# GStreamer and OMAP4

**Rob Clark**

# GStreamer and OMAP4

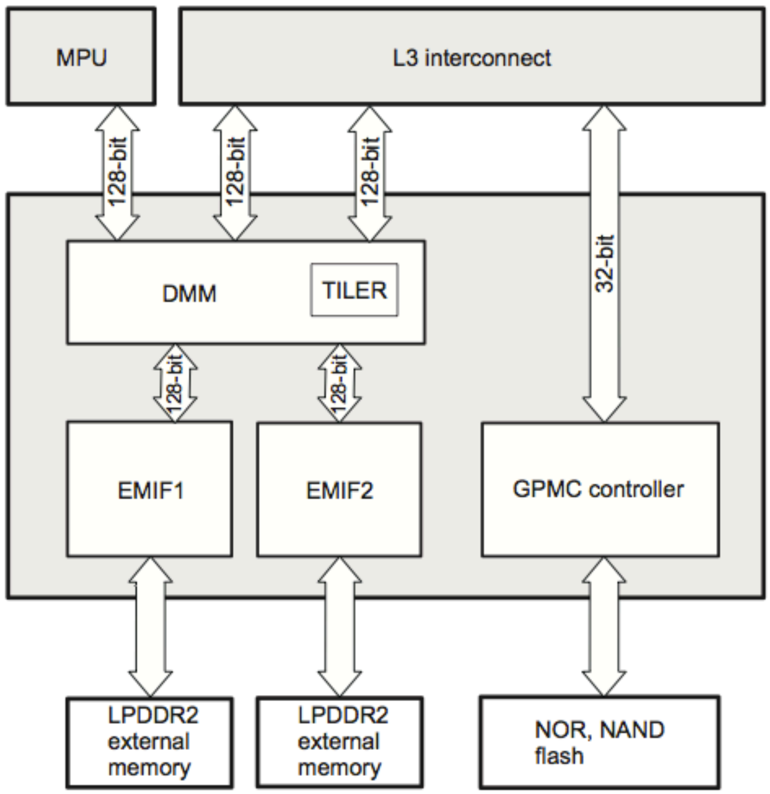
- Overview of OMAP4 Multimedia
- OMAP4 Multimedia with GStreamer
- OpenMAX on OMAP4

# Overview of OMAP4 Multimedia

# DMM/TILER

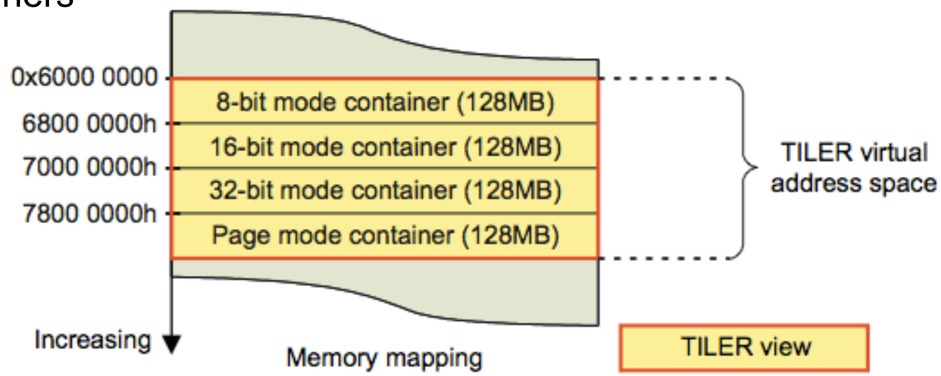
- Resolves Memory Fragmentation
  - Provides contiguous virtual memory for codecs, camera, and display
  - Removes the need for IVA-HD, DSS, and ISS to have own MMU
- Increased 2D Block Transfer Efficiency
  - Provides efficient handling of 2D data mapped in tiles like YUV macroblocks
  - Reduces number of SDRAM page accesses per block
  - Increases utilization of an 128b SDRAM burst
  - Optimize multi-channel memory transfers
- Rotation
  - Provides free rotation/mirroring for display/ camera
  - 0°/90°/180°/270° rotation with horizontal or vertical reflection

Note: Subsystem name is DMM (Dynamic Memory Manager). When most people refer to TILER they actually refer to DMM



# Non-TILER Address Space

- Non-TILER Address Space (what the ARM sees):
  - A single view comprised of four 128MiB containers
  - 2D: 8b, 16b, and 32b containers
    - 0x6000 0000 to 0x77ff ffff
    - 16KiB stride for 8b container, 32KiB stride for 16b and 32b container
    - NV12 puts Y plane in 8b container, and UV plane in 16b container
    - Framebuffer in ARGB32 could use 32b container (if rotation of GUI is desired)

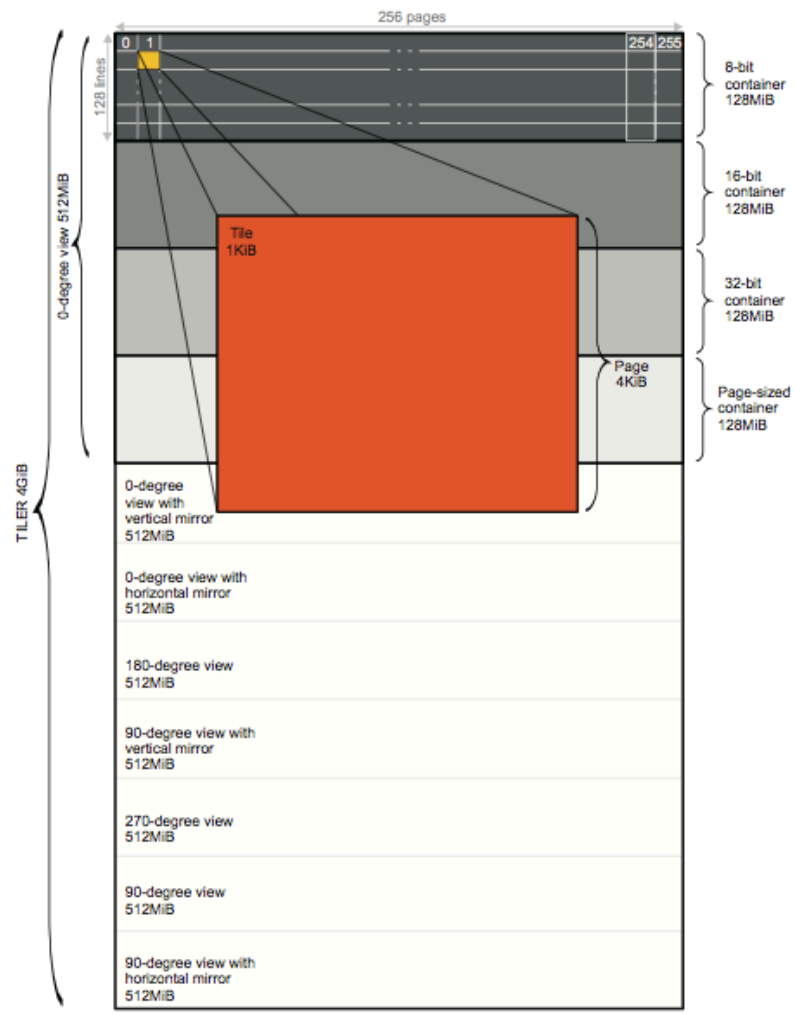



  - 1D: paged mode container used for compressed bitstream buffers typically
  - A physical address in this range is sometimes referred to as SSPtr (System Space Pointer)

- Notes about how NV12 YUV buffers are mapped
  - Individual planes (Y and UV) are themselves physically contiguous, but as separate buffers
  - For software compatibility, NV12 buffers are mapped into virtually contiguous pages, ie. one page per row
    - Because of 2D TILED transformation, not entire 4kb stride need be backed by physical memory (so actual memory requirement is not 4KiB * height * 1.5)
  - A virtual address for SSPtr is sometimes referred to as VSPtr (Virtual Space Pointer)

# TILER Address Space

- A separate 4GiB address space consisting of the 512MiB view repeated 8 times for all possible combinations of 0°/90°/180°/270° rotation with optional horizontal or vertical mirror
- A TILER address is sometimes referred to as TSPtr (TILER Space Pointer)
- The DSS or ISS can be configured to be programmed with a TSPtr instead of normal physical address to achieve rotation of displayed and/or captured image



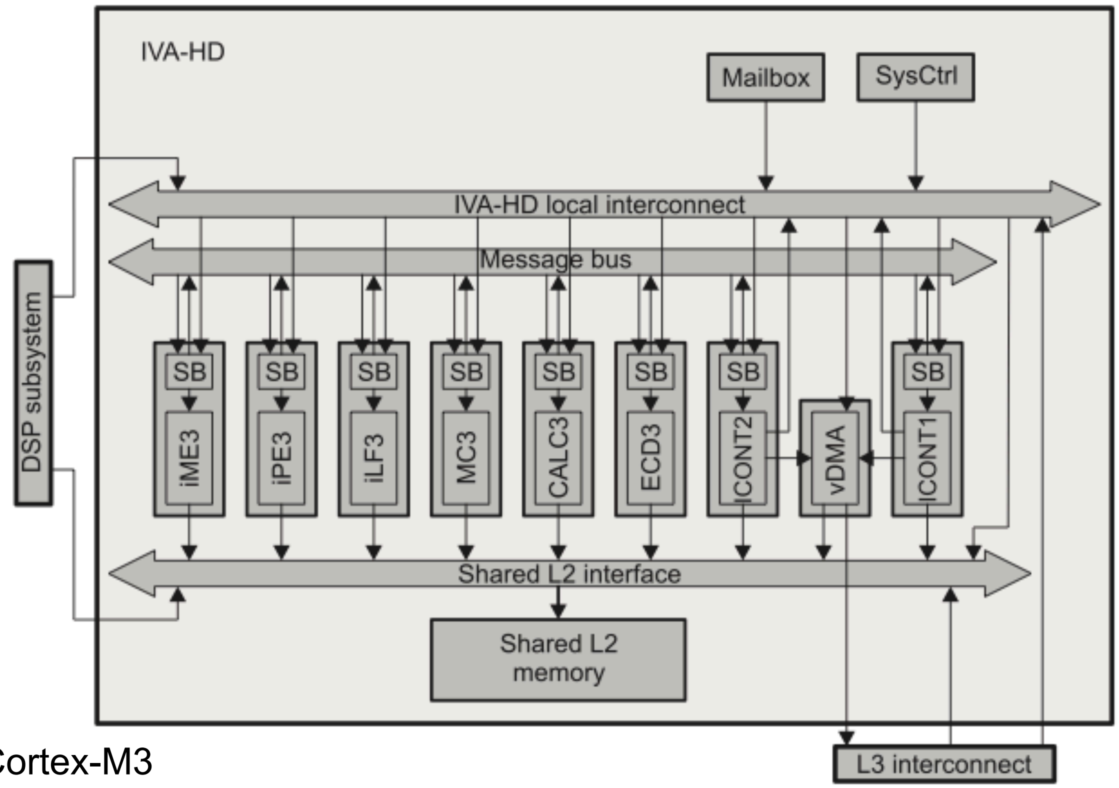
# IVA-HD

- 1080p30 / 1080i60 encode/decode
- Fully hardware accelerated codecs (without any intervention of DSP):
  - H.264 BP/MP/HP encode/decode
  - MPEG-4: SP/ASP encode/decode
  - DivX 5.x and higher encode/decode
  - H.263 Profile 0/3 decode, profile 0 encode
  - MPEG-2 SP/MP encode/decode
  - MPEG-1 encode/decode
  - VC-1/WMV9 encode/decode
  - On2® VP6/VP7 decode
  - RealVideo® 8/9/10 Decode
  - JPEG/MJPEG baseline encode/decode
  - H.264 Annex H MVC (stereo) up to 720p30

# IVA-HD Block Diagram

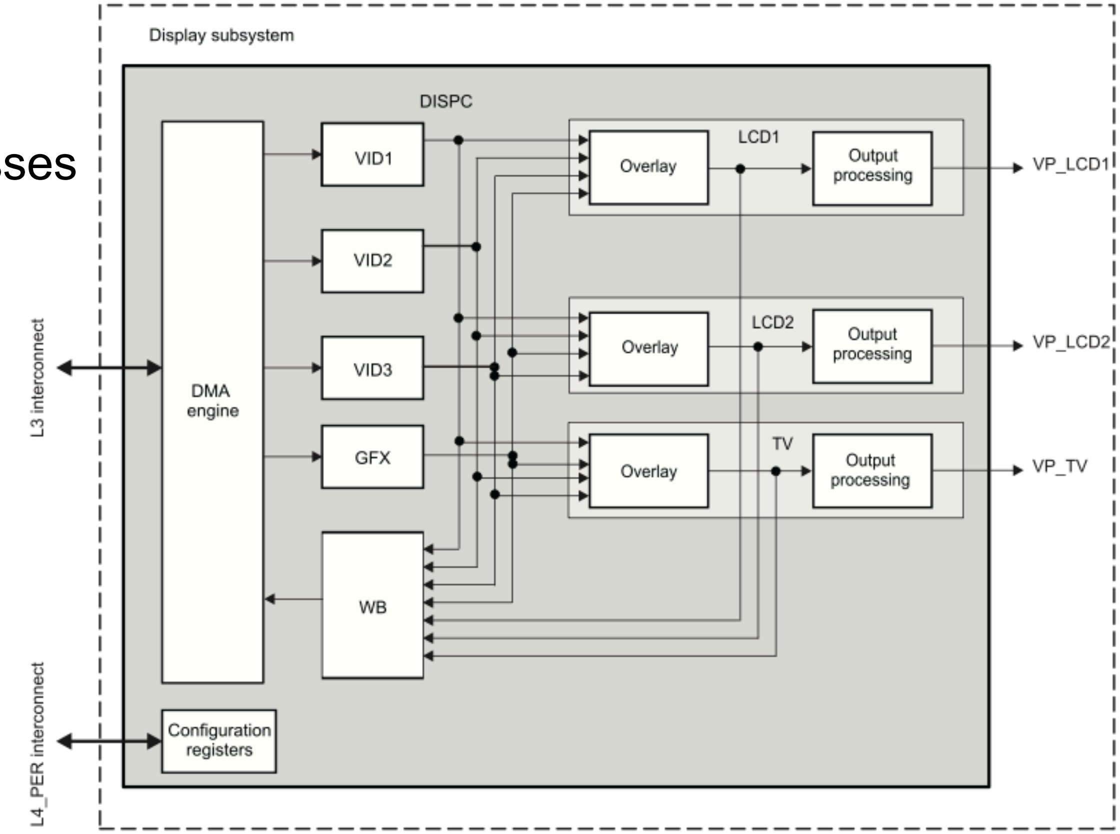
- SyncBox's (SB) and message bus for synchronizing various engines and sequencers
- ICONT1 & ICONT2: ARM968E-S™
  - ICONT1: primary sequencer
  - ICONT2: DMA processor and secondary sequencer
- vDMA: video DMA engine
- ECD3: entropy coder/decoder engine
  - Encodes/decodes bitstream
  - Supports Huffman and arithmetic codes
- MC3: motion compensation engine
- CALC3: transform and quantization

  calculation engine
- iLF3: loop filter engine
- iME3: motion estimation engine
- iPE3: intraprediction estimation engine
- Shared L2 interface and memory
- Controlled by Ducati subsystem: dual Cortex-M3



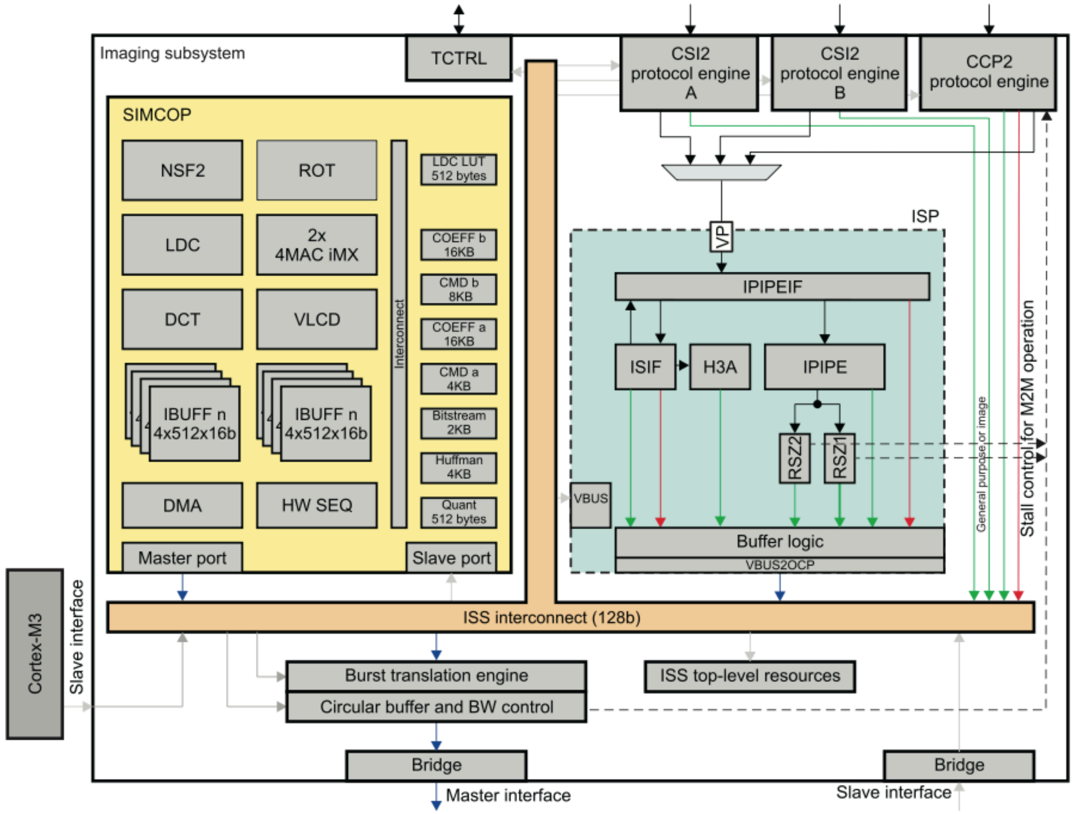
# DSS – Display SubSystem

- Largely similar to OMAP3 plus a few new features
  - NV12 support for video overlays
  - Support for TILER addresses (TSPtr)
  - Additional video overlay (VID3)
  - Writeback (WB) pipe for output to memory



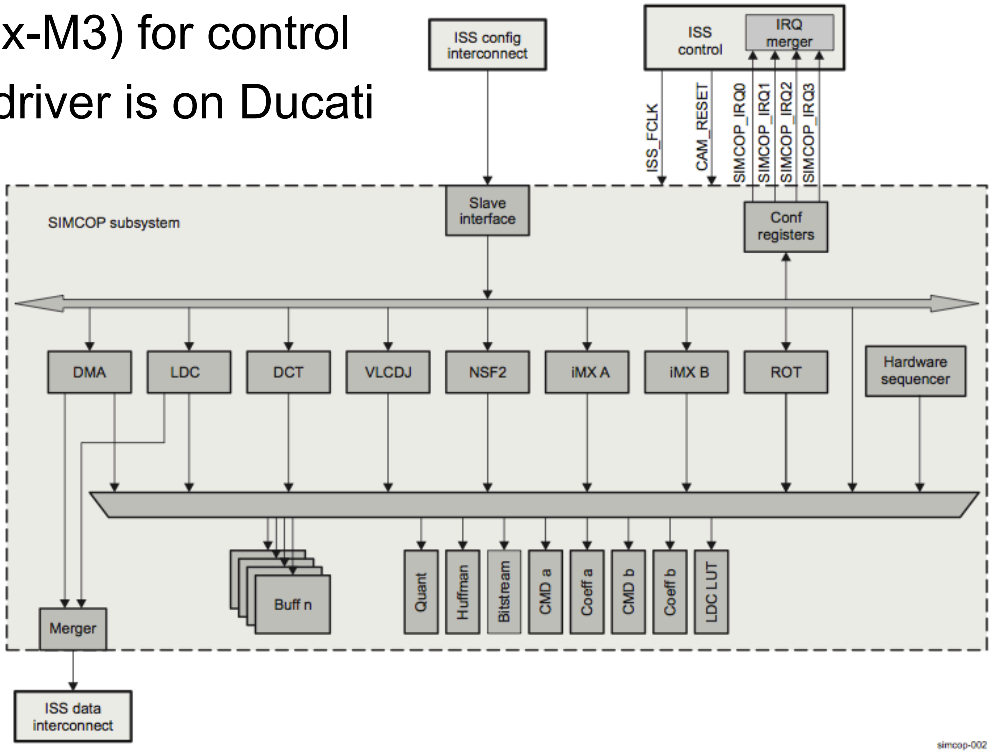
# ISS – Imaging SubSystem

- ISP: Image Signal Processor
  - Similar to OMAP3
  - Additional resizer (RSZ) allows simultaneous JPEG and video capture (for example)
- SIMCOP:
  - New block for image processing (see next slide)



# SIMCOP: Still IMage CoProcessor

- Macroblock based memory to memory processing engine
  - Fetch data to local memories
  - Process by one or more processing engines
  - Store back to system memories
  - Closely coupled to Ducati (Cortex-M3) for control functions (which is why camera driver is on Ducati vs. Linux v4l2 driver)
  - VLCDJ: JPEG encode/decode
  - NSF2: High ISO noise filter
  - LDC: Lens Distortion Correction
  - DCT: Discrete cosine transform
  - Two iMX4: general purpose imaging accelerators



# OMAP4 Multimedia with GStreamer

# Challenges presented by OMAP4 (1/3)

- To avoid memcpy's, all YUV buffers are strided
  - To realize the performance benefits of TILER 2D buffers, YUV buffers require 4KiB rowstride
  - Additionally, codecs rely on display for cropping codec edges
    - Same buffer used internally by codec for reference frames is also returned to display, to avoid a memcpy
    - Additionally cropping ensures proper alignment of macroblocks
  - Similarly with some camera algo's, such as VSTAB
    - Frame by frame notification to encoder and display to crop to stabilized frame within larger buffer

# Challenges presented by OMAP4 (2/3)

- In some cases, the display must perform additional postproc functions
  - Mirror decoder output around horizontal axis for VP6 from Flash container
  - VC-1 range mapping
  - Text or graphic (ex. FD boxes) overlay composition
  - For legacy video sink elements, a combination of ISS resizer and/or DSS WB pipe could be used for post-processing
    - But how to auto-plug this?
- Different codecs have various minimum # of buffer requirements
  - For example, H.264 has minimum buffer requirements that vary based on resolution
  - If the video sink element is allocating a fixed number of buffers, it must query the upstream element for minimum buffer requirements

# Challenges presented by OMAP4 (3/3)

- Most existing camera apps hard-code pipeline:
  - v4l2src and sw based encoder elements
  - Makes it difficult to just drop in plug-ins and fully leverage ISS and IVA-HD

- In some cases, differences in encoded bitstream format
  - For example asfdemux vs VC-1 decoder

# Current Solutions

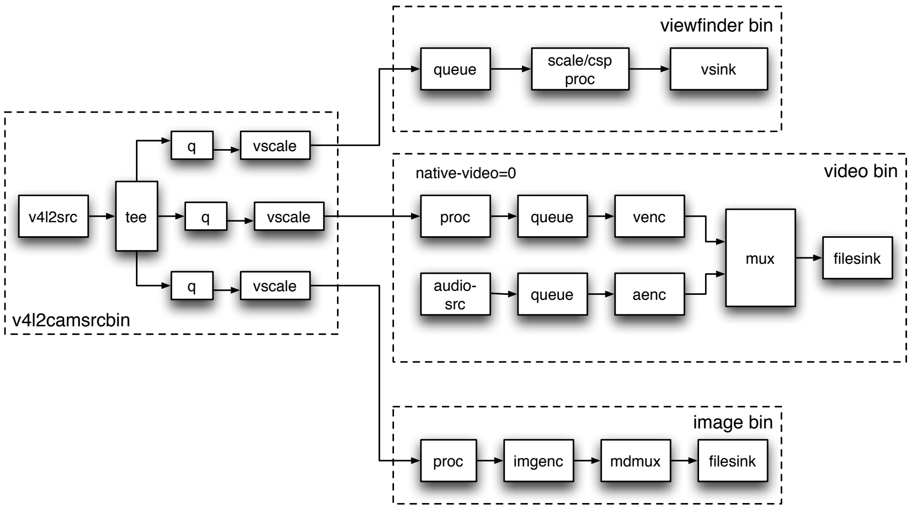
- Caps:  video/x-raw-yuv-strided
  - But would be good to combine this with stereo and better interlaced support
  - Support for rgb/gray via pseudo-fourcc's would be nice cleanup too
    - video/x-raw ?
  - http://gstreamer.freedesktop.org/wiki/NewGstVideo
- Events: GST_EVENT_CROP
  - Downstream serialized event to pass cropping information to display and encoders
  - No solution yet for VC-1 range mapping or VP6 mirroring
  - No solution yet for non-destructive text/graphic overlays
  - Won't work properly if video-sink does not handle the crop event
- Queries: GST_QUERY_BUFFERS
  - Upstream query from video sink to get minimum number to request, for given caps, the minimum number of buffers

# Ideas to better handle postproc functions

- Introduce interface(s) for postproc functions
  - If video sink does not implement the interfaces, playbin2 can plug a sw fallback element
  - What about camera scenarios with a tee element?  If display supports cropping but encoder does not or vice versa?
- Or a query to find which events are supported by downstream elements?
  - Pros: easy to extend with new events later
  - Cons: handling in case of tee isn't quite right.. We need to know if *any* branch of the tee cannot perform particular postproc functions on its own
- Or just put it all in the caps negotiation
  - Pros: existing negotiation mechanism to determine if buffer consumer can perform postproc functions, or if fallback to sw element is required
  - Cons: caps get bigger and bigger; difficult to extend in the future
  - Maybe a GParamFlags with bitmask to define avail postproc functions?

# OMAP4 Camera

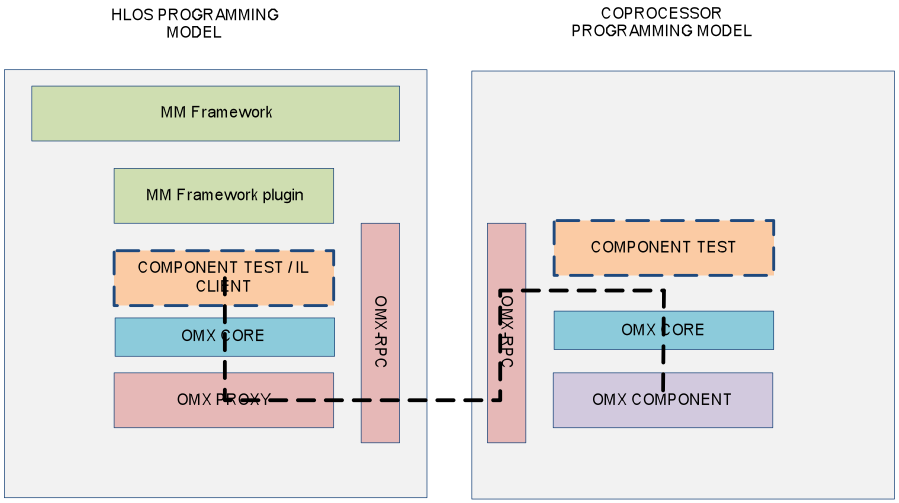
- For playback, pipeline is well abstracted by playbin2
  - Something similar is needed for camera/capture: camerabin
  - But camerabin is currently too limited, and not defacto standard (yet)
- Solution is to keep enhancing camerabin:
  - Split capture buffer "plumbing" from image and video encode pipelines:
  - And autoplug highest ranked camsrcbin… fallback to v4l2camsrcbin
  - An OMAP4 specific camsrcbin would expose enhanced ISS features: simultaneous video/jpeg capture, VSTAB, 3A, LDC, face detect, etc
    - Most already in photography interface
  - Autoplug encoders based on application requested caps filters
  - See: http://dev.omapzoom.org/?p=gstreamer/gst-plugins-bad.git;a=shortlog;h=refs/tags/L24.10



# OpenMAX on OMAP4

# Overview of OpenMAX on OMAP4

- Distributed OpenMAX (domx)
  - An RPC shim for an IL client on Chiron to use an OMX IL component on Ducati
- Design goals
  - Fully transparent to IL-Client and OMX components
  - Work with OMX-Core available in the system
  - Symmetric framework
  - Distributed implementation
- Features
  - Supports remote execution of OpenMAX IL 1.X components on AppM3 transparently from Chiron in **Non-tunnel mode**
  - Supports Multiple instances of OMX component
  - Supports TILER allocated buffers (both paged mode and 2D buffers in NV12 format)
  - Supports new buffers to be used at runtime without preannouncement.
  - Using optimized RCM modes for callbacks
  - Manages buffer mapping, cache coherence



# OpenMAX Buffer Passing (1/2)

- To strictly obey the OpenMAX spec would require buffers to be memcpy'd
  - OpenMAX buffers are pre-negotiated before transition to executing
  - But GStreamer does not give the decoder/encoder any way to know number of buffers or access buffer data ptr before transition to idle (OMX_UseBuffer())
- A solution will be part of OpenMAX 1.y: non-pre-announced (NPA) buffers
  - Pass NULL in on OMX_UseBuffer() call, and then free to reassign pBuffer pointer
  - This is what is used on OMAP4 gst-openmax branch
- But this introduces a problem of reference counting

# OpenMAX Buffer Passing (2/2)

- Codecs and Locked Buffers
  - To avoid an internal memcpy, the decoders will lock a buffer to use as a future reference frame
  - But then also return the buffer to be displayed
  - With NPA there is no longer any guarantee that the IL client (ie. GStreamer) will not free, reuse, or write to the buffer that the decoder is still holding
- Solution: custom buffer flag and event
  - A readonly flag on the returned buffer triggers gst-openmax to increment the refcnt of the corresponding GstBuffer
  - A corresponding refcount event is used to inform when the buffer is no longer used by the codec, which triggers gst-openmax to unref the corresponding GstBuffer

# OpenMAX Buffer Padding

- Need a way to indicate to IL client the actual size of buffer vs region of interest
- Current solution is a bit messy (so don't consider as final solution)
  - Use OMX_TI_IndexParam2DBufferAllocDimension to retrieve the required buffer size and alignment
  - While OMX_IndexParamPortDefinition on output port still indicates actual size of video picture within larger padded buffer
    - Fails ungracefully with IL client not aware of custom param
  - nOffset gives offset to valid picture within frame
    - nStride = top * nStride + left ← for NV12
- Preferred solution:
  - Set width/height from caps on input port (OMX_IndexParamPortDefinition)
  - Retrieve padded nFrameWidth/Height and nBufferAlignment on output port
  - Introduce OMX_IndexConfigRegionOfInterest to retrieve cropping