DSS Overview

Agenda

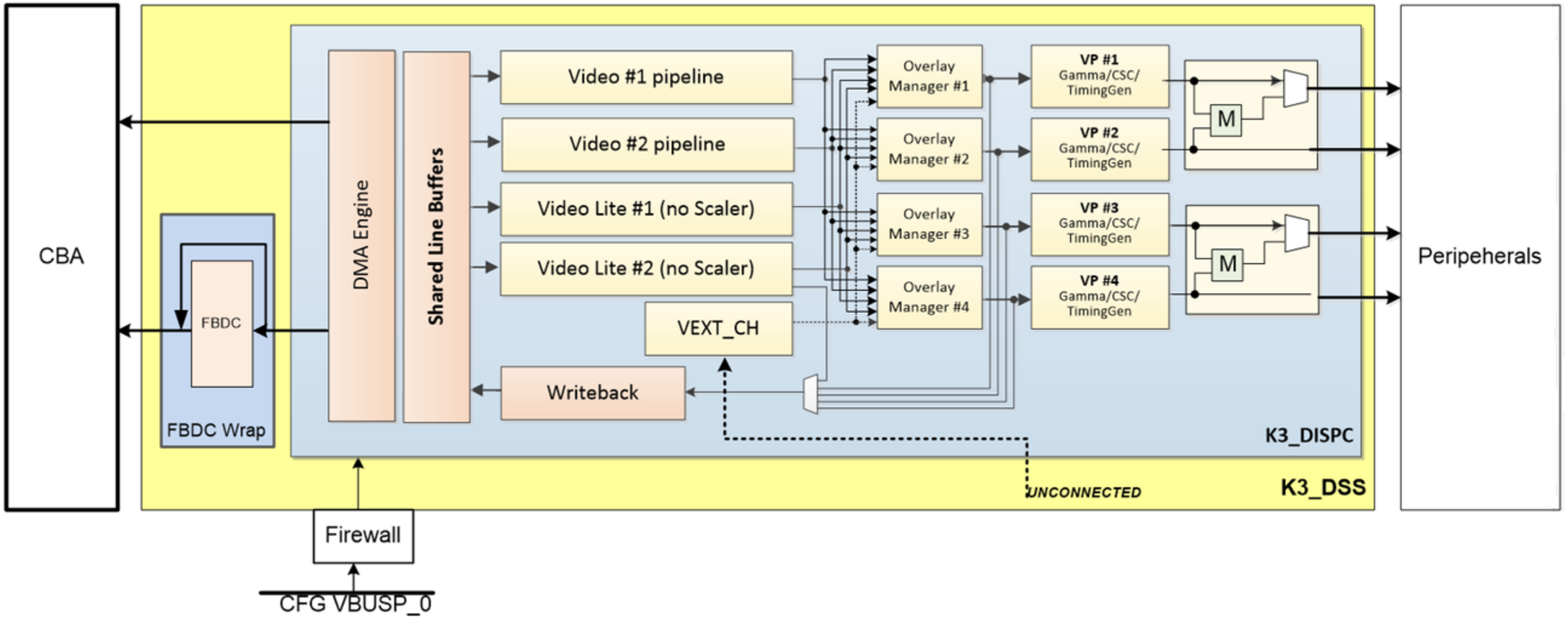
- What is DSS?
- DSS Hardware Overview
  - Display Controller
  - Video Pipeline
  - Write back Pipeline
  - Overlay Manager
  - Video Port
  - Display Interfaces: eDP, DSI, DPI
  - DSS Interrupt Architecture
- DSS Driver Overview
- DSS Advance Features
  - Safety Checks
  - FIFO Buffer Mapping
  - Security Isolation
- Display Sharing

What is DSS?

Introduction

- DSS: Display Sub System
- Displays video frames from memory to LCD
- Provides the logic to output video frames stored in the memory to the external display interfaces.
- Performs multi-layer composition for the display output
- Supports a set of industry standard display interfaces to drive wide range of display panel resolutions
- Supports Scaling, Color Space Conversion, Blending, Color Keying

DSS Block Diagram for J721E

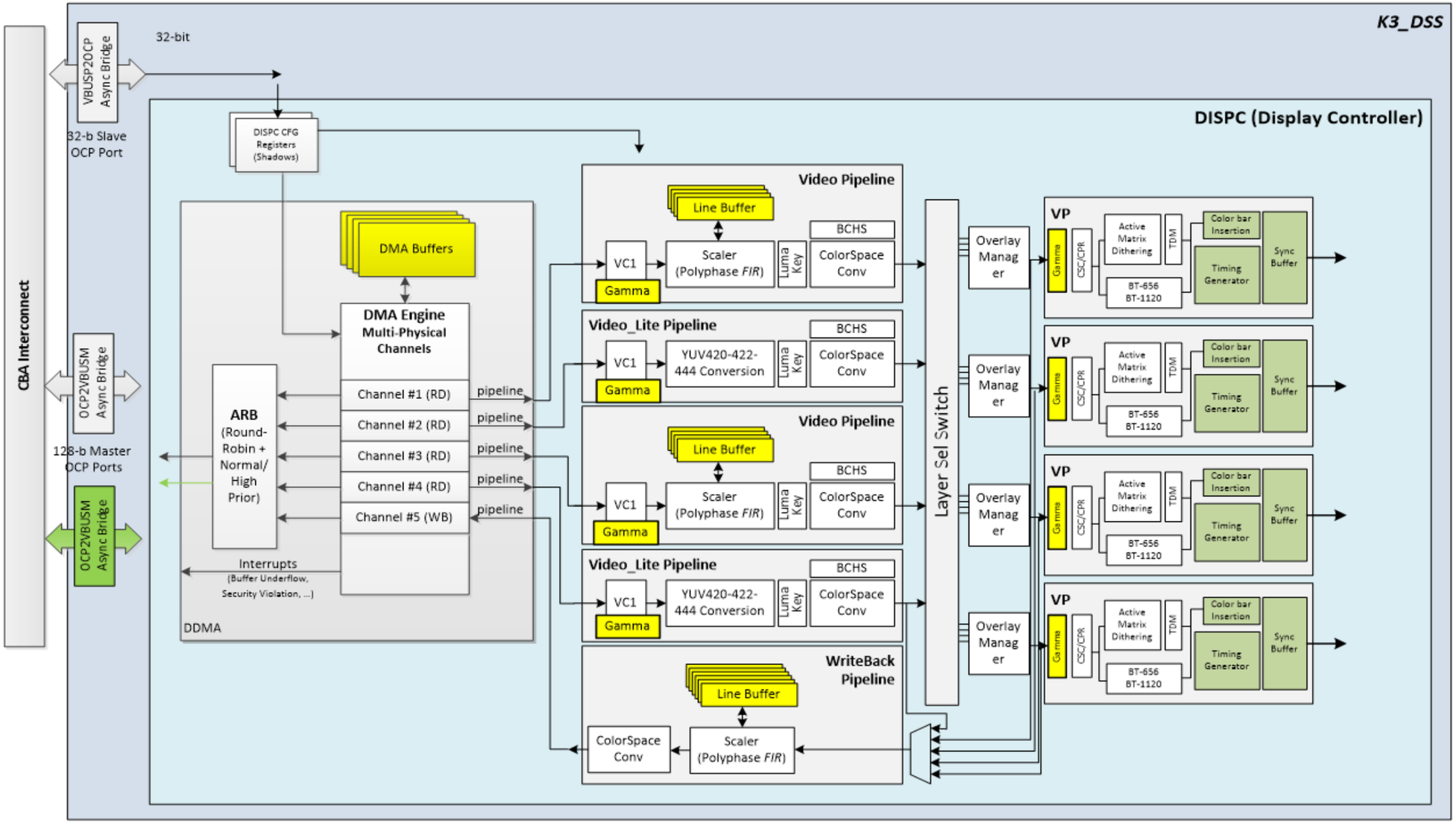


DSS Hardware Overview

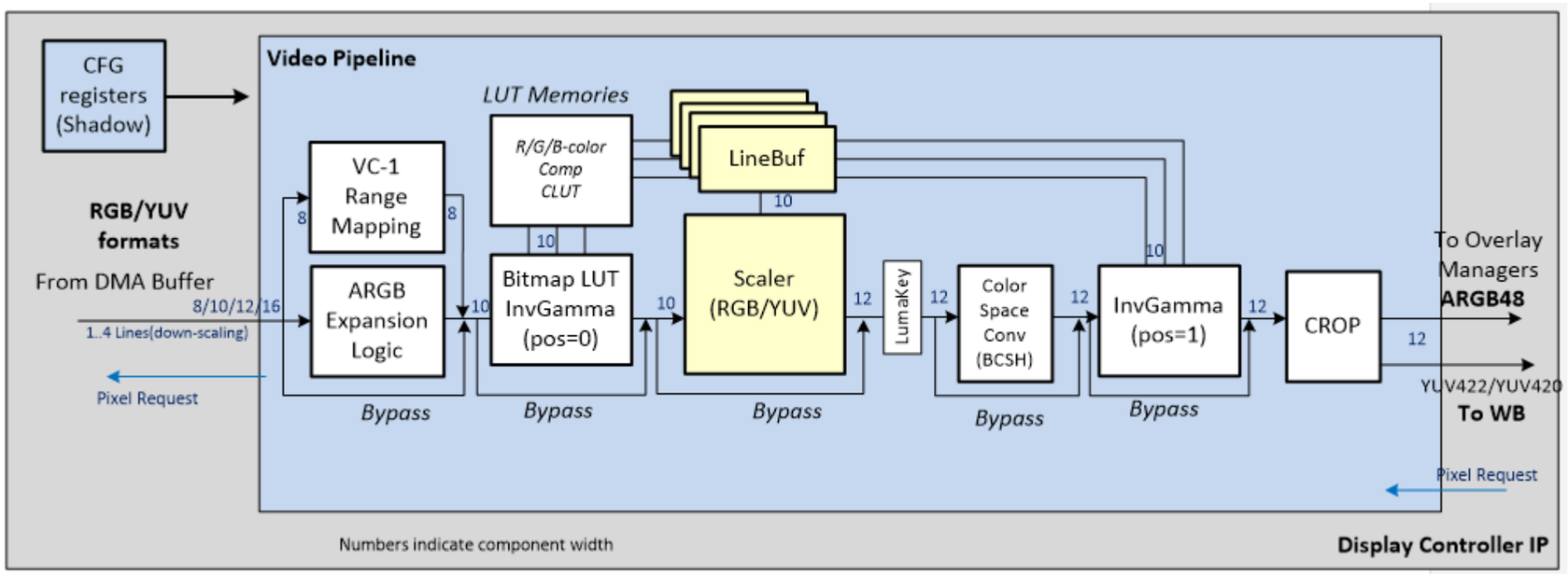
Display Controller

- Four Video Pipelines
  - Two Video pipelines - Supporting 8/10-bit/component RGB or 8/10/12-bit/component YUV source data formats with 3/5-tap 16-phase scaler capable of 0.25x to 16x resizing.
  - Two Video_Lite pipelines - Supporting 8/10-bit/component RGB or YUV source data formats (including YUV420/422 source data)
- Four Overlay Managers
  - Fully mapped to all input pipelines
  - Multi-layer alpha blending
- Four Video Ports
  - Support up to four simultaneous 4K display outputs (600MHz pixel clock)
  - 24/30/36-bit per pixel RGB output
  - Additional 8/10-bit per component YUV422 embedded sync stream (BT1120/BT656) on every output
- One optional Write-back pipeline with a scaler to perform memory to memory operations or to capture one of the display outputs.
- DMA controller capable of supporting up to 8K-wide input source frame

Display Controller Architecture

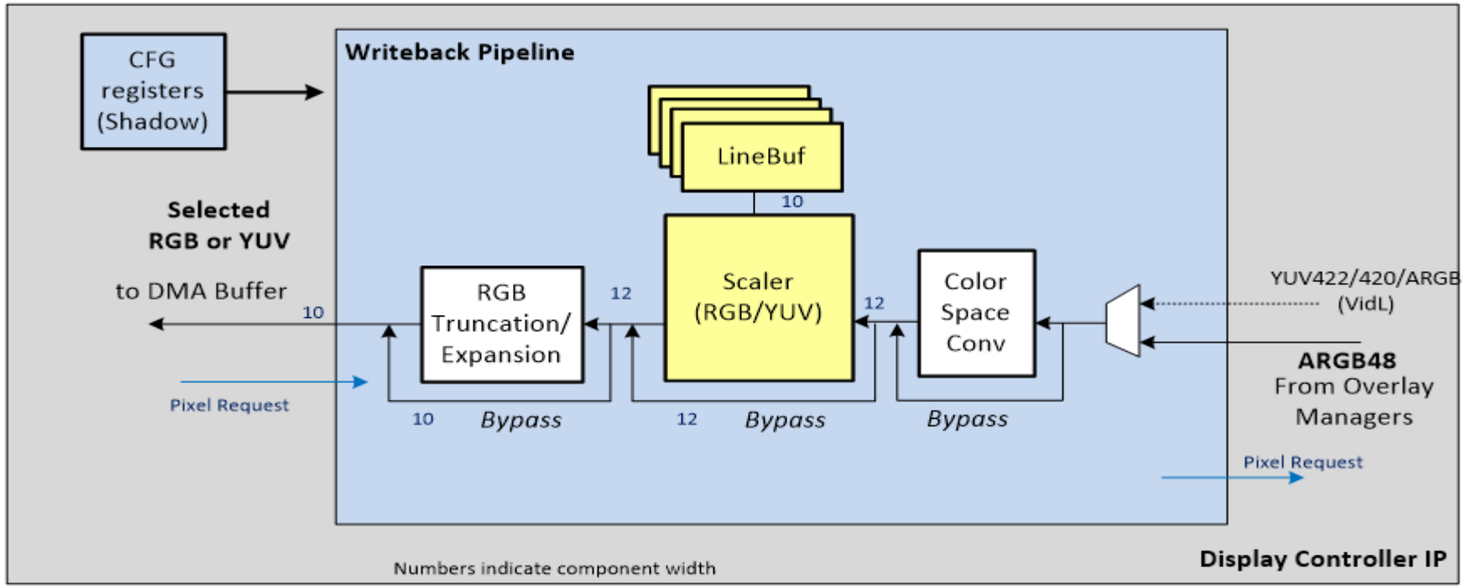


Video/Video Lite Pipeline



- The video pipeline is connected to the video DMA buffer controller for the input port and to the overlay manager. It performs all functionalities of a Graphics pipeline.
  - Polyphase-filter based resizer
  - Programmable BCHS (Brightness/Contrast/Hue/Saturation) control
  - Luma-key support (for Blue-Ray application)
  - Crop support at the output of the video pipeline
- Video Lite is identical to the Video pipeline except polyphase-filter based resizer is not included

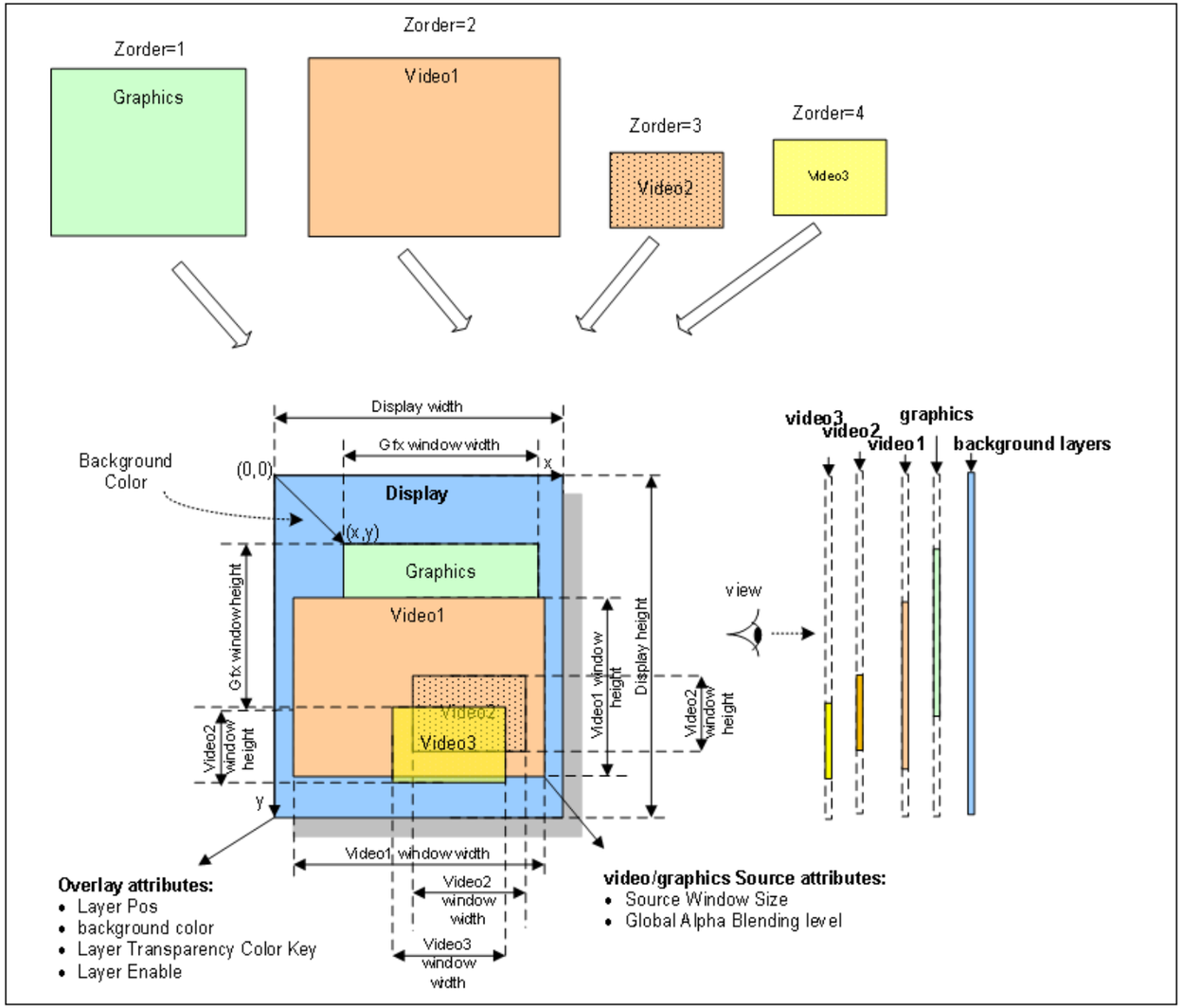
Write back Pipeline



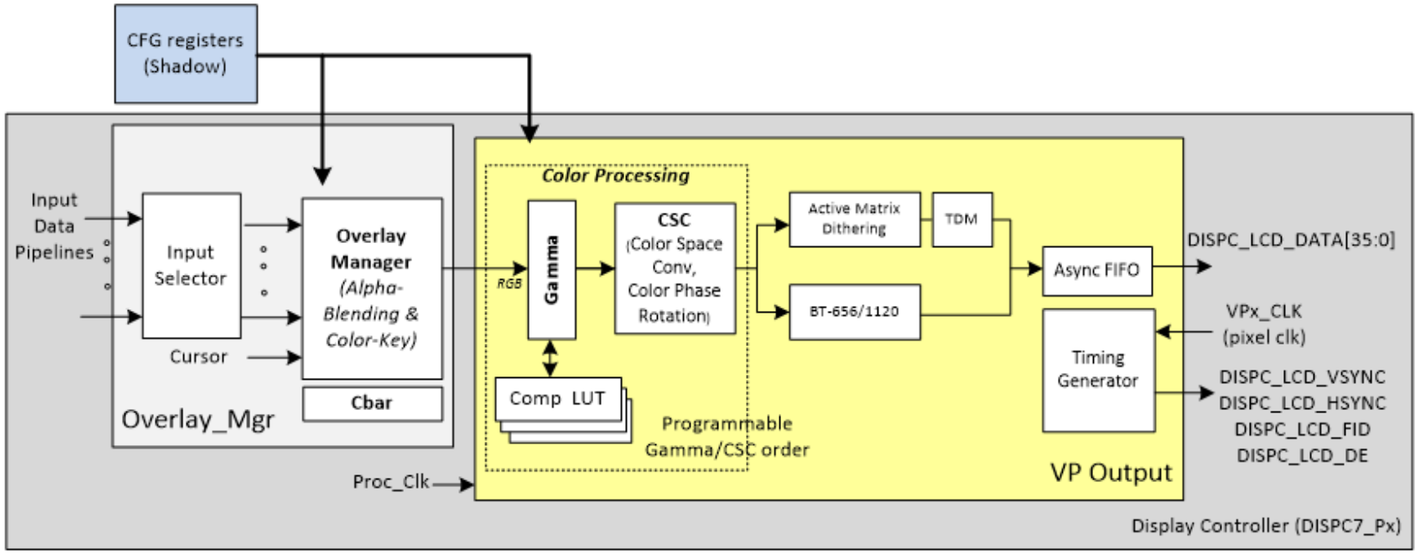
- The write-back pipeline is connected to
  - a pipeline output (video lite 2 pipeline only for J721E)
  - the output of one of the overlay manager, either in M2M (memory-to-memory) or Capture mode.
- Write back pipeline consists of
  - Color Space Conversion Unit
  - Resizer unit
  - RGB truncation logic

Overlay Manager

- Composites selected input layers together to generate the final display output frame based on
  - Priority rule following the display Z-order (selected by configuring the input selector)
  - Color key configurations (destination and source transparency)
  - Alpha Blending values (using the Alpha component of each pixel or a global alpha value set by the user)
  - Size and position attributes of the source window data
- Each overlay manager supports a simple internal color bar insertion to enable testing of display output interface without using the frame buffer data from the memory



Video Port

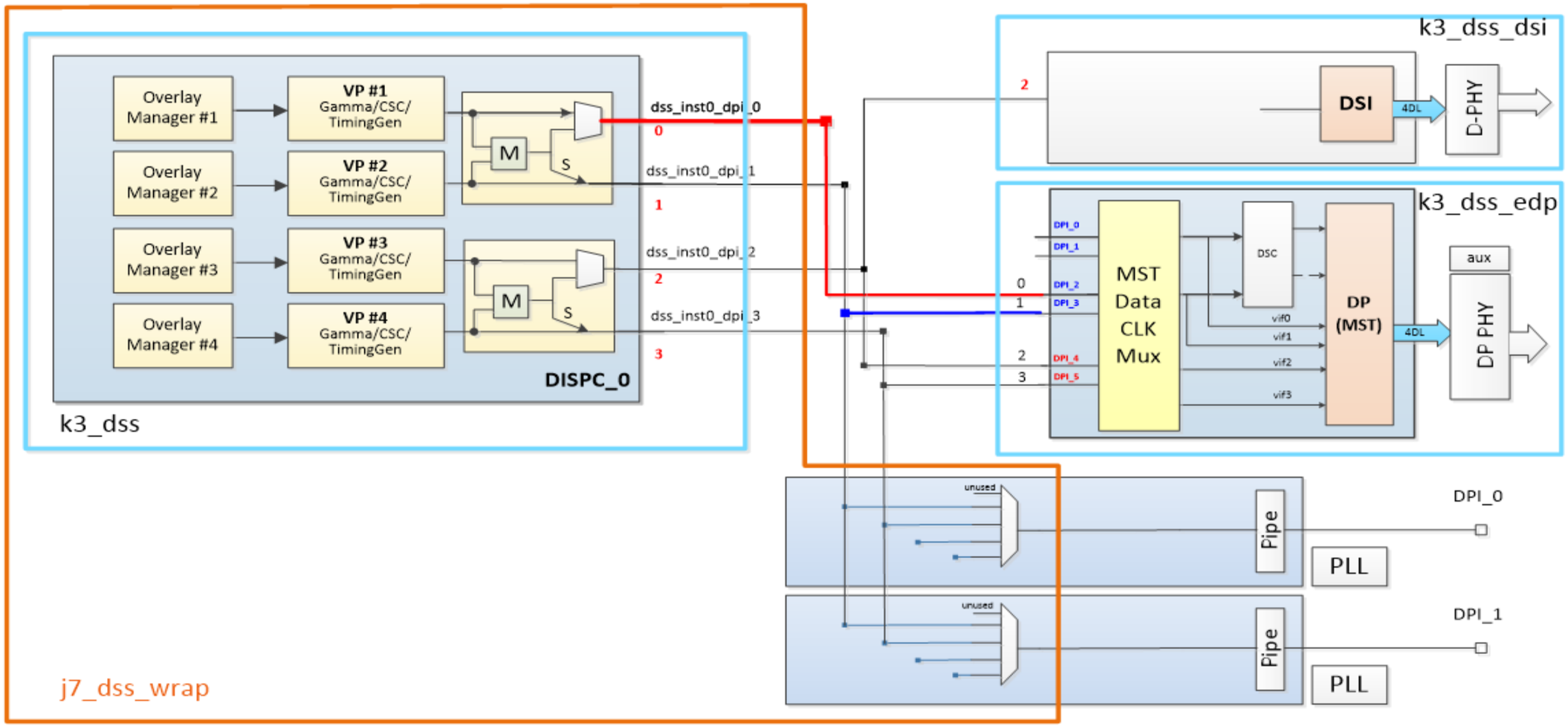


- 36-bit per pixel on the RGB output interface (12-bits per component)
- Independent programmable timing generators for each display output to support – up to **600 MHz** pixel clock video formats
- Fully programmable Color Space Conversion matrix to serve as color phase rotation (CPR) and/or as Brightness/Contrast/Saturation control on the combined output
- Selection between RGB(progressive) and YUV422(progressive/interlaced) output pixel format. YUV4:2:2 only available when BT656/1120 output is enabled.

Display Interfaces

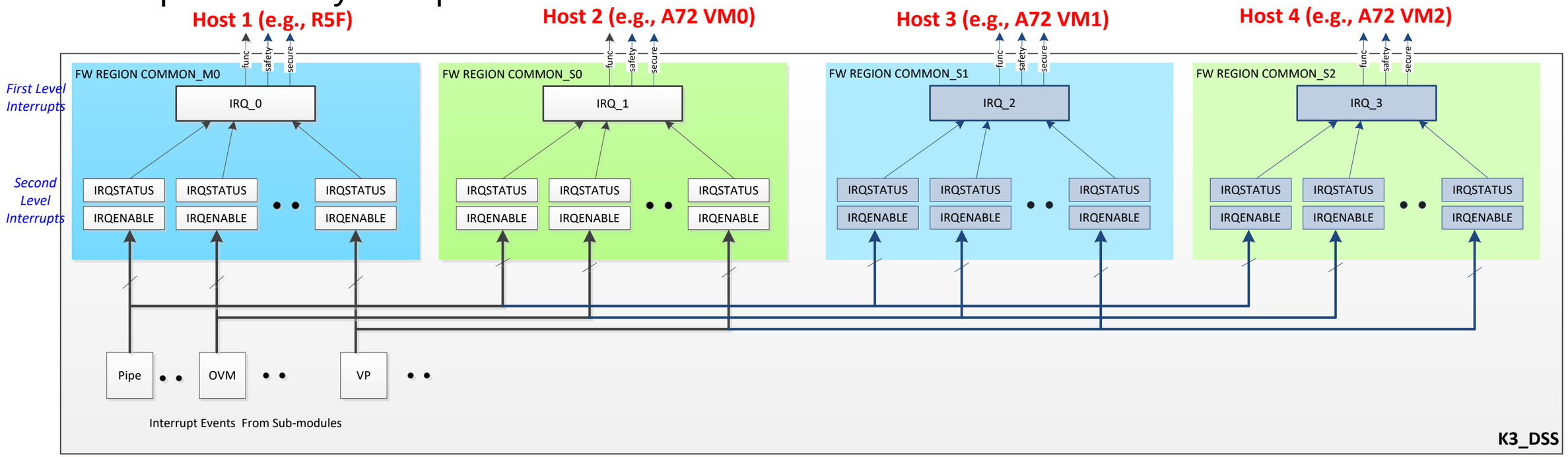
- J721E supports below display interfaces:
  - One Display Port/embedded Display Port (DP/eDP)
  - One MIPI Display Serial Interface (DSI) port
  - Two Display Parallel Interface (DPI) ports.
- Future devices will have multiple DP/eDP and DSI ports
- The DP/eDP interface supports the following features:
  - VESA Display Port (DP) 1.3 specification compliant with DSC(1.1)/FEC support
  - VESA embedded Display Port (eDP) 1.4 specification compliant
  - Physical link rates up to HBR3 (up to 8.1 Gbps per lane)
  - Video pixel clock rate range from 25-600MHz
- DSI interface supports following features:
  - 4,2, or 1 data lane configurations
  - MIPI Display Serial Interface (DSI) 1.3 specification compliant with DSC(1.1) support
  - D-PHY 1.2 Physical link rates up to 2.5 Gbps per lane
  - Video pixel clock rate range from 25-400MHz (up to 3K Display Panels)
  - 4,3,2, or 1 data lane configurations

Connectivity Diagram for J721E



DSS Interrupt Architecture

- DSS supports multiple interrupt outputs each of which can be set independently
- There are separate interrupt lines for different category of events:
  - Functional
  - Safety
  - Security
- There are four common register regions which allows fully independent monitoring/control of the interrupt events by multiple hosts

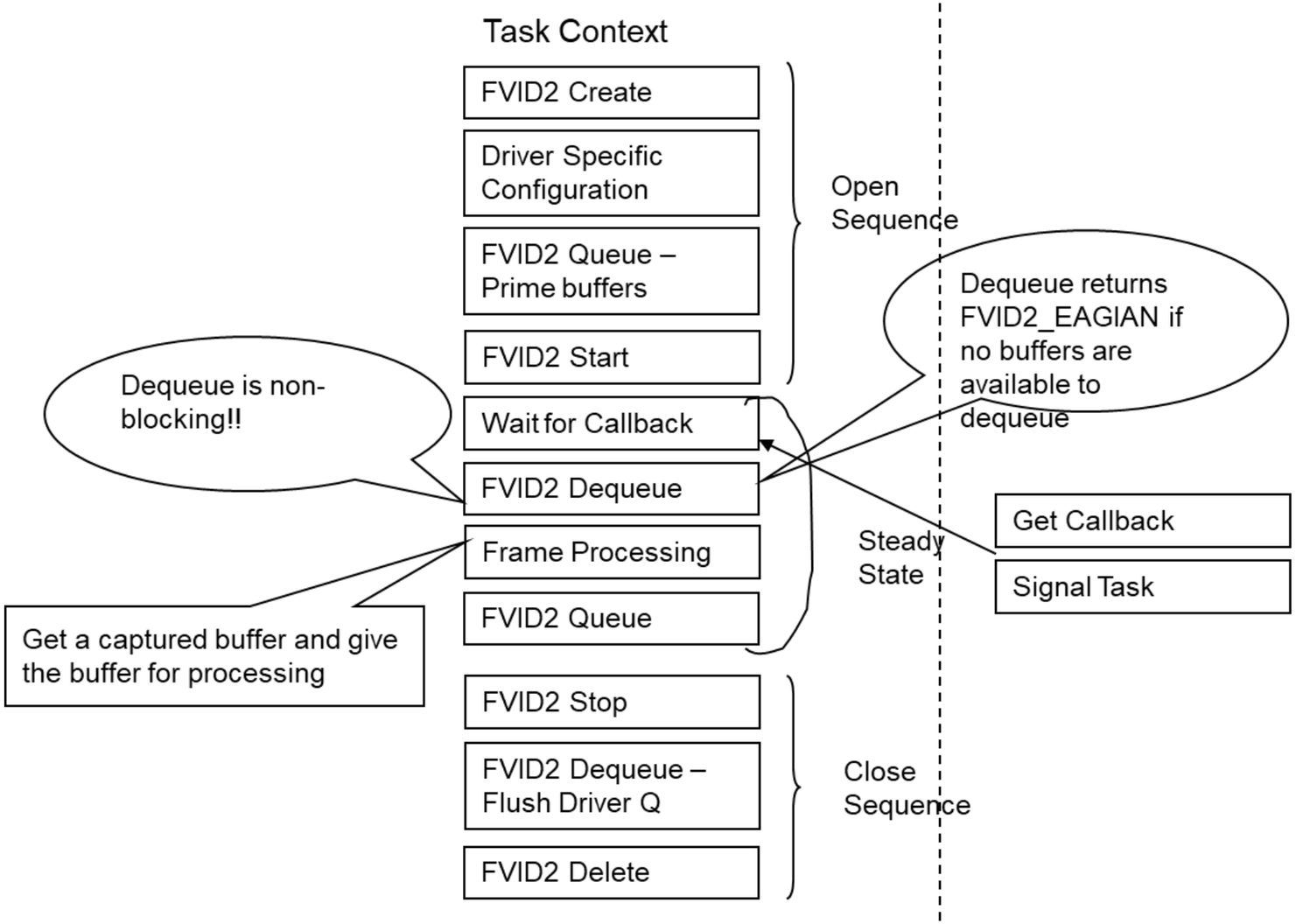


DSS Driver Overview

FVID2 Introduction

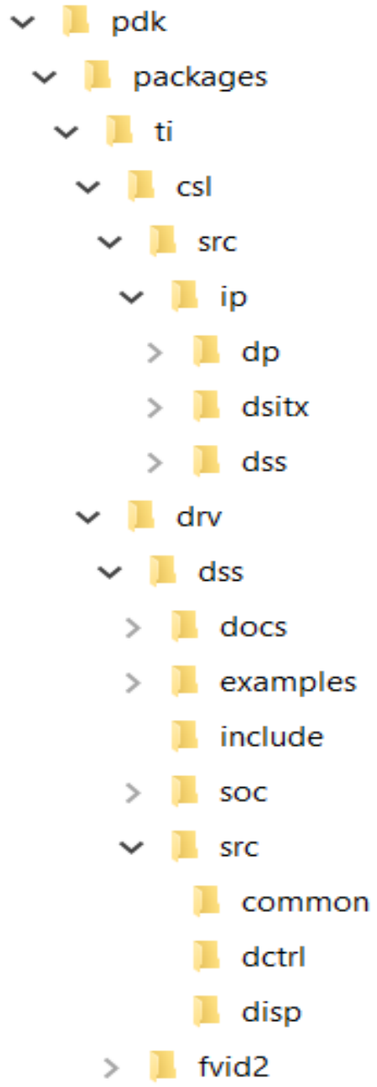
- What is FVID2?
  - Provides interface to streaming operations like queuing of buffers to driver and getting back a buffer from the driver
  - Abstracts the underlying hardware for the video application with a standard set of interface
  - Gives a same look and feel for video applications across different SOC
  - Interface is independent of OS/Hardware/Driver
  - FVID2 is currently supported on BIOS
- What is not FVID2?
  - Not the actual driver
  - Does not define hardware specific APIs and structures
- FVID2 Interfaces
  - Fvid2_init, Fvid2_deInit
  - Fvid2_create, Fvid2_delete
  - Fvid2_start, Fvid2_stop
  - Fvid2_queue, Fvid2_dequeue
  - Fvid2_processFrames, Fvid2_getProcessedFrames
  - Fvid2_control

Typical Application Flow



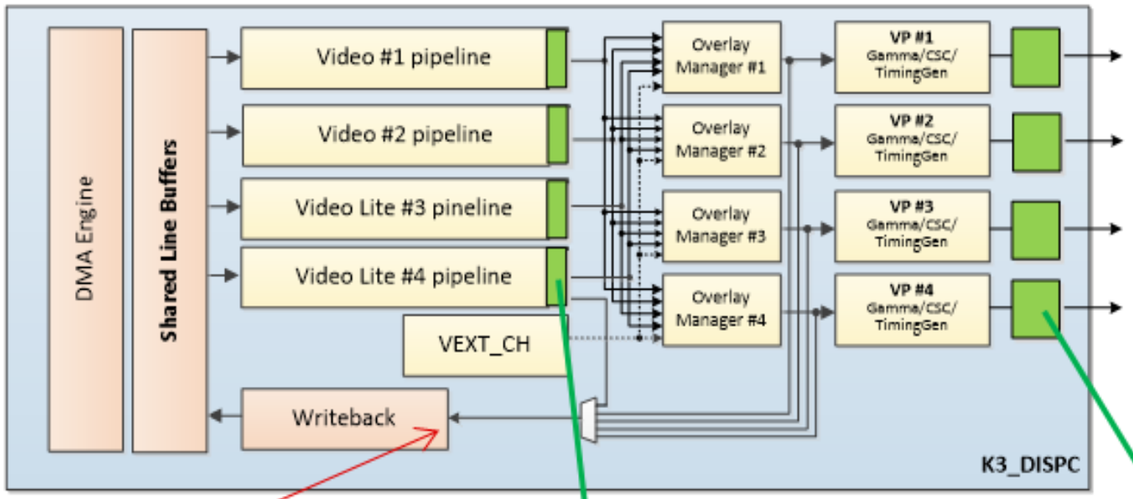
Directory Structure

- DSS CSL RL and CSL FL files are present in <pdk>/packages/ti/csl/src/ip/dss
- FVID2 driver is present in <pdk>/packages/ti/drv/fvid2
- DSS driver is present in <pdk>/packages/ti/drv/dss



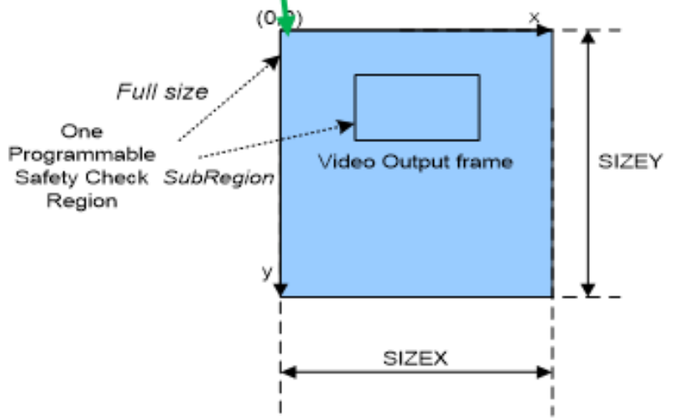
DSS Advance Features

DSS Safety Checks

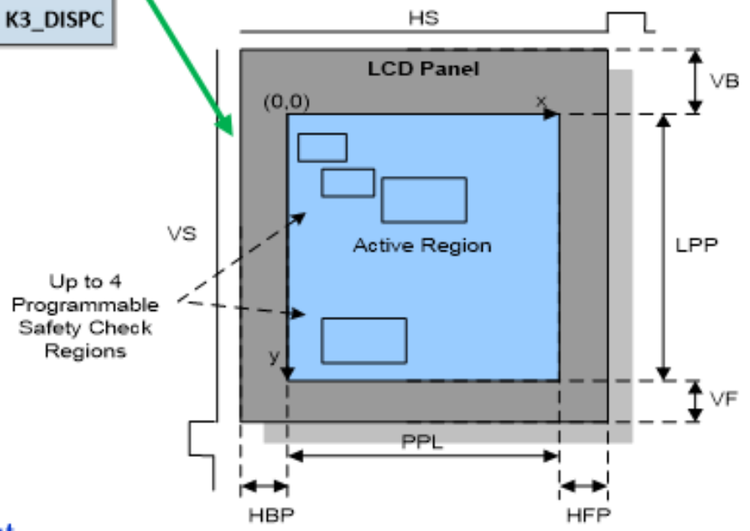


- CRC generation/comparison on multiple programmable ROIs in each pipe and VP
- Freeze Frame Detection based on CRC comparison over multiple frames

WB to DDR -
ROI data capture
for post processing



Safety region at Video Pipe Output



Safety regions at VP output

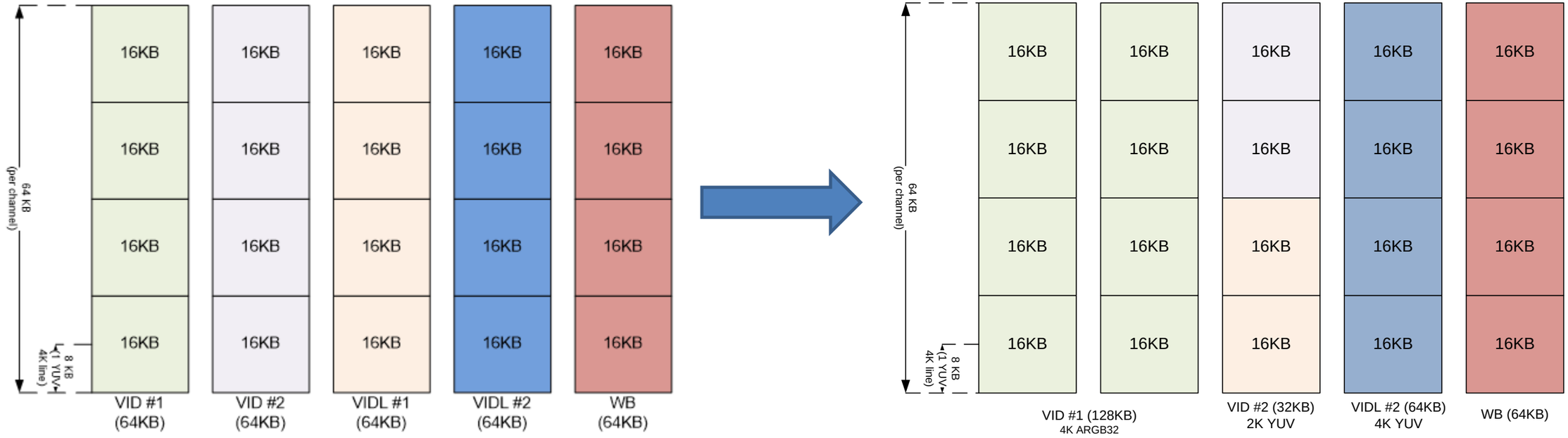
FIFO Buffer Mapping (1/2)

- When Scaler is enabled, the DMA needs to send multiple lines simultaneously (in parallel) to the VID pipeline.
- The number of lines could be up to 4 lines in parallel (the case when there is downscaling by 4).
- The buffer in this case needs to be used in “ping-pong” mode.
- Hence, the requirement on the buffer memory size is to have space to store at least eight full lines (The size in bytes of a line depends on the resolution as well as the pixel format)
- The below table lists the size of a line in bytes for the common resolutions and formats

Resolution	Format	1-line (in KB)	8-lines (in KB)
1920 (1080p)	ARGB32 (32bpp)	7.5	60
	RGB24p (24bpp)	5.625	45
	YUV422 (16bpp)	3.75	30
2560 (2.5K)	ARGB32 (32bpp)	10	80
	RGB24p (24bpp)	7.5	60
	YUV422 (16bpp)	5	40
4096 (4K)	ARGB32 (32bpp)	16	128
	RGB24p (24bpp)	12	96
	YUV422 (16bpp)	8	64

FIFO Buffer Mapping (2/2)

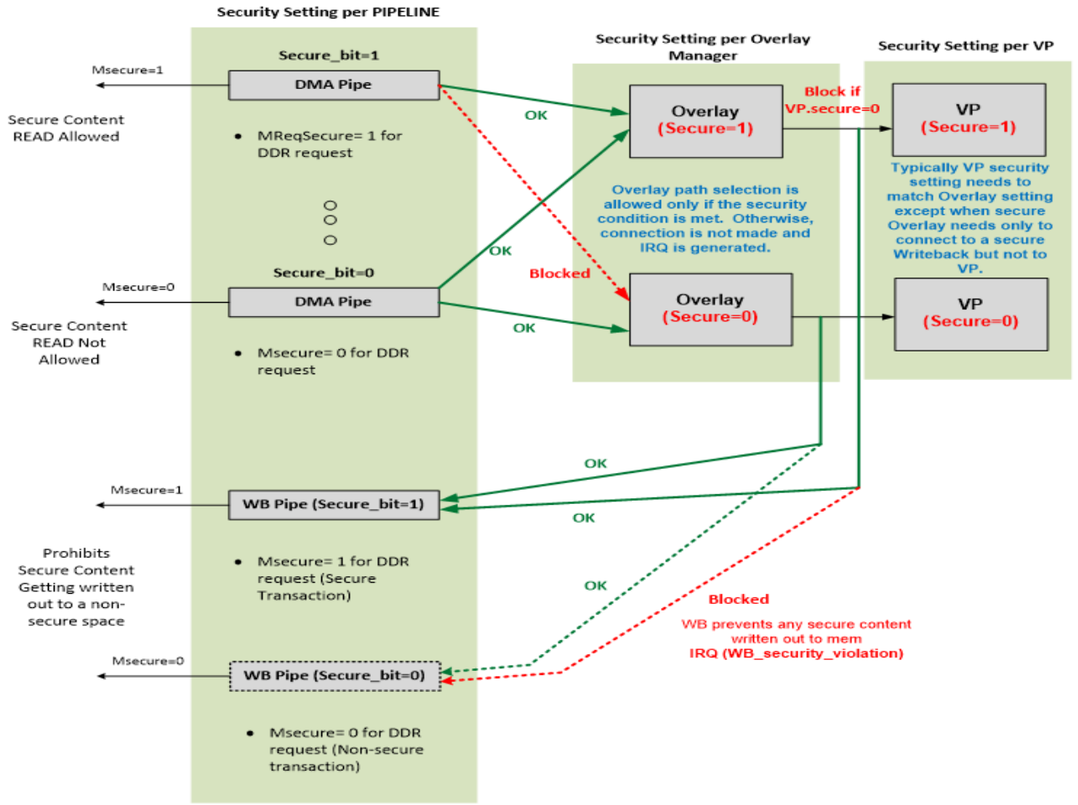
- Each channel in DMA is allocated a uniform buffer size of 64 Kilo Bytes at reset.
- To support resolutions larger than 4K YUV or 2K ARGB (with ping-pong configuration), the default buffer size of 64KB per RD channel is not sufficient as demonstrated above.
- In such cases, the design allows dynamic repartition of the allocated buffers across the different RD channels. In this way each RD channel can have a variable buffer size allocated to it, starting from a minimum of 16KB to a maximum of 256KB, in 16KB increments.



Security Isolation

- Configuration Firewall – Each Common Region, Pipe, Overlay & Video Port has 64KB FW region
- Secure Mode setting – Hardware based connection check/illegal connection prevention
- Multiple sets of Interrupt Logic for secure/non-secure host handling

Interface name	Region name	Addr size	Byte offset	Request ID
dss_inst0_vbusp_cfg	dispc_0_COMMON_M	16	0x0	0
dss_inst0_vbusp_cfg	dispc_0_COMMON_S0	16	0x10000	0
dss_inst0_vbusp_cfg	dispc_0_VIDL1	16	0x20000	0
dss_inst0_vbusp_cfg	dispc_0_VIDL2	16	0x30000	0
dss_inst0_vbusp_cfg	dispc_0_VID1	16	0x50000	0
dss_inst0_vbusp_cfg	dispc_0_VID2	16	0x60000	0
dss_inst0_vbusp_cfg	dispc_0_OVR1	16	0x70000	0
dss_inst0_vbusp_cfg	dispc_0_VP1	16	0x80000	0
dss_inst0_vbusp_cfg	dispc_0_OVR2	16	0x90000	0
dss_inst0_vbusp_cfg	dispc_0_VP2	16	0xa0000	0
dss_inst0_vbusp_cfg	dispc_0_OVR3	16	0xb0000	0
dss_inst0_vbusp_cfg	dispc_0_VP3	16	0xc0000	0
dss_inst0_vbusp_cfg	dispc_0_OVR4	16	0xd0000	0
dss_inst0_vbusp_cfg	dispc_0_VP4	16	0xe0000	0
dss_inst0_vbusp_cfg	dispc_0_WB	16	0xf0000	0
dss_inst0_vbusp_cfg	dispc_0_COMMON_S1	16	0x100000	0
dss_inst0_vbusp_cfg	dispc_0_COMMON_S2	16	0x110000	0

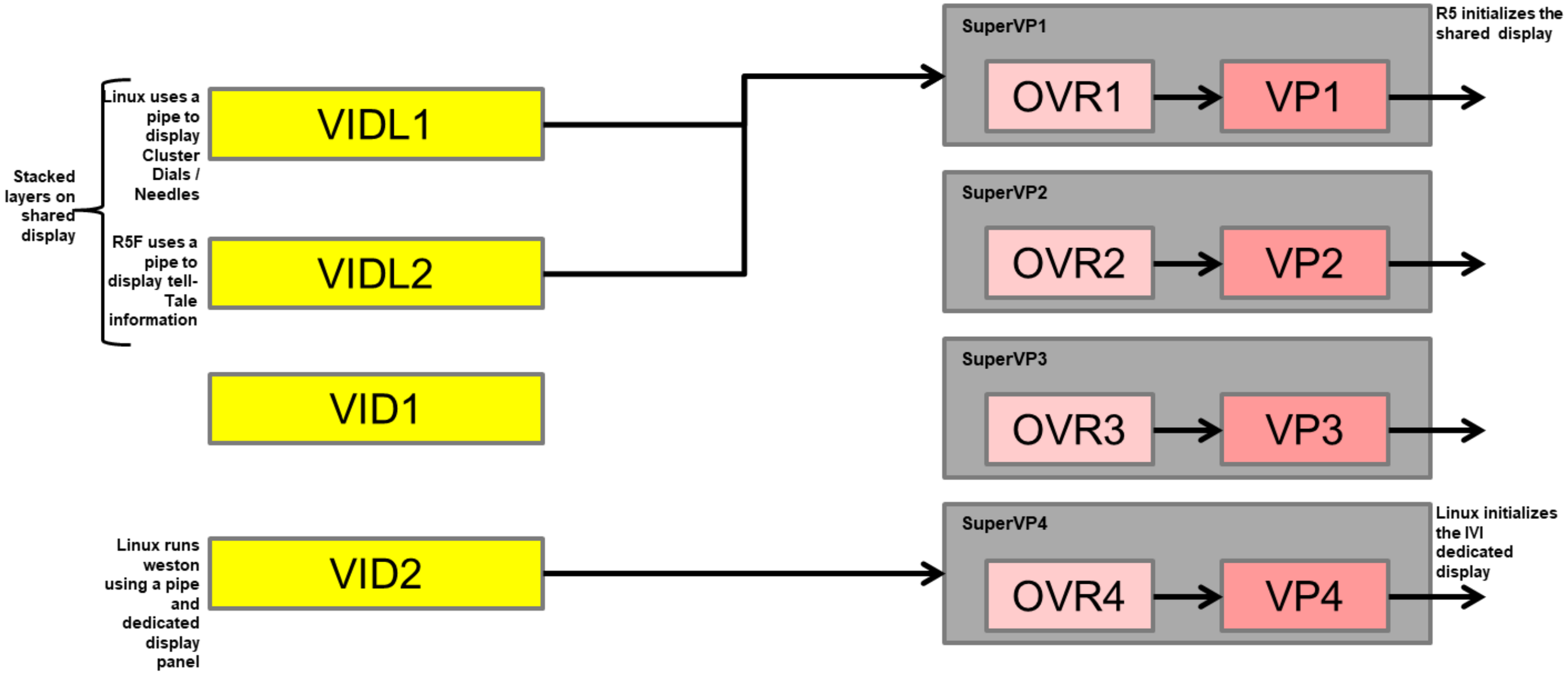


Display Sharing

Display Sharing for K3 SoCs

- Cockpit Use case: Simultaneous Safety and QM Requirements on Display
  - Safety content is generated by a safety certified OS running on a dedicated core
  - QM content is usually generated by HLoS (Linux)
  - There may be multiple instances of QM SW running on different cores / same core with virtualization support
- Static partitioning of Resources
  - Resources are allocated at system boot time
- VP thread as Atomic Resource Partition Entity
  - VP Thread : A VP, Overlay-Manager and all pipes terminating at the VP
  - Any core can drive one / multiple end-to-end VP threads
  - All pipe, overlay manager and VP register updates for a thread shadowed by asserting VPx.GO bit
  - All interrupts for pipes and VP in a thread used by the driving core
- Distributed DSS Initialization
  - R5F boots up first and initializes DSS
  - Any core who owns a VP thread initializes the thread specific setup
  - Pipes are initialized and setup on-demand
- Pipe Subscription and IPC based Frame update requests
  - VP thread owner can
    - Reserve a set of pipes to be used by local applications
    - Export a set of pipes to a remote core via a local display-server
  - Remote cores probe a display-server to get a list of exported pipes
  - Remote cores send frame update requests to display-server for exported pipes

Design: Logical HW Split



Thank You!