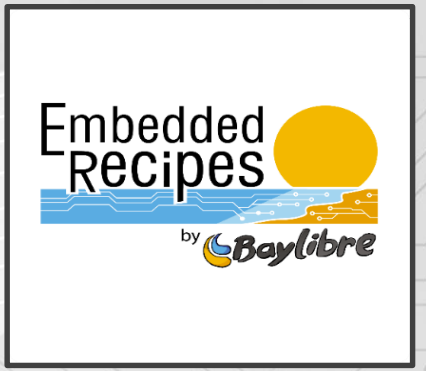


Flicker-free transition of display context across various stages from bootloader to user-space

Date: 2025-05-14

Devarsh Thakkar (Texas Instruments)

About us: TI Processors and Open source

Decades of contribution and collaboration

Ingrained culture to give back to the community

Upstream FIRST!

Focus on long term, sustainable and quality products

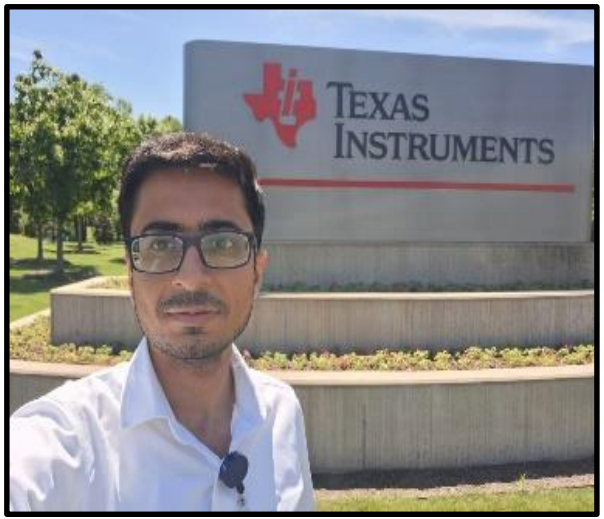
Upstream and opensource ecosystem in device architecture

Upstream FIRST mentality!

Introduction to the Speakers

Devarsh Thakkar, (MGTS) Software Engineering Manager at Texas Instruments

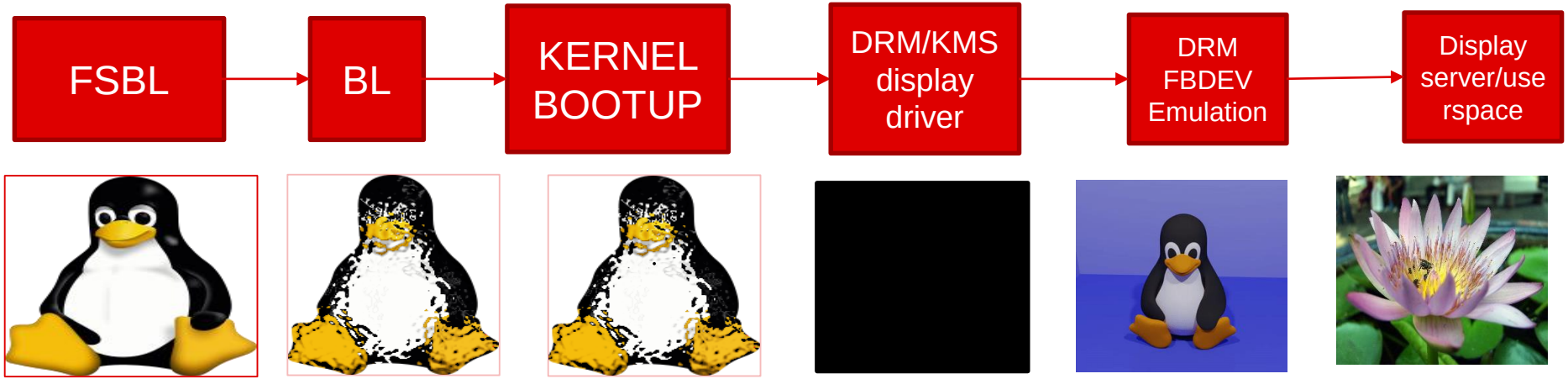
Devarsh Thakkar works as an Embedded Linux developer at Texas Instruments He has ~12 years of experience in software development ranging from open source bootloaders to the Linux kernel, middleware frameworks and applications His expertise lies in Audio/Video related multimedia frameworks, Linux media subsystems, Linux device drivers and applications He has made contributions to open source projects such as U – boot, Linux Kernel and Gstreamer and also presented in international conferences.



Overview

- Problem statement
  - Display screen flickers multiple times while system is booting
- Goals (Mainly the use-cases to be supported)
  - Preserve bootloader splash with smooth transition to intermediate animation and to GUI
  - Preserve bootloader splash with smooth transition to GUI
  - Preserve boot animation with smooth transition to GUI
- Solution
  - Preserve IP, power and clock-domain state
    - Across bootloader stages
    - From bootloader to Kernel
    - During kernel bootup
    - During kernel to OS transition
- It’s a generic problem
  - Audio
  - Boot KPIs
- Current Status
- Upstream Gaps

Problem Statement



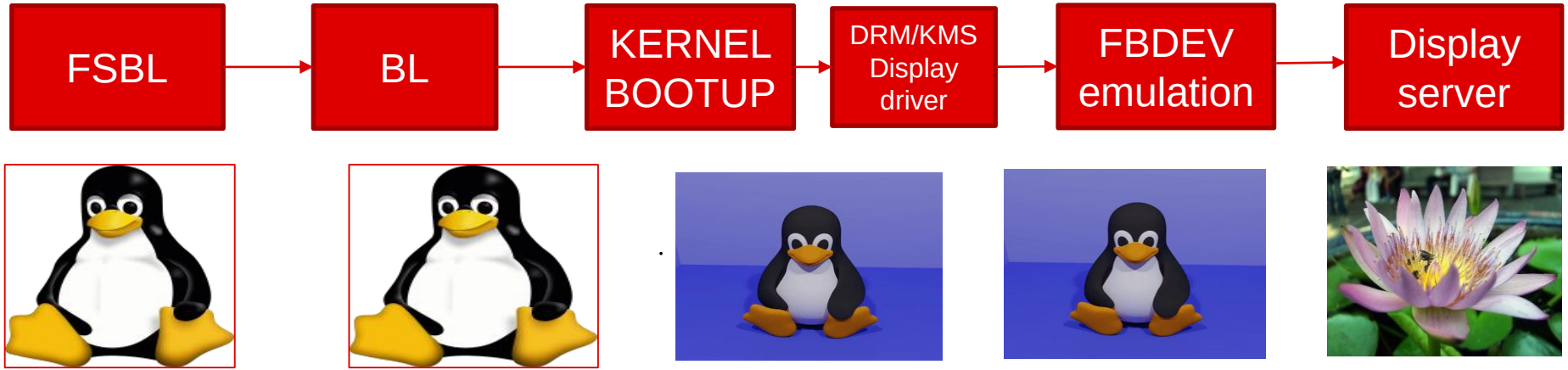
Problems

- **MEMORY:** Display Framebuffer Memory
  - At the next stage,
    - Display framebuffer memory gets reset/re-used by some other driver
  - New FB memory allocated in each phase
- **Video related IP STATE:** Display controller, display bridges, GPIOs, PWMs
  - Power and clock domains get toggled
  - IPs can get soft reset on each stage when corresponding drivers get probed

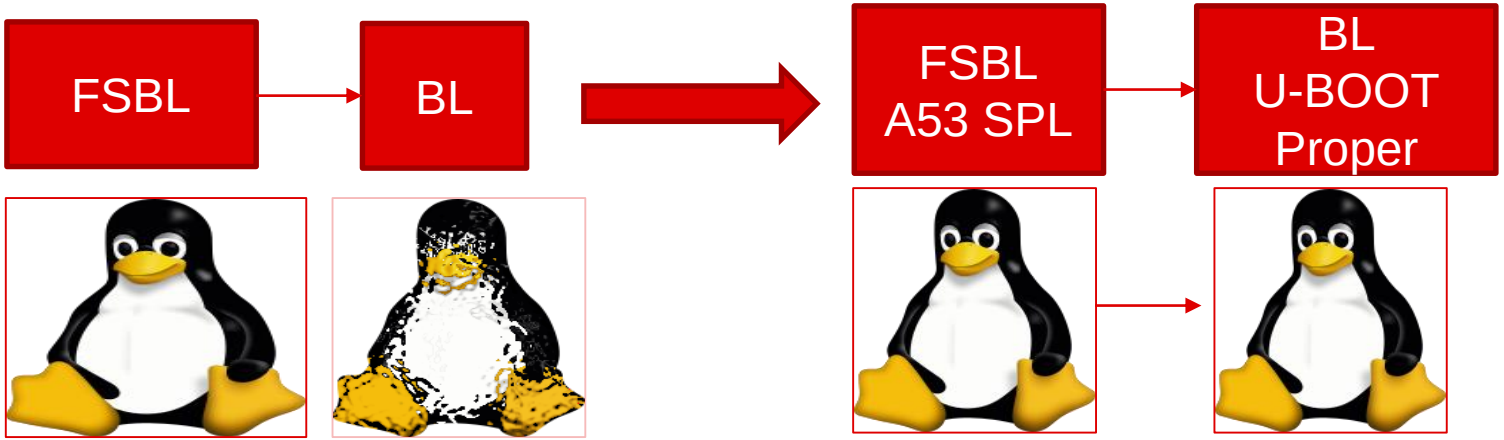
Goals

- Enable smooth transition of display contexts while device is booting up
  - **UC #1:** Preserve boot splash screen with smooth transition to intermediate display context e.g. another logo, animation context (psplash) until OS boots up and have smooth transition to OS GUI
  - **UC #2:** Preserve boot splash screen setup by early bootloader until OS boots up and have smooth transition to OS GUI
  - **UC #3:** Preserve boot animation until OS boots up and have smooth transition to OS GUI
    - While at it, minimize memory foot-print and reduce kernel bootup time so display comes up fast.
    - Taking example of TI's K3 DSS with OLDI based display as an example:

UC #1 – With psplash animation



Across bootloader boot-stages



Across bootloader boot-stages

- Pass video bloblist across each bootloader stage
  - CONFIG_SPL_BLOBLIST=y
  - CONFIG_BLOBLIST=y
- Leave the power domain on
  - Pass DM_FLAG_LEAVE_PD_ON
  - Remove method does not power off display power domains
- More details covered at :
  - Early_display_using_uboot_presentation
  - Early_display_using_Uboot_Video

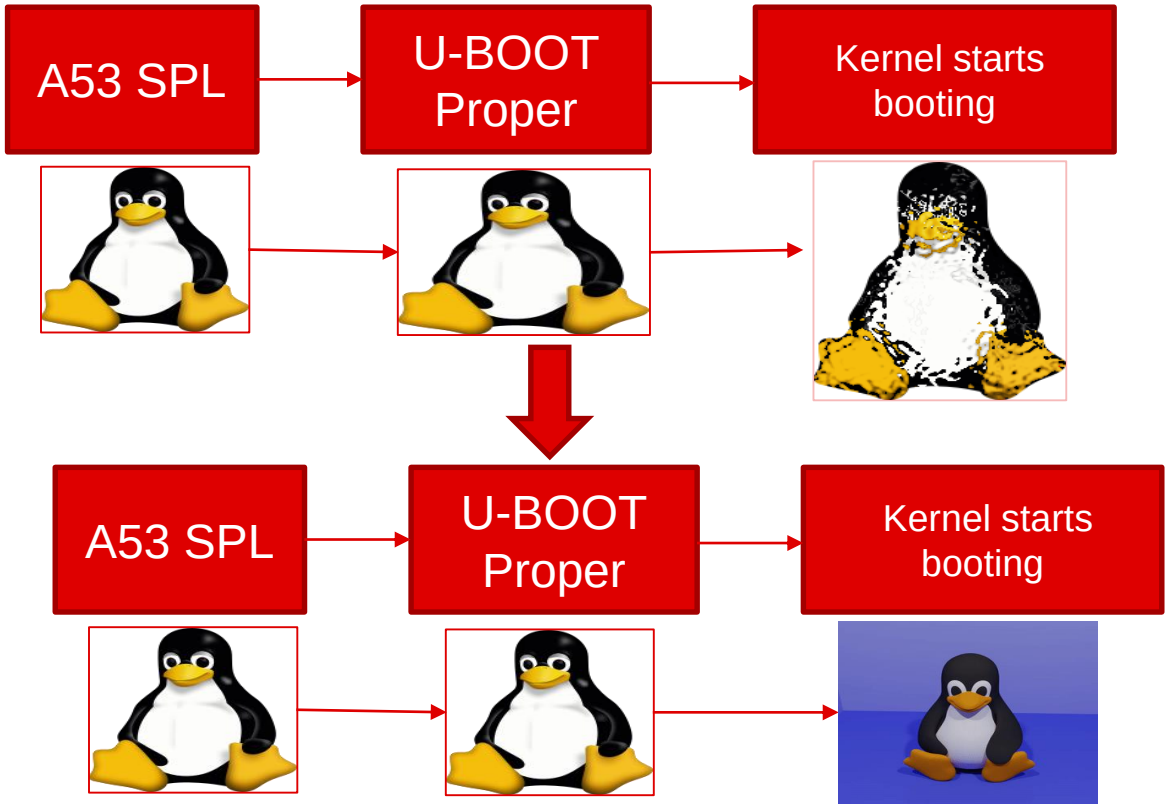
Video-uclass.c

```
static int video_post_probe(struct udevice *dev)
if (xpl_phase() == PHASE_SPL && CONFIG_IS_ENABLED(BLOBLIST)) {
        struct video_handoff *ho;
        ho = bloblist_add(BLOBLISTT_U_BOOT_VIDEO, .);
        ho->fb = gd->video_bottom;
        /* Fill aligned size here as calculated in video_reserve() */
        ho->size = gd->video_top - gd->video_bottom;
        ho->xsize = priv->xsize;
        ho->ysize = priv->ysize;
        ho->line_length = priv->line_length;
        ho->bpix = priv->bpix;
        ho->format = priv->format;
}
```

tidss-drv.c

```
U_BOOT_DRIVER(tidss_drv) = {
....
....
#if CONFIG_IS_ENABLED(VIDEO_REMOVE)
    .flags = DM_FLAG_OS_PREPARE,
#else
    .flags = DM_FLAG_OS_PREPARE | DM_FLAG_LEAVE_PD_ON,
#endif
};
```

While kernel is booting

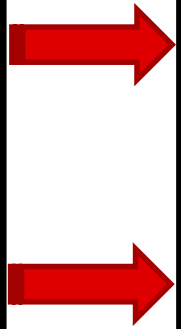


While Kernel is booting

- Reserve framebuffer memory area as a reserved-memory node
- Simple-framebuffer
  - Helps display animation before display driver is probed
  - Preserves clock and power domains while kernel is booting and display driver is not probed
    - By default genpd provider driver marks all domains as unused during bootup, so kernel has no knowledge about already powered-on peripherals.
      - These domains can get disabled at any time.
    - Driver core attaches the power-domain mentioned in DT
    - Supports attaching to multiple power-domains too
  - Re-uses framebuffer memory set up by bootloader
  - Enabled by CONFIG_FB_SIMPLE=y and it creates /dev/fb0
  - Psplash or fbdev console can use this as an early framebuffer

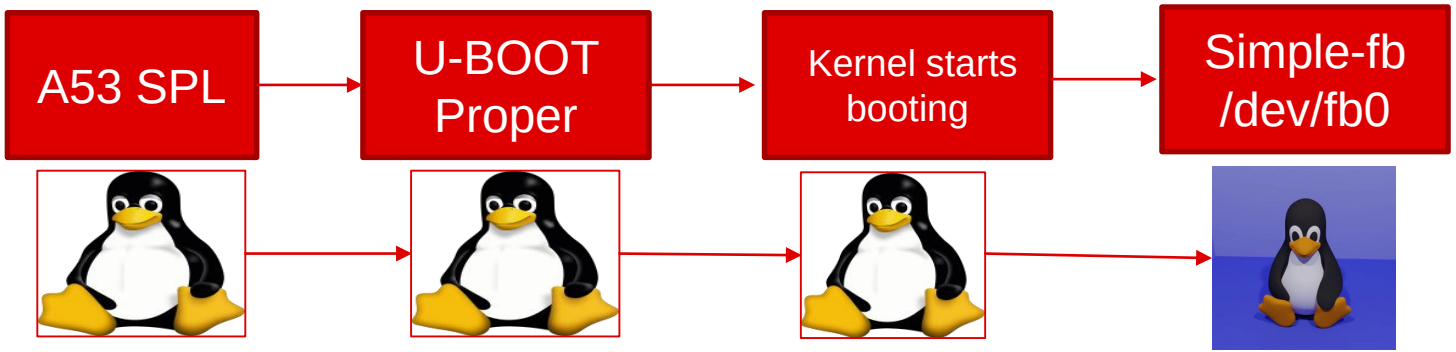
From bootloader to kernel – Using simple-fb

```
// Enable simple-framebuffer and reserve FB area using fdt APIs
int ft_board_setup(void *blob, struct bd_info *bd)
{
fdt_simplefb_enable_and_mem_rsv(blob);

fdt_add_fb_mem_rsv(blob);

}};
}};
```



```
video-uclass.c
// base device-tree with simple-framebuffer node (Given by user)
 chosen {
        framebuffer0: framebuffer@0 {
                compatible = "simple-framebuffer";
                power-domains = <&k3_pds 186 TI_SCI_PD_EXCLUSIVE>;
                clocks = <&k3_clks 186 6>,
                        <&dss0_vp1_clk>,
                        <&k3_clks 186 2>;
                display = <&dss0>;
                status = "disabled"; // U-boot sets this to enabled using fdt API and adds FB region
        };

  //Automatically created by U-boot using  fdt API
        framebuffer: framebuffer@ff700000 {
           reg = <0x00 0xff700000 0x00 0x008ca000>;
           no-map;
        }
 };
```

While Kernel is booting



Avoiding power-off on probe deferral

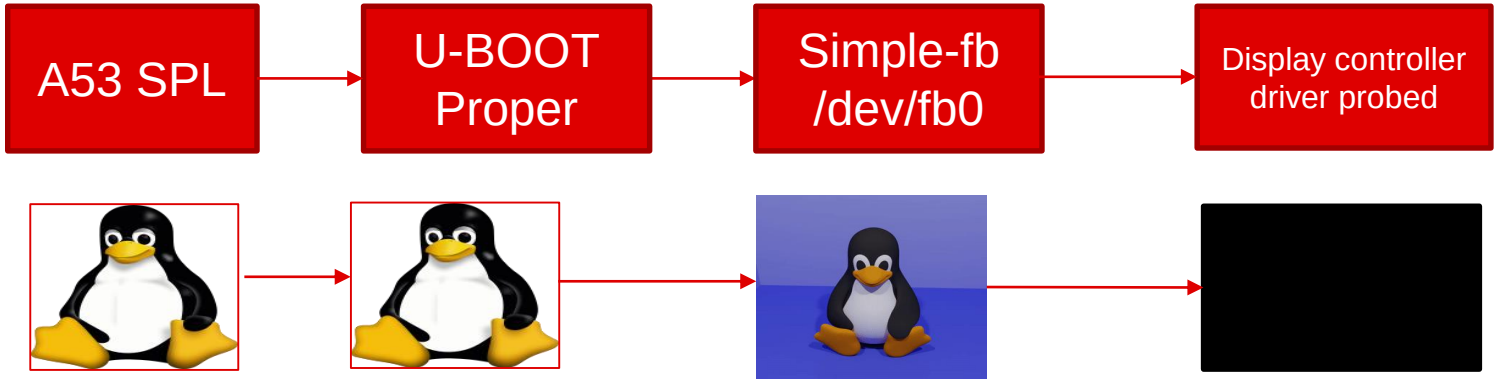
- What if we just reserve the bootloader splash memory ?
- Kernel boots up but has no knowledge of already powered-on peripherals
- Although display is in powered-on state it will mark it’s power domain as unused
- While probing the driver framework automatically takes reference to device-power domain (powers-on) and on probe exit it powers off the device if probe failed (deferred probe)
- Extra reference taken by simple-fb helps here

```
static int platform_probe(struct device
*_dev)
{
        if (drv->probe) {
                ret =
dev_pm_domain_attach(_dev, true);
                 ret = drv->probe(dev);
                 if (ret)
                 dev_pm_domain_detach(_dev,
true);
        }
}
```

What about simple-drm?

- Simple-drm
  - Similar to simplefb this preserves clock and power domains by taking an extra reference.
  - Simple-drm takes ownership of the bootloader buffer and sets it up to be used as drm client
    - Any drm based application can also use this to render an animation using /dev/dri/cardX
  - Simple-drm also supports fbdev emulation which will use a separate buffer to map it to /dev/fbX
    - It allocates another buffer which is used as back buffer for fbdev emulation and bootloader framebuffer contents need to be copied to this buffer to preserve the splash screen as otherwise user will see a black screen.
- Disadvantages :
  - Higher memory footprint as compared to simplefb if using /dev/fbX with fbdev emulation.

After the display controller probes



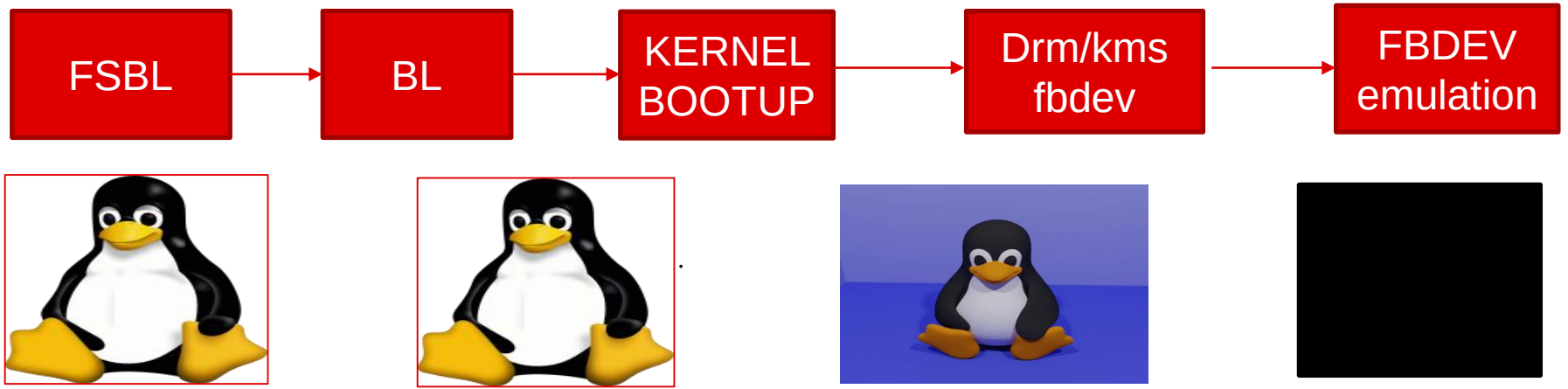
Display related IPs may get reset during probe

Avoid reset for video IPs

- Detect if display active on probe
- Keep power-domains and active clocks enabled
  - So that on probe-deferral driver core does not power-off
- Until new modeset request comes up, continue to reuse bootloader splash context without re-initializing display hardware
- On first atomic_commit callback, reset the display controller so that display server can cleanly take control of display IP
- Same thing needs to be done for all IPs in chain
- More details at :
  - [PATCH 0/2] drm/tidss: Delay reset if we have a splash-screen - Tomi Valkeinen

```
// Check if display controller is active
dispc_is_idle(struct dispc_device *dispc)
{
        return REG_GET(dispc, DSS_SYSSTATUS, 9, 9);
}
//Check which all CRTC's are active
 bool enabled = VP_REG_GET(dispc, vp_idx,
                                        DISPC_VP_CONTROL, 0, 0);
                dispc->tidss->boot_enabled_vp_mask |=
BIT(vp_idx);
// * Keep the CRTC clk enabled */
                        ret =
clk_prepare_enable(dispc->vp_clk[vp_idx]);
// *Keep the display controller GENPD on*
if (dispc->tidss->boot_enabled_vp_mask) {
                dev_dbg(dev,
                        "Bootloader splash-screen detected, leaving
DSS active.\n");
                pm_runtime_get_noresume(dev);
```

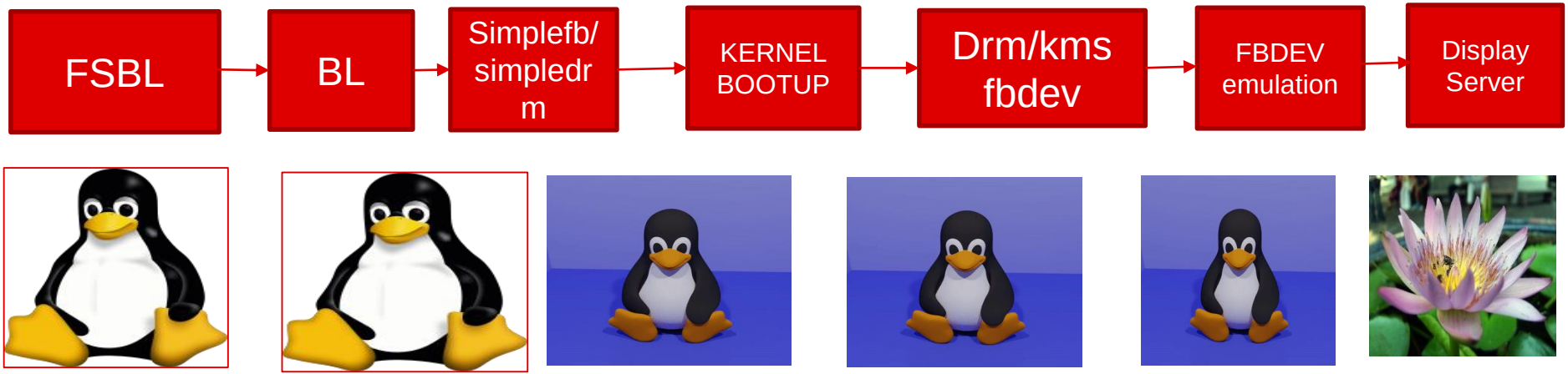
After fbdev emulation starts



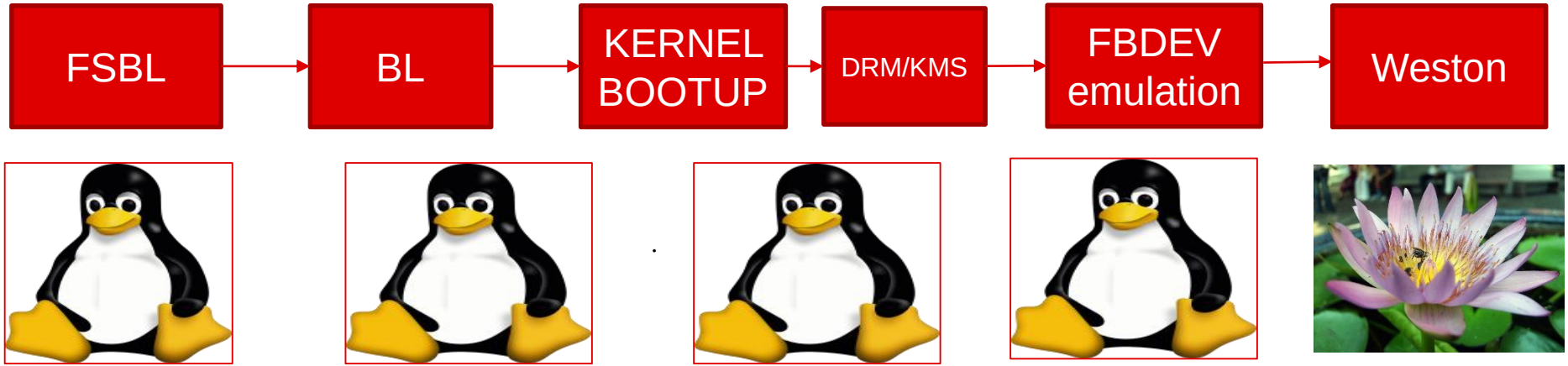
After fbdev emulation starts

- FBDEV emulation uses separate buffer for the atomic commit
- Display driver can implement the function to copy simplefb contents of bootloader splash to fbdev buffer memory
  - Copy is required since there is no direct way to map the bootloader memory to a drm buffer struct and use it to pass to fbdev directly
  - Export function pointer to copy simplefb contents to fbdev buffer
- FBDEV emulation driver calls this function during probe and before the modeset happens.
- Splash buffer contents are displayed seamlessly until app updates new buffer
- Other option is to disable fbdev emulation altogether
- Can fbdev emulation directly use the bootloader supplied buffer ??

After fbdev emulation starts



Goals #2 – Preserve bootloader splash



Power-domain toggle problem – genpd sync state

- Have to keep power and clock domains on without using simple-fb
- Platform power domain driver marks power domains as active based on if device is already active during bootup
- GenPD framework adds a special flag called stay_on domains to those domains and keeps them enabled until sync_state call happens
- ti_sci_pd_boot_state_rfc
- [PATCH 00/11] pmdomain: Add generic ->sync_state() support to genpd - Ulf Hansson

```
//GENPD provider driver
while (!of_parse_phandle_with_args(np, "power-domains",
                                   "#power-domain-cells",
                                   index, &args)) {
        is_on = ti_sci_pm_pd_is_on(pd_provider, pd->idx);
        pm_genpd_init(&pd->pd, NULL, !is_on);
}
int pm_genpd_init(struct generic_pm_domain *genpd,
        struct dev_power_governor *gov, bool is_off)
{
        genpd->status = is_off ? GENPD_STATE_OFF :
GENPD_STATE_ON;
        genpd->stay_on = genpd_may_stay_on(!is_off);
}
```

Power-domain toggle problem – genpd sync state

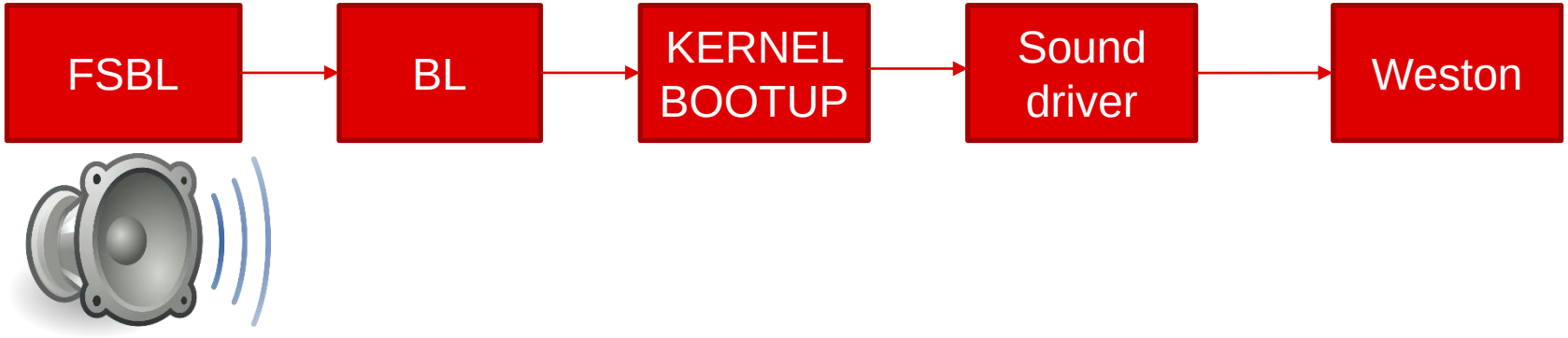
- Sync state gets triggered when all PD consumer drivers get probed successfully
- This can be triggered via sysfs too manually after system is up
  - *echo –n 1 > /sys/devices/platform/bus@f0000/44043000.system-controller/44043000.system-controller:power-controller/state_synced*
- In case sync state callback is not enabled, there is a idle timeout after which it gets triggered automatically and drops reference for all probed drivers – This should serve as backup
- More details at :Re: [PATCH 00/11] pmdomain: Add generic ->sync_state() support to genpd - Ulf Hansson

```
void of_genpd_sync_state(struct device *dev)
{
        list_for_each_entry(genpd, &gpd_list,
gpd_list_node) {
            if (genpd->provider == &np-
>fwnode) {
        genpd_lock(genpd);
        genpd->stay_on=false;
        genpd_power_off(genpd, false, 0);
        genpd_unlock(genpd); }
        }
}
```

Challenges with genpd_sync_state

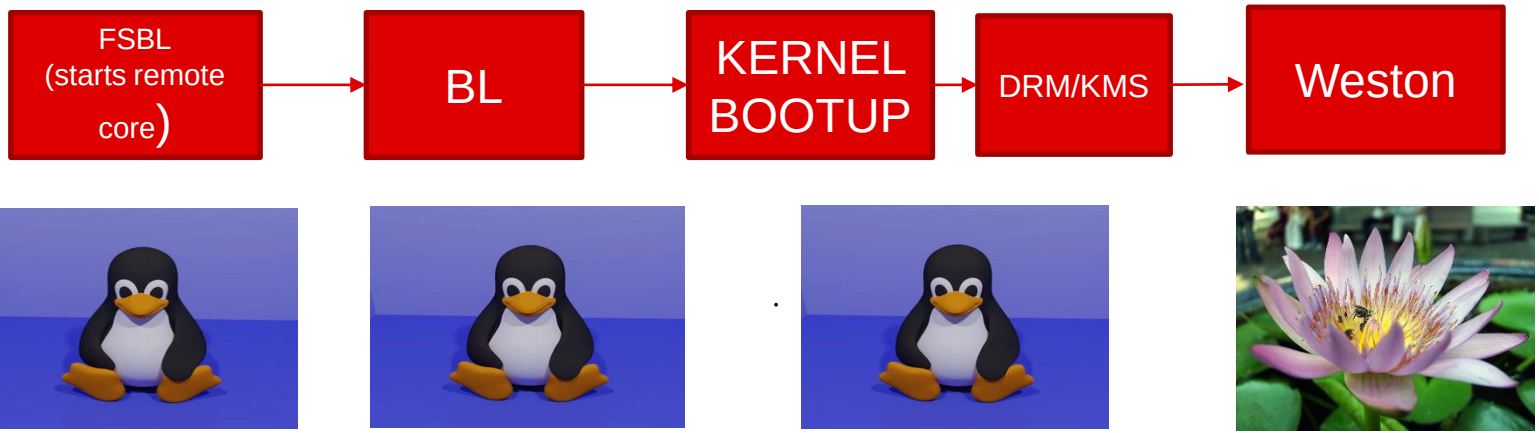
- At boot-time the genpd provider driver needs to query each of the peripherals to know their power-state one-by-one
- The power-domains are generally managed by a separate entity possibly on a remote core running a separate firmware and linux needs to query via IPC
  - This adds to increase in boot-time (in several milliseconds)
- POSSIBLE SOLUTION 1:
  - [TI SPECIFIC] Get combined status of all power-domains using a new TISCI API
  - [Generic] Get combined status of all power-domains using a new SCMI API
- POSSIBLE SOLUTION 2:
  - Separate dt-flag to specify which of the peripherals need to be queried ??:
    - For e.g. we have always-enabled flag, similarly there could be boot-enabled flag.

Generic problem….

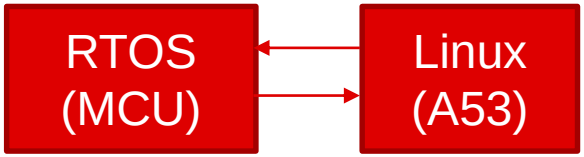


- Early audio tone set by bootloader, that need to be continued to play until tone completes while system is booting up
  - Power, clock and IP states for all devices in pipeline need to be preserved
  - Genpd sync state helps here too….
- Boot time optimizations
  - Skip initialization of already initialized IPs until it is really required to re-initialize saves boot times.

Goals #3 – With remote core animation



Goals #3 – With remote core animation



- MCU core driving a boot animation context and controlling the display
  - MCU cores generally boot earlier in the system and have interrupts support so can render animations
- We protect the MCU memory context using reserved memory region similar to what was done earlier
- Disable the FBDEV emulation
- Smooth hand-off
  - Linux needs to specify MCU core that it is taking control of display
  - MCU core needs to relinquish the display control and do proper cleanup and send an ACK
  - IPC mechanism – No standard API
    - Linux display driver kicks a mailbox which generates an interrupt to remote core and waits for a signal from remote core
    - Remote core does the cleanup and signals to Linux again using a mailbox interrupt.
    - Linux takes control of the display allowing modeset from weston

Open challenges/Gaps

- No direct way to free bootloader splash reserved region and give it back to kernel.

  reserved-memory {

  reg = <0xFF000000 0xC00>

  }

- Small flicker seen when Weston starts up
  - Weston creates a buffer and does a modeset with it without filling the buffer beforehand thus causing a blank screen ???
- Copy needed for preserving splash with FBDEV Emulation enabled
  - In case we just want to preserve bootloader splash until GUI comes up, there is no direct way to map the bootloader buffer into a drm framebuffer struct and pass it to fbdev directly.
  - As a workaround, copying the bootloader splash buffer to fbdev fb before doing atomic commit.
- Genpd sync state API does not support per device sync state calls to disable the stay-on power domain reference as soon as corresponding driver gets probed
  - Some discussion to extend fwdevlink API : Re: [PATCH 00/11] pmdomain: Add generic ->sync_state() support to genpd - Ulf Hansson
- No standard API for remote-core to Linux smooth hand-off

References

- Kernel Recipes 2015 - Introduction to Kernel Power Management - by Kevin Hilman
- Kernel Recipes 2017 - Overview of Generic PM Domains (genpd) - Kevin Hilman
- https://www.youtube.com/watch?v=-uJ_mNUjYpM
- https://www.youtube.com/watch?v=UvFG76qM6co&t=1910s
- [PATCH 00/11] pmdomain: Add generic ->sync_state() support to genpd - Ulf Hansson
- [PATCH 0/2] drm/tidss: Delay reset if we have a splash-screen - Tomi Valkeinen

Credits and Acknowledgement

- Texas Instruments Inc.
- Tomi Valkeinen (IdeasOnBoard oy) for reviewing the slides
- Embedded Recipes 2025

Q&A

- Contact Information:
  - Devarsh Thakkar devarsht@ti.com
  - @devarsht:matrix.org
- Also on IRC @ libera.chat #linux-ti

Learn more about TI products

- https://www.ti.com/linux
- https://www.ti.com/processors
- https://www.ti.com/edgeai

Why choose TI MCUs and processors?

Scalability

Our products offer scalable performance that can adapt and grow as the needs of your customers evolve.

Efficiency

We design products that extend battery life, maximize performance for every watt expended, and unlock the highest levels of system efficiency.

Affordability

We strive to make innovation accessible to all by creating cost-effective products that feature state-of-the-art technology and package designs.

Availability

Our investment in internal manufacturing capacity provides greater assurance of supply, supporting your growth for decades to come.