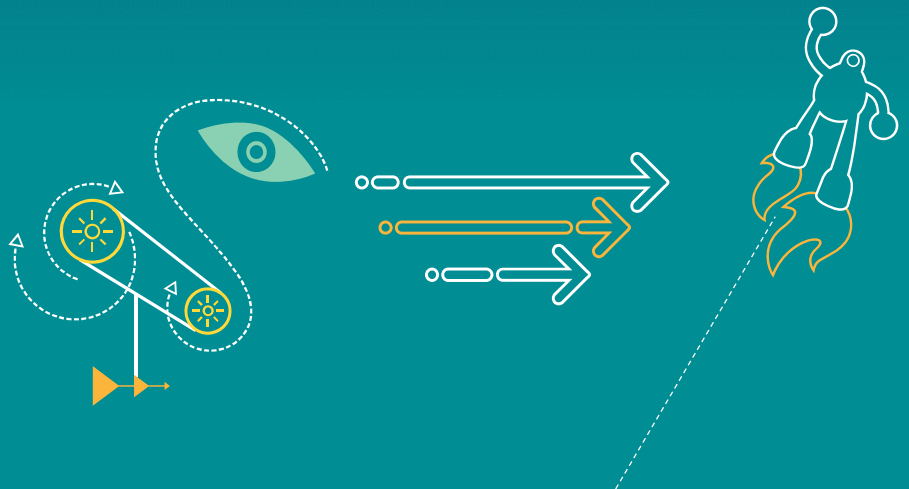
MSM8974/APQ8074/MSM8x26/ APQ8084 Linux Camera Overview

Qualcomm Technologies, Inc.



80-NA157-22 G

Confidential and Proprietary – Qualcomm Technologies, Inc.

NO PUBLIC DISCLOSURE PERMITTED: Please report postings of this document on public servers or websites to: DocCtrlAgent@qualcomm.com.

Not to be used, copied, reproduced, or modified in whole or in part, nor its contents revealed in any manner to others without the express written permission of Qualcomm Technologies, Inc.

Qualcomm Krait is a product of Qualcomm Technologies, Inc. Other Qualcomm products referenced herein are products of Qualcomm Technologies, Inc. or its subsidiaries.

Qualcomm is a trademark of Qualcomm Incorporated, registered in the United States and other countries. Qualcomm is a trademark of Qualcomm Incorporated, registered in the United States and other countries. Krait is a trademark of Qualcomm Incorporated. All Qualcomm Incorporated trademarks are used with permission. Other product and brand names may be trademarks or registered trademarks of their respective owners.

This technical data may be subject to U.S. and international export, re-export, or transfer (“export”) laws. Diversion contrary to U.S. and international law is strictly prohibited.

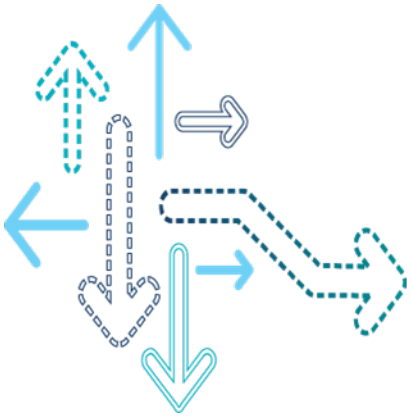
Qualcomm Technologies, Inc.
5775 Morehouse Drive
San Diego, CA 92121
U.S.A.

Revision History

Revision	Date	Description
A	Aug 2012	Initial release
B	Jun 2013	Updated
C	Sep 2013	Updated for APQ8084
D	Oct 2013	Updated software architecture details
E	Nov 2013	Updated slides 11, 12, 18, 19, 28, 29, and 77
F	Aug 2014	Added slides 42-44, 54-56, and 59; updated slides 34 and 51.
G	Jun 2015	Updated slide 13

Contents

Introduction



Objectives

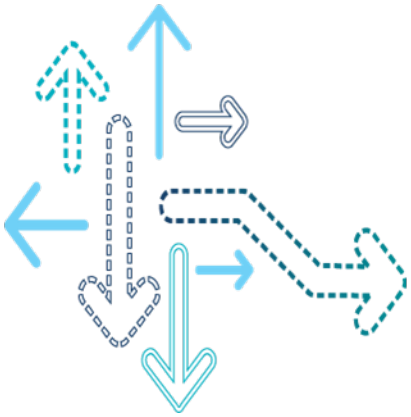
- At the end of this presentation, you will be able to understand the:
  - MSM8974 camera system features
  - Android camera system architecture, including the camera stack
  - Camera use case data flows
  - Native camera software, including the hardware and software architecture
  - Source code references
  - Logging methods

Note: Overall camera software architecture for APQ8084 and MSM8x26 is similar to that on MSM8974/ APQ8074, with the differences in supported feature set supported due to the hardware delta indicated in this presentation. For specifics on MSM8x26, see *Presentation: MSM8x26 Linux Android Multimedia Software Overview* (80-NF043-1). For simplicity, we would use the term "MSM8974" in this presentation for the baseline hardware and software architecture on these chipsets.

Scope

- Scope
  - Supported chipsets – MSM8974
  - Operating system – Android
- Prerequisite
  - MSM8974 architecture overview
  - Basic understanding of Android system and its camera software stack

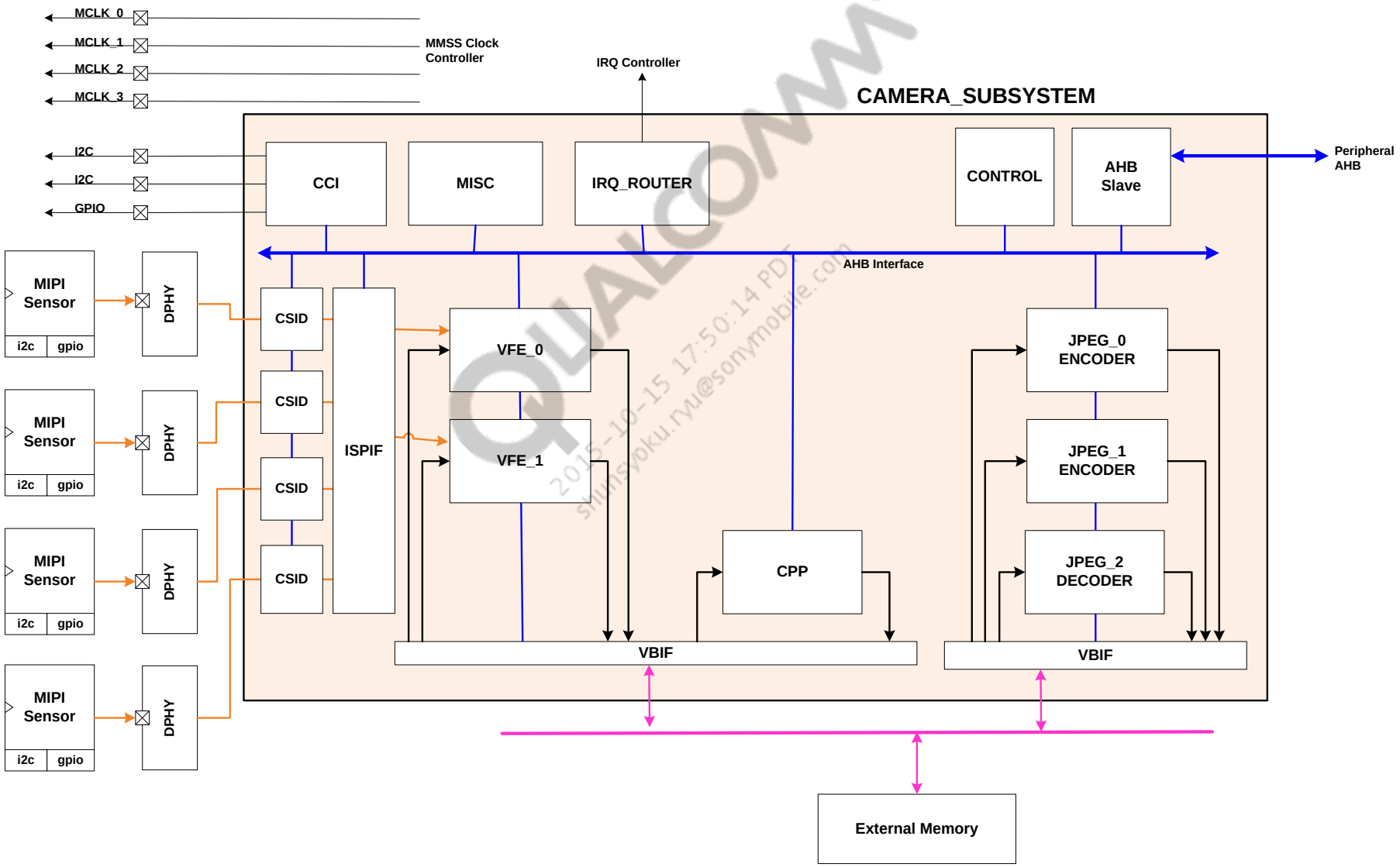
System Architecture



Major Hardware Blocks Used

- Apps processor (Qualcomm® Krait™)
  - Hosts Linux OS
  - Loads and executes camera driver software that controls all camera-related hardware blocks, sets up and controls clocks, as well as configures GPIO for camera sensor
- Camera Control Interface (CCI)
  - Used for dedicated I2C for camera subsystem
- Video Front End (VFE4)
  - Image processing hardware for processing camera frame data
  - MSM8974 has two VFEs (one with ~21 MP maximum, and one with ~13.5 MP maximum input)
  - VFE4 maximum clock is 266 MHz in Normal mode and 320 MHz in Turbo mode
- Offline JPEG encoder
  - Hardware block for JPEG encoding
  - MSM8974 has two JPEG encoder hardware
  - JPEG encoder maximum clock is 266 MHz in Normal mode and 320 MHz in Turbo mode
- Camera Postprocessor (CPP)
  - Supports flip/rotate, denoise, smooth/sharpen, crop, upscale features for full-size VFE output frames
- JPEG decoder hardware
  - Hardware block for JPEG decoding
  - Performance – 166 Mpps

MSM8974 Camera Hardware Subsystem Top-Level Diagram



Camera ISP Comparison – MSM8960 vs MSM8974

	MSM8960	MSM8974
Number of ISPs	1 (21 MP)	2 (21 MP, 13.5 MP)
Maximum output sizes (constraint on width)	▪ Enc – 5376x4032 ▪ View – 1920x1080 + 10%	▪ ISP1 Enc – 5376x4032 ▪ ISP1 View – 2560x2048 + 10% ▪ ISP2 Enc – 4288x3216 ▪ ISP2 View – 2560x2048 + 10%
Maximum sensor input PCLK	320 MHz, 1 Gbps/ln MIPI (DPHY 1.0)	320 MHz per VFE 1.5 Gbps/ln MIPI (DPHY 1.1)
Number of simultaneous cameras supported	3 (1 Bayer, 2 SOCs)	4 (2 Bayer/YUV via VFEs, 2 via RDI)
Camera port MIPI lane configuration	4/2; 2/1/1 lane modes	4/4/1/1; 4/4/2/1; 4/4/4 modes
Sensor formats supported	▪ Raw – 12-bit Bayer ▪ Processed – YUV ▪ DPCM8, DPCM6 ▪ CSID2.0 – 1 pix, 3 raw, 4 VC/DT, 3D merge	▪ Raw – 12-bit Bayer ▪ Processed – YUV ▪ DPCM8, DPCM6 ▪ CSID3.0 – 2pix, 3raw, 4 VC/DT, 3D merge (x2)
Special operational modes	High-speed raw capture, ideal raw, embedded status line, native 3D, concurrent camera support	High-speed raw capture, ideal raw, embedded status line, native 3D, concurrent camera support, dual ISP
JPEG hardware codec	▪ EncHW – 266 MPps ▪ DecHW – 50 MPps	▪ 2x 320 MPps each (>600 MPps total) ▪ DecHW – 166 MPps
Hardware postprocessing	NA	CPP (Wavelet denoise→Upscale→ASF sharpening→Rotation)

Differences Between MSM8974, APQ8084, and MSM8x26

	APQ8084	MSM8974 v2	MSM8x26
Number of ISPs	Same as MSM8974	2 (21 MP, 13.5 MP)	1 (12 MP)
Maximum output sizes (constraint on width)	Same as MSM8974	▪ ISP1 Enc – 5376x4032 ▪ ISP1 View – 2560x2048 + 10% ▪ ISP2 Enc – 4288x3216 ▪ ISP2 View – 2560x2048 + 10%	▪ Enc – 4288x3216 ▪ View – 1440x900 + 10%
VFE and CPP max speed (nominal/high clock)	465 MHz/600 MHz	▪ V2.1 – 266 MHz/320 MHz ▪ V2.2 – 266 MHz/400 MHz	266 MHz/320 MHz
Number of PHYs	Same as MSM8974	3x combo PHY	1x combo PHY, 1x 2-lane PHY
Number of CSIDs	Same as MSM8974	4	2
Number of simultaneous cameras supported	Same as MSM8974	4 (2 Bayer/YUV via VFEs, 2 via RDI)	2 (1 Bayer/YUV via VFE, 1 via RDI)
Camera port MIPI lane configuration	Same as MSM8974	4/4/1/1; 4/4/2/1; 4/4/4 modes	4/1
3D merging support	Yes	Yes	No
JPEG	Same as MSM8974	2x JPEG encoder hardware, 1x JPEG decoder hardware	1x JPEG encoder hardware
VFE/CPP changes introduced in APQ8084	LSC moved after ABF, 40x30 grid LSC, larger ABF kernel, improved demosaic, dual AF kernels for BF stats, hardware chroma aberration correction, local tone mapping; improved ASF with level-based sharpening control	NA	NA
Hardware denoising	Yes	Yes	No

Note: See the APQ8084 Camera Subsystem Enhancements Compared to MSM8974 section for APQ8084 specifics.

Camera Interfaces on MSM8974

- Up to 4 MIPI cameras can be connected to MSM8974/APQ8074 hardware
  - There are 3 combo PHYs and 4 CSIDs
  - Each combo PHY supports up to 4-lane configuration
  - Two PHYs can support up to 4-lane cameras each, and the third combo PHY can be configured to support either of the following
    - One camera of 4 lanes
    - One camera of 2 lanes and one camera of 1 lane
    - Two cameras of 1 lane each
  - Up to two cameras can be used concurrently, with each camera using one VFE

Maximum Data Rates in Normal Mode Operation

- For MSM8974, each VFE's maximum speed is 266 MHz in normal clock mode
- Each MIPI lane's PHY limitation is 1.5 Gbps ($1.5 * 10^9$ bps)
- Raw capture interface data rate does not depend on VFE clock
- On 4-lane PHY interface, maximum data per lane is (266 x bpp)/number of lanes; bpp→sensor output bits per pixel
  - For 12-bit data rate per lane – (266 * 12)/4 – 798 Mbps
  - For 10-bit data rate per lane – (266 * 10)/4 – 665 Mbps
  - For 8-bit data rate per lane – (266 * 8)/4 – 532 Mbps

Number of lanes	Raw interface data rate	Pixel interface data rate		
		8 bpp sensor output	10 bpp sensor output	12 bpp sensor output
4 lanes	6 Gbps (PHY limitation) (4 x 1.5 Gbps)	2128 Mbps (VFE limitation) (4 x 532 Mbps)	2660 Mbps (VFE limitation) (4 x 665 Mbps)	3192 Mbps (VFE limitation) (4 x 798 Mbps)
2 lanes	3 Gbps (PHY limitation) (2 x 1.5 Gbps)	2128 Mbps (VFE limitation) (4 x 532 Mbps)	2660 Mbps (VFE limitation) (4 x 665 Mbps)	3 Gbps (PHY limitation) (2 x 1.5 Gbps)
1 lane	1.5 Gbps (PHY limitation)	1.5 Gbps (PHY limitation)		

Note: If required, more data per camera can be processed by using both VFEs together. For the maximum data rate limitation on chipsets other than MSM8974, use the maximum clock value shown in slide 12.

Maximum Data Rates in High Clock Operation Mode

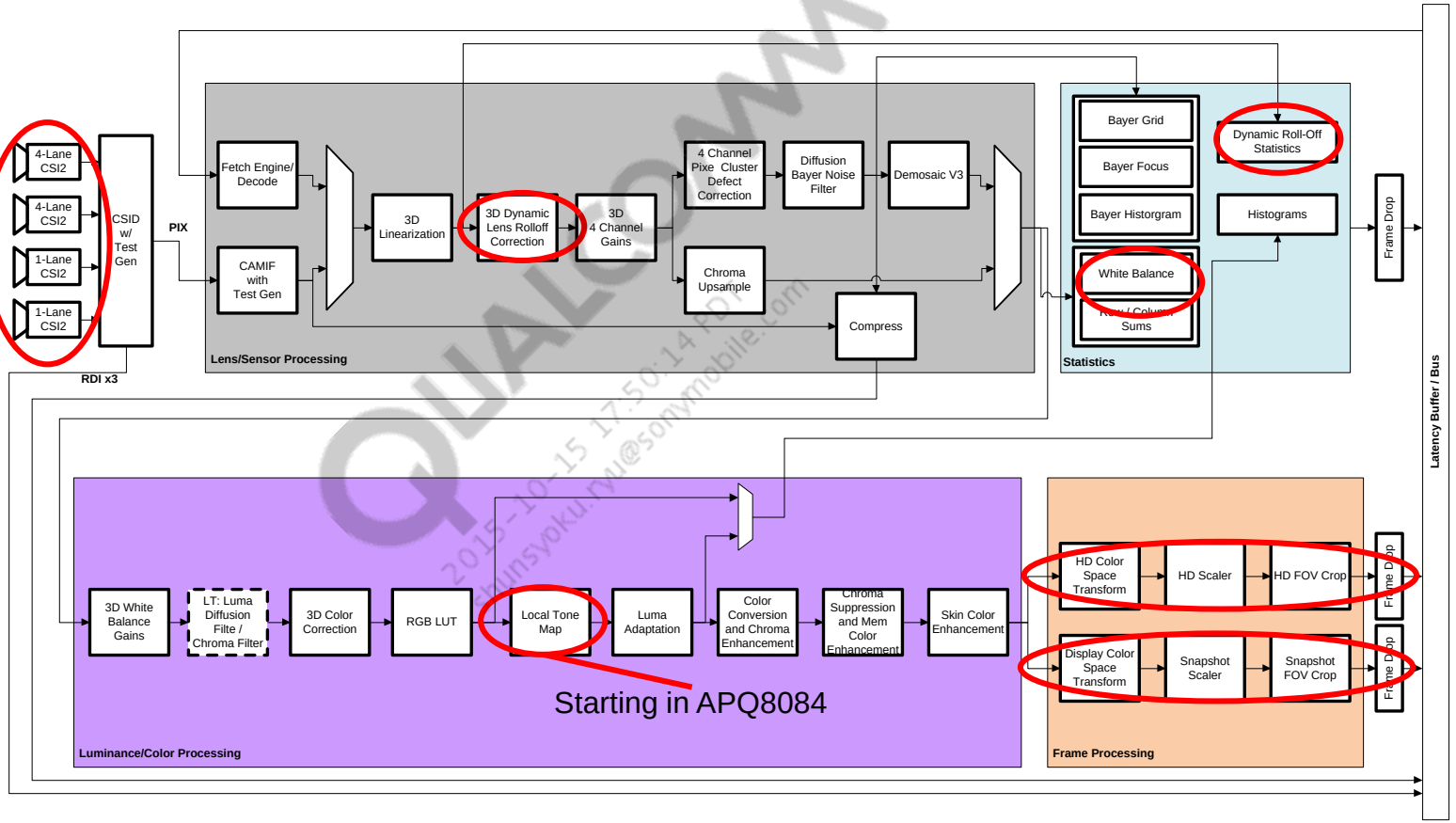
- For MSM8974, each VFE's max speed is 320 MHz in high clock mode
- On 4-lane PHY interface, maximum data per lane is (320 x bpp)/number of lanes; bpp→sensor output bits per pixel
  - For 12-bit data rate per lane – (320 * 12)/4 – 960 Mbps
  - For 10-bit data rate per lane – (320 * 10)/4 – 800 Mbps
  - For 8-bit data rate per lane – (320 * 8)/4 – 640 Mbps

Number of lanes	Raw interface data rate	Pixel interface data rate		
		8 bpp sensor output	10 bpp sensor output	12 bpp sensor output
4 lanes	6 Gbps (PHY limitation) (4 x 1.5 Gbps)	2560 Mbps (VFE limitation) (4 x 640 Mbps)	3200 Mbps (VFE limitation) (4 x 800 Mbps)	3840 Mbps (VFE limitation) (4 x 960 Mbps)
2 lanes	3 Gbps (PHY limitation) (2 x 1.5 Gbps)	2560 Mbps (VFE limitation) (4 x 640 Mbps)	3 Gbps (PHY limitation) (2 x 1.5 Gbps)	3 Gbps (PHY limitation) (2 x 1.5 Gbps)
1 lane	1.5 Gbps (PHY limitation)	1.5 Gbps (PHY limitation)		

Note: If required, more data per camera can be processed by using both VFEs together. For the maximum data rate limitation on chipsets other than MSM8974, use the maximum clock value shown in slide 12.

ISP Pipeline and Key Updates on MSM8974 Compared to MSM8960 ISP

VFE core



CPP core (new)

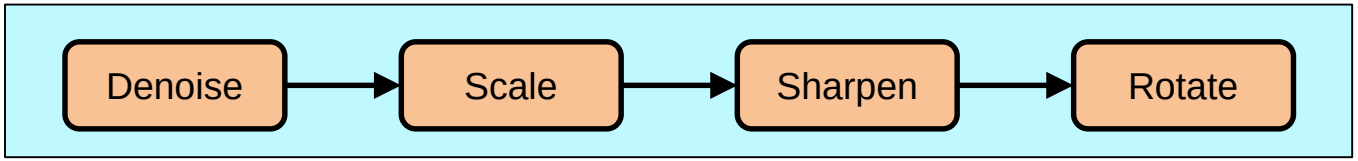
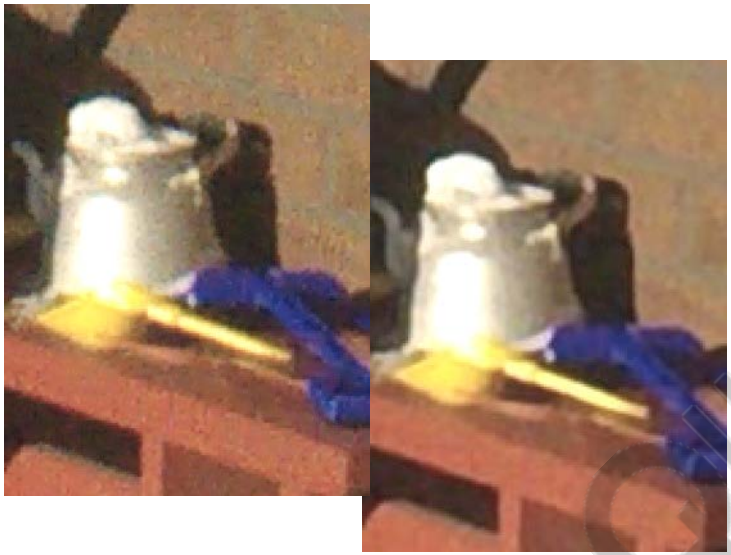
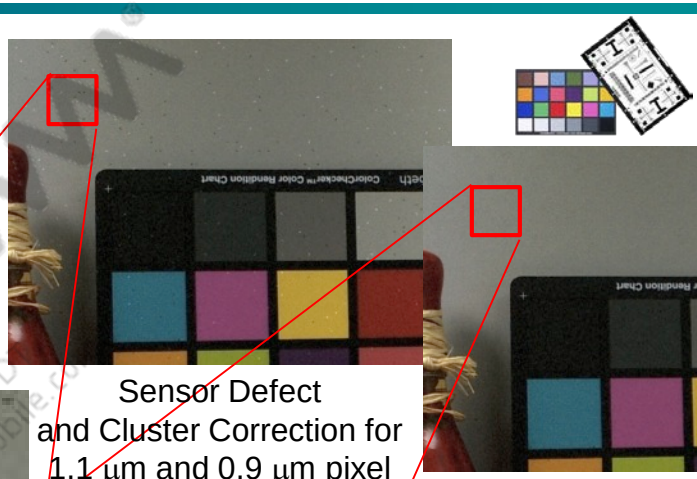


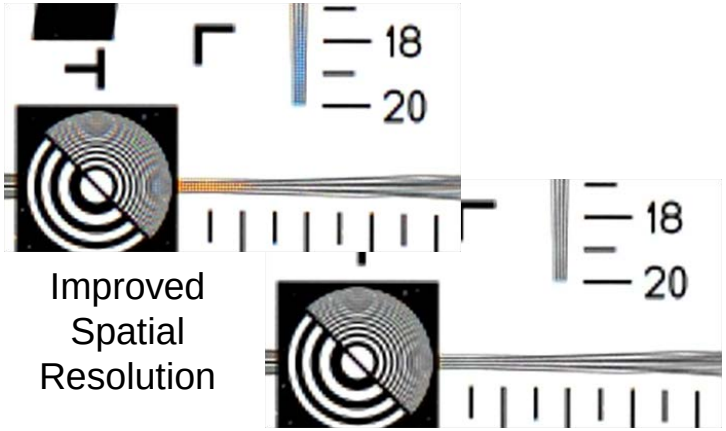
Image Quality Improvements in MSM8974 Camera



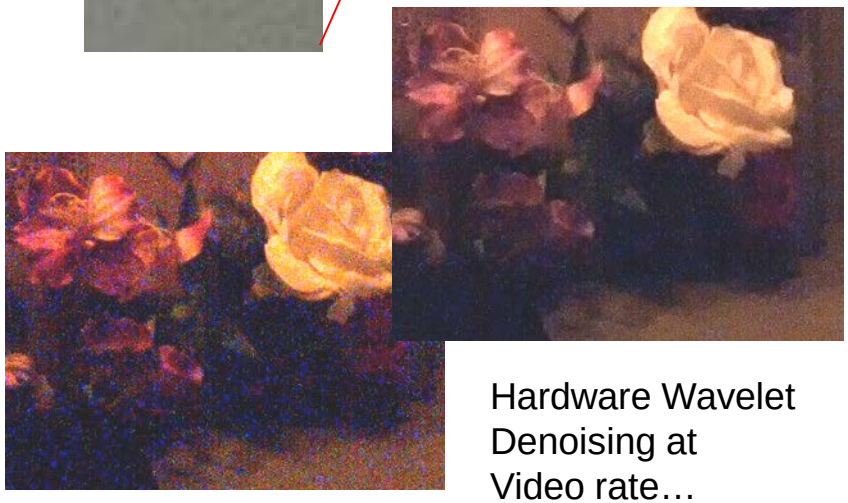
Improved Demosaic and Spatial Filter



Sensor Defect
and Cluster Correction for
1.1 μm and 0.9 μm pixel



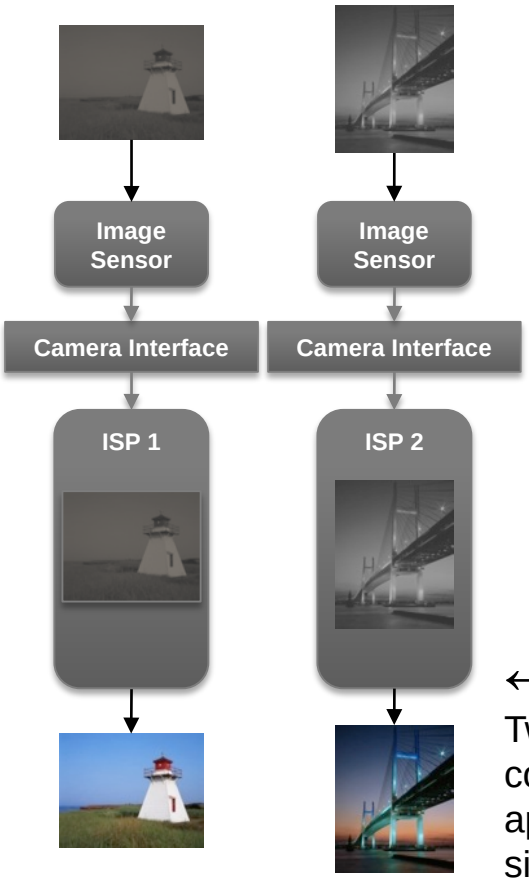
Improved
Spatial
Resolution



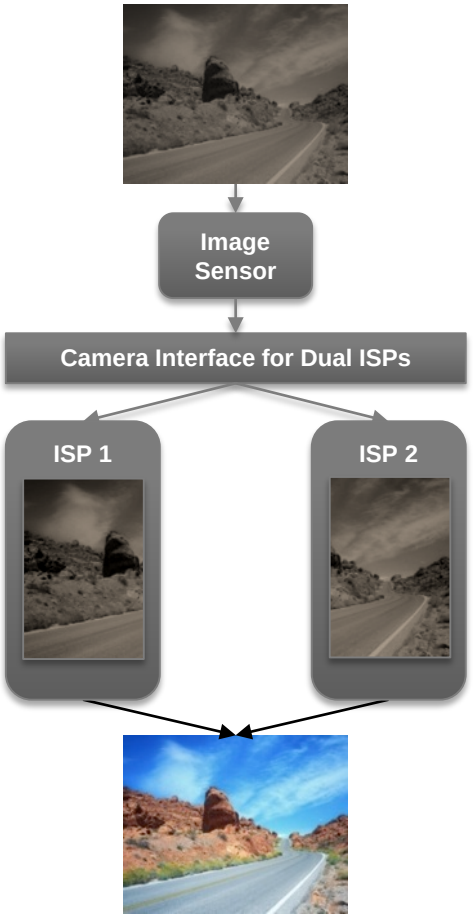
Hardware Wavelet
Denoising at
Video rate…

Performance Improvements in MSM8974 Camera

- Next Gen Camera Solution – Dual ISP
  - ISPs can be used in parallel for high throughput or used individually for concurrent processing
  - Single ISP capability ~320 MPps, Dual ISP up to >600 MPps throughput



Dual ISP Modes	Description
Concurrent front/ back cameras	Two camera previews operating concurrently
High performance	One camera using both ISP pipelines to support high MP sensors at high frame rates
3D imaging	Two cameras in tandem providing stereo 3D (Feature not available in software as of Nov'13)



← Two ISP pipelines for concurrent camera applications like simultaneous camera with videotelephony

→ Two ISP pipelines for increased pixel processing throughput; also supports much larger MP

Camera Hardware Responsiveness in MSM8974

- JPEG hardware encoder tracks PCLK/fps – 600 MPps
- Hardware acceleration for commonly used features
  - Hardware upscaling for digital zoom up to 8x
  - Hardware scaler/rotator
  - Powerful Wavelet NR moved to hardware
    - Tracks video rate
- Dedicated camera control interface
  - Dedicated I2C camera controller with FIFO command queue (nonblocking)
  - High granular control and dedicated camera precision I/O synchronized to sensor timing
    - Exposure
    - Mode switches
    - Mechanical shutter
    - Xenon flash
    - Focus

Updates in Lens Rolloff Correction Processing

- New lens rolloff correction module provides much denser mesh-based correction compared to MSM8960. Combined with new rolloff stats module, it can:
  - Correct spatial color tinting in images
  - Offer better performance than fixed-table calibration methods
  - Fix module-to-module color shading variation without per-unit calibration

MSM8974 Lens Rolloff Correction Example

Without correction

With new correction

Older ASF vs MSM8974 ASF Performance






Original image

Older ASF-processed image

MSM8974 ASF-processed image

Older ASF vs MSM8974 ASF Performance (cont.)

Original image

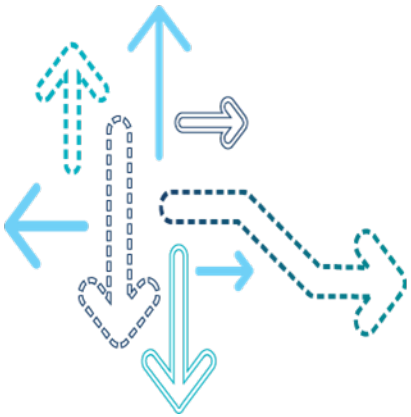
Older ASF-processed image

MSM8974 ASF-processed image

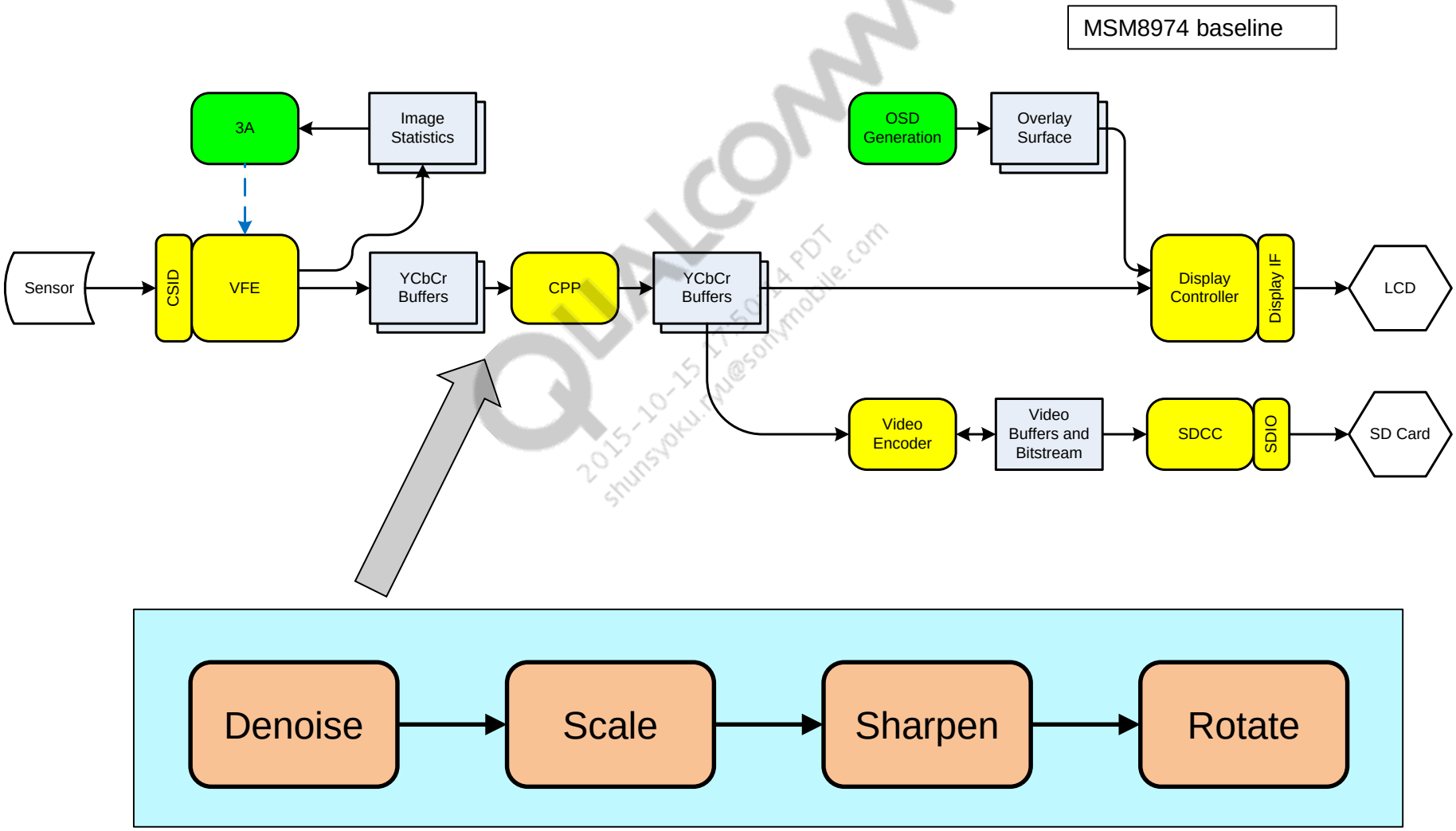
Hardware Noise Reduction in MSM8974

Hardware Noise Reduction in MSM8974 (cont.)

APQ8084 Camera Subsystem Enhancements Compared to MSM8974

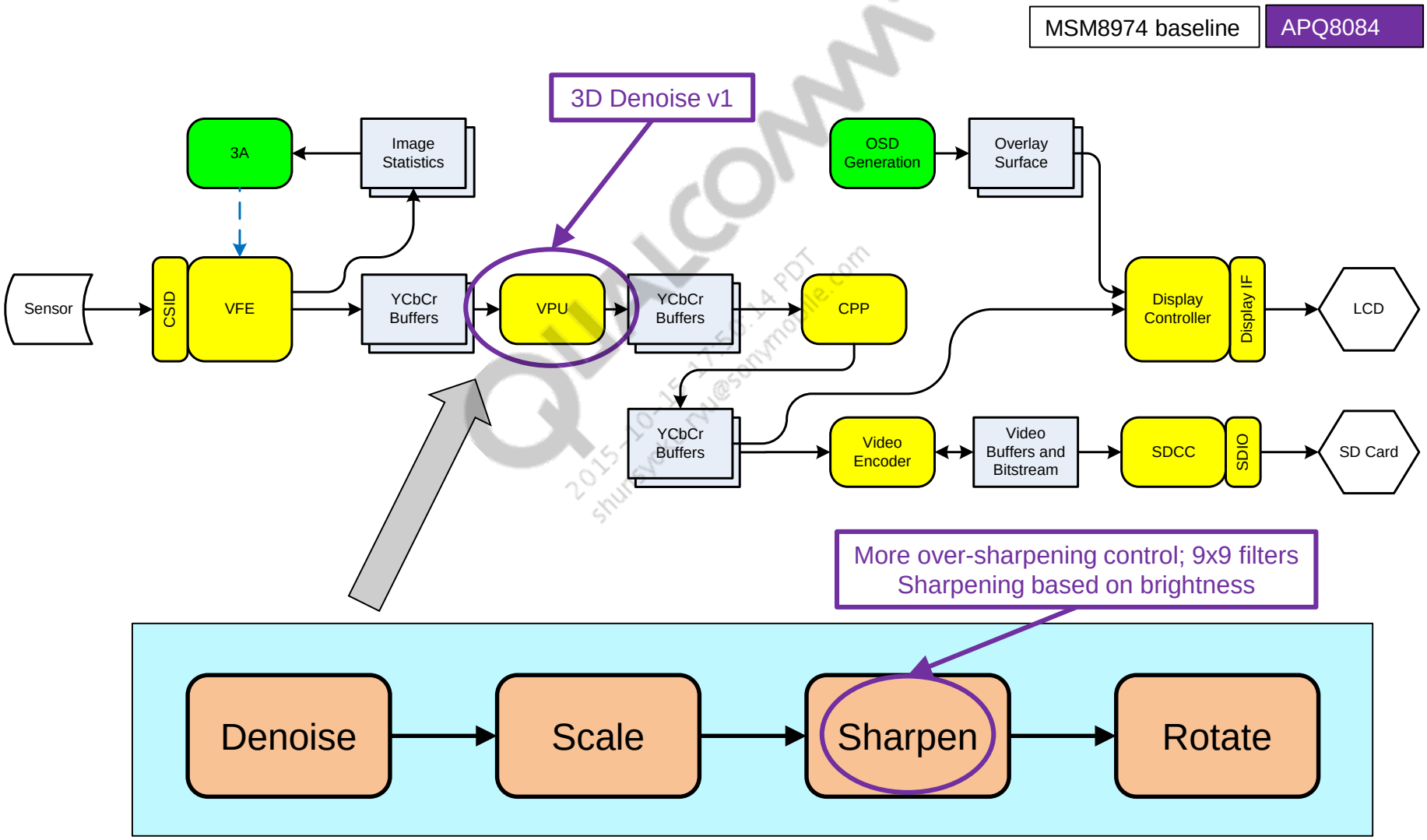


Additional Updates in APQ8084 Camera Subsystem



CPP core – 2D 3D denoising, scaling, sharpening, rotation

Additional Updates in APQ8084 Camera Subsystem (cont.)

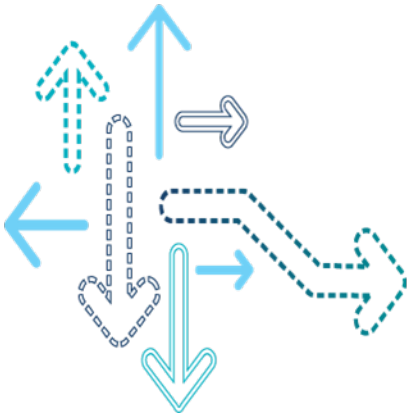


CPP core – 2D 3D denoising, scaling, sharpening, rotation

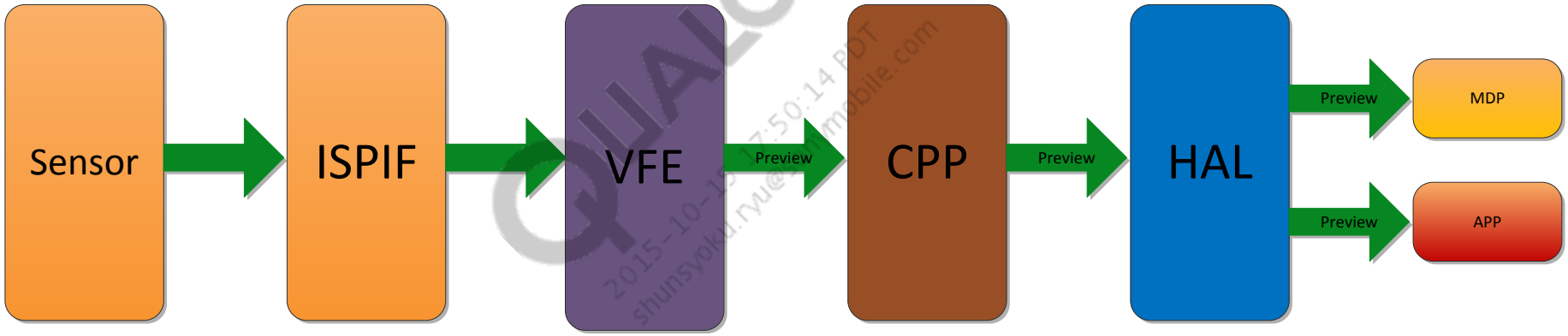
Additional Details for APQ8084 Camera Subsystem Improvements

- Improved demosaic to reduce color artifacts; better/smoother green pixel interpolation; directional interpolation for red and blue
- Mesh rolloff – 40x30 programmable grids; effective mesh of up to 320x240 with interpolation
  - MSM8974 has 12x9 programmable grids, 96x72 effective
- AF statistics
  - Double-buffered, dual AF kernels to analyze larger/finer details of the image at the same time and fast/accurate AF
  - Region of interest is divided into a grid with up to 18x14 regions; larger kernel allows better frequency response in handling large resolution sensors
- Chroma Aberration Correction (CAC) to suppress chrominance artifacts along edges
- Local tone mapping to boost shadow details, improve contrast and appearance of sharpness/dynamic range
- New ASF module, including level-based sharpening control
- Temporal denoise using new hardware block Video Processing Unit (VPU)

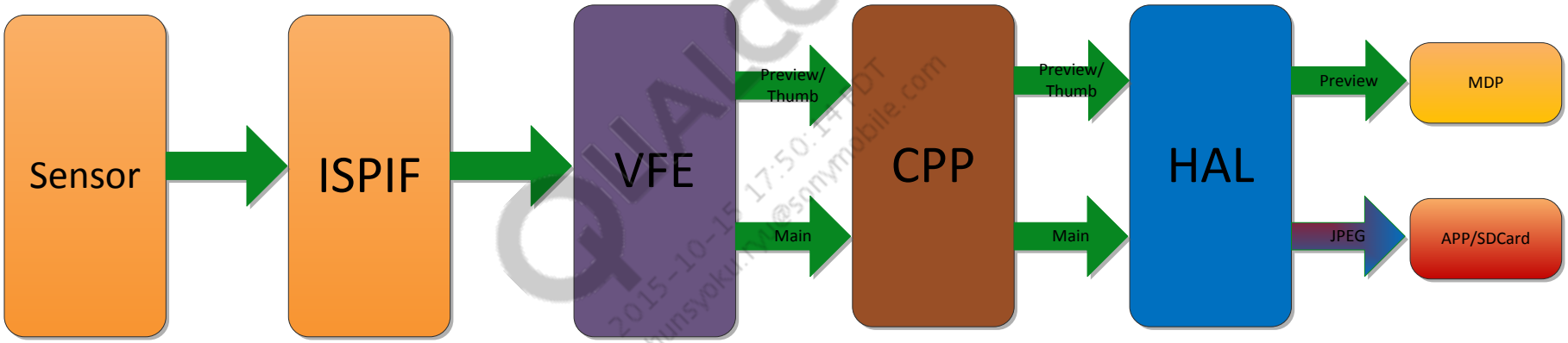
Data Flows



Data Flows – Camera Preview



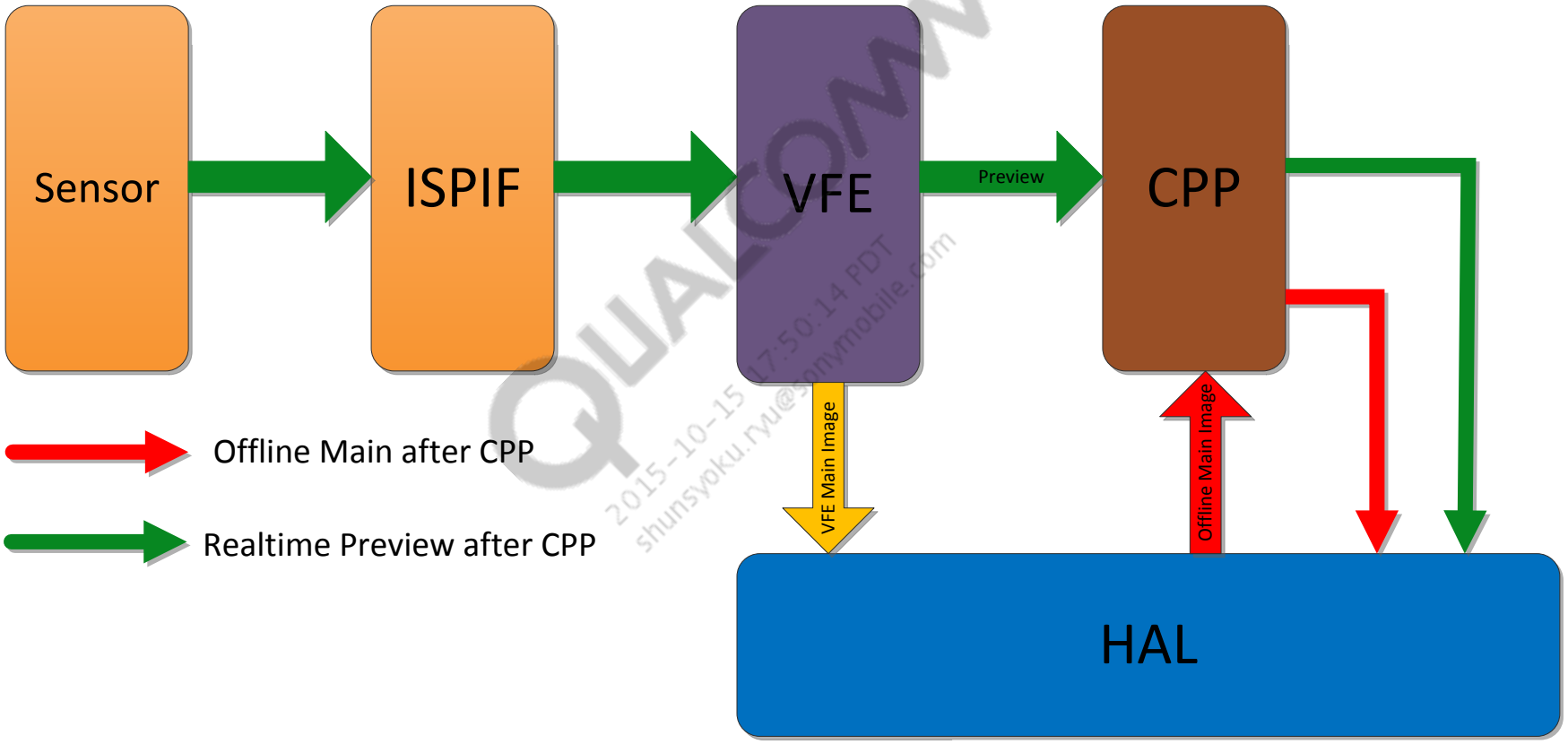
Data Flows – Camera Snapshot (non-ZSL)



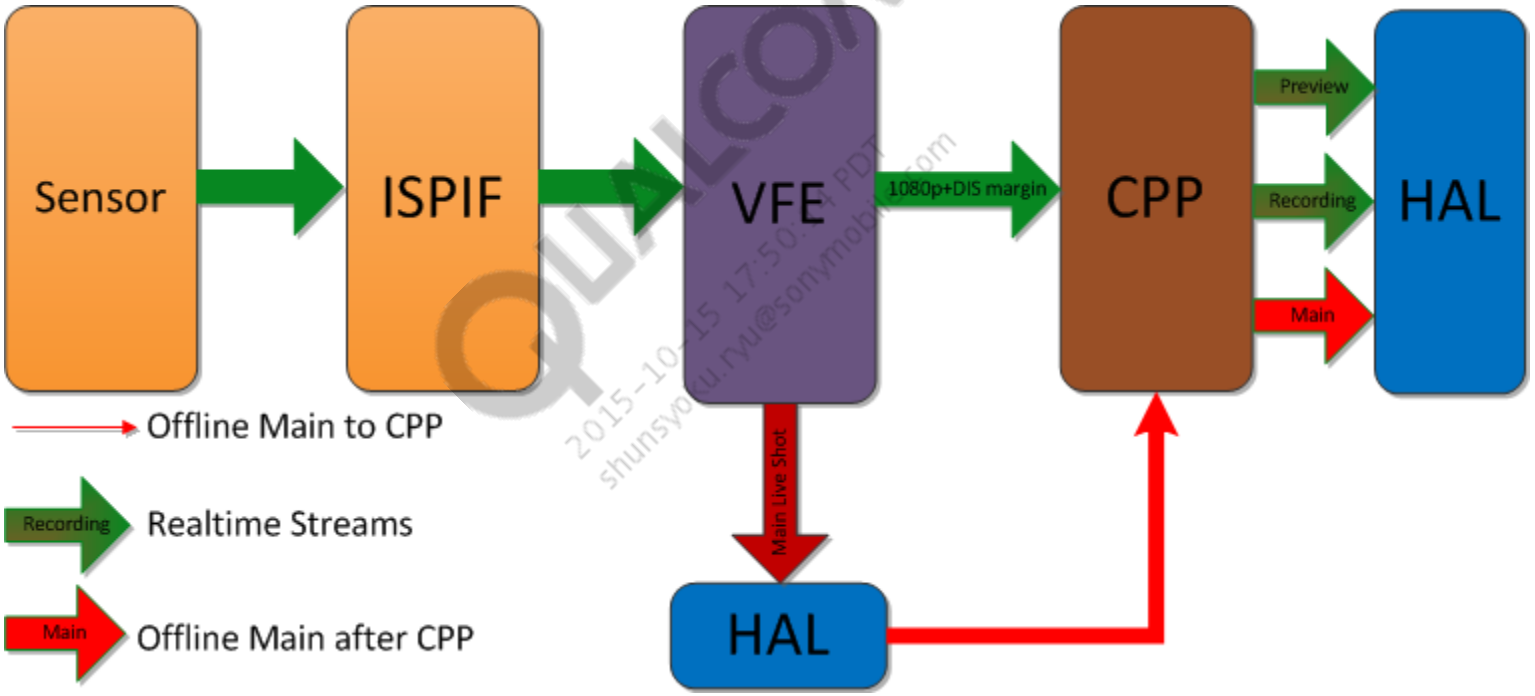
Notes:

- This is a legacy snapshot; preview is stopped while taking the snapshot.
- Preview will not be restarted before CPP finishes main/thumbnail processing.

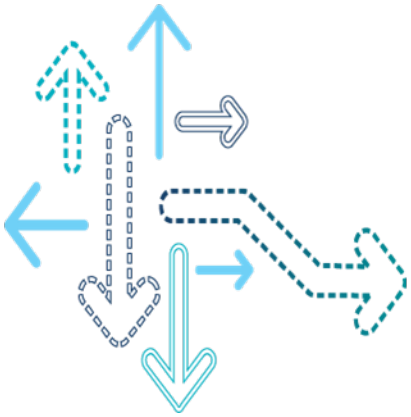
Data Flows – Zero Shutter Lag (ZSL)



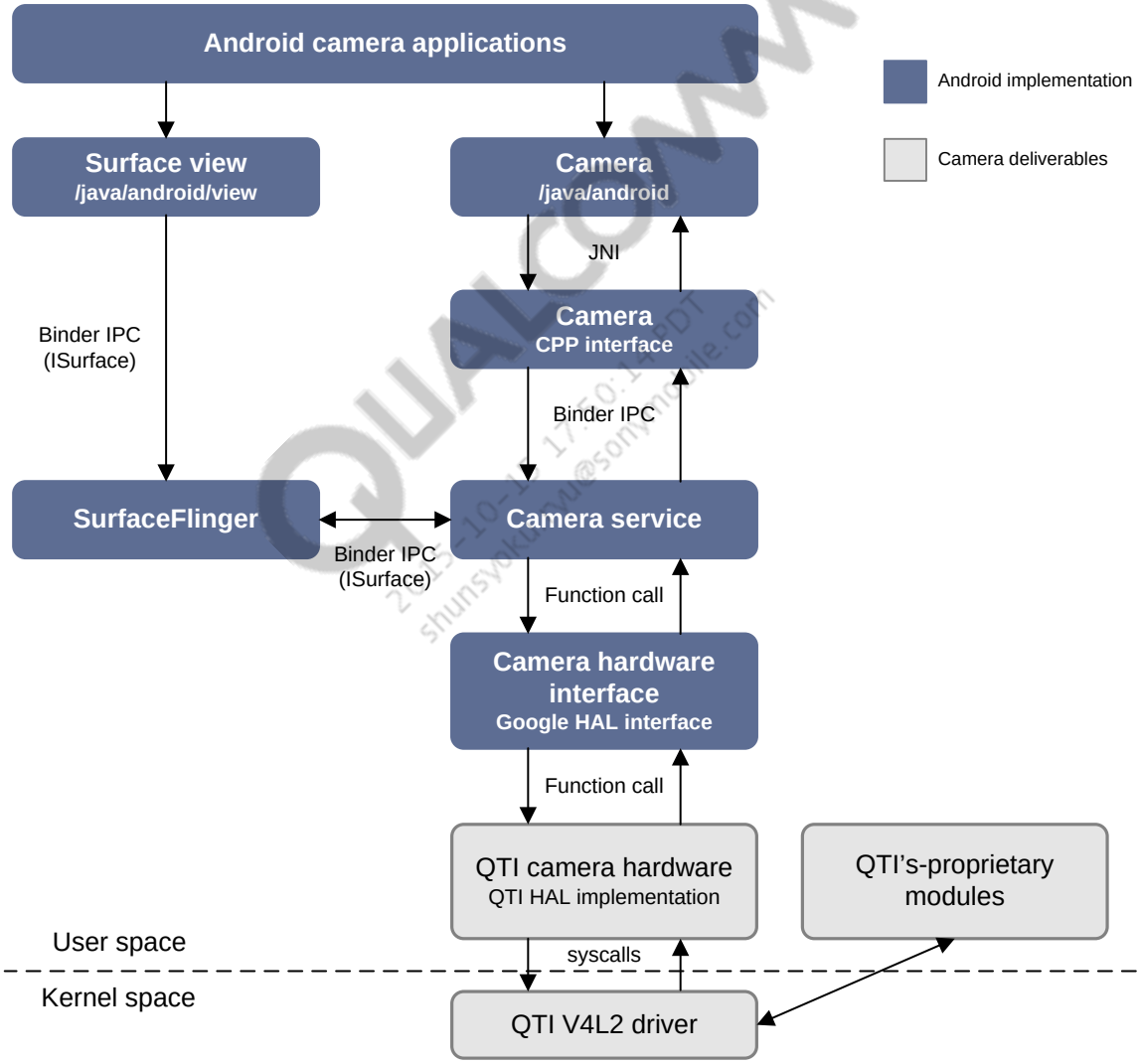
Data Flows – Camcorder



Software Architecture



Software Architecture – Android Camera



Android Camera Stack (Top to Bottom)

- android.hardware.Camera
  - Java object for camera operation
  - Thin wrapper of android_hardware_Camera.cpp
- android_hardware_Camera.cpp
  - JNI interface for camera operation
- frameworks/av/camera/Camera.cpp
  - Proxy for remote object in camera service layer
- ICamera.cpp/ICamera.h
  - Binder interface for IPC call marshaling code
- ICameraClient
  - Marshalized object appearing on camera service layer

Android Camera Stack (Top to Bottom) (cont.)

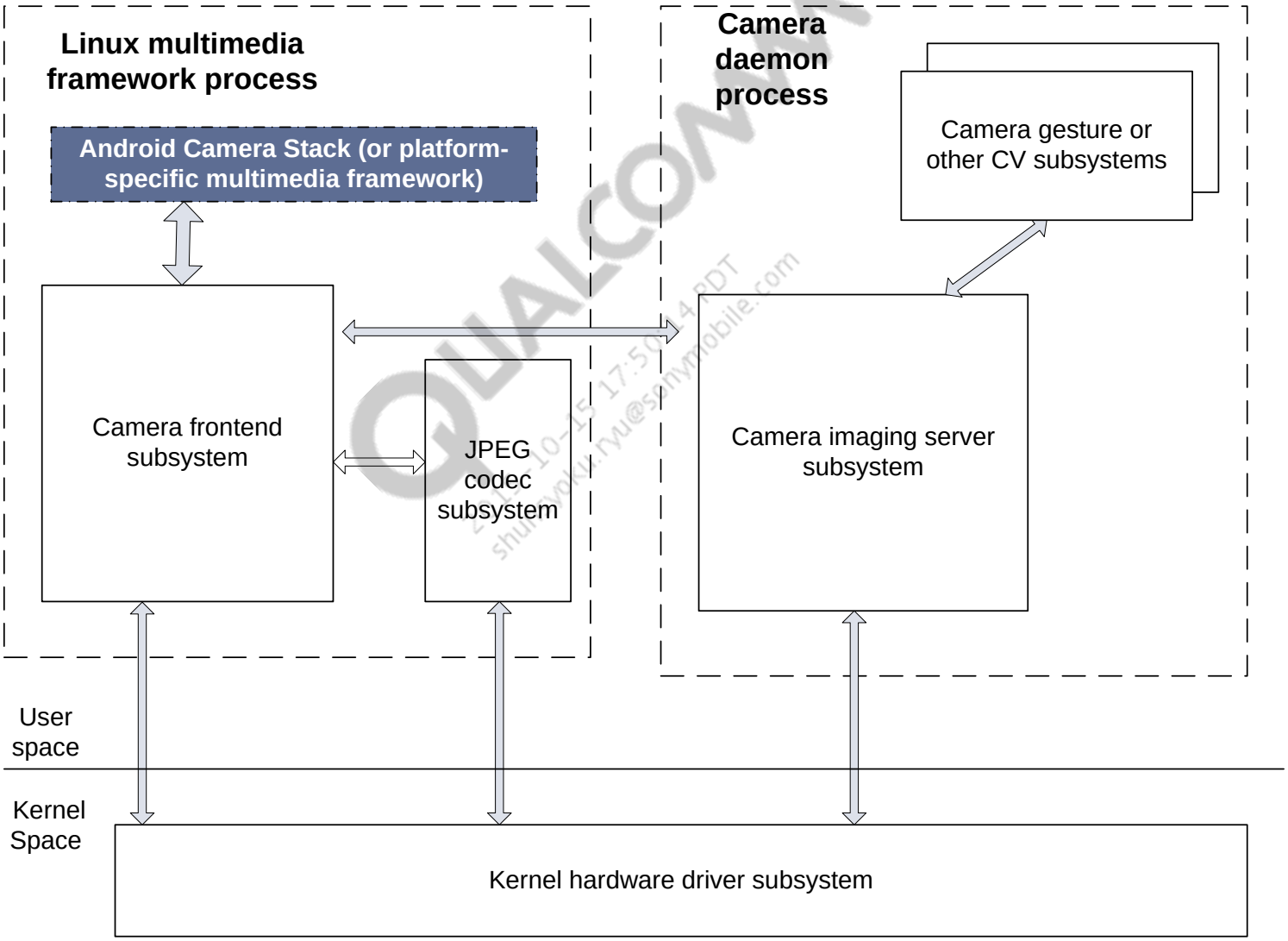
- CameraService (libcamerservice)
  - Camera service layer interacts with:
    - HAL for camera control APIs
    - SurfaceFlinger for delivering preview frames on LCD
- CameraHardwareInterface (derived object)
  - HAL for actual camera hardware
  - QTI's HAL implements Google HAL APIs to hook up QTI native camera driver in kernel
- QTI Linux V4L2 camera driver (kernel)
  - Controls access to camera hardware
  - Proxy to access the QTI-proprietary modules on user space

* **Note**: Refer *Presentation: Camera Frontend Code Walkthrough for MSM8974/APQ8074/MSM8x26 Linux Camera Software* (80-NF499-2) to *Video: Jpeg Code Walkthrough for MSM8974/APQ8074/MSM8x26 Linux Camera Software* (VD80-NF499-10) for deep dive training on QTI camera software

Linux Camera Software

- QTI's camera solution for Android platform delivers
  - Open source
    - Kernel space (V4L2-based image streaming driver framework)
      - All hardware drivers are implemented as V4L2 subdevices; those hardware subdevices can be open/closed and controlled in the daemon domain with system privilege access rights
      - Source code is under kernel\drivers\media\platform\msm\camera_v2
      - Dtsi files are under kernel\arch\arm\boot\dts\
      - Dtsi documentation is under kernel\Documentation\devicetree\bindings\media\video\
    - User space (QTI HAL source code)
      - Android uses this interface to integrate with QTI's camera subsystem
      - Source code is under hardware\qcom\camera\QCamera2
  - Proprietary
    - User space – All QTI-proprietary solutions run as a Linux daemon containing the following main modules:
      - Camera server
      - Camera Media Control (MCTL) framework
      - Individual modules for ISP, sensor, stats, post processing, etc.
    - Source code is under vendor\qcom\proprietary\mm-camera (mm-camera2 within this directory is used for MSM8974)
    - JPEG source code is under vendor\qcom\proprietary\mm-still

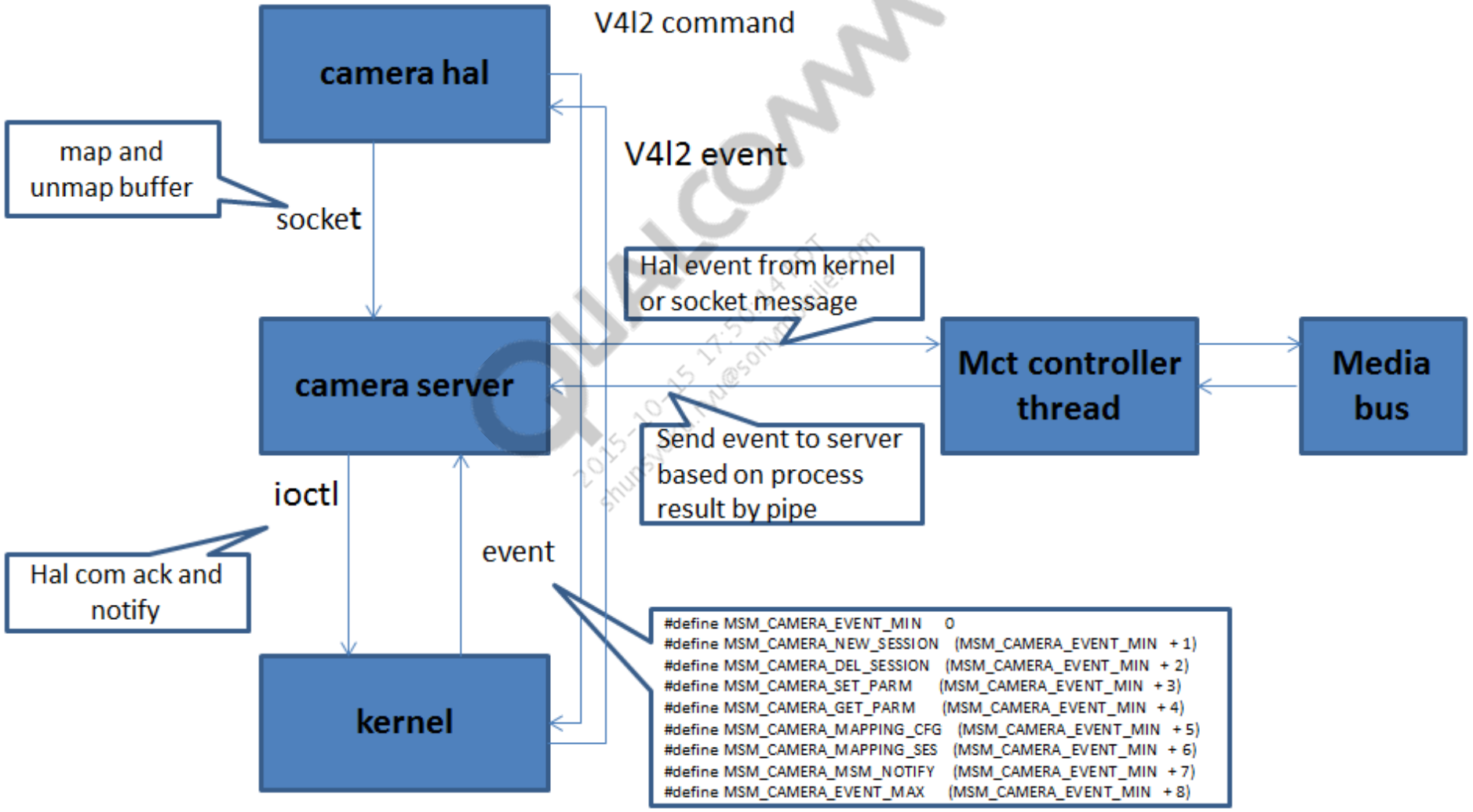
Software Architecture – Camera Architecture Overview



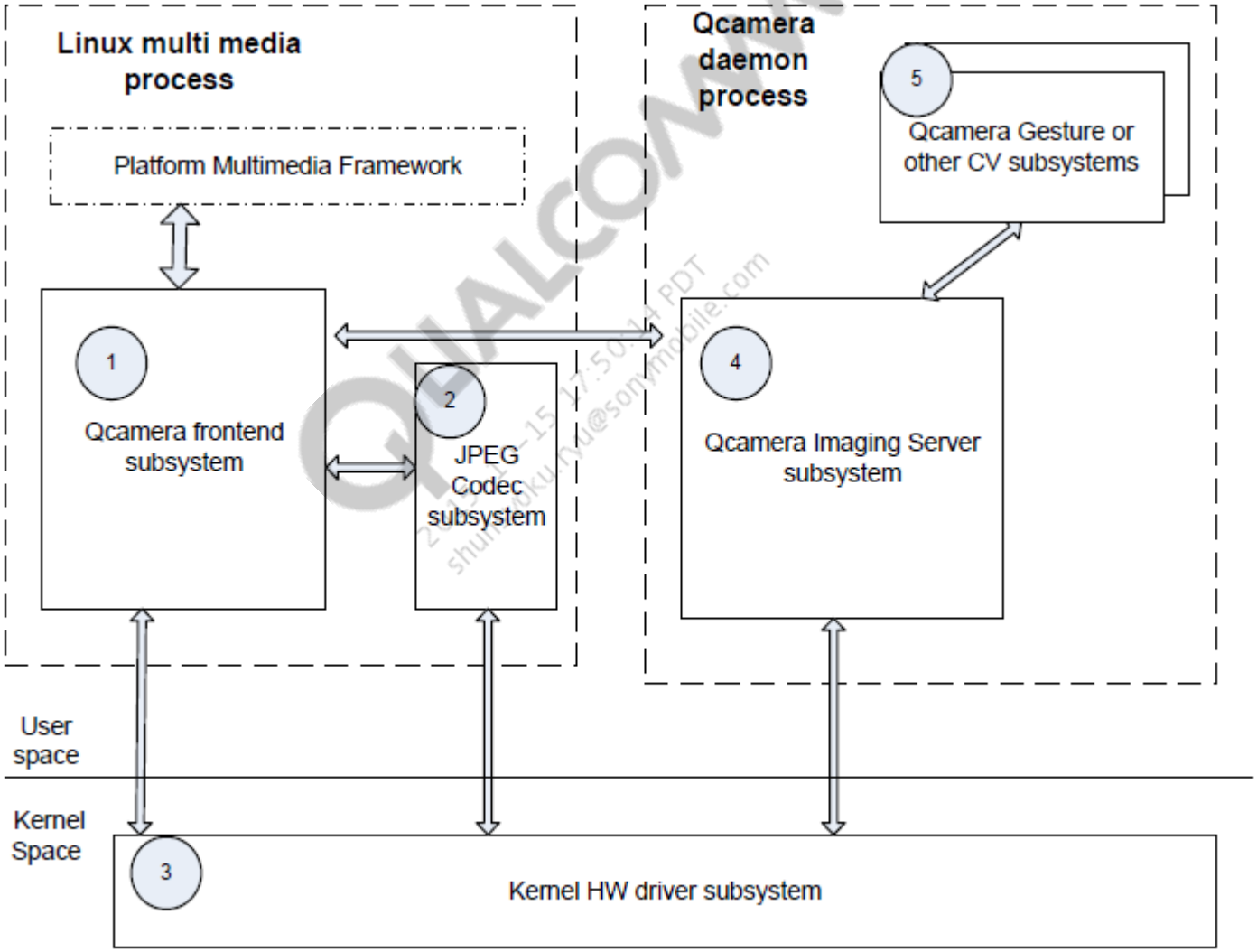
Memory Buffer Sharing Mechanism – Domain Socket

- Domain sockets are used to share buffer information via file descriptors between the media server (camera HAL) and mm-qcamera-daemon (mm-camera) processes.
- Types of buffer information exchanged:
  - CAM_MAPPING_BUF_TYPE_CAPABILITY – Camera capability buffer
  - CAM_MAPPING_BUF_TYPE_PARM_BUF – Camera parameters buffer
  - CAM_MAPPING_BUF_TYPE_STREAM_BUF – Stream buffers
  - CAM_MAPPING_BUF_TYPE_STREAM_INFO – Stream information buffer
  - CAM_MAPPING_BUF_TYPE_OFFLINE_INPUT_BUF – Offline reprocess input buffer
- Relevant source code pointers:
  - Socket implementation – QCamera2\stack\mm-camera-interface\src\mm_camera_sock.c
  - Message transmitter – QCamera2\stack\mm-camera-interface\src\mm_camera_stream.c
  - Message receiver – mm-camera\mm-camera2\server-imaging\server.c

Camera Communication



Camera Subsystems and Processes



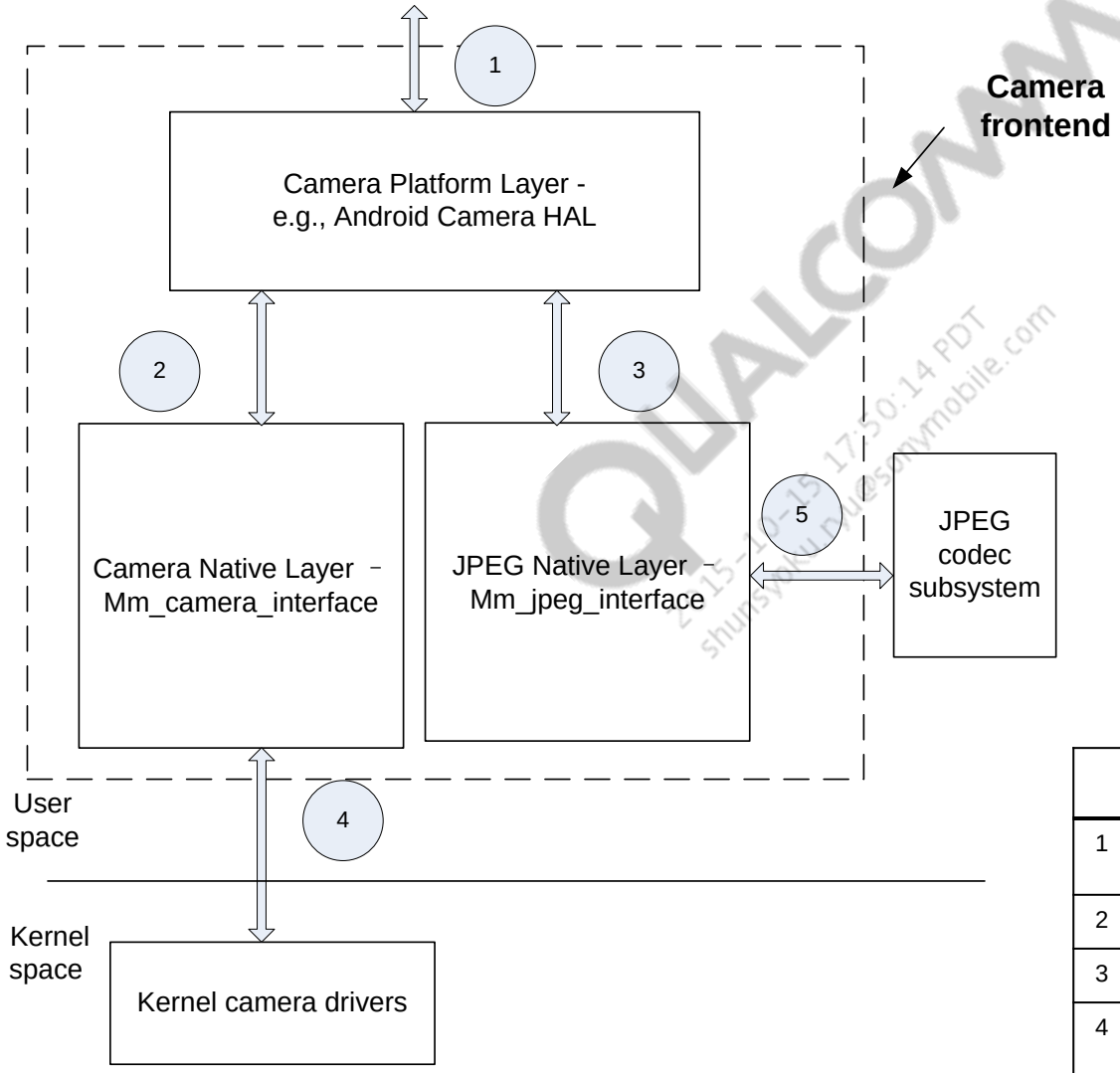
Camera Subsystems and Processes (cont.)

Subsystem ID	Subsystem name	Functionalities
1	Qcamera frontend/HAL	Provides adaptation layer to specific multimedia framework, such as camera HAL, for Android MMF Also, provides a native camera operation interface (independent from any platform)
2	Qcamera still codec	Provides JPEG encoding and decoding capabilities
3	Qcamera kernel driver framework	Provides Linux kernel driver and framework for hardware config, data flow management, and camera session control
4	Qcamera imaging server	Provides proprietary hardware configuration, 3A, and other imaging process functionalities
5	Qcamera gesture and other CV frameworks	Provides camera-based gesture detect and tracking functionalities

Each subsystem is executed by different processes on Linux:

- Subsystems 1 and 2 will be executed by the Linux platform's multimedia platform, e.g., the Android MediaServer process.
- Subsystem 3 will be executed by the Linux kernel.
- Subsystems 4 and 5 will be executed by the Qcamera Imaging Server Daemon; it is a standalone daemon process in the QuIC Linux platform that provides camera, imaging, gesture, and computer vision services.

Software Architecture – Camera Frontend



	Interface name	Description
1	Platform camera API (Android camera 1.0/2.0 API)	APIs defined by various MM platforms
2	mm_camera _interface	Native camera APIs
3	mm_jpeg _interface	Native jpeg codec APIs
4	V4L2 APIs	Standard video for Linux V2 APIs
5	OMX APIs	Standard OpenMAX APIs

Constructs/Concepts Used at HAL/mm-camera-interface Layers

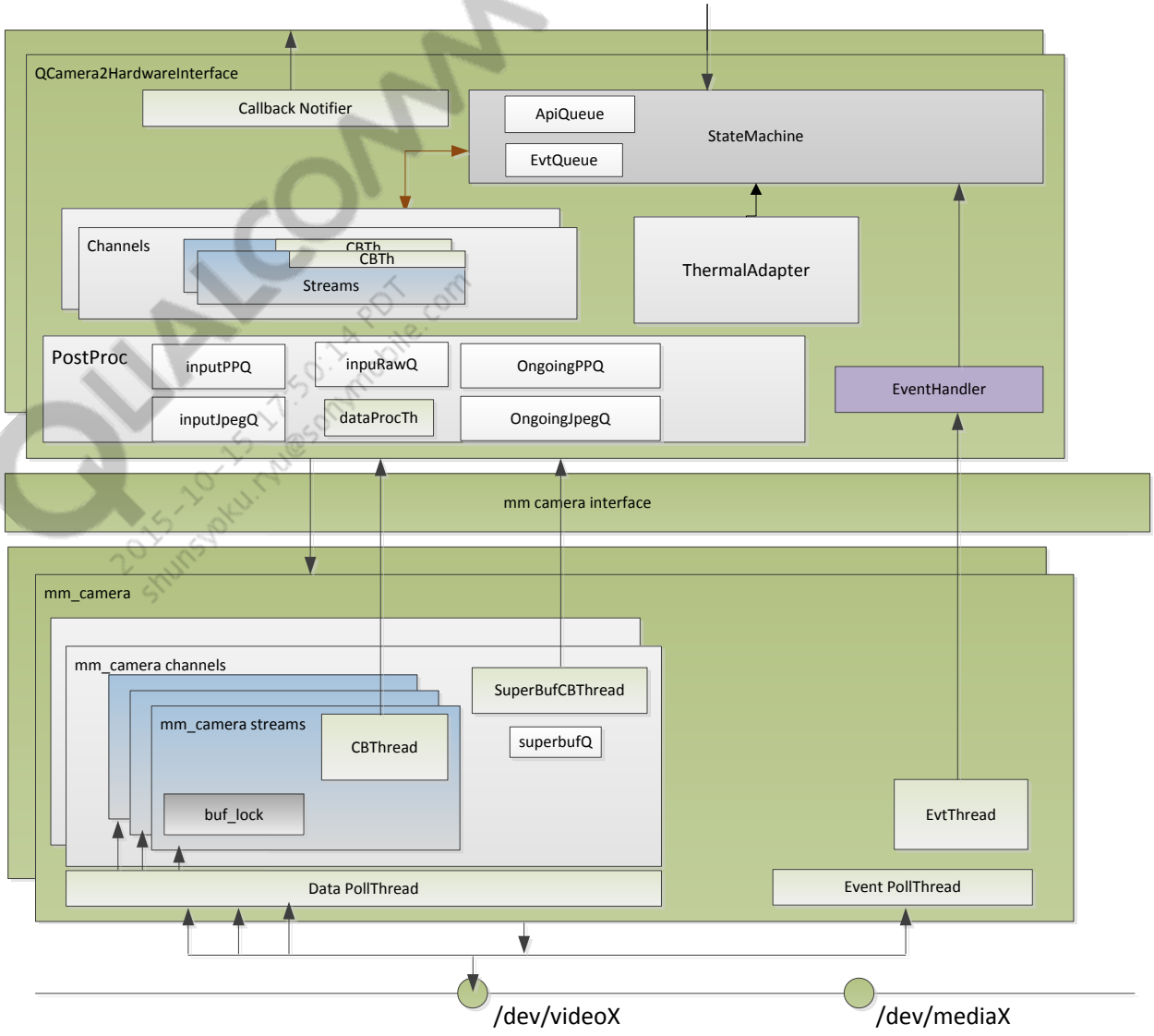
- Channel – A loose concept to bundle multiple image streams together. Currently only one channel is supported.
- Stream – The minimum streaming element. Each stream can only have one format. It is the interface to exchange capture image buffers between camera hardware and the application.
- Stream bundling – Within the channel, multiple streams can be bundled so that:
  - Hardware does not start the streaming till all bundled streams are turned on.
  - When the first bundled stream is turned off, hardware is stopped.

High-Level Operation Sequence

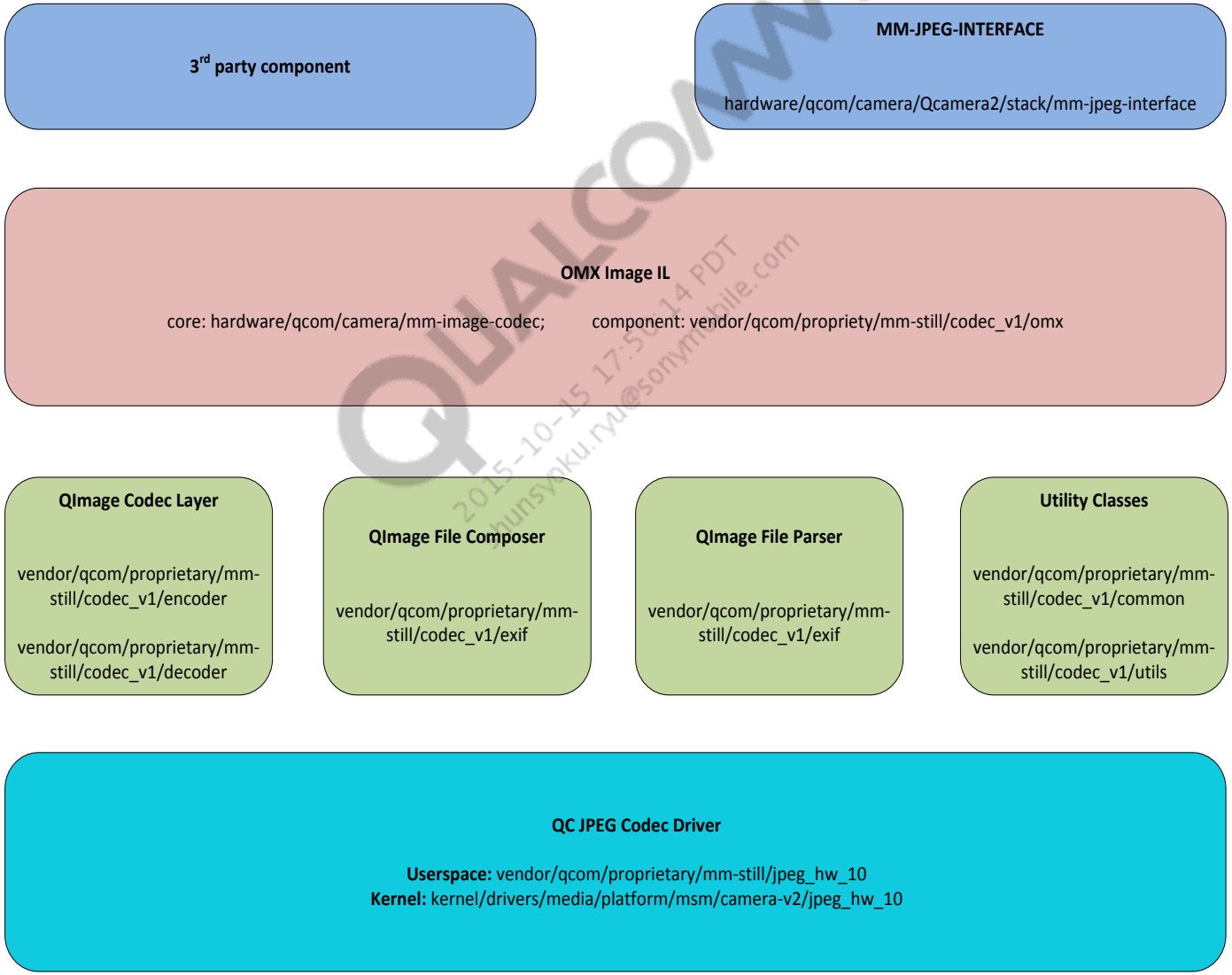
- Open camera
- Query capability
- Add camera channel
- Add streams into camera channel
- Set stream format
- Map IOMMU memory to backend daemon and wait for the mapping ack
- Bundle streams for ZSL like streaming purpose
- Start streams
- Poll and notify HAL image buffers
- Stop streams
- Unmap IOMMU memory from backend and wait for the unmap ack
- Delete stream
- Delete channel
- Close camera

Software Architecture – Camera HAL and mm-camera-interface

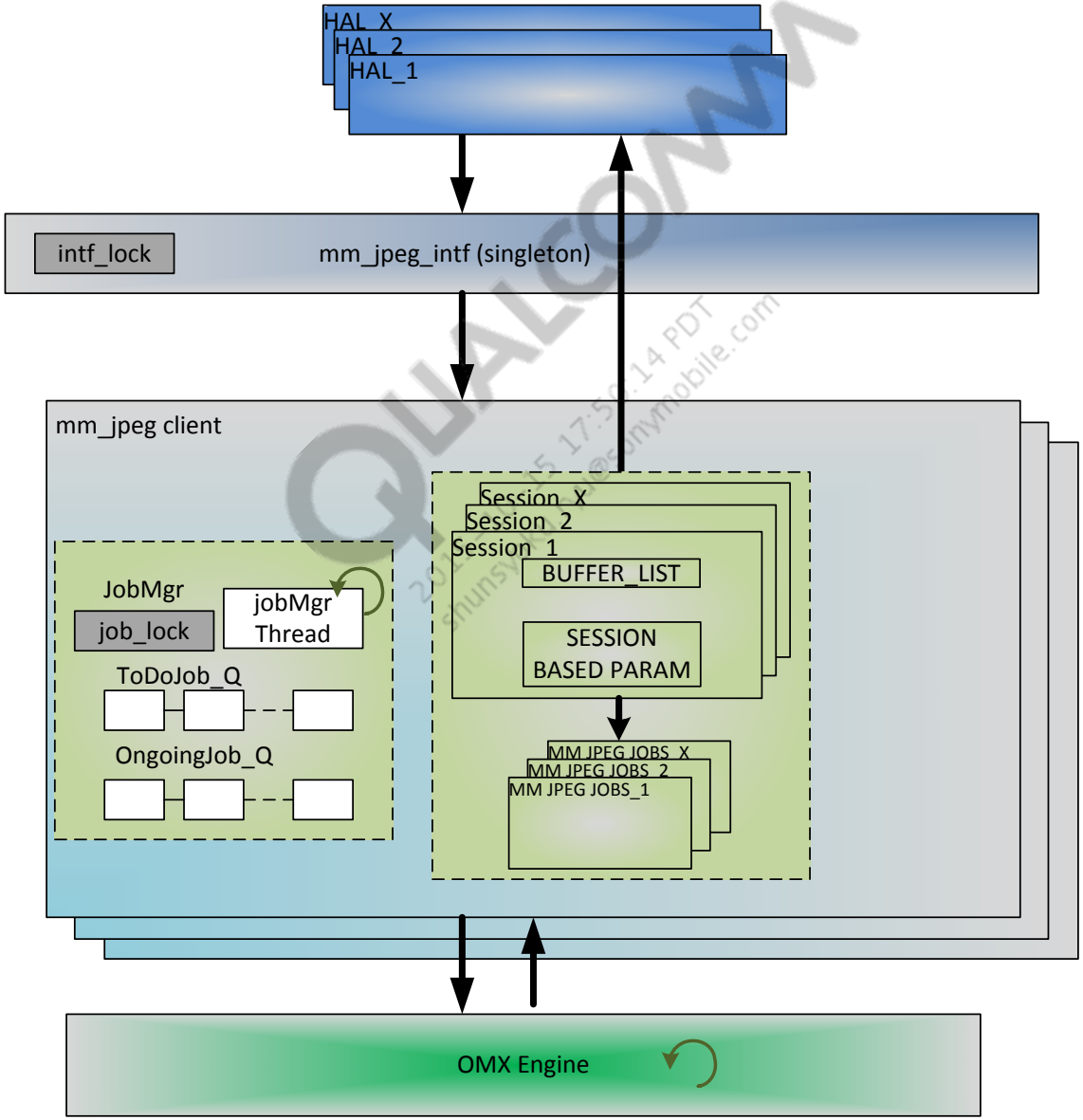
- Channels
  - ZSL
  - Capture
  - Preview
  - Snapshot
  - Video
  - Raw
  - Metadata
- Streams
  - Preview
  - Postview
  - Metadata
  - Raw
  - Snapshot
  - Video
  - Reprocessing



Software Architecture – Codec Subsystem



Software Architecture – mm-jpeg-interface

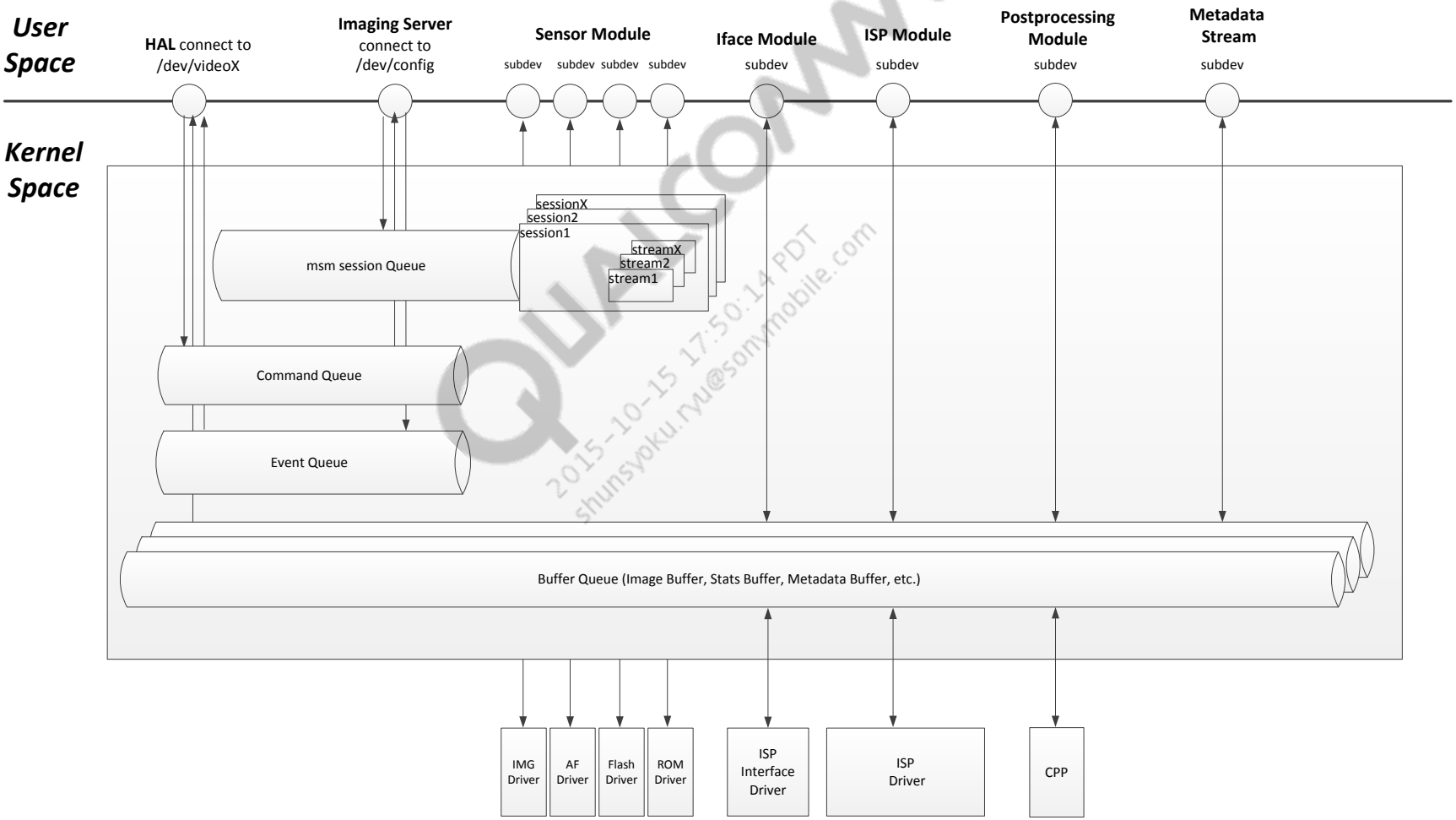


Camera Kernel Driver Framework

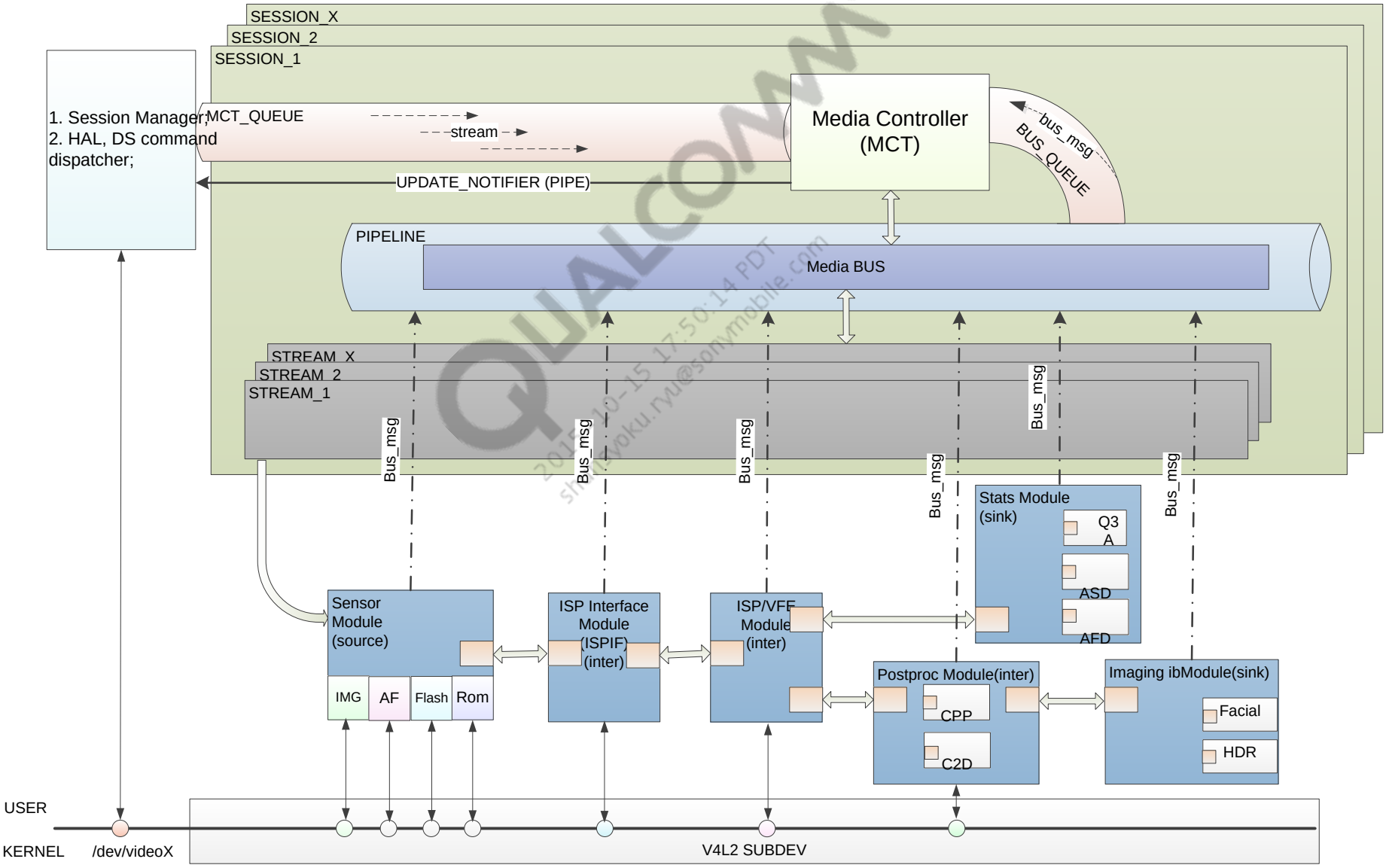
- The Camera kernel driver framework is designed based on the V4L2 framework, V4L2 video device, and subdevices are implemented to provide the following functionalities:
  - Hardware access
  - Interrupt handling
  - Buffer and event management
  - Session control

Functional modules	Description
V4L2 framework infrastructure	▪ msm.c: implement /dev/msm-config device node ▪ Camera.c: implement /dev/videoX device node ▪ msm_vb2.c: implement video buffer APIs
Sensor control and configuration	▪ Generic sensor drivers – msm_sensor.c – work with camera.c ▪ Native sensor drivers – Implement subdevices ▫ cci ▫ csiphy ▫ csid ▪ Utilities ▫ IO and i2c functions
isp	▪ Bufffer management ▫ msm_buf_mgr.c ▪ Native drivers ▫ msm_vfe.c. ▫ msm_ispif.c

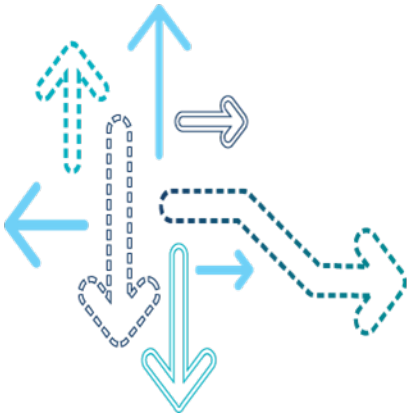
Software Architecture – Camera Kernel



Software Architecture – Imaging Server and Media Controller



Camera Image Server – Pipelines Bus/Threads



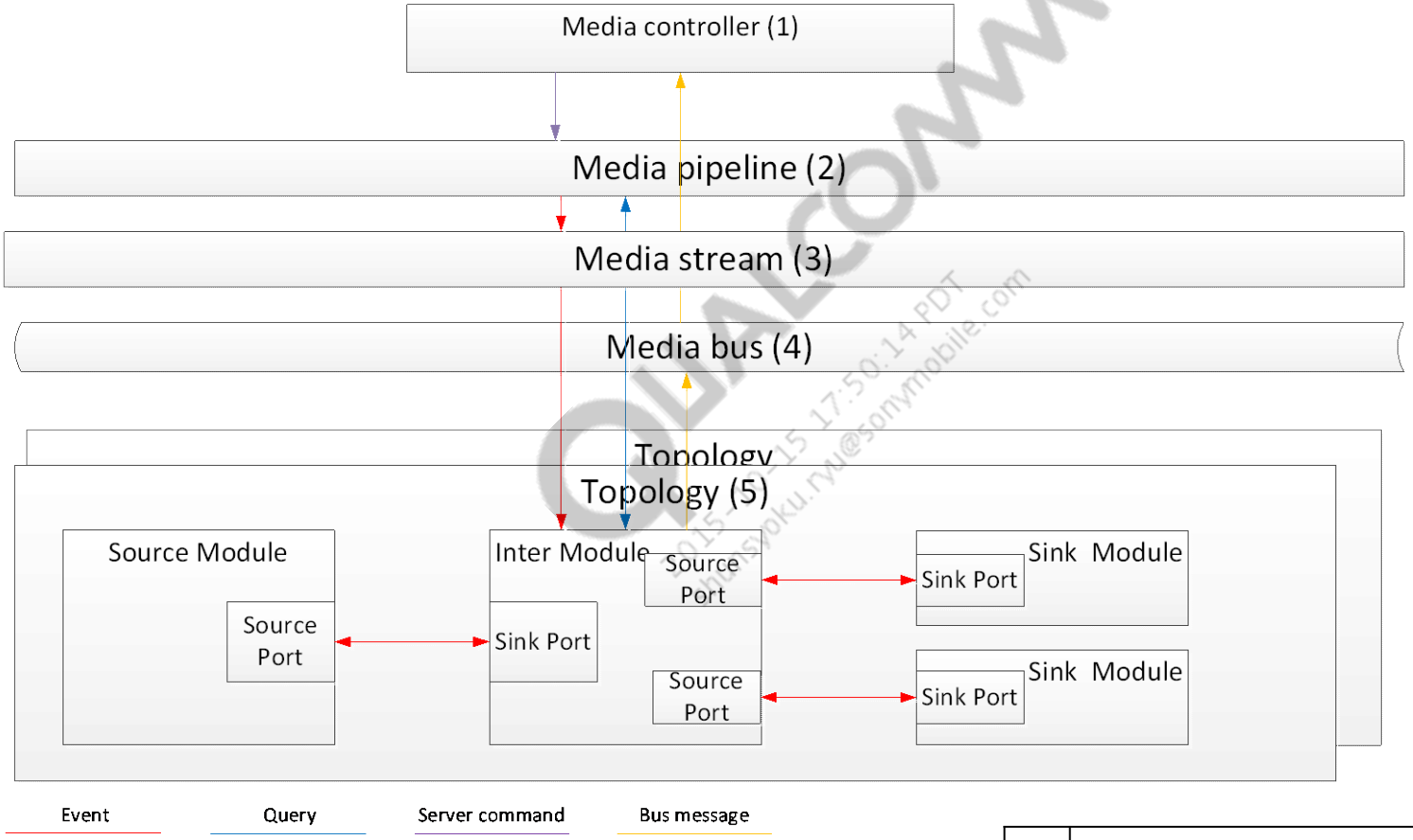
Qcamera Imaging Server Components

- Module – Software entity used to perform a particular task such as ISP module
- Port – Input (Sink) and output (Source) point of a module, which is used to transfer events between two modules; a module can have multiple ports
- Event – Data transferred among media objects
- Media controller – Control module in a media session that is responsible for module organization and communication
- Media bus – Communication channel for the modules and media controller
- Stream – Sequence of modules linked by their ports in a particular order to perform a given task; it also represents a flow of data among several modules, i.e., data transfer from sensor to ISP; a media stream can be configured, started, and stopped
- Pipeline – Collection of streams

Qcamera Imaging Server Components (cont.)

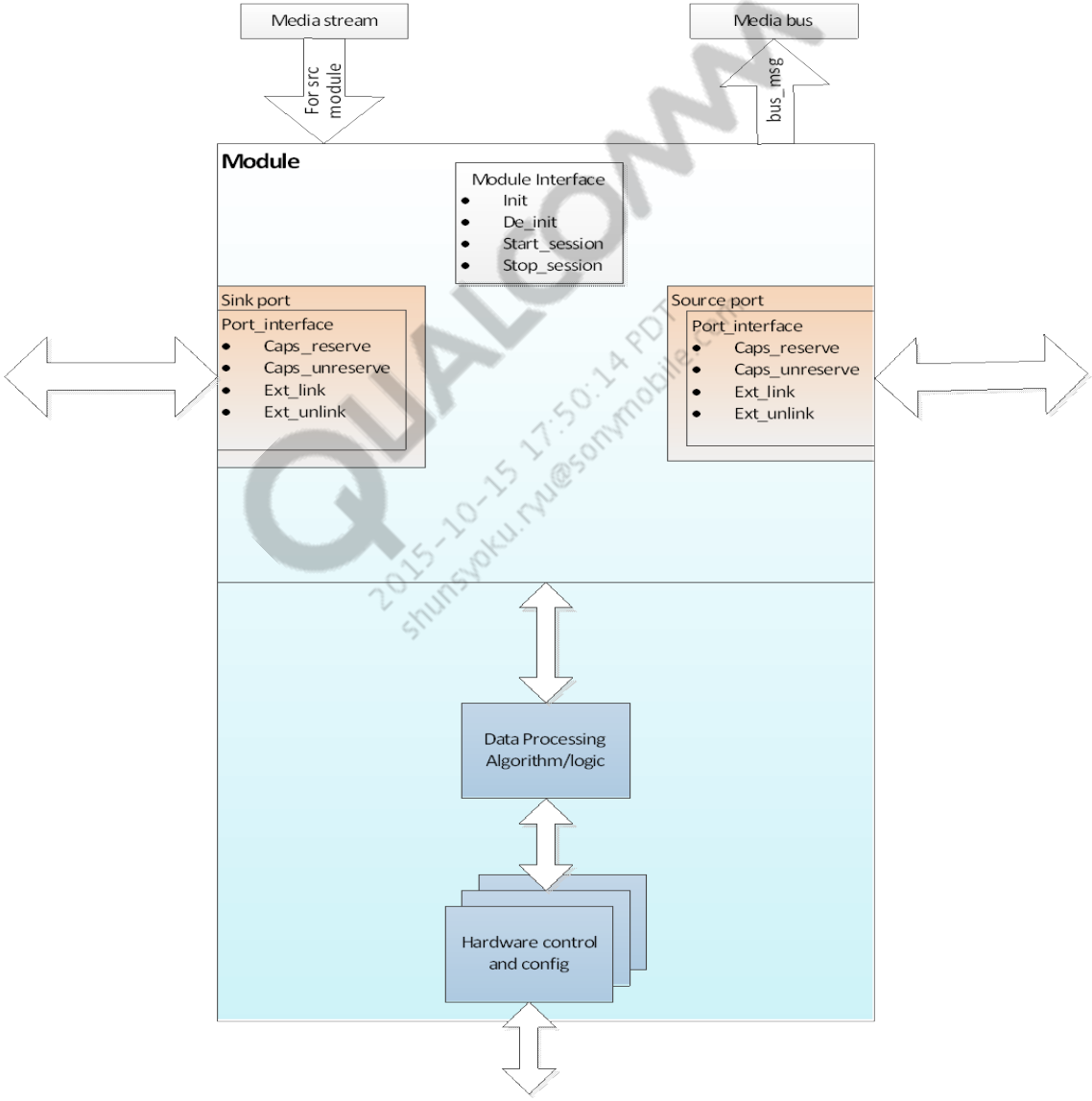
- Parent and children information
  - One object matches to one pipeline
  - One pipeline may have multiple children (streams)
  - One stream may have multiple children (modules)
  - One module may have multiple children (ports)
  - Port must have only one parent (module)
  - Module might have multiple parents (stream)
  - Stream must have only one parent (pipeline)

Software Architecture – Module Organization and Communication

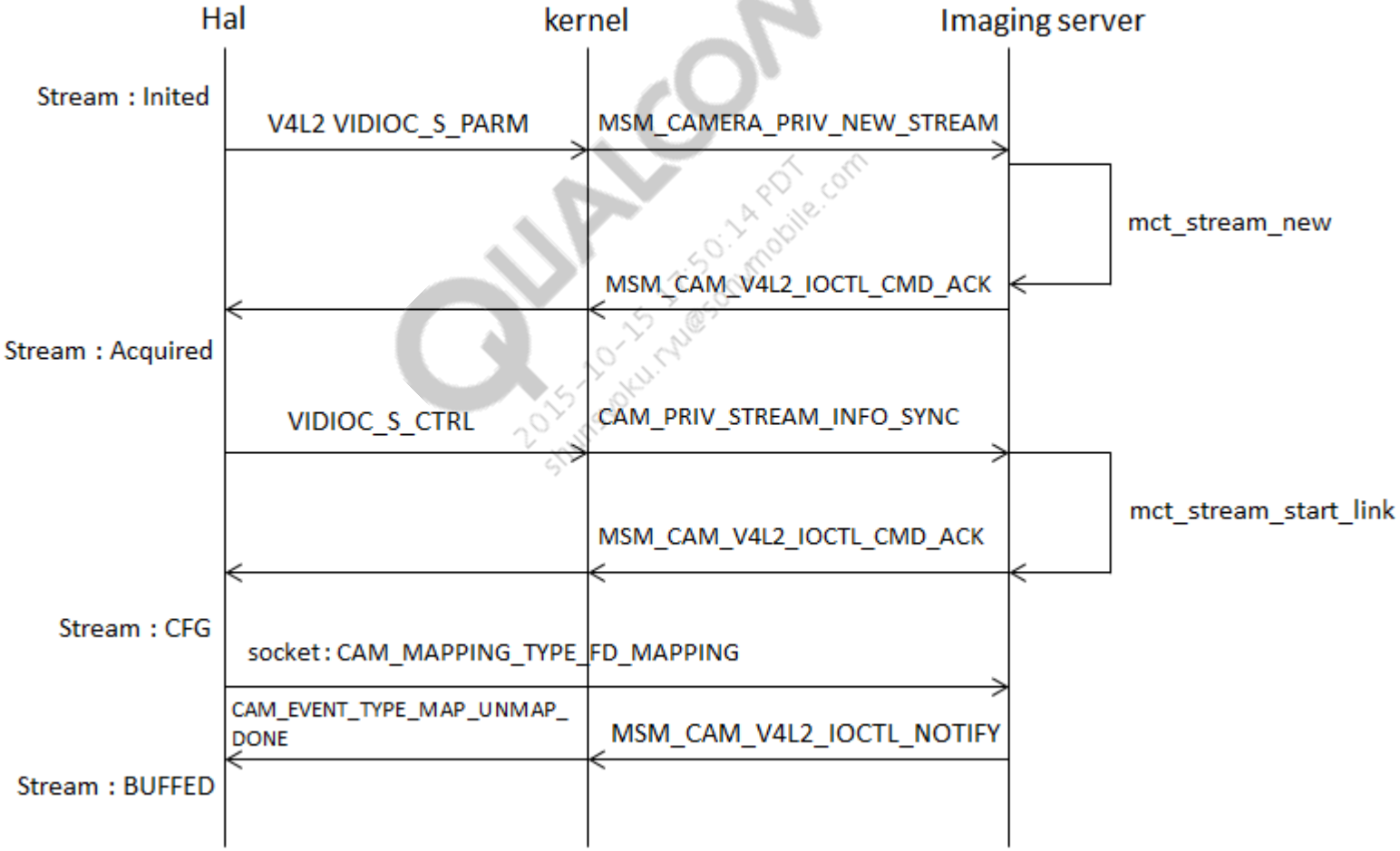


1	Manager for major communication, session tracking on camera server
2	Media pipeline wrapper
3	Individual media stream, e.g., preview, snapshot
4	Media bus wrapper
5	Topology connecting various modules for specific use case

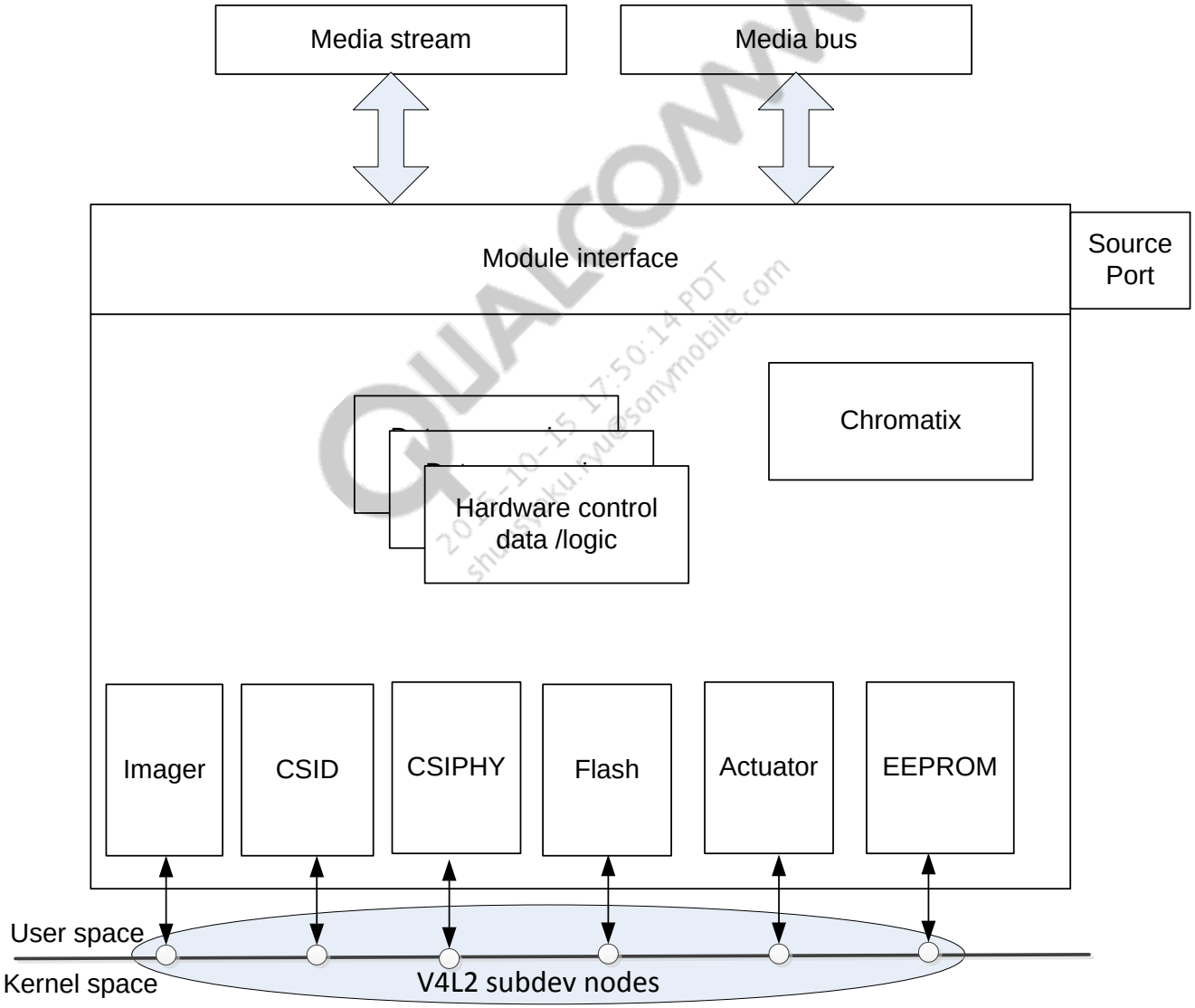
Software Architecture – Module (Generic)



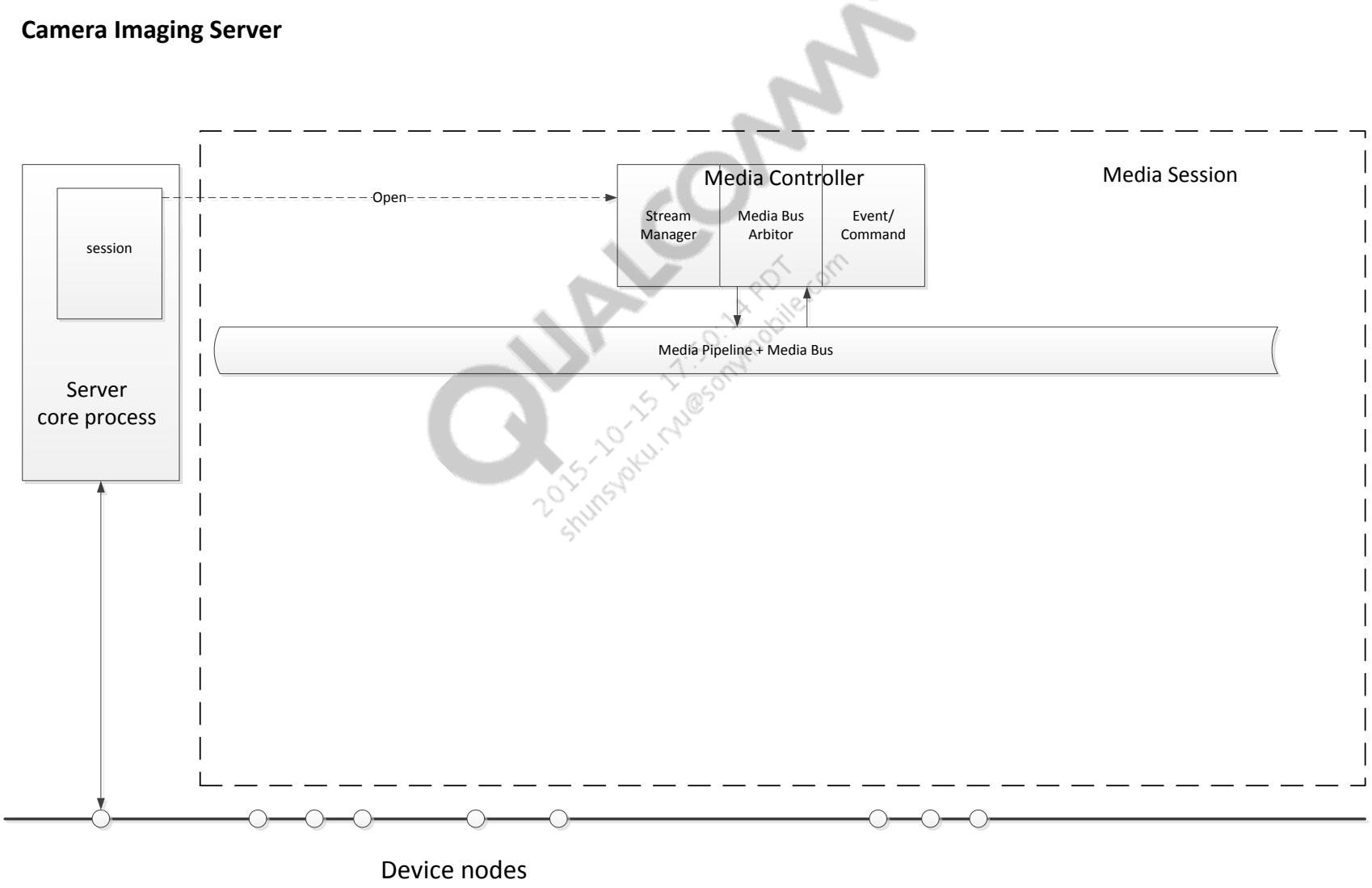
Media Stream Link Creation



Software Architecture – Sensor Module

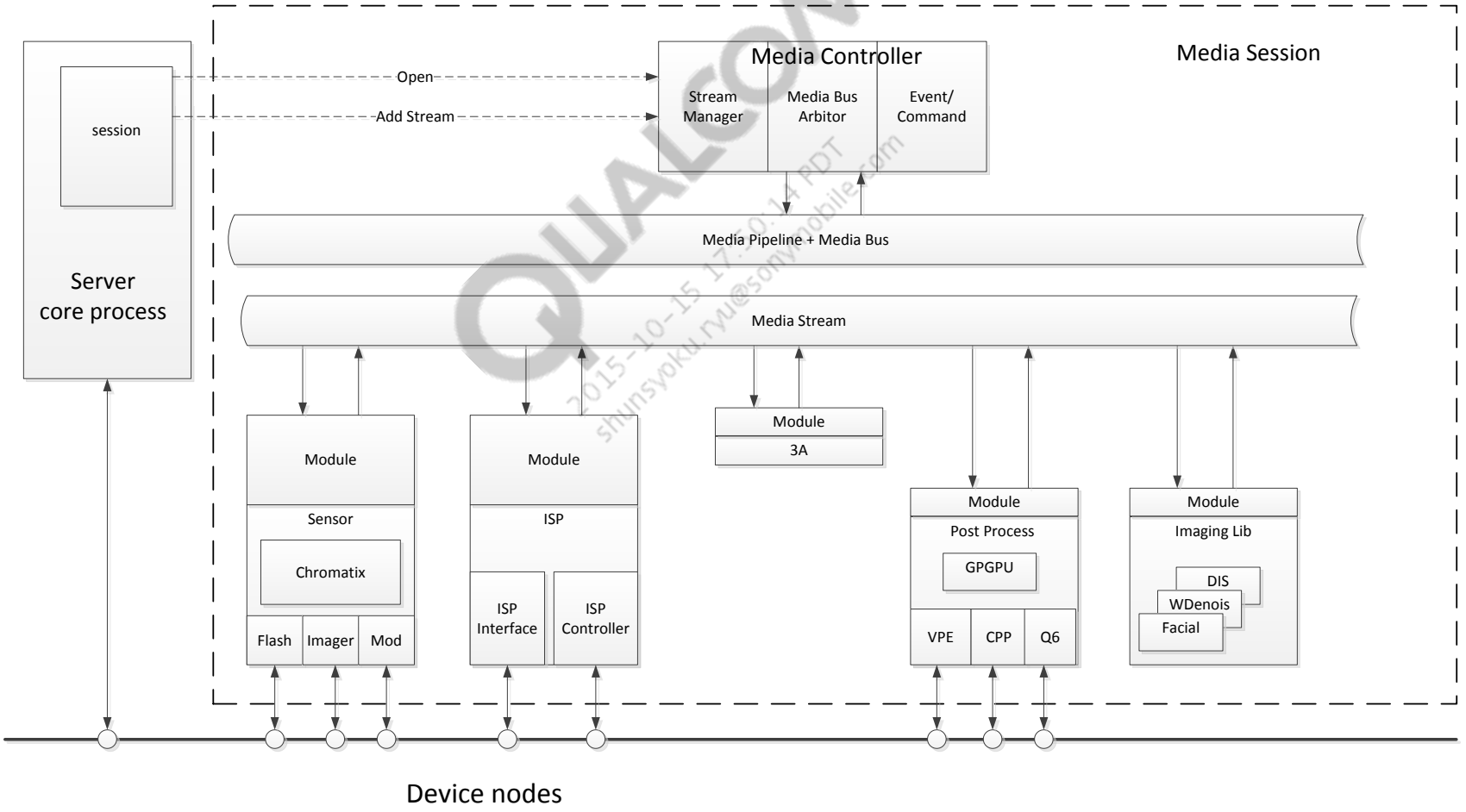


Software Architecture – Media Session Setup Example



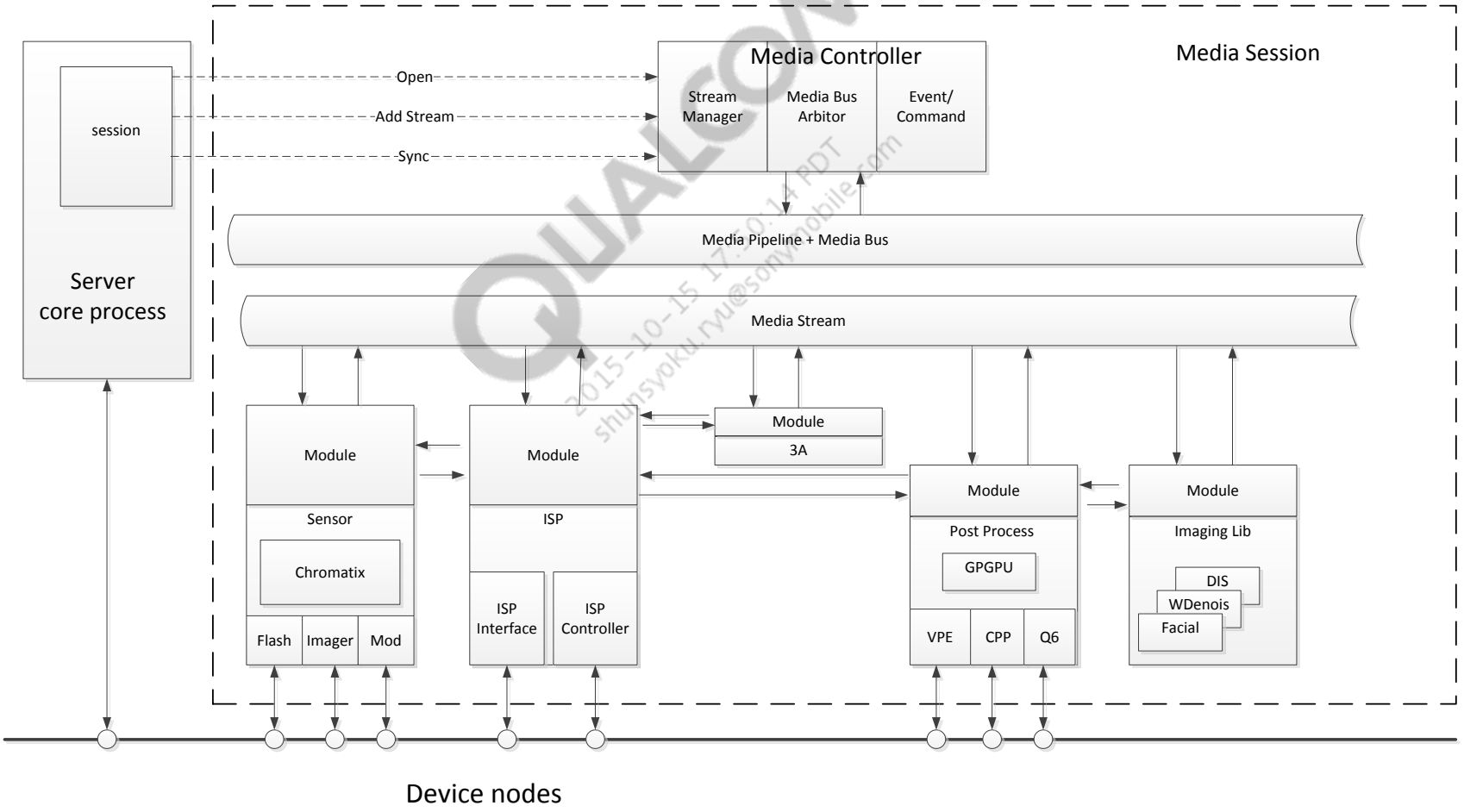
Software Architecture – Media Session Setup Example (cont.)

Camera Imaging Server



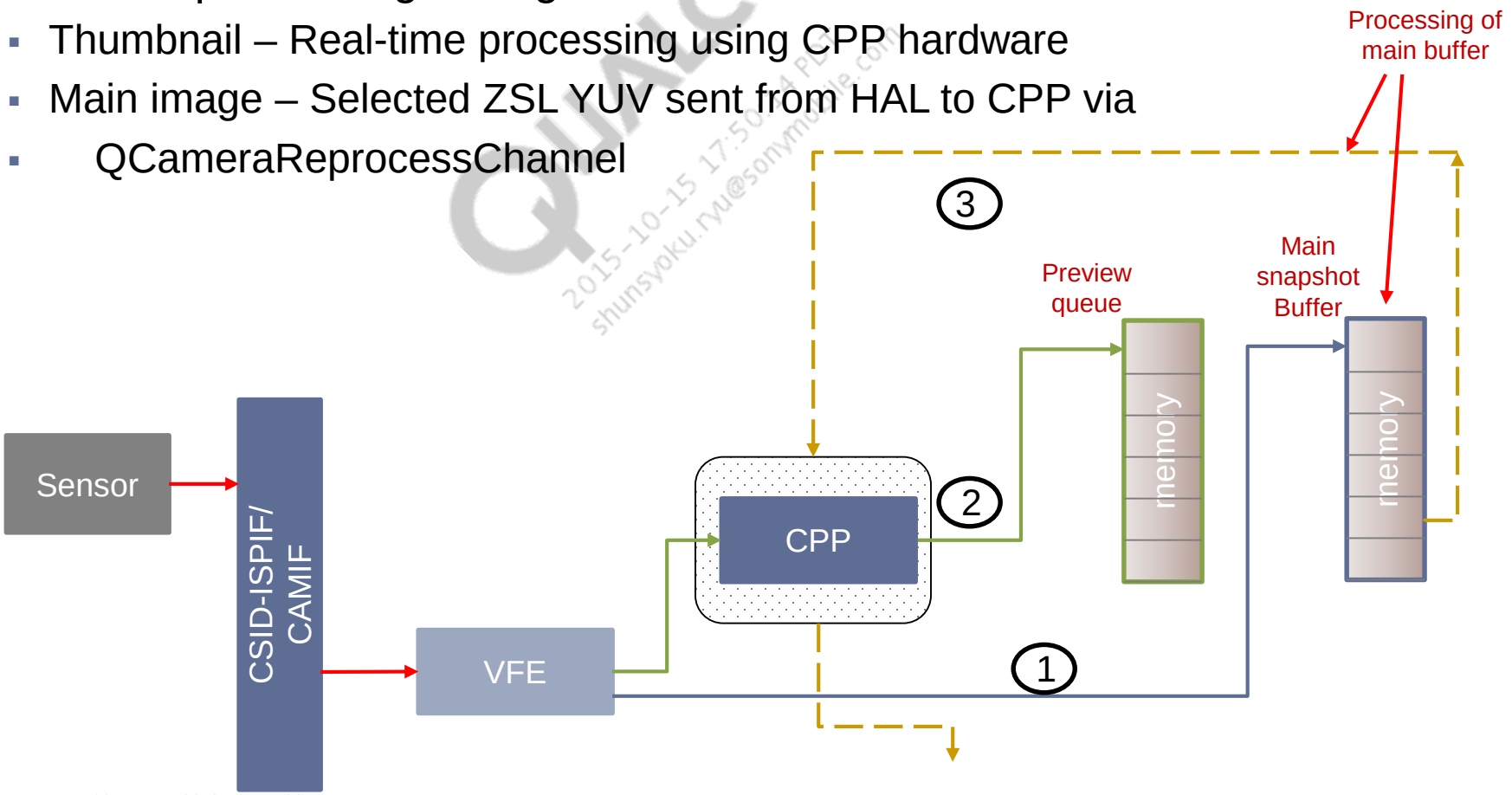
Software Architecture – Media Session Setup Example (cont.)

Camera Imaging Server



Buffer Reprocessing via CPP

- Modes – Real-time/Online (streaming use case); Offline (nonstreaming use cases)
- Uses – Noise reduction, ASF, flip/rotate, up/downscale, crop
- Online reprocessing during ZSL
  - Thumbnail – Real-time processing using CPP hardware
  - Main image – Selected ZSL YUV sent from HAL to CPP via
  - QCameraReprocessChannel



User–Kernel Space Communication

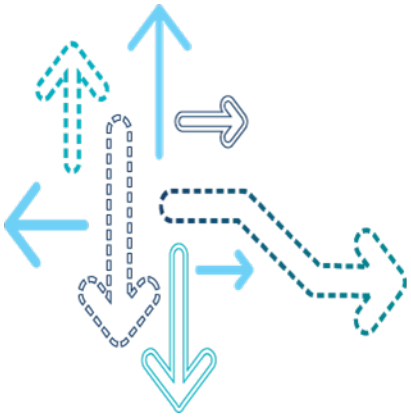
- Qualcomm camera HAL communicates with kernel through V4L2 IOCTLs on the /dev/videoX nodes
- V4L2 IOCTLs supported (this list could be revised in the future)
  - VIDIOC_S_FMT – Used to set format on the video device
  - VIDIOC_S_CTRL – Used to pass control information to device
  - VIDIOC_G_CTRL – Used to get control information from device
  - VIDIOC_QBUF – Used to queue preview buffers
  - VIDIOC_DQBUF – Used to dequeue preview buffers
  - VIDIOC_STREAMON – Used to start streaming for video device
  - VIDIOC_STREAMOFF – Used to stop streaming for video device
  - VIDIOC_QUERYCAP – Used to query device capability
  - VIDIOC_QUERYBUF – Used to query buffer information
  - VIDIOC_REQBUFS – Used to request buffer registration
  - VIDIOC_DQEVENT – Used to dequeue event from kernel

Exposed API Overview

Android application calls high-level APIs defined in Android camera architecture, as defined in http://code.google.com/android/reference/adb.html

- http://developer.android.com/reference/android/hardware/Camera.html
- QTI Camera HAL code (QCamera2HWI.cpp) translates the Android camera calls to QTI camera subsystem
- Custom parameters exposed via class QCameraParameters defined in QCameraParameters.h

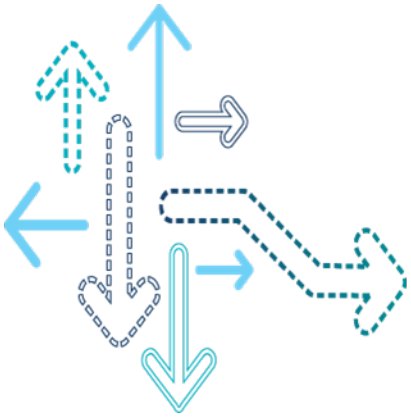
Feature Overview



Camera Feature Overview for MSM8974

- YUV and Bayer reference drivers
- Video encoding
- Zero Shutter Lag (ZSL)
- Raw snapshot capture
- Snapshot and video encode live snapshot
- 3A (AF, AE, AWB) – 3A4.0 (uses Bayer statistics)
- Continuous Auto Focus (CAF)
- JPEG encoder
- Regular JPEG decode
- EXIF/JFIF support
- White LED flash support
- Shutter sound
- Exposure modes
- ISO support
- Digital zoom
- Antibanding
- EV/brightness control
- Viewfinder special effects
- Manual white balance
- Auto Frame Rate (AFR)
- Sharpness control
- Contrast control
- Macro mode with autofocus
- Selectable zone AF
- Viewfinder histogram (G only)
- Best Shot mode
- Front/back camera support
- Facial processing library with end-to-end facial detection
- Face authentication (up to HAL)
- High Dynamic Range (HDR)
- Luma Adaptation
- Memory color enhancement and skin color enhancement
- Video stabilization (DIS, EIS)
- High frame rate
- EZ-Tune
- Wavelet noise reduction (hardware-based)
- Hardware JPEG decoding

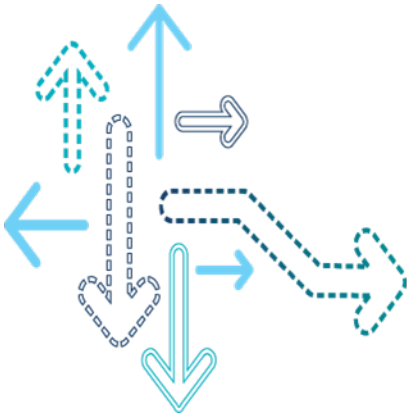
Licensee Responsibilities and Recommendations



Licensee Responsibilities and Recommendations

- Camera sensor driver
  - Especially for the sensors with YUV output for Snapshot mode
  - For Bayer driver; file a case to discuss Qualcomm support
- Camera application
  - If not using standard Android camera application
- New features/customization
  - Could be supported on case-by-case basis; file a case

Memory Requirements



Memory Requirements – Camera

	ZSL	Non-ZSL	HDR
Preview	9	8	8
Snapshot	7	2	5
Postview	NA	4	7
Snapshot Reprocess	1 (2 if any other reprocess other than WNR is enabled, like rotation)	1 (2 if any other reprocess other than WNR is enabled, like rotation)	2 (4 if any other reprocess other than WNR is enabled, like rotation)
Thumbnail Reprocess	1 (2 if any other reprocess other than WNR is enabled, like rotation)	1 (2 if any other reprocess other than WNR is enabled, like rotation)	2 (4 if any other reprocess other than WNR is enabled, like rotation)
Meta	7	7	7

Note: Above numbers are for MSM8974 builds from Oct'13. See QCamera2HardwareInterface::getBufNumRequired() for the latest data for specific chipset/build combination.

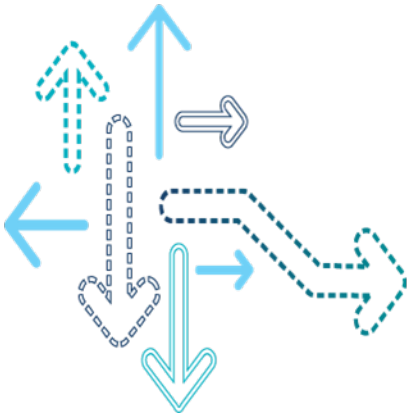
Memory Requirements and Latencies – JPEG Encoder

- JPEG encoder
  - Memory usage not fixed
    - Depends on input dimension and user's configuration
  - Buffers involved
    - Input frame buffer – Size depends on image size
      - (H2V2 – width * height * 1.5)
      - (H2V1, H1V2 – width * height * 2)
      - (H1V1 – width * height * 3)
    - Output bitstream buffer – Supplied by the user, can be of any size; two buffers of 8 kB or more recommended to reduce turnaround overhead

Memory Requirements and Latencies – JPEG Decoder

- JPEG decoder
  - Memory usage not fixed
    - Depends on output dimension and user's configuration
  - Buffers involved
    - Input bitstream buffer – Supplied by the user; can be of any size that is a multiple of 512 bytes
      - Since two such buffers are necessary, the absolute minimum is 1024 bytes
      - Two buffers of 16 kB or more are recommended to reduce turnaround overhead
    - Working buffer – Two buffers of (original width * 16) bytes
    - Output frame buffer – Size depends on the size of the desired output dimension and format
      - (H2V2 – width * height * 1.5)
      - (H2V1, RGB565 – width * height * 2)
      - (RGB888 – width * height * 3)
  - Buffers of Types 1 and 3 are allocated by the user instead of the library

Debugging/Troubleshooting Support Guidelines



Enabling mm-camera Logs

- General logs – Do '#define LOG_DEBUG' in mm-camera2\includes\camera_dbg.h before the use of LOG_DEBUG
- Enable appropriate log level in files mentioned below for various module specific logs:
  - Sensor - mm-camera2\media-controller\modules\sensors\module\sensor_dbg.h
  - CPP - mm-camera2\media-controller\modules\pproc-new\cpp\cpp_log.h
  - Imaging Library - mm-camera2\media-controller\modules\imglib\components\include\img_dbg.h

Enabling Kernel Logs

- Debug logs for files under camera_v2 folder are protected by CONFIG_MSMB_CAMERA_DEBUG flag; it is recommended to enable this flag for individual files as needed instead of enabling it globally from a file such as camera.h, as too many logs could cause issues with the camera operation
- Some files need additional flags set for logging, e.g.:
  - CSID logs – Set DBG_CSID to 1 in sensor\csid\msm_csid.c
  - CSIPHY logs – Set DBG_CSIPHY to 1 in sensor\csiphy\msm_csiphy.c
- VFE register dump using msm_camera_io_dump_2() in isp\msm_isp_axi_util.c

Enabling Qualcomm Camera HAL Logs

- These are enabled by default in QCamera*.cpp files. Logs can be selectively disabled, or can be isolated using grep filter.
- To enable mm-camera-interface logs, edit the stack\mm-camera-interface\inc\mm_camera_dbg.h file as follows:
  - #define LOG_DEBUG 1

Capturing the Logs to the Log Files

- To save user space logs
  - adb logcat -v threadtime > USER_SPACE_LOG_FILE_NAME
- To save kernel logs
  - adb shell cat /proc/kmsg > KERNEL_SPACE_LOG_FILE_NAME

Dumping VFE Output in HAL

- To dump VFE output in HAL, issue the following commands from an adb shell:

```
setprop persist.camera.dumpimg xxx
Chmod 777 /data
```

- The xxx above is a decimal integer, whose 32-bit hex counterpart represents:
  - Any of the bit-0 to bit-7 needs to be set, corresponding to flags:
    - QCAMERA_DUMP_FRM_PREVIEW – 0x1
    - QCAMERA_DUMP_FRM_VIDEO – 0x2
    - QCAMERA_DUMP_FRM_SNAPSHOT – 0x4
    - QCAMERA_DUMP_FRM_THUMBNAIL – 0x8
    - QCAMERA_DUMP_FRM_RAW – 0x10
    - QCAMERA_DUMP_FRM_JPEG – 0x20
  - Bit-8 to bit-15 specifies how many frames to skip during dumping; default is no skip
  - Bit-16 to bit-31 specifies how many frames to dump; default is 10
- Function QCamera2HardwareInterface::dumpFrameToFile() from HAL\QCamera2HWICallbacks.cpp is used to dump the files under /data folder
- For file naming conventions, see the dumpFrameToFile() function above

Dumping YUV Input to JPEG Encoder

- To dump YUV input going to the JPEG encoder:
  1. Define MM_JPEG_DUMP_INPUT in QCamera2/stack/mm-jpeg-interface/src/ mm_jpeg.c
  2. From adb shell, issue the following command:

     ```
     mkdir /data/jpeg
     ```

  3. From adb shell, issue the following command:

     ```
     chmod 777 /data/jpeg
     ```

  4. Take a picture after compilation, updating the phone binary.
  5. Store YUV frames for full image and thumbnail at /data/jpeg.
- Function mm_jpeg_session_encode() from QCamera2/stack/mm-jpeg-interface/src/ mm_jpeg.c is used to dump the files under /data/jpeg folder
- For file naming conventions, see the mm_jpeg_session_encode() function above

Information Expected from Customers for a New Project

- Number of cameras
- Schematic diagram of how the cameras are connected to MSM
- Provide following information about each camera
  - Sensor vendor name, sensor model number
  - Module vendor name for each sourcing supplier (we prefer only 1, to reduce total color tuning time)
  - For each of preview, snapshot, video modes, specify output data format, bit depth, resolution, min-max fps, number of data lanes used and data rate per lane
  - What kind of actuator mechanism is used, e.g., voice coil, piezo, etc.?
  - Milestones in terms of hardware verification, camera driver development, tuning schedule (applicable only for Bayer output sensors), etc.?
  - For Bayer sensors only, where QQI is expected to write the driver, provide:
    - Sensor datasheet
    - AF actuator datasheet
    - Sensor vendor recommended register settings for preview and snapshot
    - Sensor vendor Point of Contact (PoC) information
    - Schedule when 15 camera modules can be shipped to Qualcomm
    - Camera module connector's pinout, electrical, mechanical specs

Information Expected from Customers for a New Project (cont.)

- If you use Bayer sensor, will you use QTI 3A?
- Which postprocessing functions will be used, e.g., HDR?
- Any special camera features, use cases, or concurrencies expected beyond what's exposed by QTI at Android's camera API layer and supplied in QTI modified Android camera app?
- Do you plan to modify QTI camera stack for any kind of customization? If yes, provide details.

References

Document	
Qualcomm Technologies	
Presentation: MSM8x26 Linux Android Multimedia Software Overview	80-NF043-1
Presentation: Camera Frontend Code Walkthrough for MSM8974/APQ8074/MSM8x26 Linux Camera Software	80-NF499-2
Presentation: Kernel Code Walkthrough for MSM8974/APQ8074/MSM8x26 Linux Camera Software	80-NF499-3
Presentation: Qcamera Server And Mctl Code Walk Through for MSM8974/APQ8074/MSM8x26 Linux Camera Software	80-NF499-4
Presentation: Sensor Module Code Walkthrough for MSM8974/APQ8074/MSM8x26 Linux Camera Software	80-NF499-5
Presentation: Stats Module Code Walkthrough for MSM8974/APQ8074/MSM8x26 Linux Camera Software	80-NF499-7
Presentation: Imaging Library Module Code Walkthrough for MSM8974/APQ8074/MSM8x26 Linux Camera Software	80-NF499-8
Presentation: Pproc Module Code Walkthrough for MSM8974/APQ8074/MSM8x26 Linux Camera Software	80-NF499-9
Presentation: JPEG Code Walkthrough for MSM8974/APQ8074/MSM8x26 Linux Camera Software	80-NF499-10
Presentation: Isp/Ispif Code Walk-Through for MSM8974/APQ8074/MSM8x26 Linux Camera Software	80-NF499-11

References (cont.)

Document	
Qualcomm Technologies	
Video: MSM8974/APQ8074/MSM8x26/APQ8084 Linux Camera Overview	*VD80-NA157-22
Video: Camera Frontend Code Walkthrough for MSM8974/APQ8074/MSM8x26 Linux Camera Software	*VD80-NF499-2
Video: Kernel Code Walkthrough for MSM8974/APQ8074/MSM8x26 Linux Camera Software	*VD80-NF499-3
Video: Qcamera Server And Mctl Code Walk Through for MSM8974/APQ8074/MSM8x26 Linux Camera Software	*VD80-NF499-4
Video: Sensor Module Code Walkthrough for MSM8974/APQ8074/MSM8x26 Linux Camera Software	*VD80-NF499-5
Video: Stats Module Code Walkthrough for MSM8974/APQ8074/MSM8x26 Linux Camera Software	*VD80-NF499-7
Video: Imaging Library Module Code Walkthrough for MSM8974/APQ8074/MSM8x26 Linux Camera Software	*VD80-NF499-8
Video: Pproc Module Code Walkthrough for MSM8974/APQ8074/MSM8x26 Linux Camera Software	*VD80-NF499-9
Video: Jpeg Code Walkthrough for MSM8974/APQ8074/MSM8x26 Linux Camera Software	*VD80-NF499-10
Video: Isp/Ispif Code Walk-Through for MSM8974/APQ8074/MSM8x26 Linux Camera Software	*VD80-NF499-11
Resources	
Android Debug Bridge	http://code.google.com/android/reference/adb.html
Camera: Android Developers	http://developer.android.com/reference/android/hardware/Camera.html

***Note:** Refer to the corresponding 80- versions for the latest updates on contents

References (cont.)

Acronyms	
Term	**Definition**
ABF	Adaptive Bayer Filter
ACE	Advanced Chroma Enhancement
AF	Auto Focus
ASF	Adaptive Spatial Filter
AWB	Auto White Balance
BLC	Black Level Correction
CAC	Chromatic Aberration Correction
CCM	Color Correction Matrix
CCT	Color Correction Table
CPP	Camera Post Processing
CS	Chroma Suppression
DPC	Defective Pixel Correction
FD	Face Detection
FIR	Finite Impulse Response
GIC	Green Imbalance Correction
IIR	Infinite Impulse Response
LSC	Lens Shading Correction
LTM	Local Tone Mapping
MCTL	Media Control

References (cont.)

Acronyms	
Term	**Definition**
MCTL	Media Control
MCE	Memory Color Enhancement
MIPI	Mobile Industry Processor Interface
PVL	Preferred Vendor List
RNR	Radial Noise Reduction
RDI	Raw Dump Interface
SCE	Skin Color Enhancement
SNR	Skin Noise Reduction
TNR	Temporal Denoise
VFE	Video Front-End of Camera Firmware
WNR	Wavelet Noise Reduction

Questions?

https://support.cdmatech.com