Demystifying the Embedded Linux Graphics Stack: An Easy Introduction for Beginner

Parthiban

Linumiz

whoami

- Parthiban, Linumiz GmbH
  - Embedded Linux and Zephyr RTOS development, consulting, training
  - Embedded Linux development: BSP, u-boot, Linux Kernel, Yocto Project, Buildroot
  - Zephyr: SoC, Board support, drivers
  - www.linumiz.com
- Living in **Berlin**, Germany

Linumiz

Linumiz and its offering

Linumiz is a domain-agnostic embedded software engineering and consulting company specializing in Linux and Zephyr RTOS. Our expertise spans board bring-up, Board Support Package (BSP) development, device driver implementation, and over-the-air (OTA) software updates, primarily for ARM-based systems but not limited to them. We empower businesses with tailored embedded solutions, ensuring reliability, scalability, and seamless hardware-software integration.

Embedded Linux

- Board support package
- Driver Development
- Board Bring up
- Custom Linux build
- Application Development

Zephyr RTOS

- Board porting
- Firmware Development
- Driver development
- Update management
- Upstream activities

Consulting

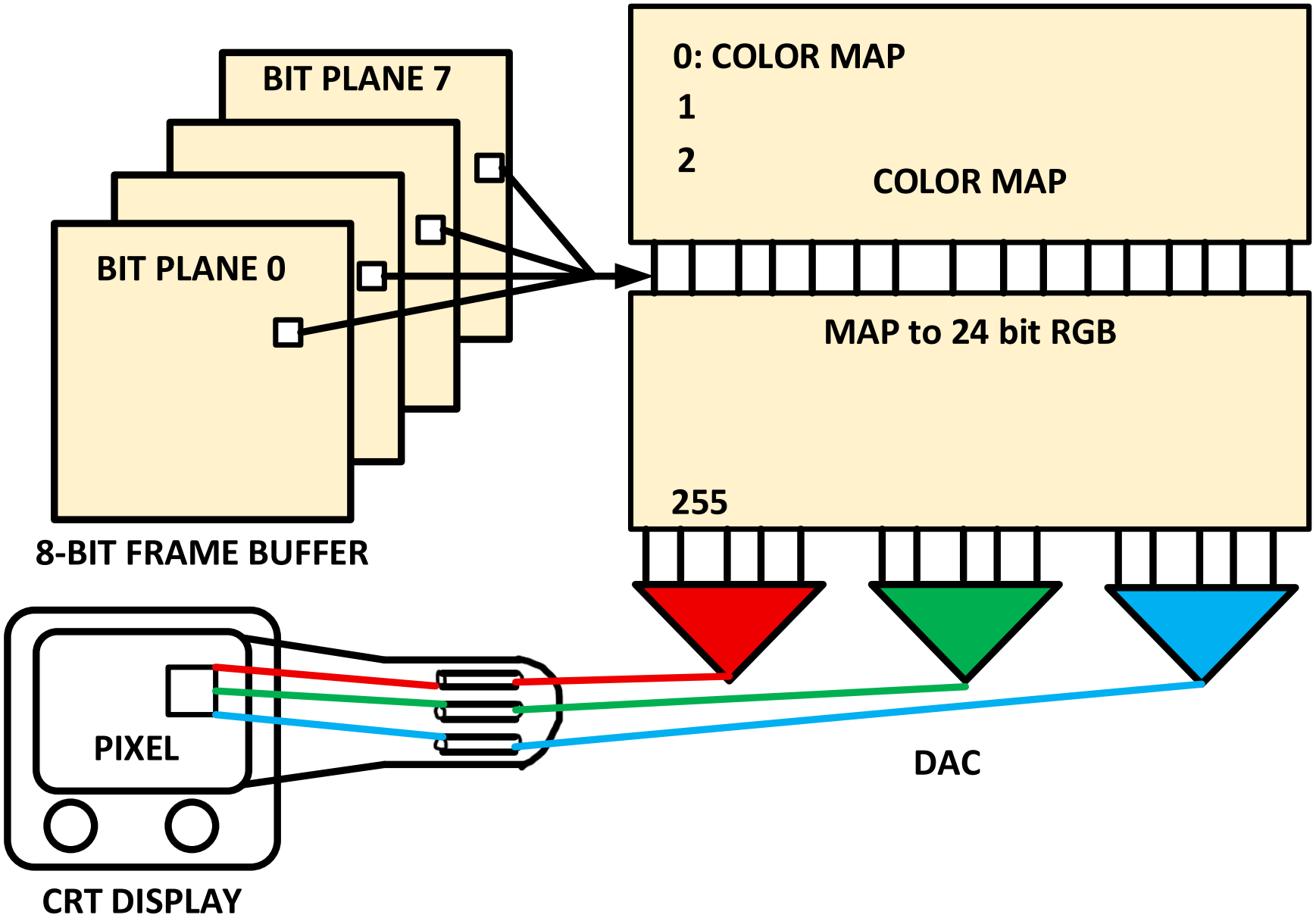
- HW recommendation
- Architecture design
- System development
- Support for specific issues
- Support Services

Device security

- Static analysis
- Dynamic analysis
- Device hardening
- Root of Trust (RoT)
- Security recommendations

Linumiz

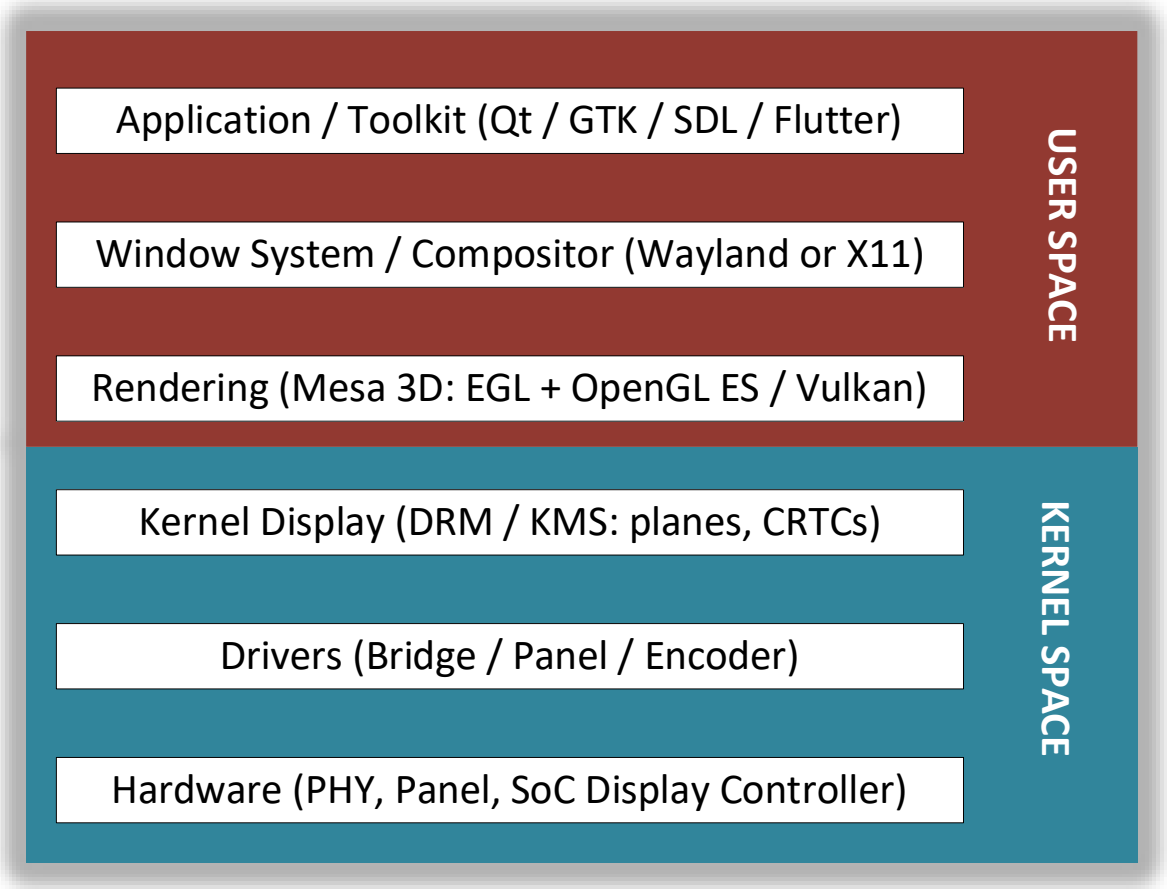
Color Raster Display



Linumiz

Memory ⟶ Photons: Linux Graphics Dataflow

- Stack: App → Compositor/Mesa (userspace) → DRM/KMS → Drivers → Hardware (kernel).
- Handoff line: userspace makes buffers; kernel scans out to the display.
- Buffers: wl_shm (CPU RAM) vs dmabuf (zero-copy GPU/Video) — dmabuf enables direct scanout.
- Atomic KMS + planes = tear-free flips; target PRIMARY for UI (XRGB8888), OVERLAY for video (NV12/P010).



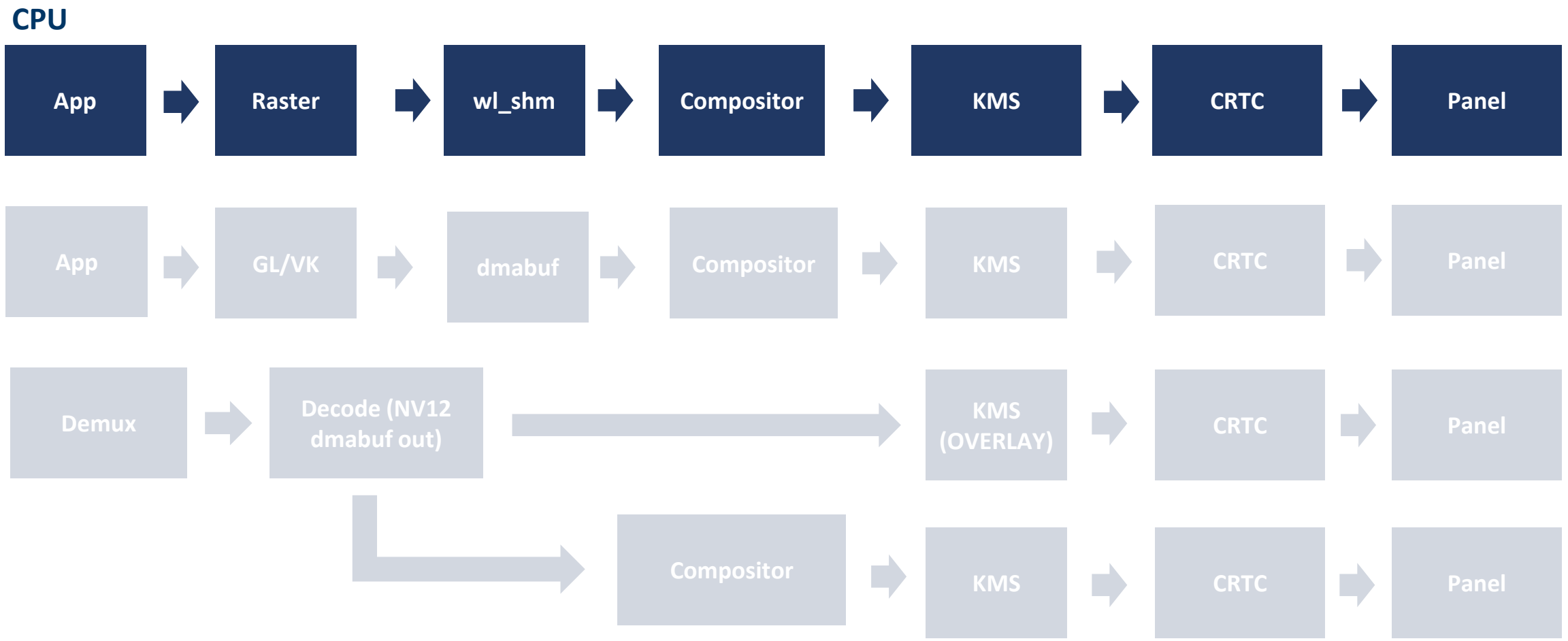
Linumiz

The Subway Map (CPU/GPU/Video Lanes)

- Three lanes: CPU/wl_shm, GPU/dmabuf, Video (NV12/P010 dmabuf).
- Boundary: Compositor → KMS/CRTC/planes; atomic flips at vblank.
- Direct scanout if fullscreen/fit + plane supports format+modifier; else GPU composite.
- Prefer dmabuf and proper fences; UI=XRGB8888→PRIMARY, Video=NV12/P010→OVERLAY.

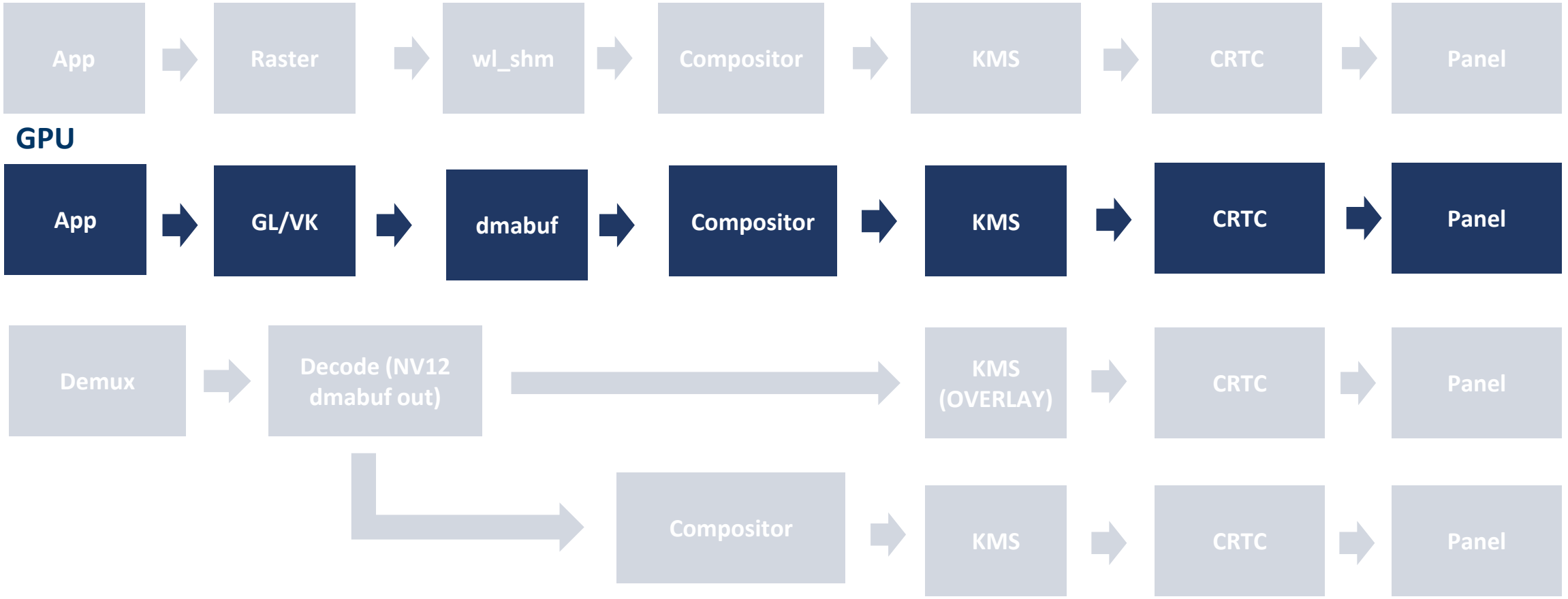
Linumiz

The Subway Map (CPU/GPU/Video Lanes) – Cont.,



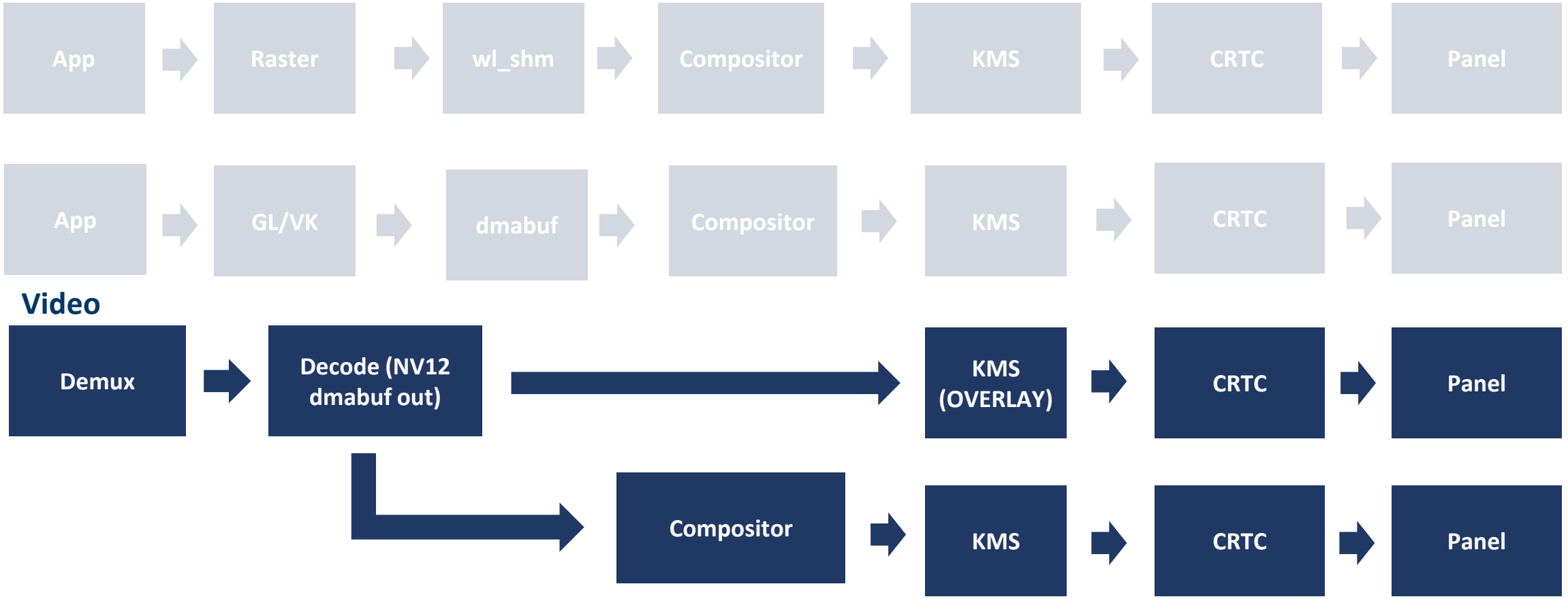
Linumiz

The Subway Map (CPU/GPU/Video Lanes) – Cont.,



Linumiz

The Subway Map (CPU/GPU/Video Lanes) – Cont.,



Linumiz

Pixels, Buffers, Formats (Ground Zero)

- dmabuf (zero-copy handle)
  - Your planes/buffers can be shared as a dmabuf FD and referenced by the FB.
- FB = (format, pitches, offsets, modifier); it’s just metadata.
- RGB UI → PRIMARY; NV12 video → OVERLAY (if supported).
- Modifiers must match plane support; otherwise a copy/resolve happens.
- NV12 needs even width/height; beware alignment.
- If plane can’t accept it (format/scale/rotation), compositor/GPU converts to RGB

Pixels, Buffers, Formats (Ground Zero) – Cont.,

WHAT LIVES IN MEMORY RGB (XRGB8888)

p0 -> pixel rows
stride0 bytes/row

(one plane)

NV12 (YUV 4:2:0)

Y plane (luma)
Y_pitch bytes

UV plane (chroma)
UV_pitch bytes

(two planes: Y + interleaved UV)

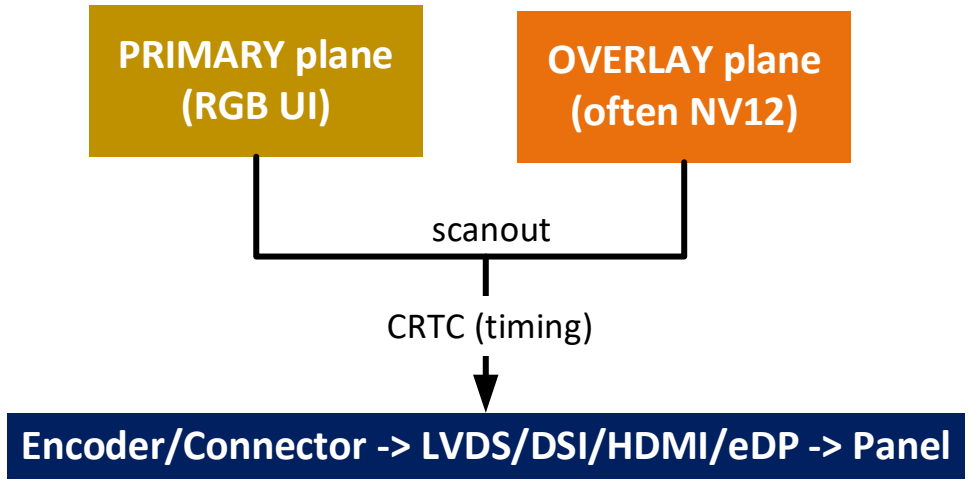
TURN INTO A DRM FRAMEBUFFER (metadata only)

format: XRGB8888
pitches: [stride0]
offsets: [0]

format: XRGB8888
pitches: [stride0]
offsets: [0]

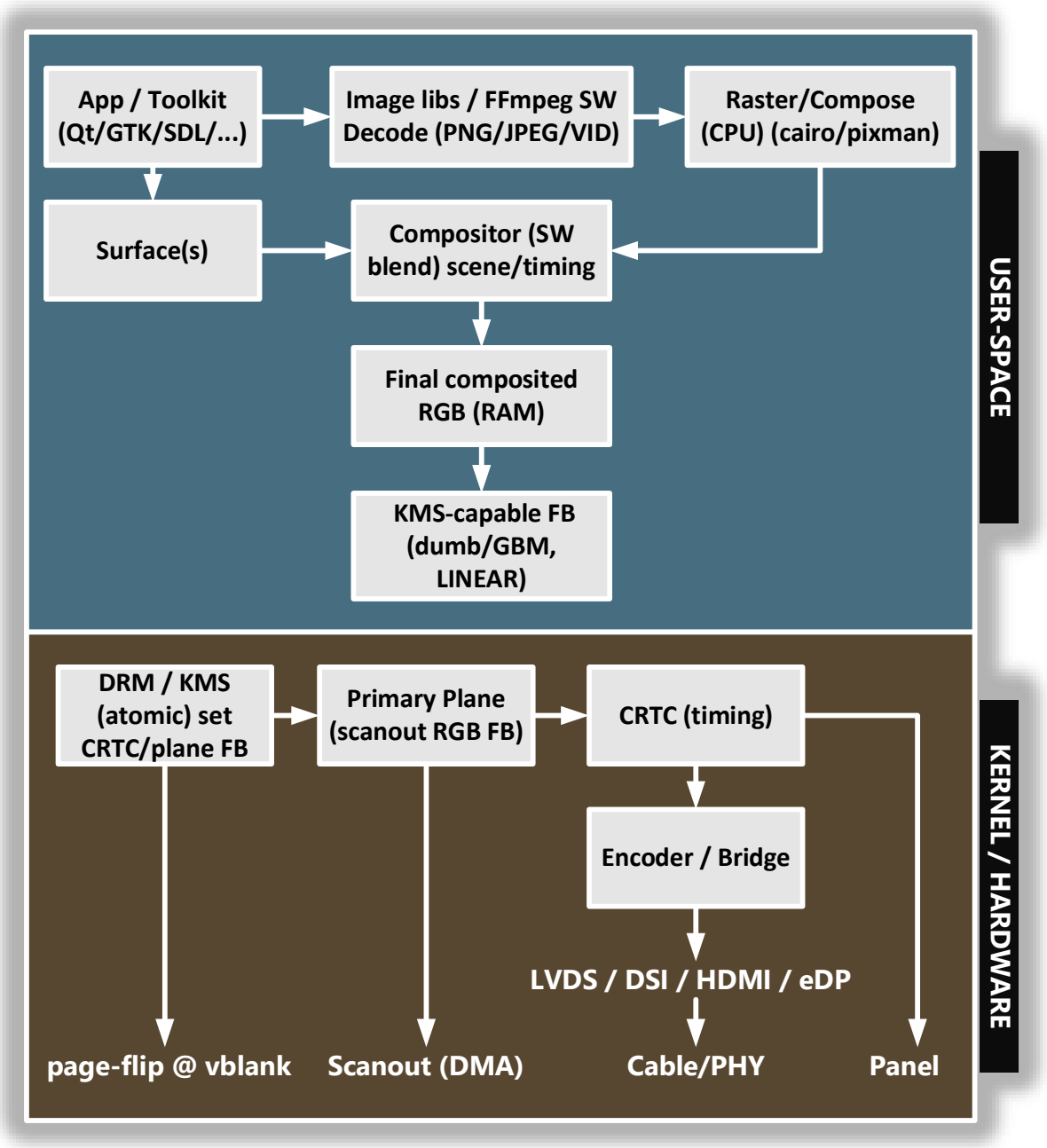
modifier: LINEAR / TILED / COMPRESSED (e.g., AFBC)

(FB is a descriptor pointing at memory; no pixels inside)

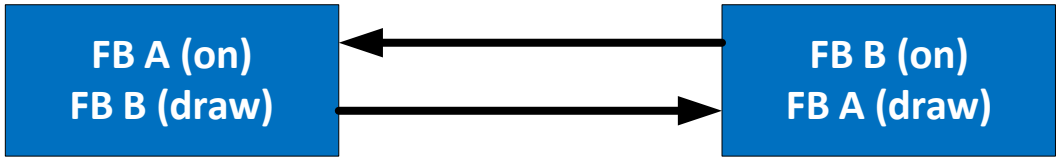


Linumiz

CPU-only rendering path



Double-buffering (tear-free page-flips at vblank)

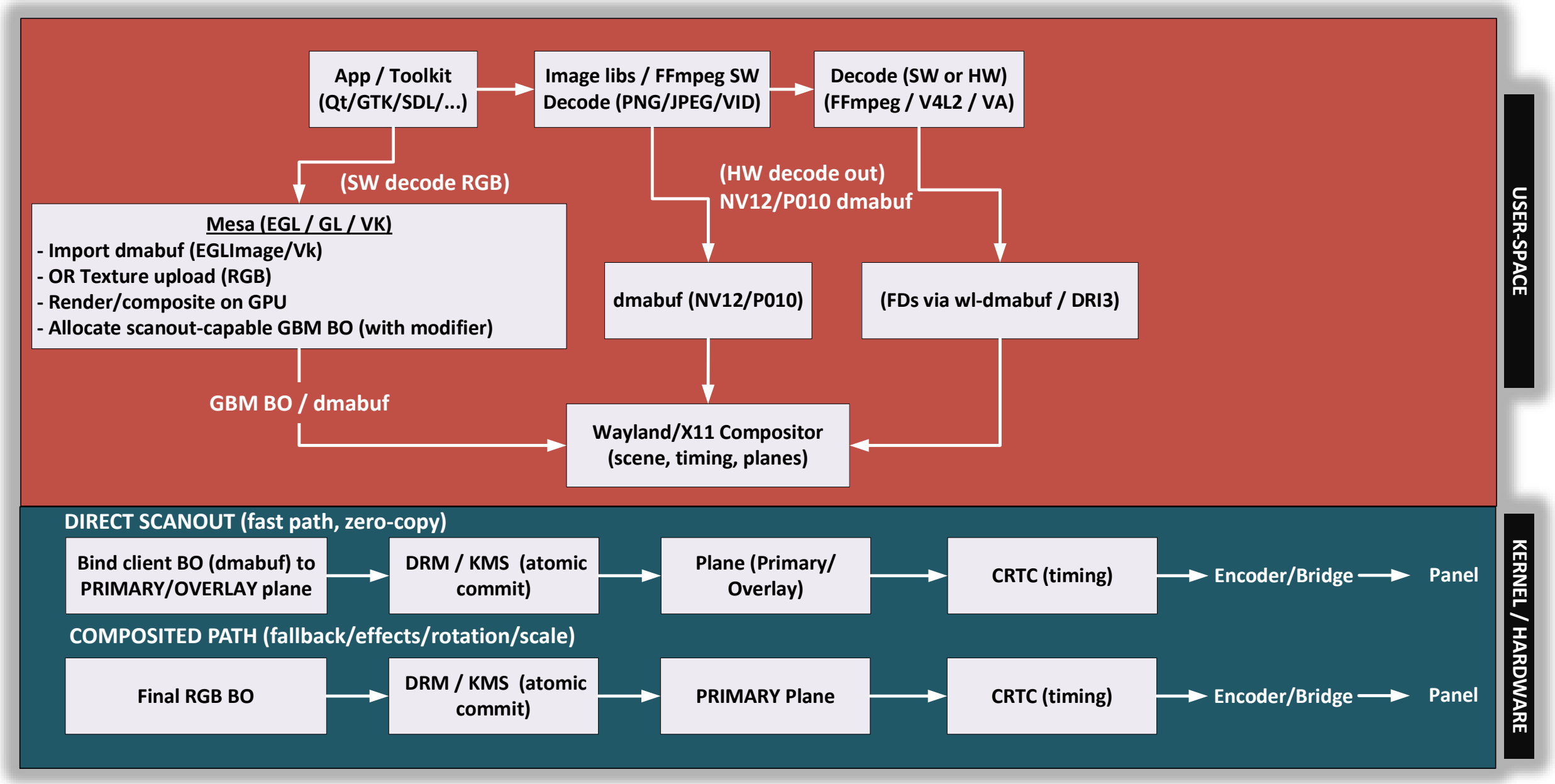


Linumiz

GPU - Accelerated path

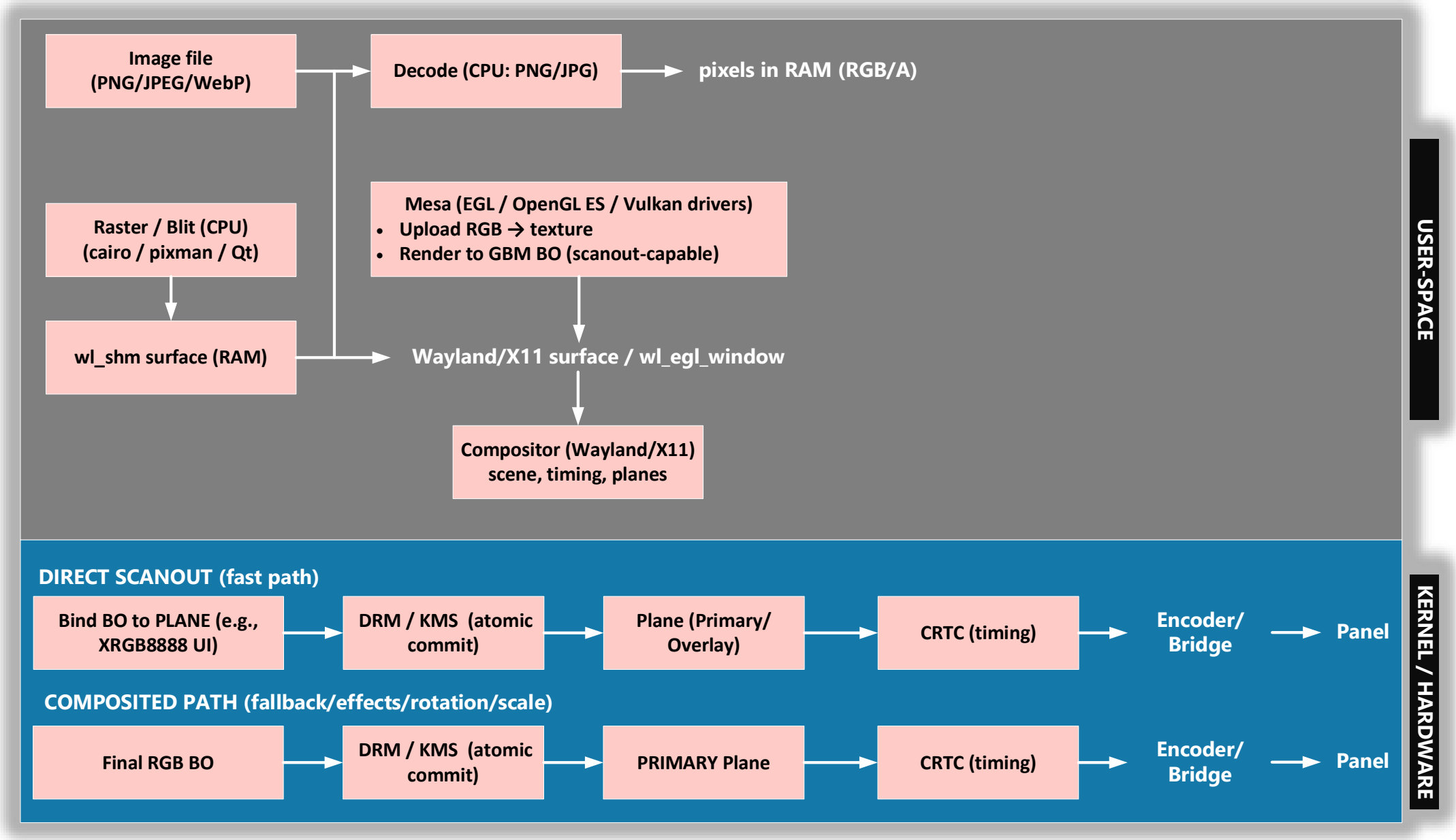
- wl-dmabuf / DRI3 pass buffer FDs from apps to the compositor (zero-copy)
- Choose formats/modifiers the plane supports for direct scanout (e.g., XRGB8888 for UI, NV12/P010 for video overlay).
- If any constraint fails, compositor GPU-composites to a final RGB BO and scans that out via PRIMARY.

GPU - Accelerated path - Cont.,



Linumiz

Opening an image

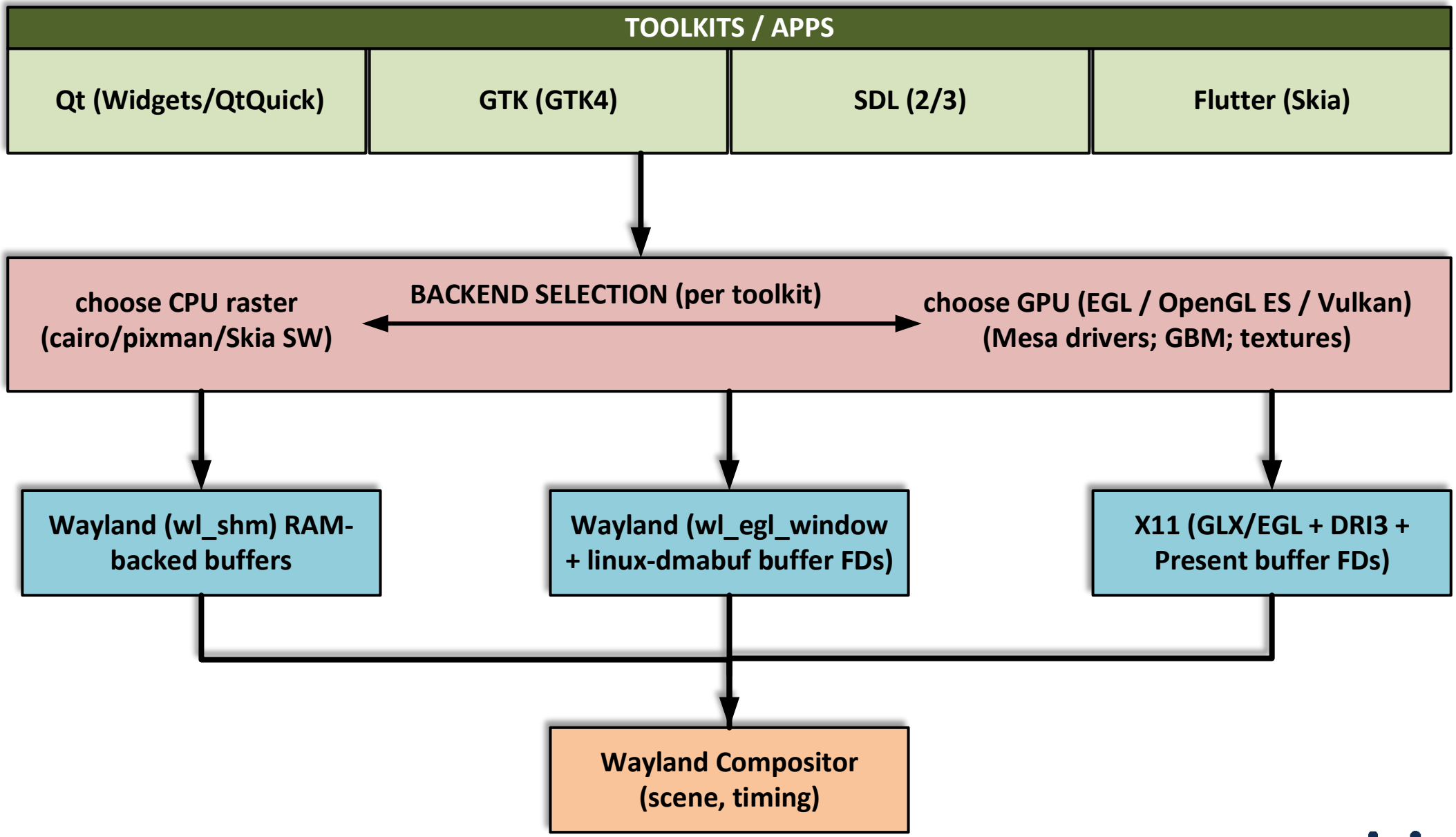


Linumiz

Toolkits & App Frameworks (user-space only)

- CPU path → wl_shm (Wayland) / XShm (X11) = RAM buffers; compositor likely copies/composites.
- GPU path → linux-dmabuf/DRI3 FDs = zero-copy capable; compositor may direct-scanout.
- Env knobs (examples): Qt: QT_QUICK_BACKEND, QT_QPA_PLATFORM; GTK: GSK_RENDERER, GDK_BACKEND; SDL:
- SDL_VIDEODRIVER/SDL_RENDER_DRIVER; Flutter: prefers Skia GPU, falls back to software.

Toolkits & App Frameworks (user-space only) – Cont.,

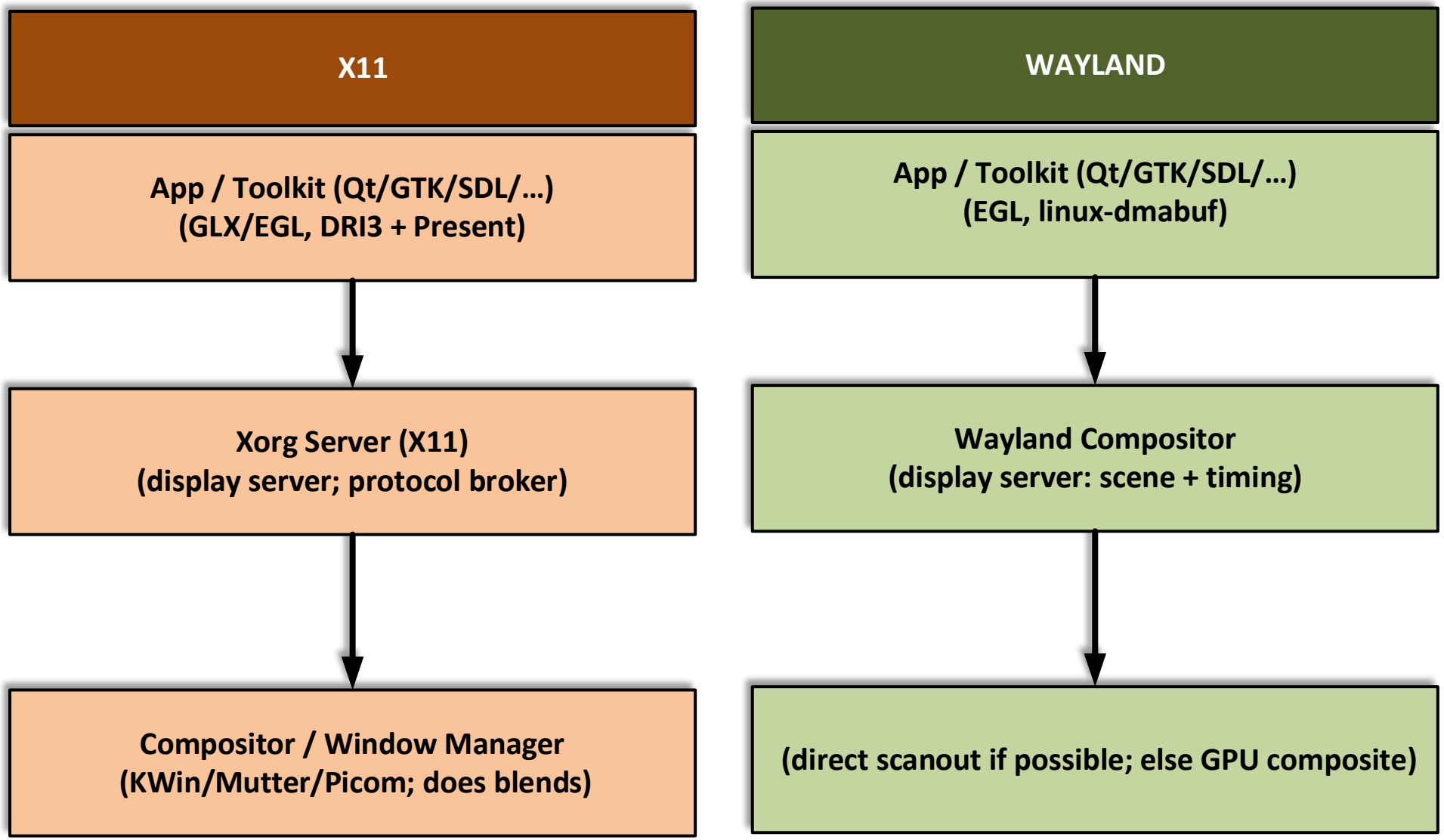


Linumiz

Window systems: X11 vs WAYLAND

- X11: Client → Xorg → Compositor → KMS (extra hop; more copies/latency likely)
- Wayland: Client → Compositor → KMS (fewer hops; simpler)
- Buffer passing: X11 uses DRI3+Present; Wayland uses linux-dmabuf (+ feedback for formats/modifiers)
- Embedded wins on Wayland because:
  - fewer hops → lower latency
  - dmabuf + plane support → more direct scanout, fewer GPU composites
  - better damage/presentation-time integration

Window systems: X11 vs WAYLAND – Cont.,

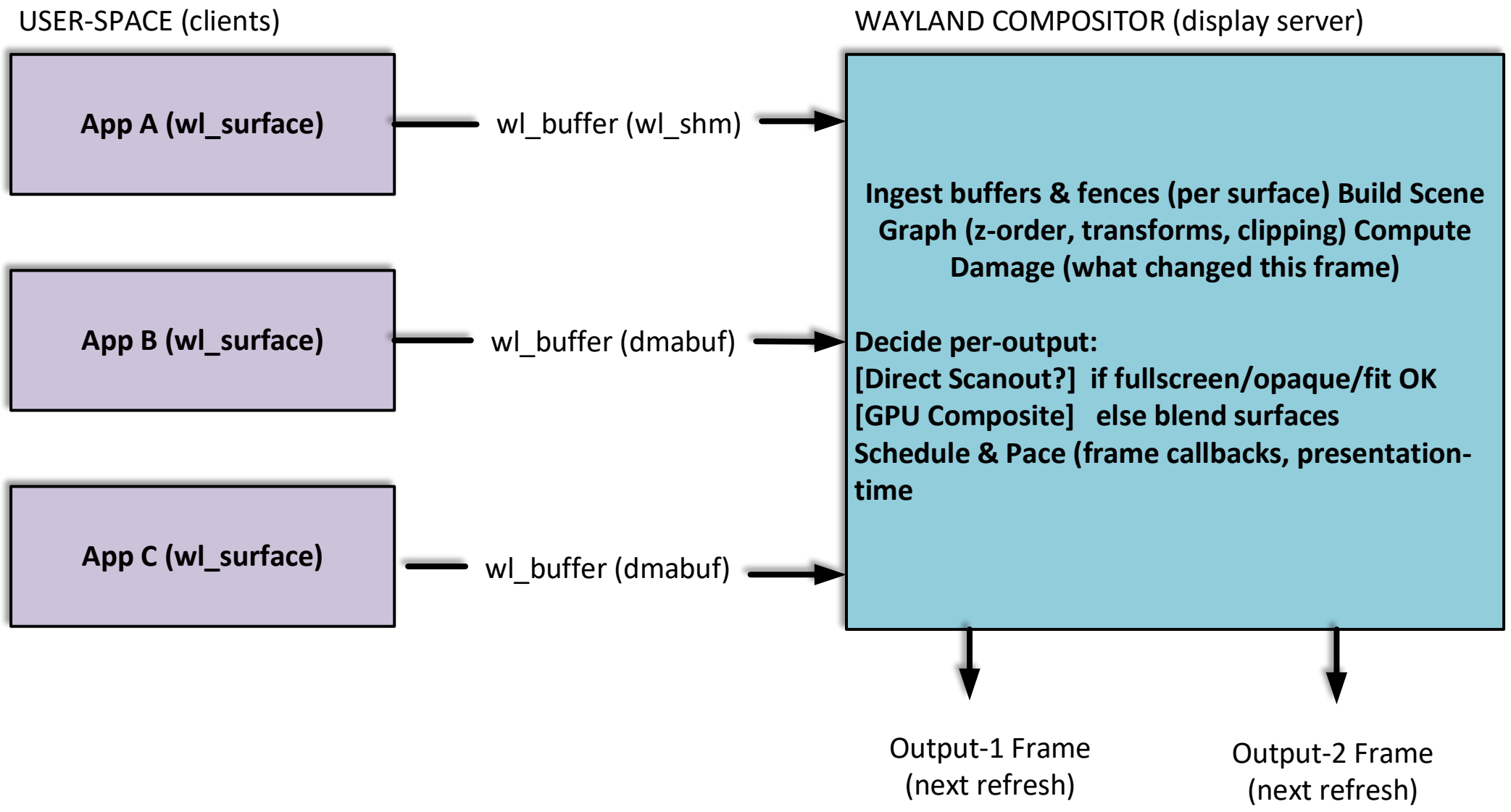


Linumiz

The compositor’s job

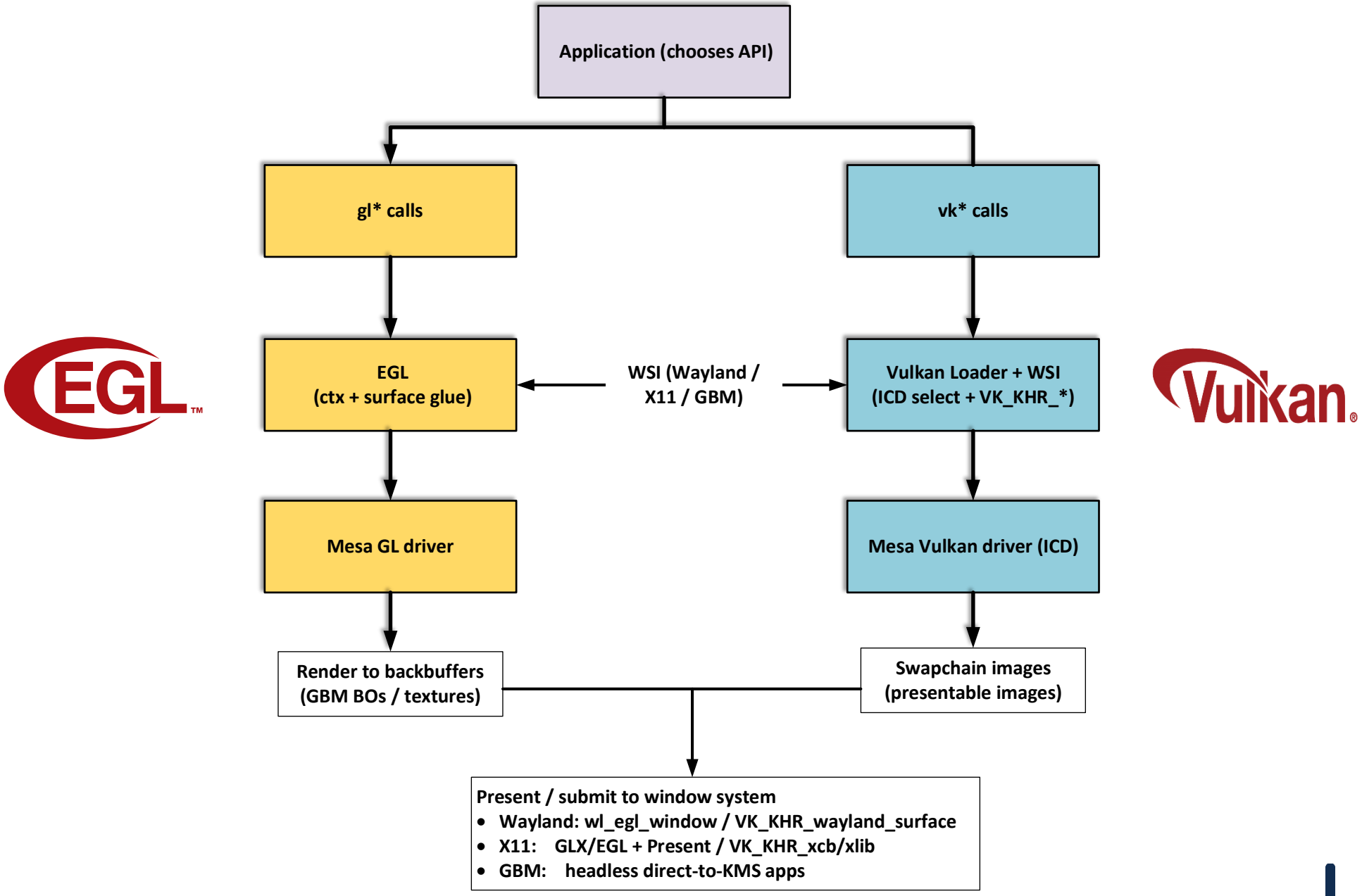
- The compositor decides when your frame shows: it aligns presents to the next refresh and throttles clients via frame callbacks.
- It decides how your frame shows: direct scanout skips GPU blending (lower latency), composite adds a render pass.
- It limits work with damage: only redraws changed regions to keep frame time under budget.

The compositor's job – Cont.,



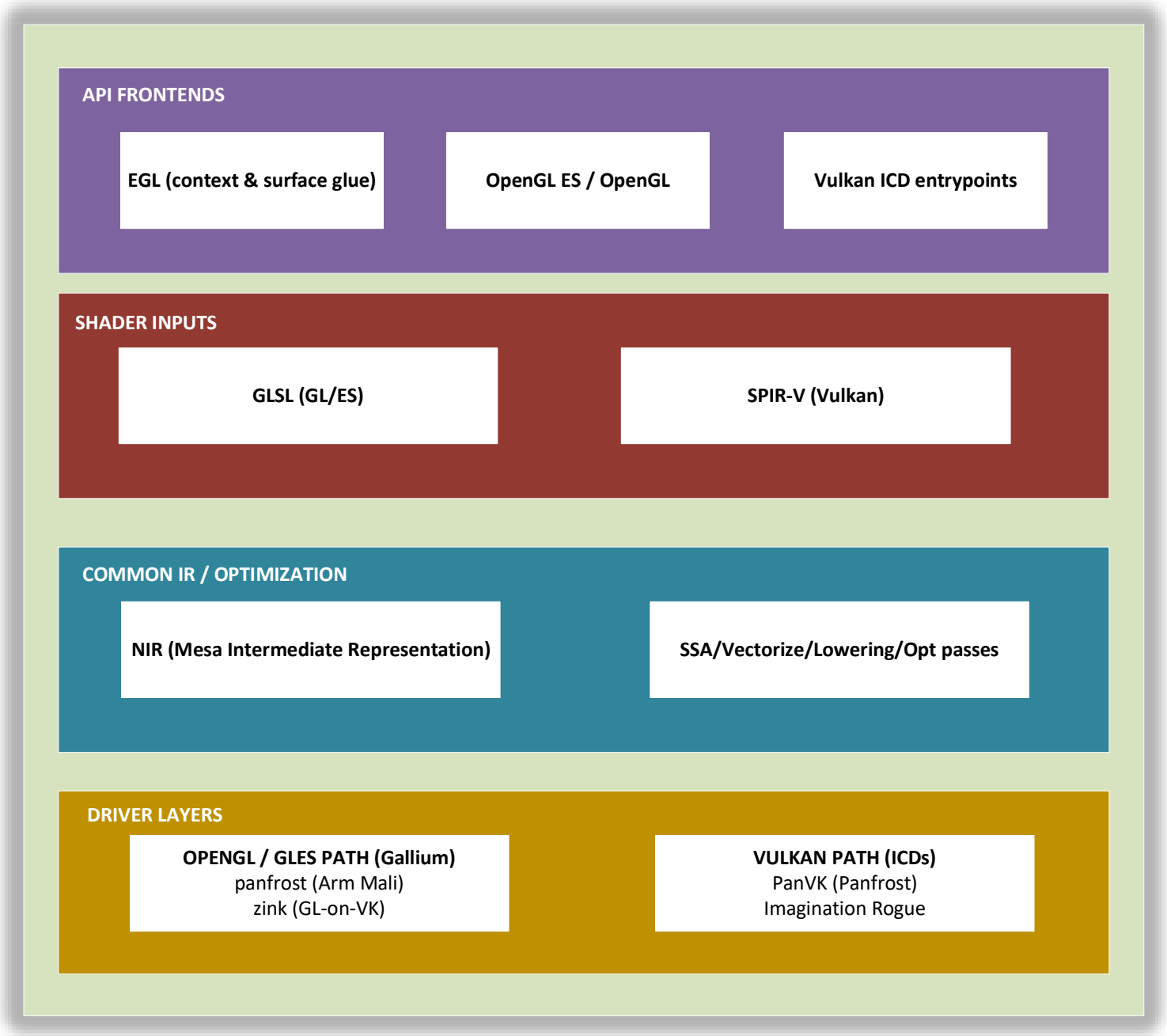
Linumiz

Khronos & Graphics APIs (EGL, OpenGL ES, Vulkan, WSI)



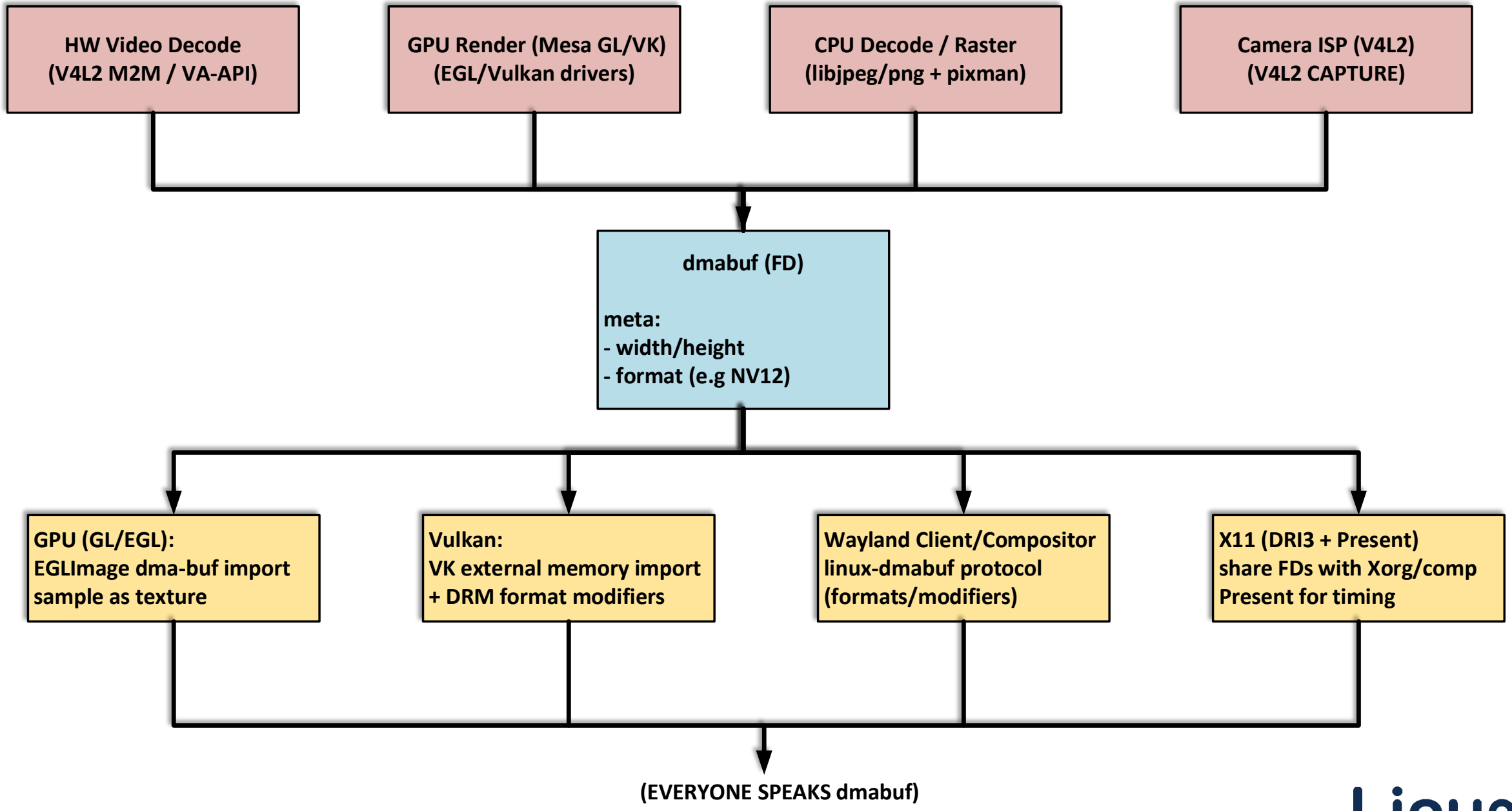
Linumiz

Mesa 3D in a Nutshell



Linumiz

Sharing buffers efficiently (dma-buf, modifiers, fences)

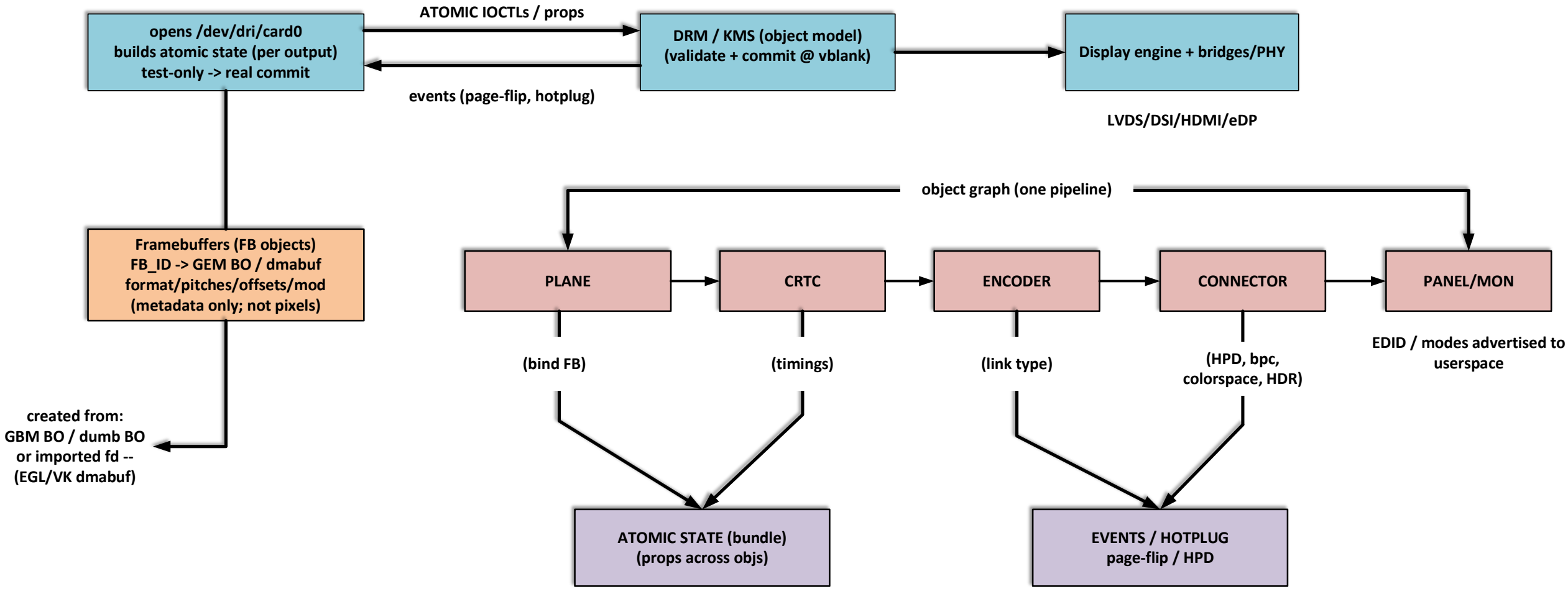


Linumiz

Sharing buffers efficiently (dma-buf, modifiers, fences) – Cont.,

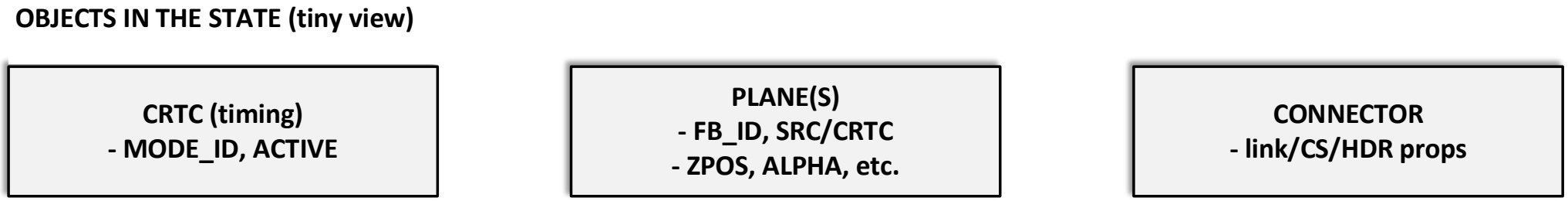
DIRECT SCANOUT	COMPOSITE
Fullscreen or plane-fitting rect	GPU shader (sample dmabufs, YUV->RGB, blend)
Supported format/modifier on plane	Produce final RGB BO (GBM) for scanout
No disallowed transform (rotation/scale beyond caps)	Possible extra VRAM copy/resolve

DRM/KMS Objects



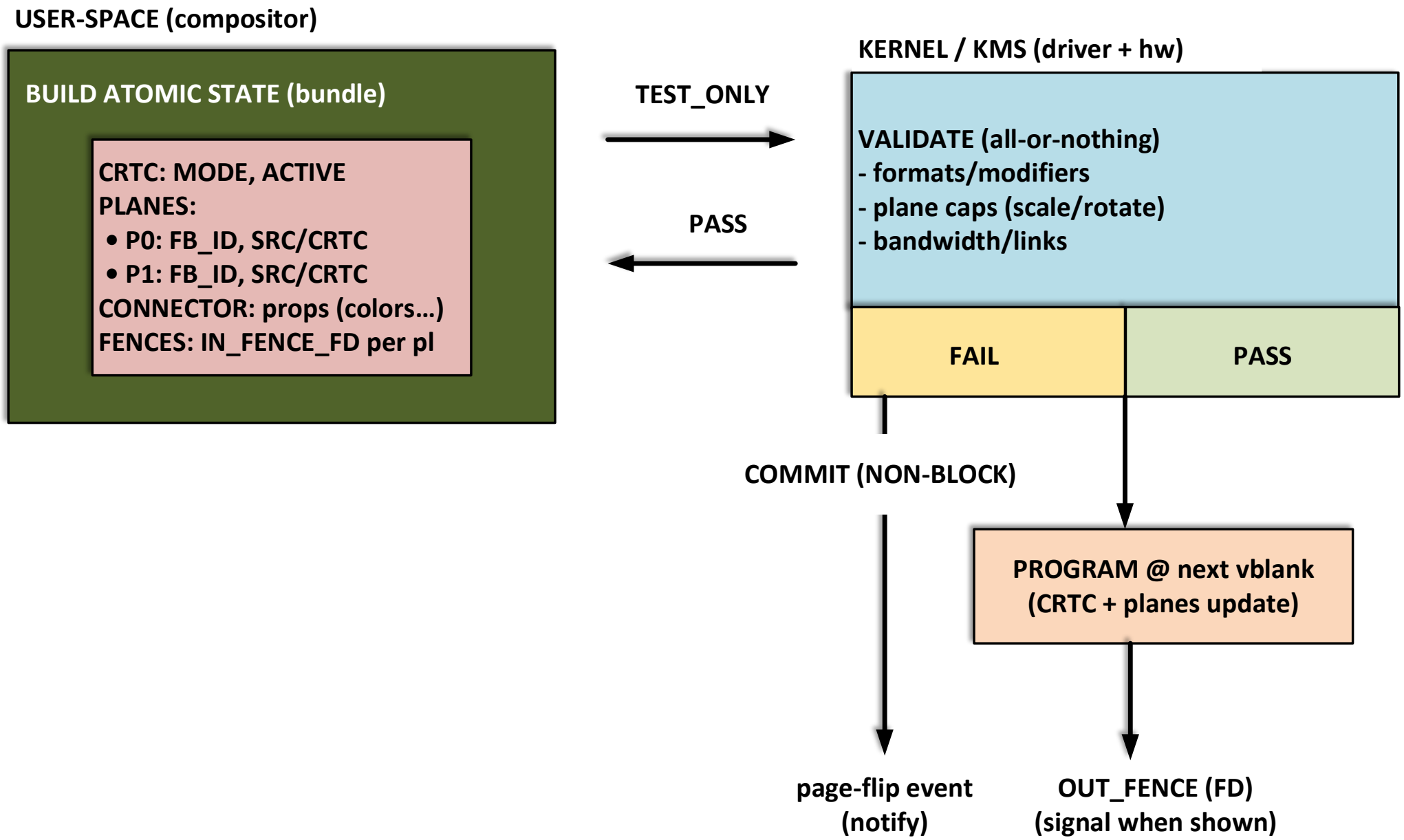
Linumiz

Atomic KMS & Mode setting



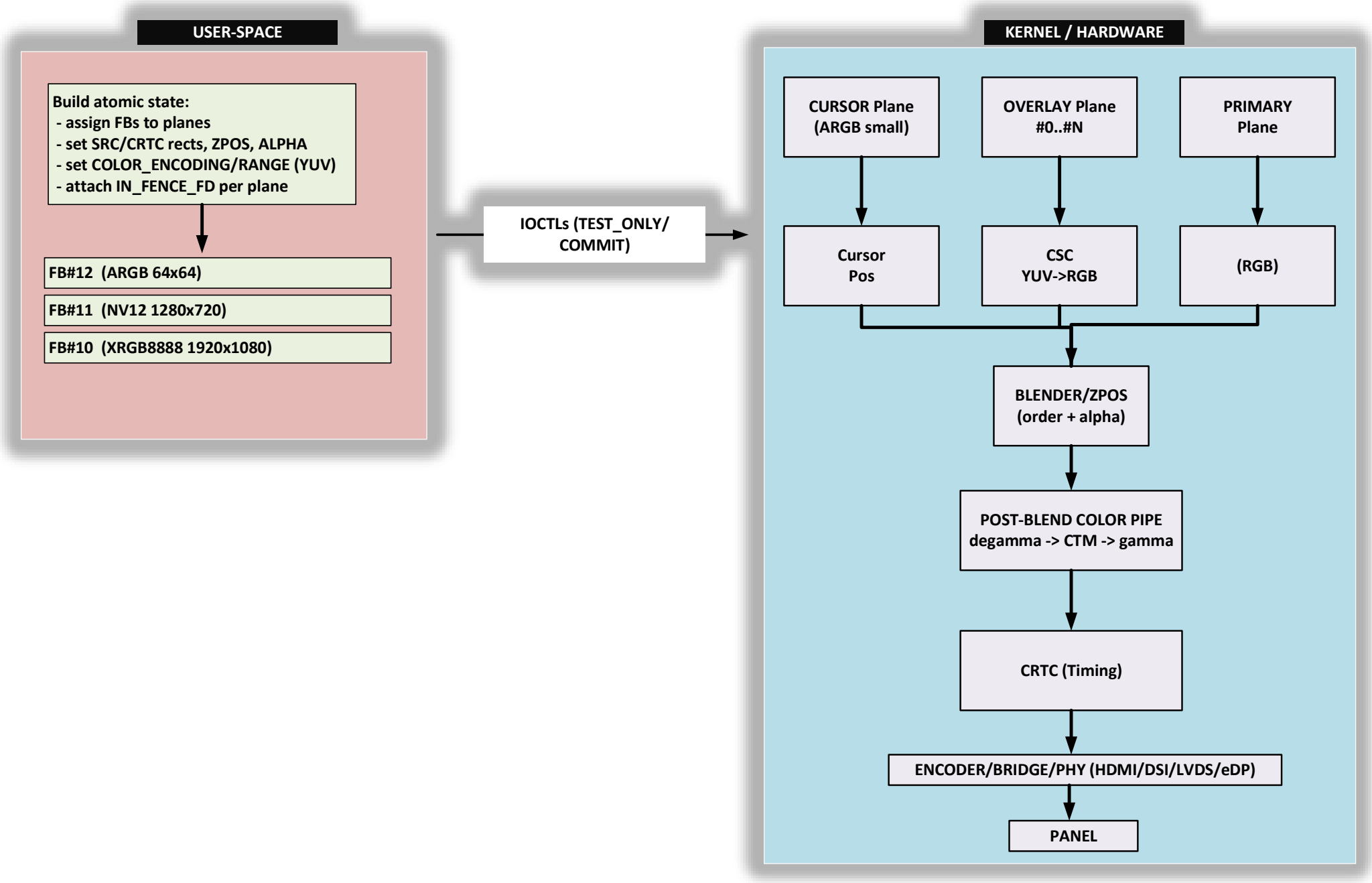
- "Atomic" = one transaction for mode + all planes → predictable, tear-free updates.
- Always TEST_ONLY first (fast fail), then COMMIT non-blocking and pace by vblank events.
- Attach producer fences as IN_FENCE_FD; use OUT_FENCE/page-flip events to schedule the next

Atomic KMS & Mode setting – Cont.,



Linumiz

Display Controller & Planes



Linumiz

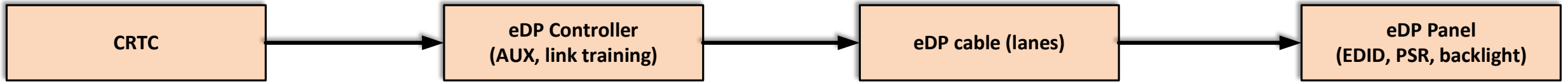
Physical Links (LVDS/DSI/DBI/HDMI/eDP)

[LVDS]

[DSI]

[HDMI]

[eDP]



[DBI]

Linumiz

How DRM Knows the Pipeline

```
soc {
    de: display-engine@01000000 {
        compatible = "allwinner,sun50i-a133-de3";
        reg = <0x01000000 0x100000>;
        clocks = <&ccu CLK_DE>;
        resets = <&rst RST_DE>;
        power-domains = <&pwrc PG_DE>;
        status = "okay";
    };

    tcon_lcd: tcon@02000000 {
        compatible = "allwinner,sun50i-a133-tcon-lcd";
        reg = <0x02000000 0x10000>;
        clocks = <&ccu CLK_TCON_LCD>, <&ccu CLK_TCON_LCD_BUS>;
        clock-names = "tcon-lcd", "ahb";
        resets = <&rst RST_TCON_LCD>;
        status = "okay";

        ports {
            #address-cells = <1>;
            #size-cells = <0>;

            port@0 {
                reg = <0>;
                tcon_lcd_out: endpoint {
                    remote-endpoint = <&dsi_in>;
                };
            };
        };
    };
```

Linumiz

How DRM Knows the Pipeline – Cont.,

```
mipi_dsi: dsi@03000000 {
    compatible = "allwinner,sun50i-a133-mipi-dsi";
    reg = <0x03000000 0x10000>;
    clocks = <&ccu CLK_MIPI_DSI>, <&ccu CLK_MIPI_DSI_BUS>;
    clock-names = "dsi", "ahb";
    resets = <&rst RST_MIPI_DSI>;
    status = "okay";

    ports {
        #address-cells = <1>;
        #size-cells = <0>;

        port@0 {                                // input from TCON
            reg = <0>;
            dsi_in: endpoint {
                remote-endpoint = <&tcon_lcd_out>;
            };
        };

        port@1 {                                // output to the panel
            reg = <1>;
            dsi_out: endpoint {
                remote-endpoint = <&panel_in>;
                data-lanes = <0 1 2 3>;         // 4 data lanes
            };
        };
    };
};
```

Linumiz

How DRM Knows the Pipeline – Cont.,

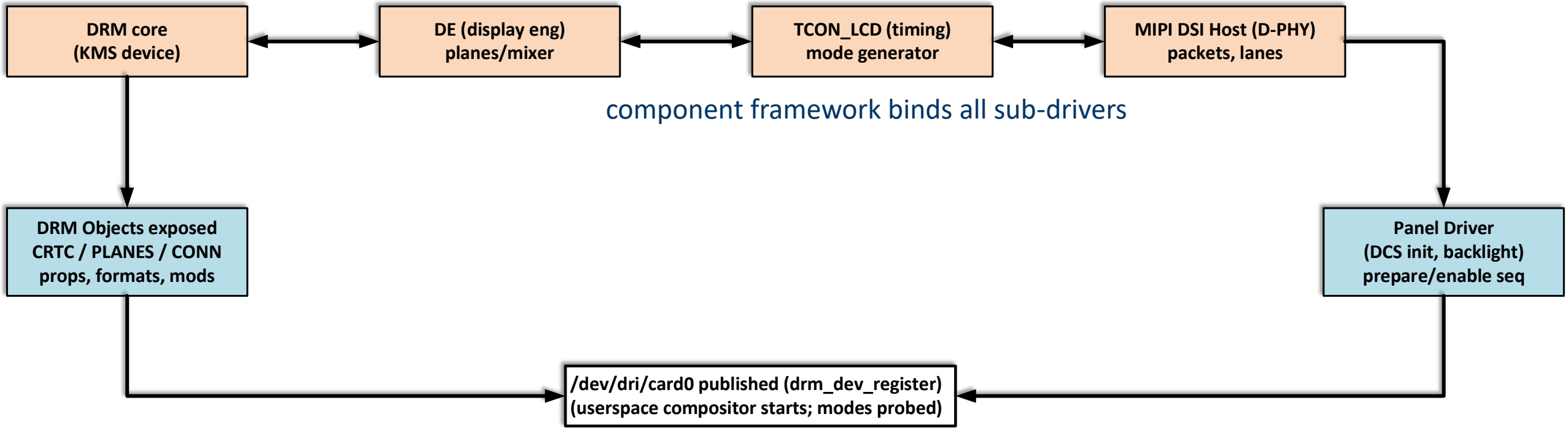
```
        panel@0 {
            compatible = "panel-maker,model-800x1280-dsi";
            reg = <0>;
            power-supply = <&reg_lcd>;
            backlight = <&backlight>;
            reset-gpios = <&pio PG 12 GPIO_ACTIVE_LOW>;
            enable-gpios = <&pio PG 13 GPIO_ACTIVE_HIGH>;
            status = "okay";

            port {
                panel_in: endpoint {
                    remote-endpoint = <&dsi_out>;
                };
            };
        };
    };
};
```

Linumiz

How DRM Knows the Pipeline – Cont.,

What the kernel binds & how DRM “learns” the pipeline

Kernel Components (probing via DT compatibles



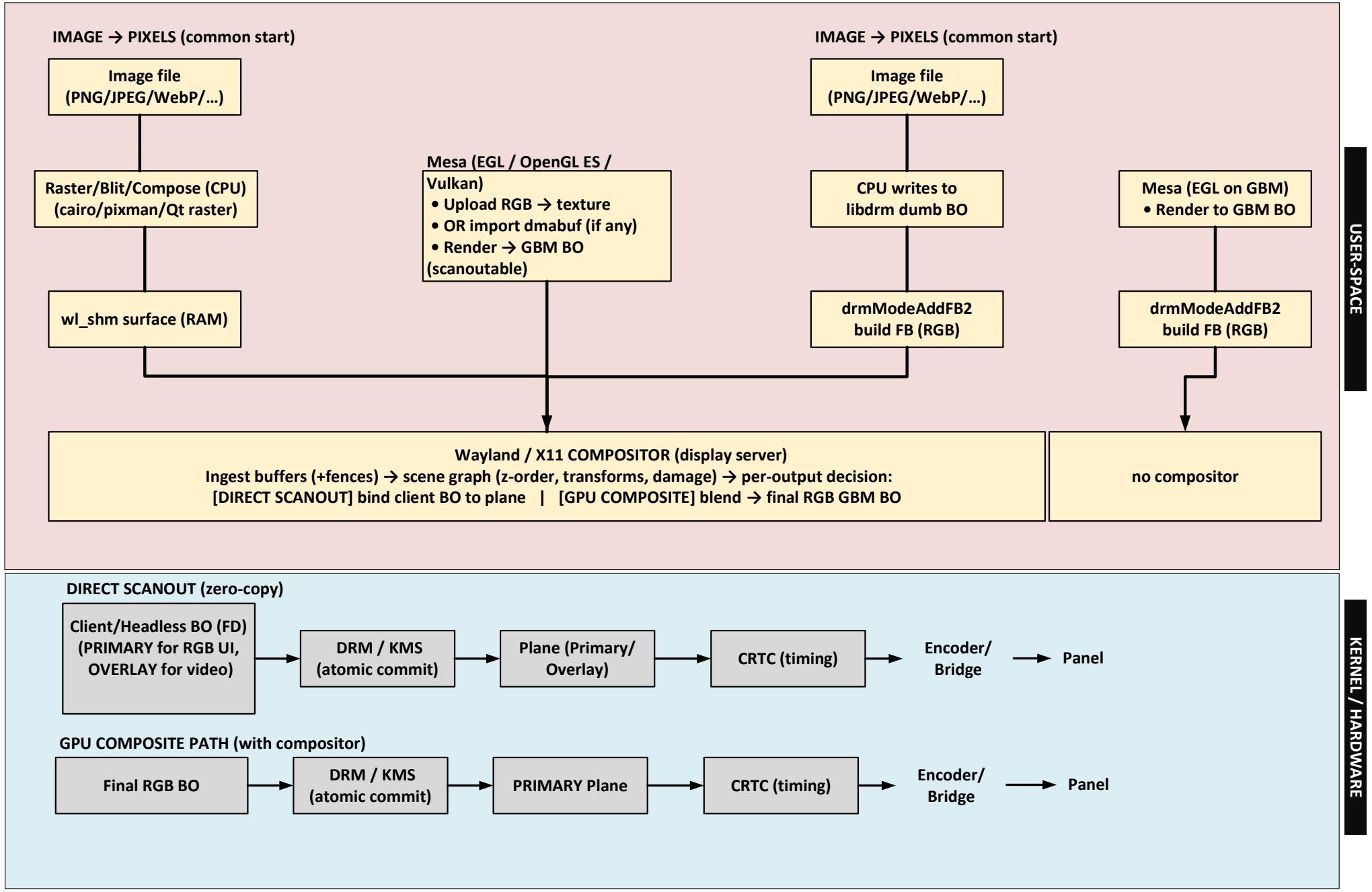
- DT “ports/endpoint” pairs define links: TCON → DSI → Panel.
- component framework waits until DE, TCON, DSI, Panel are all probed, then binds them and registers a single DRM device.
- Userspace then sees: one connector (DSI), one CRTC, a set of planes (from DE).

Linumiz

Memory→Photons

- wl_shm: RAM buffers (CPU path) → compositor must copy/composite.
- GBM BO: scanout-capable GPU buffer (dmabuf-exportable).
- Headless CPU: write into a dumb BO, add FB, atomic commit (no compositor).
- Headless GPU: EGL on GBM → GBM BO → atomic commit (no compositor).
- Direct scanout requires coverage + supported (FORMAT, MODIFIER) + within plane scale/rotate caps.
- Fences: attach producer fences as IN_FENCE_FD; pace via page-flip/out-fences.

Memory→Photons – Cont.,



Linumiz

Userspace (apps, toolkits, compositors, GL/VK)

Tool	What it does / helps
glxinfo -B	OpenGL renderer/driver summary (X11)
eglinfo	EGL configs (Wayland/X11/GBM)
glmark2	Quick GL/ES sanity/perf smoke test
vulkaninfo	Full Vulkan caps (ICD, features, formats)
vkcube	Simple Vulkan present test (WSI sanity)
vkconfig	Enable VK validation layers, configs
kmscube	GBM+EGL → KMS flip (no compositor path)

Linumiz

Kernel / System (KMS, links, timing, tracing)

Tool/Path	What it does / helps
modetest (-c -p -a / -s / -P)	Probe KMS; set modes; bind planes
drm_info	Caps: planes, formats, modifiers, properties
cat debugfs .../dri/0/state	Live DRM state (CRTC/planes/FBs)
dmesg | grep -i drm	Probe logs, atomic failures, HPD/EDID
drm.debug=0x1ff (kernel arg)	Verbose DRM logs

Handy env knobs

ENV VARIABLE	What is it
MESA_DEBUG=1	Mesa verbose (driver decisions)
LIBGL_DEBUG=verbose	GL loader/driver logs
LIBGL_ALWAYS_SOFTWARE=1	Force llvmpipe (debug HW vs SW)
VK_ICD_FILENAMES=…json	Select Vulkan ICD (hw/lavapipe)
VK_LAYER_KHRONOS_validation=1	VK validation (w/ vkconfig)
WAYLAND_DEBUG=1	Protocol trace for one client

Summary

- Think in LAYERS and LANES.
- Share via dma-buf; match (FORMAT, MODIFIER) to plane caps.
- TEST_ONLY early; COMMIT non-blocking; pace by vblank events.
- Aim for direct scanout: UI→PRIMARY, video→OVERLAY.
- If it stutters, find the extra copy and remove it.

Linumiz

GLOSSARY

CORE KMS / DISPLAY OBJECTS	
DRM / KMS	Kernel Display Manager / Mode Setting: API for modes, planes, atomic flips
CRTC	Scanout + timing generator (one per output pipeline)
Plane	Hardware layer scanning a framebuffer: PRIMARY (full), OVERLAY (extra), CURSOR (small ARGB)
FB (Framebuffer)	Descriptor of pixel memory (w/h, format, pitches, offsets, modifier); not the pixels itself
Connector	External link endpoint (HDMI, eDP, DSI, LVDS); exposes modes (EDID or DT)
Encoder	Bridges CRTC to a connector/link type (HDMI/DSI/LVDS/DP)
Atomic Commit	Single transaction applying mode + plane properties at vblank (TEST_ONLY → COMMIT)

GLOSSARY

BUFFERS / SHARING / SYNC	
dma-buf	File-descriptor handle to share buffers across devices/processes (zero copy)
Modifier	Memory layout hint: LINEAR / TILED / COMPRESSED (e.g., AFBC); must match plane capabilities
Fence	Sync primitive: in-fence (wait before scanout), out-fence (signals when flip completes)
PRIME	Cross-GPU buffer sharing via dma-buf export/import
Direct Scanout	Bind client buffer directly to a plane (no composite/copy)

Linumiz

GLOSSARY

PIXEL FORMATS / COLOR	
XRGB8888	Common 32-bit RGB UI format (no alpha in scanout)
NV12 / P010	YUV 4:2:0 (8-bit / 10-bit); typical HW decoder output; great for overlay planes
Colorspace	BT.601 / BT.709 / BT.2020; defines YUV↔RGB conversion
Range	LIMITED vs FULL; affects YUV levels
sRGB	Standard RGB transfer function; linearize before blending for correct results

Linumiz

GLOSSARY

APIs / STACK TERMS	
Wayland	Display server protocol; compositor is the display server; dmabuf-friendly
X11 / DRI3	Legacy window system; DRI3 passes FDs; Present syncs frames
EGL	Glue API: creates GL ES contexts on Wayland/X11/GBM surfaces
OpenGL ES	Graphics API widely used for UIs and apps (via Mesa)
Vulkan + WSI	Low-level graphics API; WSI integrates with Wayland/X11/GBM
GBM	Userspace buffer manager for scanout-capable BOs (works with KMS)
Mesa	Open-source userspace GL/VK driver stack (Gallium/ICDs; llvmpipe/lavapipe as CPU fallbacks)

GLOSSARY

LINKS / PANELS	
HDMI / eDP	External/embedded DP links; EDID/HPD; audio (HDMI); PSR (eDP)
DSI (MIPI)	Packetized mobile link (1–4 lanes); command vs video mode
LVDS	Fixed-timing parallel link; common in industrial panels
DBI (SPI/8080)	Tiny displays; command/RAM write; not true scanout from KMS
EDID / HPD	Sink capabilities blob / hot-plug detect
TCON	Timing controller generating pixel clock/syncs

GLOSSARY

COMPOSITOR / RENDERING	
Scene Graph	Compositor’s tree of surfaces (z-order, transforms, damage)
Damage	Dirty rectangles; enables partial redraw
Overlay	Use hardware plane instead of GPU compositing
Cursor Plane	Dedicated small ARGB plane; very low-latency updates

Thank You

parthiban@linumiz.com

Linumiz