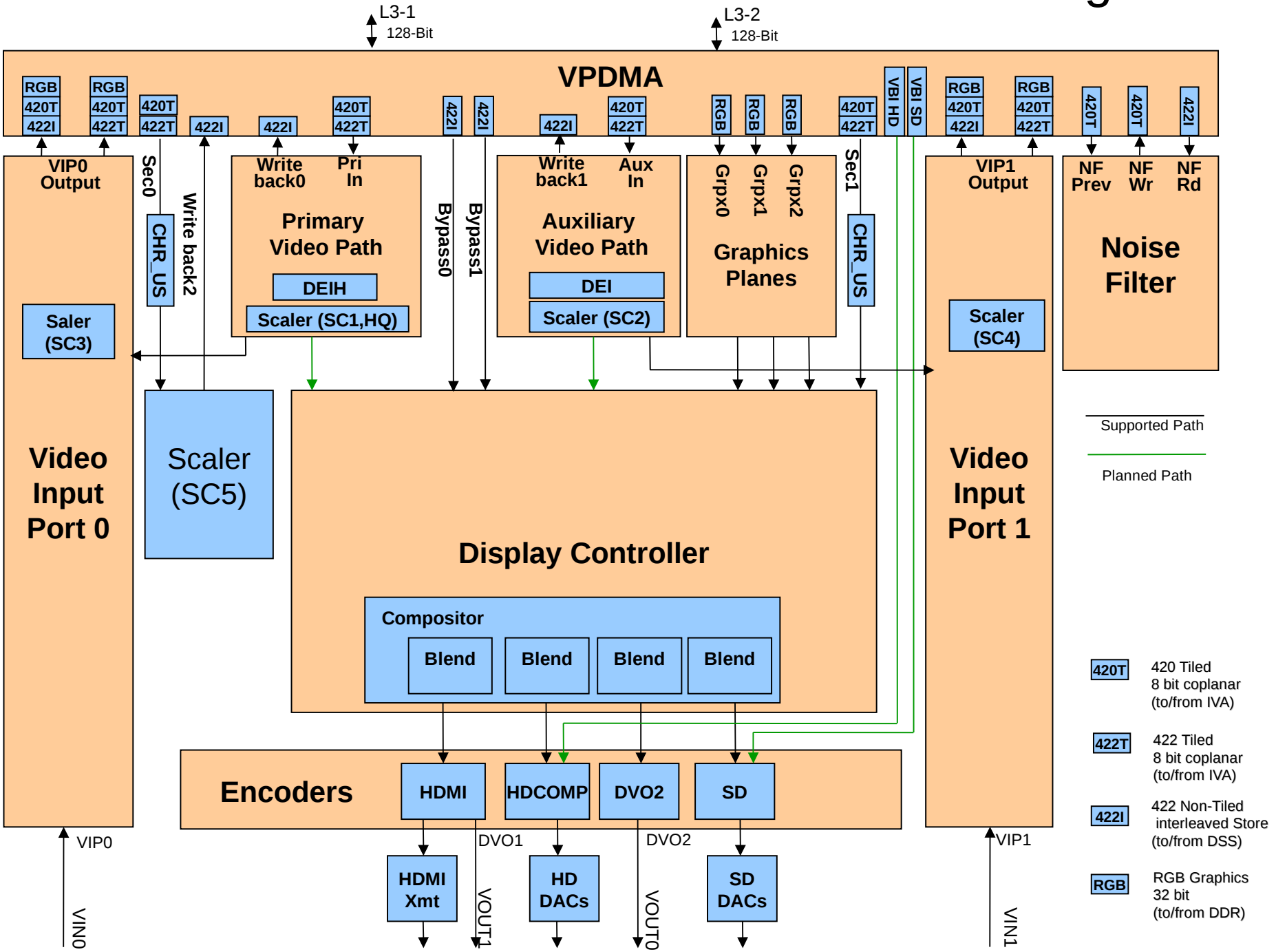


SoC HW: TI OMAP3 ISP



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 (gStreamer!) 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.

SoC Support Developments

- Initial talks with Texas Instruments Spring/Summer 2008.
- First RFC July 2008.
- Discussed during the Linux Plumbers Conference in August 2008.
- V4L-DVB mini-summit during the Linux Plumbers Conference in September 2009. Agreed on how to proceed with V4L SoC support.
- Brainstorm meeting in Oslo to discuss memory handling (videobuf) in March 2010.
- Mini-summit in June 2010 in Helsinki.
- Brainstorm meeting in Warsaw in March 2011.
- Meetings with Linaro in May and August 2011 to discuss buffer sharing and contiguous physical memory allocations.
- A V4L-DVB workshop in October 2011 and a workshop tomorrow.

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.

HDTV Timings API

- The original API used the 'preset' idea with presets for the common standards (e.g. 720p30, 1080p60, etc.) allowing you to get/set/query/enumerate supported presets.
- In addition it was possible to specify exact timings (front & backporch, sync widths, pixelclock, etc.) with the DV_TIMINGS API.
- Merged in 2.6.33.
- The preset API turned out to be insufficient and this API is now marked deprecated. It will be removed in the near future.
- The DV_TIMINGS API was extended to include timings enumeration and querying (detecting) the current timings.
- This was merged in 3.5.

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.

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.

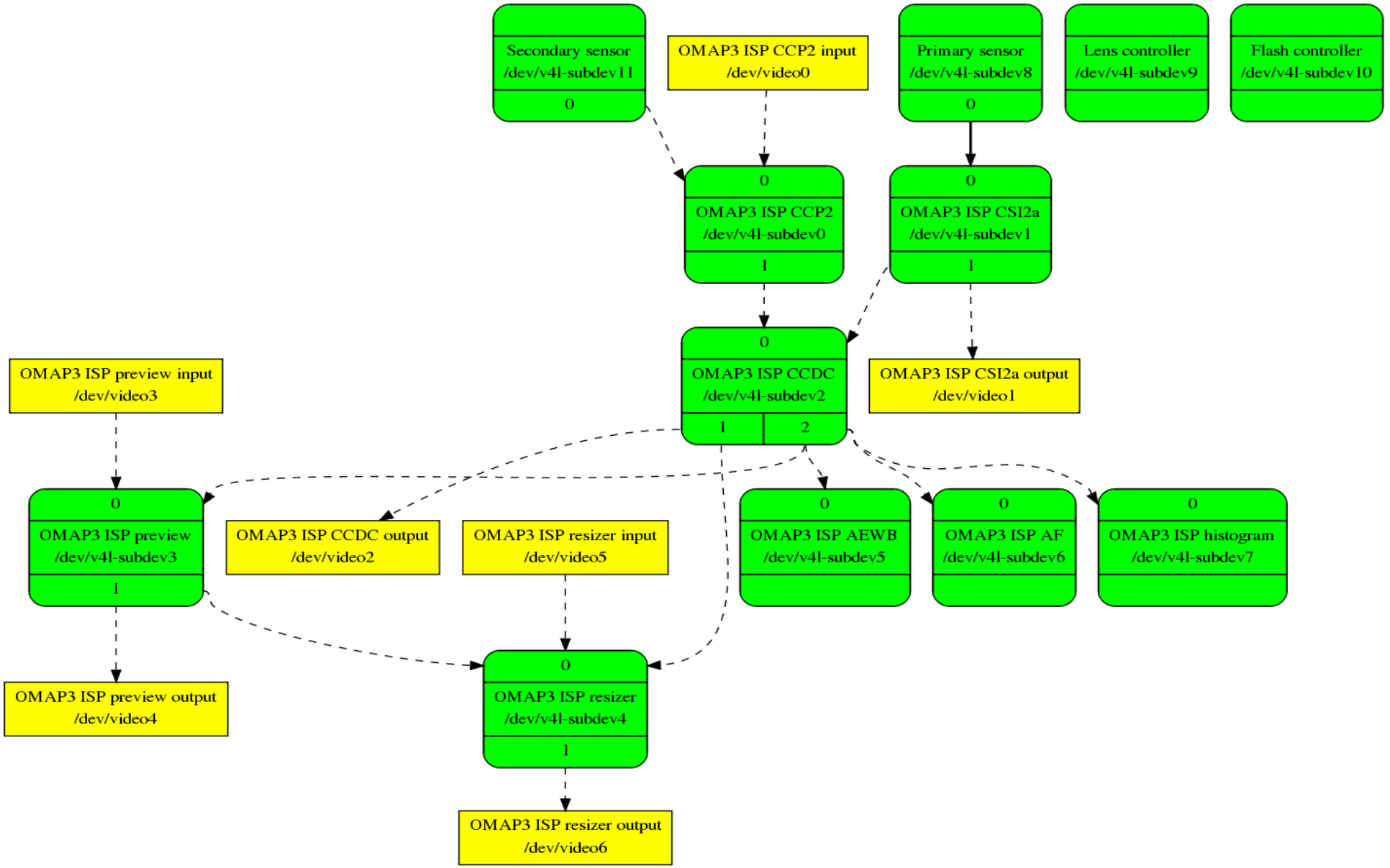
Media Controller

- 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

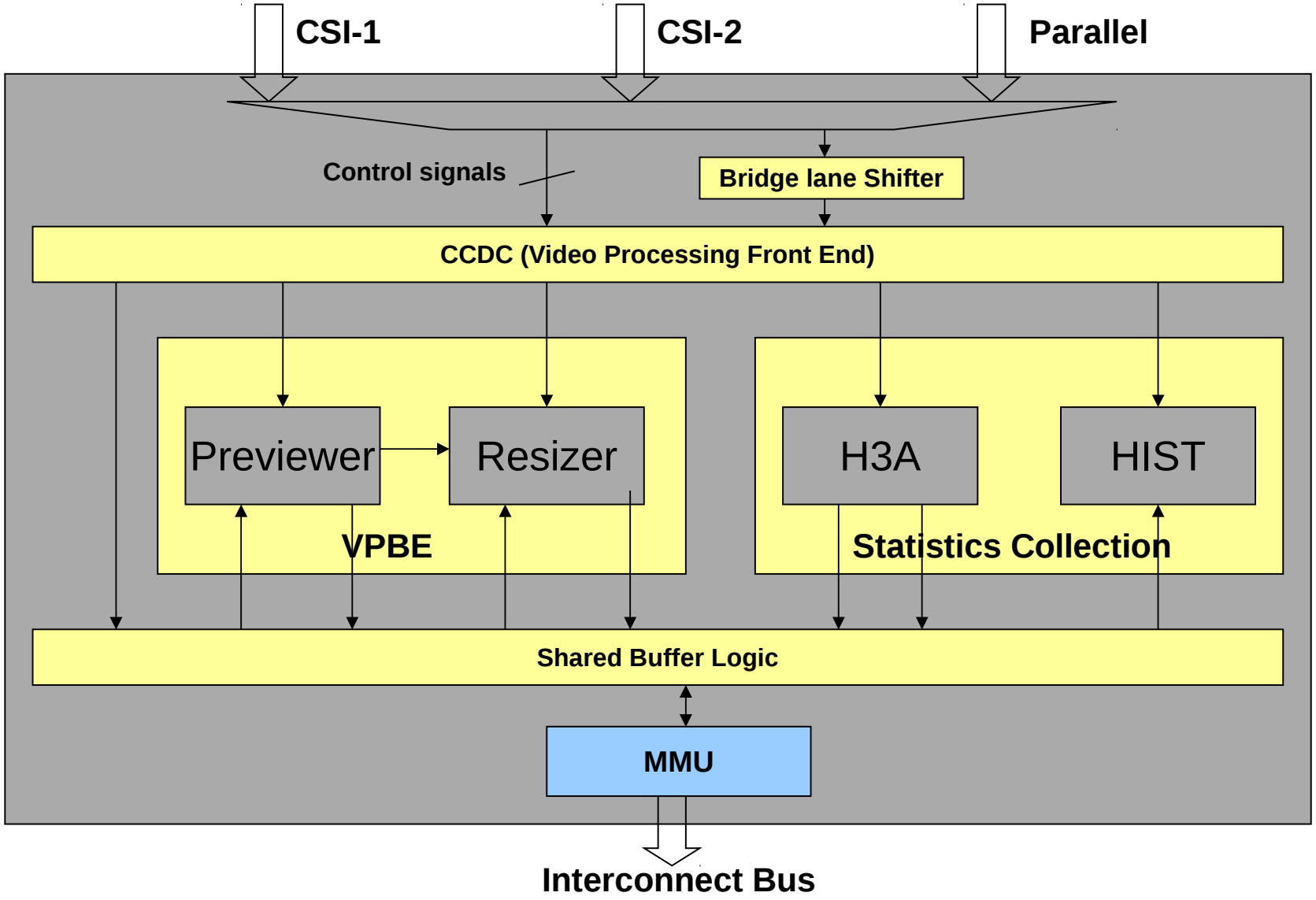
- For embedded devices the mc controls the internal data flow of the device.
- For embedded devices the /dev/v4l-subdevX device nodes allow direct control of the advanced and hardware-specific features of sub-devices.
- For each particular SoC a userspace library is required to use the hardware optimally. 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

TI OMAP3 ISP: Block Diagram



Courtesy of Texas Instruments

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.
- Added libv4l2rds for RDS decoding (including TMC). TMC decoding still needs some work.

Memory Handling

- Video hardware often requires large amounts of physically contiguous memory.
- Hard to allocated 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.

In Progress: Buffer Sharing

- A lot of work is being done through Linaro (www.linaro.org) to improve buffer handling in the kernel. Goal: zero-copy video pipelines.
- Allow easy transfer of buffers from one video device node to another, or to gpu textures or framebuffers) through the new dmabuf API (merged in 3.3). Each buffer will be represented by a file descriptor.
- Work is in progress to allow V4L2 through the videobuf2 framework to handle such buffers.
- An application that wants to use these buffers will have to ensure that both sides can understand the format of the buffer.

In Progress

- Support for DVI/HDMI/DisplayPort/VGA connectors: odds 'n ends like EDID handling, hotplug detect, rx sense detection. Expected to be merged for 3.7.
- CEC (Consumer Electronics Control) support.
- Driver improvements: v4l2-compliance tool.

Questions?

e-mail:

hverkuil@xs4all.nl

linux-media mailinglist:

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