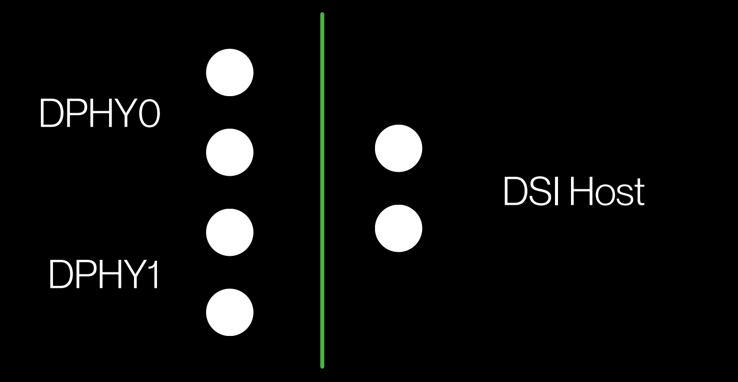
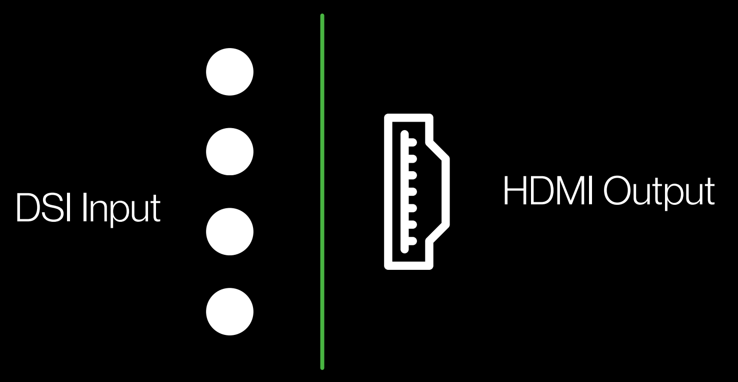
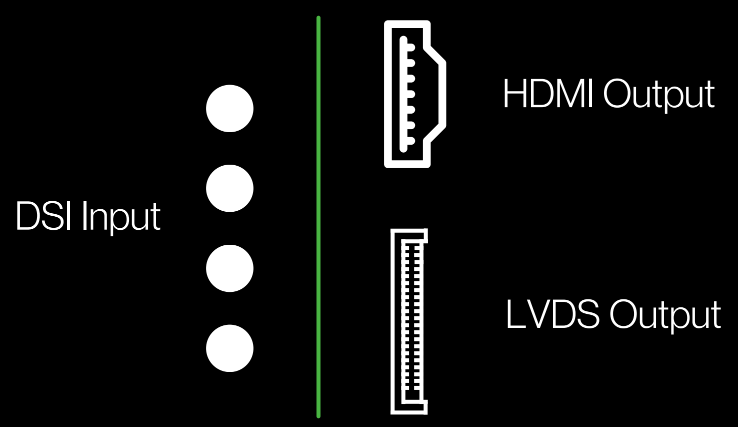
Standardising Linux DRM driver implementations by interfacing DRM Bridge as a single API

Linux Plumbers Conference 2023, Richmond, VA, Nov 14 - Jagan Teki <jagan@amarulasolutions.com>







edgeble

amarula solutions

Embedded Linux Engineer

Founder, Upstream Linux Specialist

Technical Conference Speaker

U-Boot

Build Root Making Embedded Linux Easy

yocto PROJECT

Contributions (patches)	1000+	350+	50+
Maintainer (Subsystems)	SPI/SPI Flash Allwinner sunXi SoC	MIPI DSI Bridge/Panel drivers NXP PF8X00 PMIC driver	Hardware platforms based on i.MX6/8, Rockchip, Allwinner

Jagan Teki

Runtime display switch implementation in a Linux DRM bridge subsystem: X.Org Developers Conference 2022

https://indico.freedesktop.org/event/2/contributions/76/attachments/73/114/XDC2022%20-%20DRM%20Bridge%20Switch.pdf

Supporting Complex MIPI DSI Bridges in a Linux System

- Linux Automotive Summit, Japan 2021
- Linaro Connect Virtual, 2021

https://static.linaro.org/connect/lvc21f/presentations/LVC21F-223.pdf

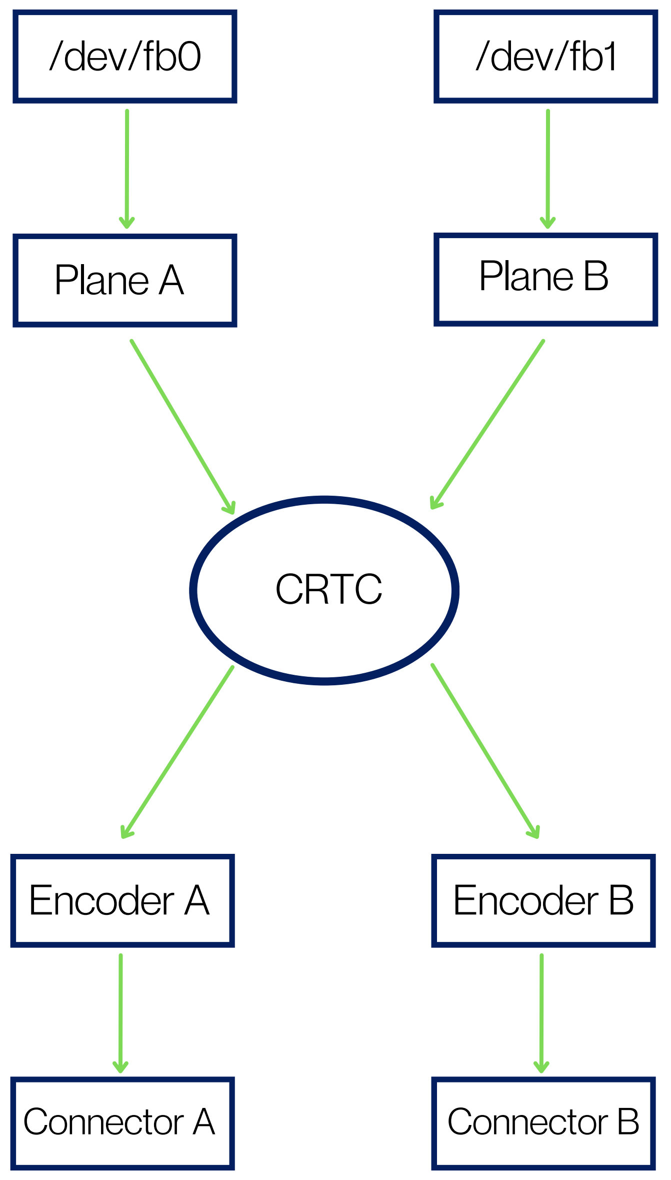
Demystifying Linux MIPI DSI Subsystem

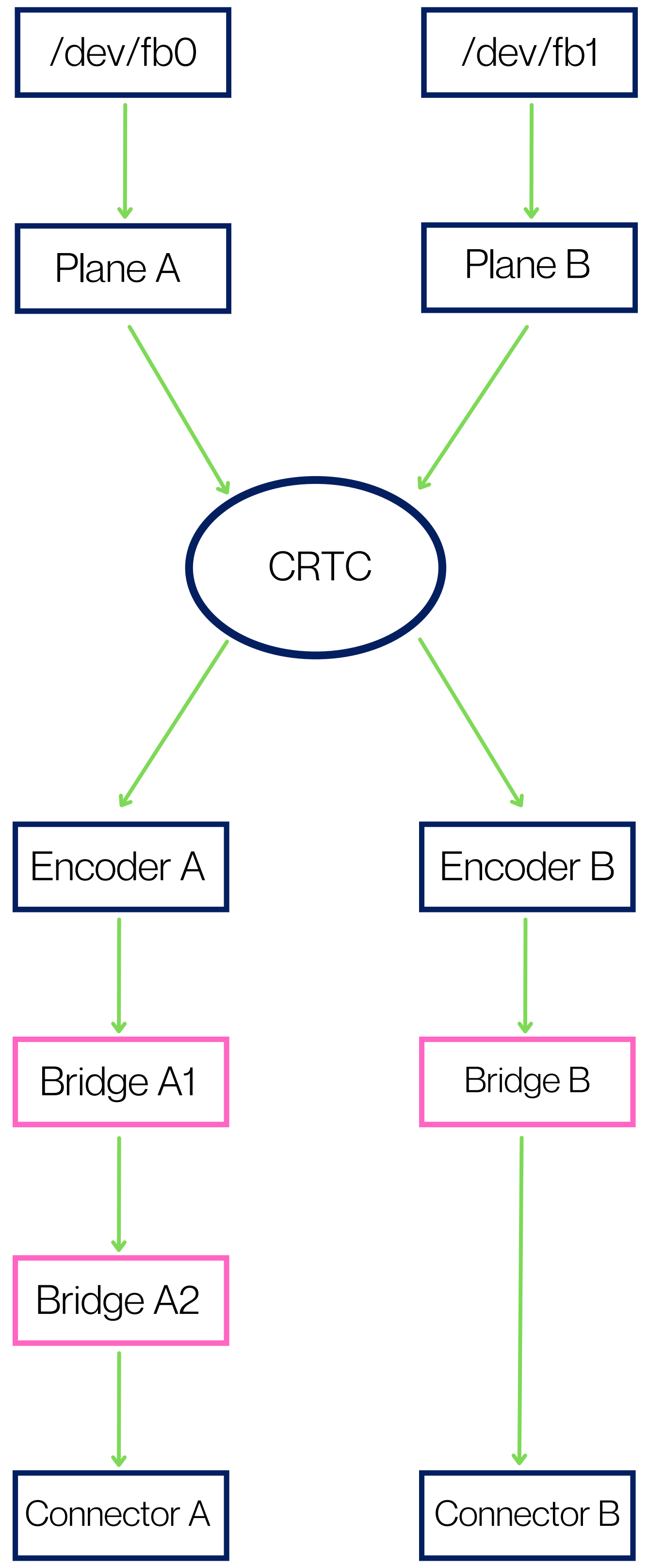
- 2019 ELC North America
- 2019 ELC Europe

https://elinux.org/images/7/73/Jagan_Teki_-_Demystifying_Linux_MIPI-DSI_Subsystem.pdf

My Previous Talks

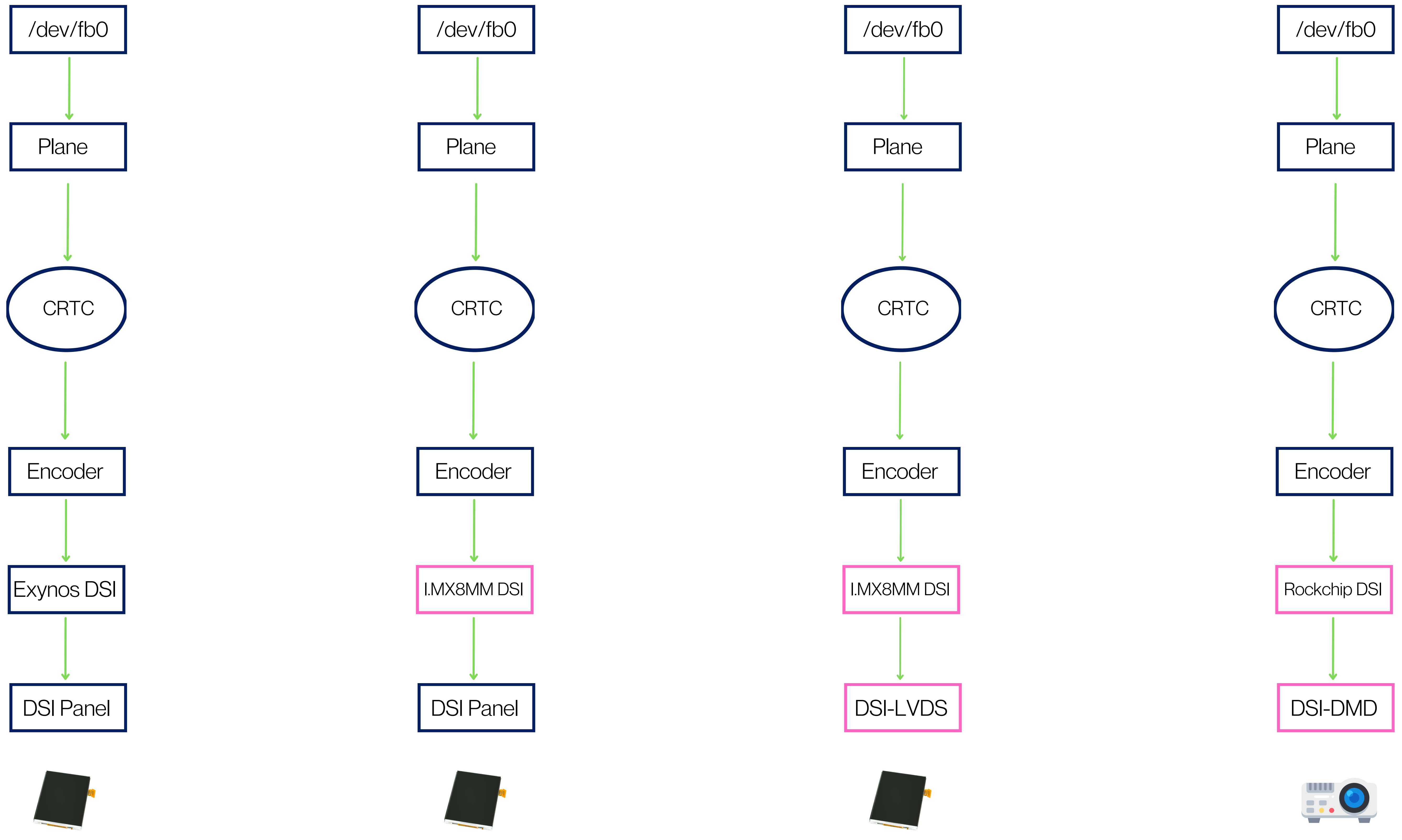
Linux DRM Bridge(In-detail)





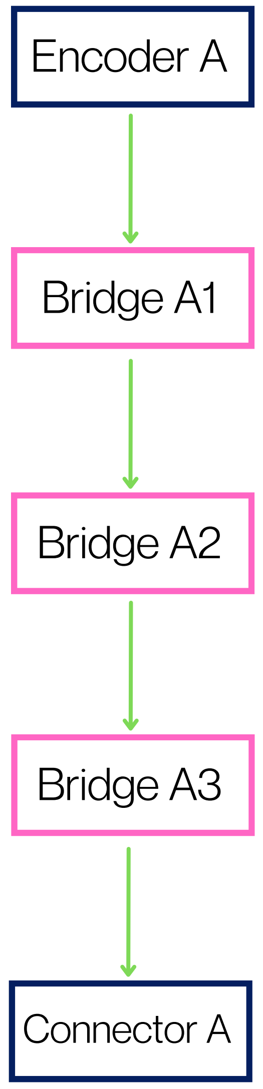
Linux-v4.0

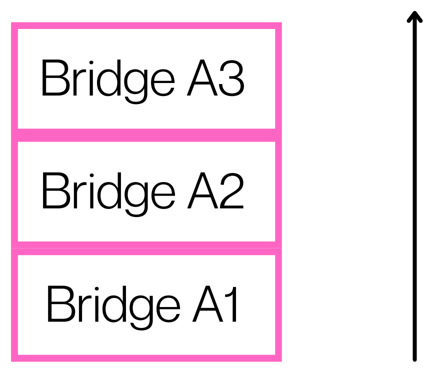
Linux KMS Display Pipeline



Display Bridge Topologies

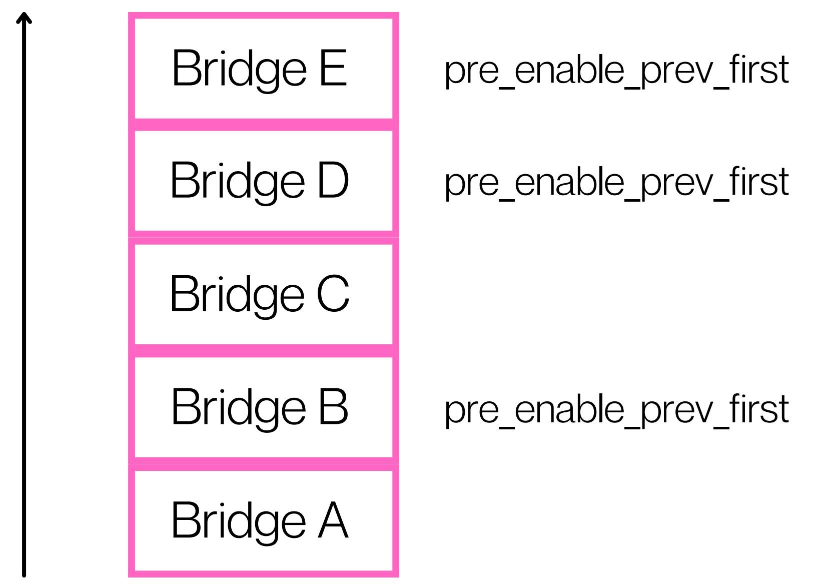
Linux DRM Bridge(recent - additions)

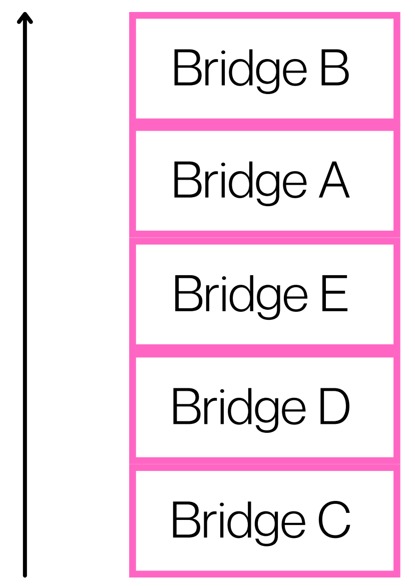




Result

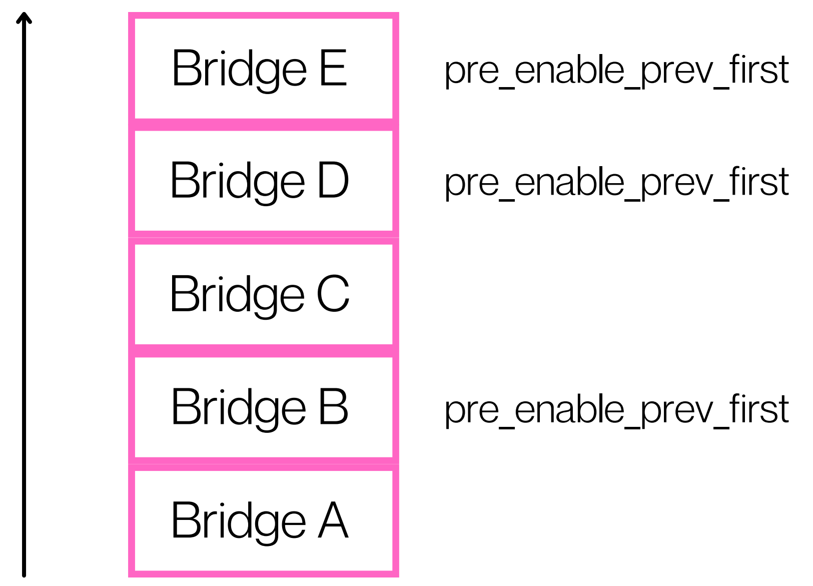
Bridge Attachment

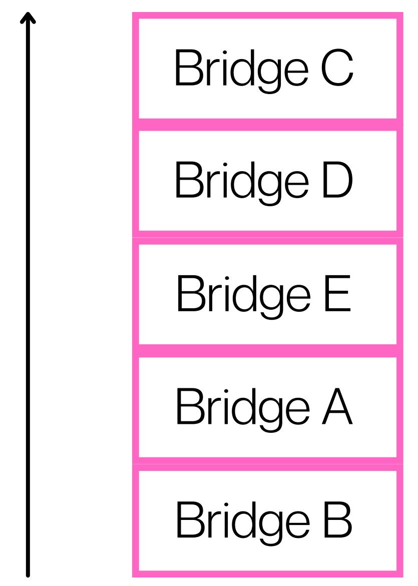




Result

Bridge pre_enable(new)

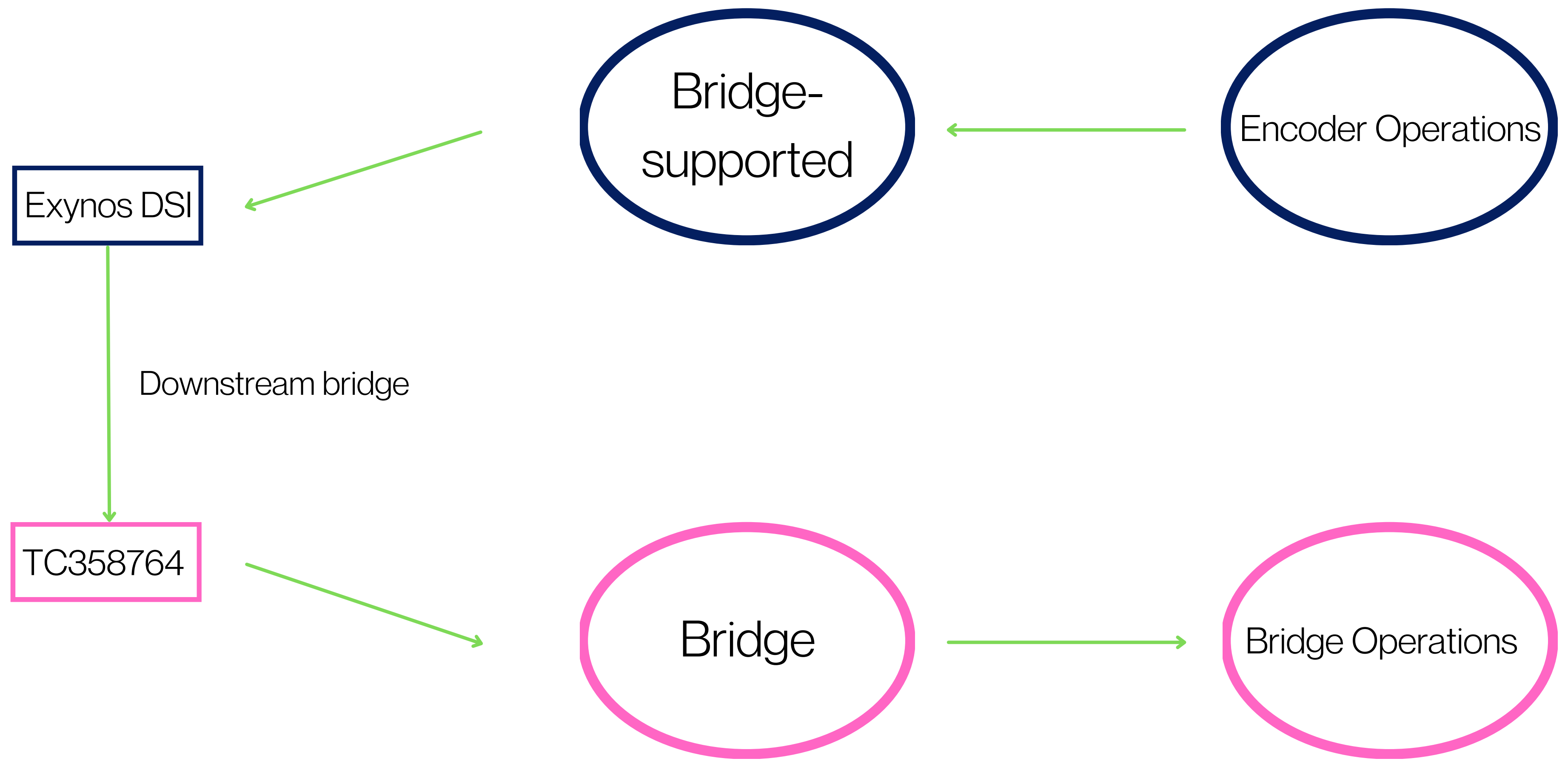




Result

Bridge post_disable(new)

Bridge as standard API - Why?



Bridge vs Bridge-supported

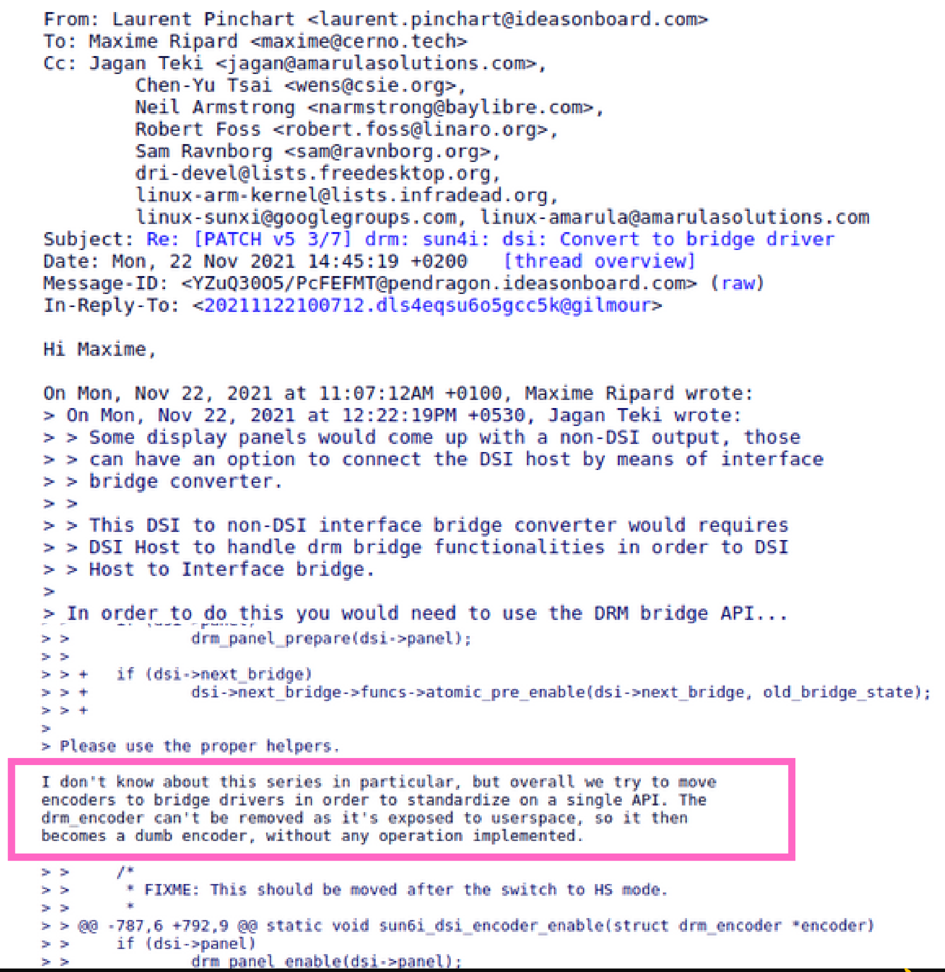
DROP ENCODER OPERATIONS

REGISTER BRIDGE HOOKS

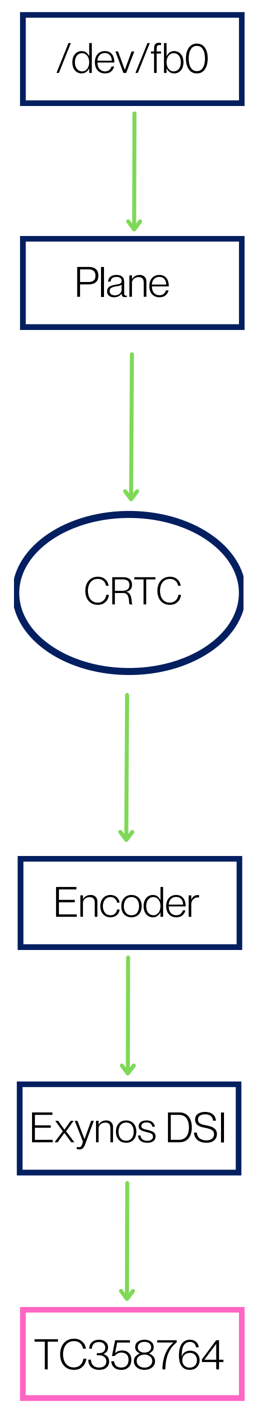
DROP BRIDGE CHAIN CALLS

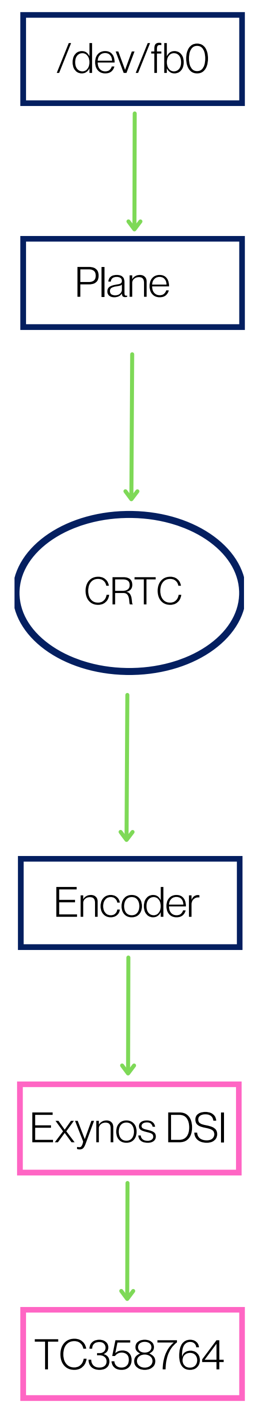
ENCODER BECOME DUMB

DRM Bridge as a Standardise API?

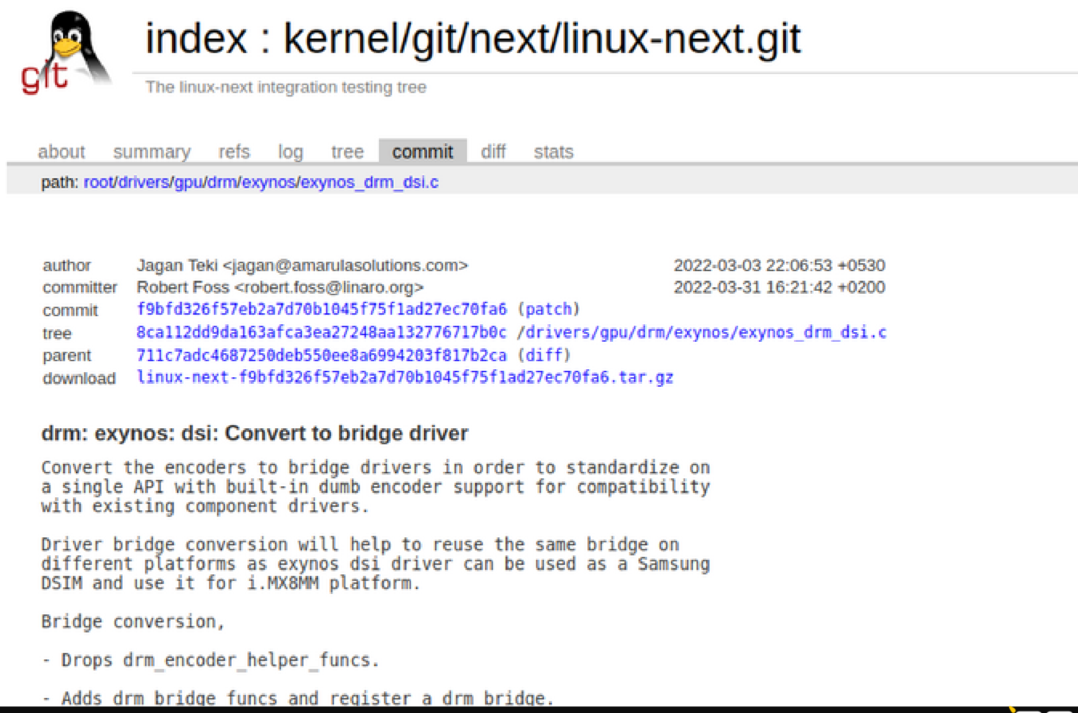


```
From: Laurent Pinchart <laurent.pinchart@ideasonboard.com>
To: Maxime Ripard <maxime@cerno.tech>
Cc: Jagan Teki <jagan@amarulasolutions.com>,
        Chen-Yu Tsai <wens@csie.org>,
        Neil Armstrong <narmstrong@baylibre.com>,
        Robert Foss <robert.foss@linaro.org>,
        Sam Ravnborg <sam@ravnborg.org>,
        dri-devel@lists.freedesktop.org,
        linux-arm-kernel@lists.infradead.org,
        linux-sunxi@googlegroups.com, linux-amarula@amarulasolutions.com
Subject: Re: [PATCH v5 3/7] drm: sun4i: dsi: Convert to bridge driver
Date: Mon, 22 Nov 2021 14:45:19 +0200   [thread overview]
Message-ID: <YZuQ3005/PcFEFMT@pendragon.ideasonboard.com> (raw)
In-Reply-To: <20211122100712.dls4eqsu6o5gcc5k@gilmour>

Hi Maxime,

On Mon, Nov 22, 2021 at 11:07:12AM +0100, Maxime Ripard wrote:
> On Mon, Nov 22, 2021 at 12:22:19PM +0530, Jagan Teki wrote:
> > Some display panels would come up with a non-DSI output, those
> > can have an option to connect the DSI host by means of interface
> > bridge converter.
> >
> > This DSI to non-DSI interface bridge converter would requires
> > DSI Host to handle drm bridge functionalities in order to DSI
> > Host to Interface bridge.
>
> In order to do this you would need to use the DRM bridge API...
> >             drm_panel_prepare(dsi->panel);
> >
> > +   if (dsi->next_bridge)
> > +           dsi->next_bridge->funcs->atomic_pre_enable(dsi->next_bridge, old_bridge_state);
> > +
>
> Please use the proper helpers.

I don't know about this series in particular, but overall we try to move
encoders to bridge drivers in order to standardize on a single API. The
drm_encoder can't be removed as it's exposed to userspace, so it then
becomes a dumb encoder, without any operation implemented.

> >     /*
> >      * FIXME: This should be moved after the switch to HS mode.
> >      *
> > @@ -787,6 +792,9 @@ static void sun6i_dsi_encoder_enable(struct drm_encoder *encoder)
> >     if (dsi->panel)
> >             drm_panel_enable(dsi->panel);
```





Linux-v5.17-rc2



index : kernel/git/next/linux-next.git

The linux-next integration testing tree

path: root/drivers/gpu/drm/exynos/exynos_drm_dsi.c

author	Jagan Teki <jagan@amarulasolutions.com>	2022-03-03 22:06:53 +0530
committer	Robert Foss <robert.foss@linaro.org>	2022-03-31 16:21:42 +0200
commit	f9bfd326f57eb2a7d70b1045f75f1ad27ec70fa6 (patch)	
tree	8ca112dd9da163afca3ea27248aa132776717b0c /drivers/gpu/drm/exynos/exynos_drm_dsi.c	
parent	711c7adc4687250deb550ee8a6994203f817b2ca (diff)	
download	linux-next-f9bfd326f57eb2a7d70b1045f75f1ad27ec70fa6.tar.gz	

drm: exynos: dsi: Convert to bridge driver

```
Convert the encoders to bridge drivers in order to standardize on
a single API with built-in dumb encoder support for compatibility
with existing component drivers.

Driver bridge conversion will help to reuse the same bridge on
different platforms as exynos dsi driver can be used as a Samsung
DSIM and use it for i.MX8MM platform.

Bridge conversion,

- Drops drm_encoder_helper_funcs.

- Adds drm_bridge_funcs and register a drm_bridge.
```

DRM Bridge as a Standardise API

Bridge as standard API - Impact

```c
static void exynos_dsi_enable(struct drm_encoder *encoder)
{
        struct exynos_dsi *dsi = encoder_to_dsi(encoder);
}

static void exynos_dsi_disable(struct drm_encoder *encoder)
{
        struct exynos_dsi *dsi = encoder_to_dsi(encoder);
}

static void
exynos_dsi_mode_set(struct drm_encoder *encoder,
                    struct drm_display_mode *mode,
                    struct drm_display_mode *adjusted_mode)
{
        struct exynos_dsi *dsi = encoder_to_dsi(encoder);

        drm_mode_copy(&dsi->mode, adjusted_mode);
}

static const struct drm_encoder_helper_funcs exynos_dsi_encoder_helper_funcs = {
        .enable = exynos_dsi_enable,
        .disable = exynos_dsi_disable,
        .mode_set = exynos_dsi_mode_set,
};
```

```c
static void exynos_dsi_post_disable(struct drm_bridge *bridge)
{
        struct exynos_dsi *dsi = bridge_to_dsi(bridge);
}

static void exynos_dsi_mode_set(struct drm_bridge *bridge,
                                const struct drm_display_mode *mode,
                                const struct drm_display_mode *adjusted_mode)
{
        struct exynos_dsi *dsi = bridge_to_dsi(bridge);
}

static int exynos_dsi_attach(struct drm_bridge *bridge,
                             enum drm_bridge_attach_flags flags)
{
        struct exynos_dsi *dsi = bridge_to_dsi(bridge);

        return drm_bridge_attach(bridge->encoder, dsi->out_bridge, NULL, flags);
}

static const struct drm_bridge_funcs exynos_dsi_bridge_funcs = {
        .pre_enable                = exynos_dsi_pre_enable,
        .enable                    = exynos_dsi_enable,
        .disable                   = exynos_dsi_disable,
        .post_disable              = exynos_dsi_post_disable,
        .mode_set                  = exynos_dsi_mode_set,
        .attach                    = exynos_dsi_attach,
};
```

Encoder Ops gone - Bridge Ops up

```
struct exynos_dsi {
         struct drm_encoder encoder;
         struct mipi_dsi_host dsi_host;
         struct list_head bridge_chain;
};

static void exynos_dsi_disable(struct drm_encoder *encoder)
{
        struct exynos_dsi *dsi = encoder_to_dsi(encoder);
        struct drm_bridge *iter;

        list_for_each_entry_reverse(iter, &dsi->bridge_chain, chain_node) {
                if (iter->funcs->disable)
                        iter->funcs->disable(iter);
        }

        exynos_dsi_set_display_enable(dsi, false);

        list_for_each_entry(iter, &dsi->bridge_chain, chain_node) {
                if (iter->funcs->post_disable)
                        iter->funcs->post_disable(iter);
  }
```

```
struct exynos_dsi {
         struct drm_encoder encoder;
         struct mipi_dsi_host dsi_host;
         ~~struct list_head bridge_chain;~~
};

static void exynos_dsi_disable(struct drm_encoder *encoder)
{
        struct exynos_dsi *dsi = encoder_to_dsi(encoder);
        ~~struct drm_bridge *iter;~~

        ~~list_for_each_entry_reverse(iter, &dsi->bridge_chain, chain_node) {~~
                ~~if (iter->funcs->disable)~~
                        ~~iter->funcs->disable(iter);~~
        }

        exynos_dsi_set_display_enable(dsi, false);

        ~~list_for_each_entry(iter, &dsi->bridge_chain, chain_node) {~~
                ~~if (iter->funcs->post_disable)~~
                        ~~iter->funcs->post_disable(iter);~~
  }
```

(explicit) Bridge Chain-calls gone

DRM Bridge as a Standardise API - Advantage

SINGLE BRIDGE DRIVER

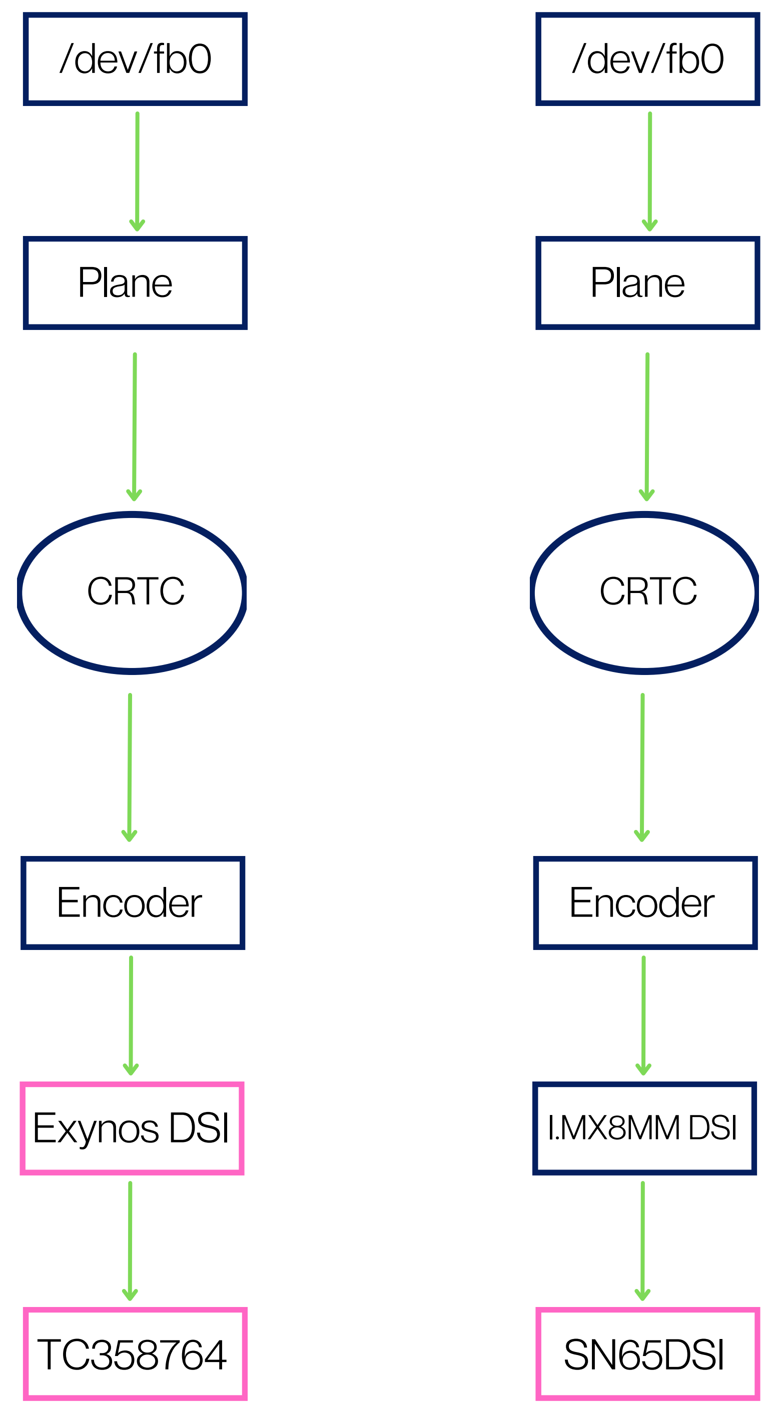
DROP ENCODER OPERATED DRIVER

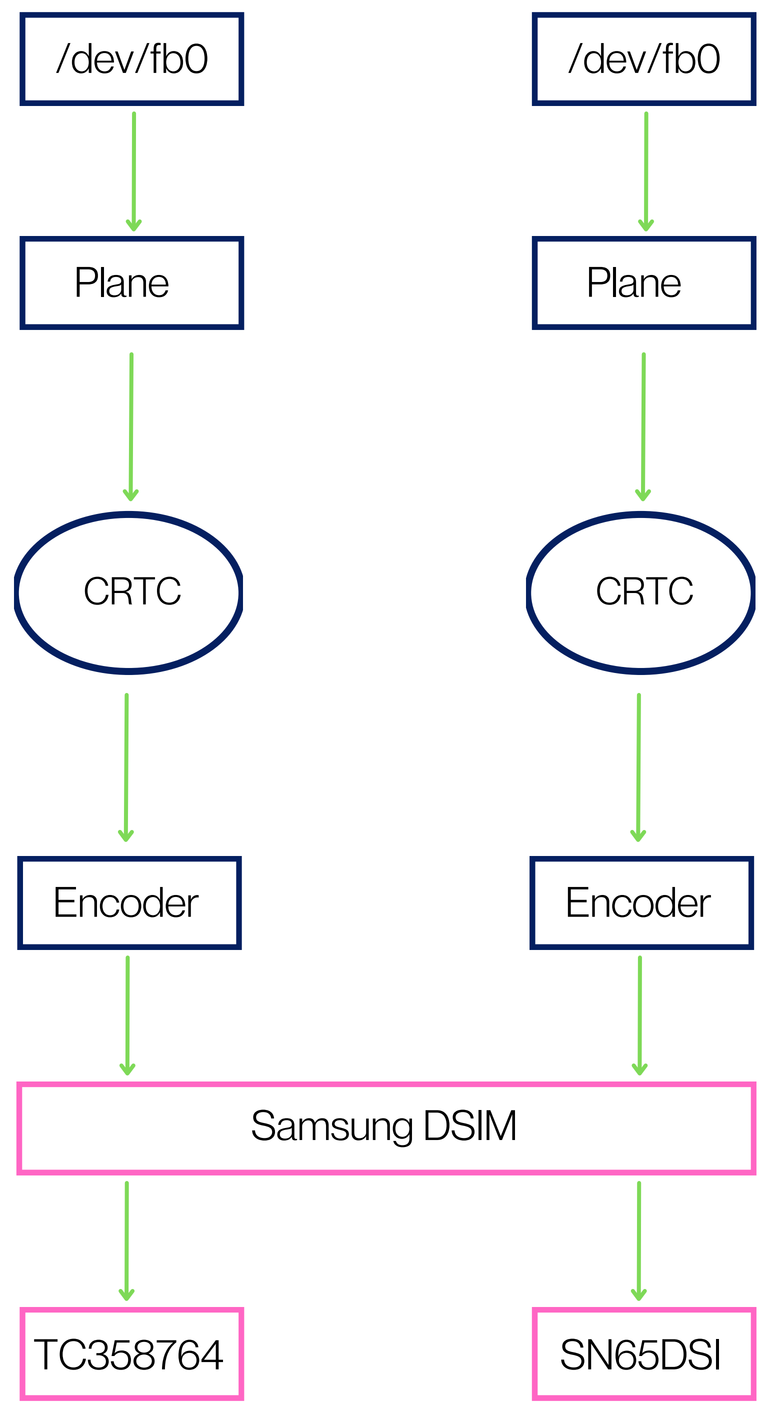
SUPPORT CHAIN OF BRIDGE HARDWARE

MAINTENANCE EASY FOR LONG RUN

DRM Bridge as a Standardise API - Advantage

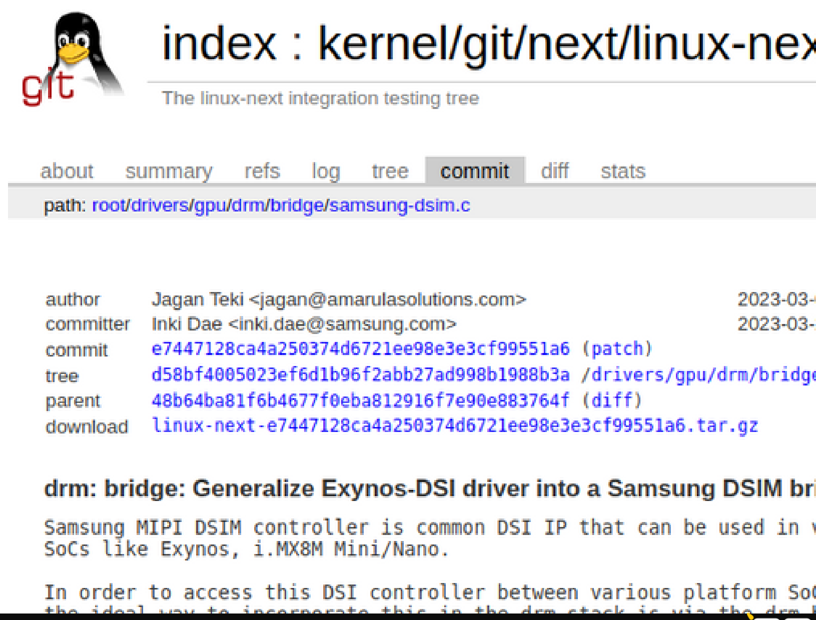
DRM Bridge as a Standardise API - 1 Bridge Driver





Linux-v5.17-rc2

DRM Bridge as a Standardise API - Common Bridge


index : kernel/git/next/linux-nex
The linux-next integration testing tree
about summary refs log tree commit diff stats

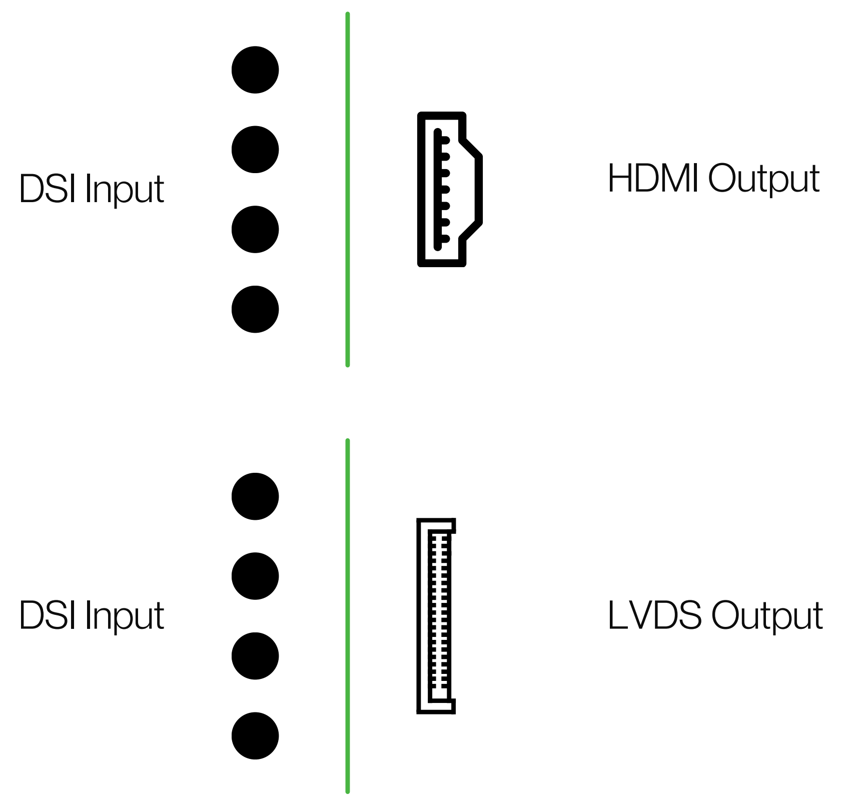
path: root/drivers/gpu/drm/bridge/samsung-dsim.c

author	Jagan Teki <jagan@amarulasolutions.com>	2023-03-
committer	Inki Dae <inki.dae@samsung.com>	2023-03-
commit	e7447128ca4a250374d6721ee98e3e3cf99551a6 (patch)	
tree	d58bf4005023ef6d1b96f2abb27ad998b1988b3a /drivers/gpu/drm/bridg	
parent	48b64ba81f6b4677f0eba812916f7e90e883764f (diff)	
download	linux-next-e7447128ca4a250374d6721ee98e3e3cf99551a6.tar.gz	

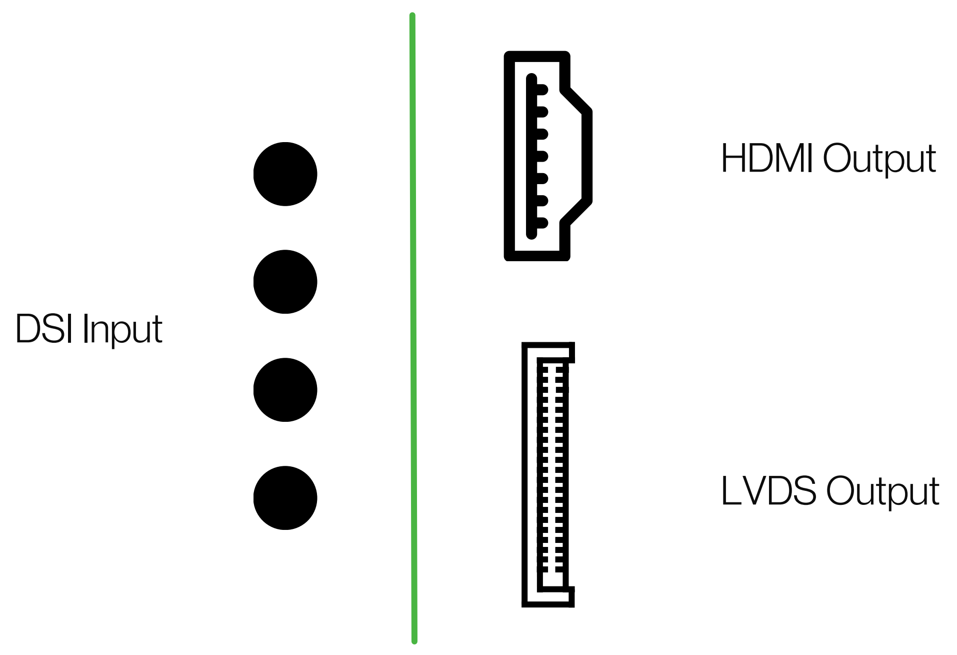
drm: bridge: Generalize Exynos-DSI driver into a Samsung DSIM br

```
Samsung MIPI DSIM controller is common DSI IP that can be used in v
SoCs like Exynos, i.MX8M Mini/Nano.

In order to access this DSI controller between various platform SoC
```

Linux DRM Bridge - Any Potential Enhancement?



1x1 - bridge conversion

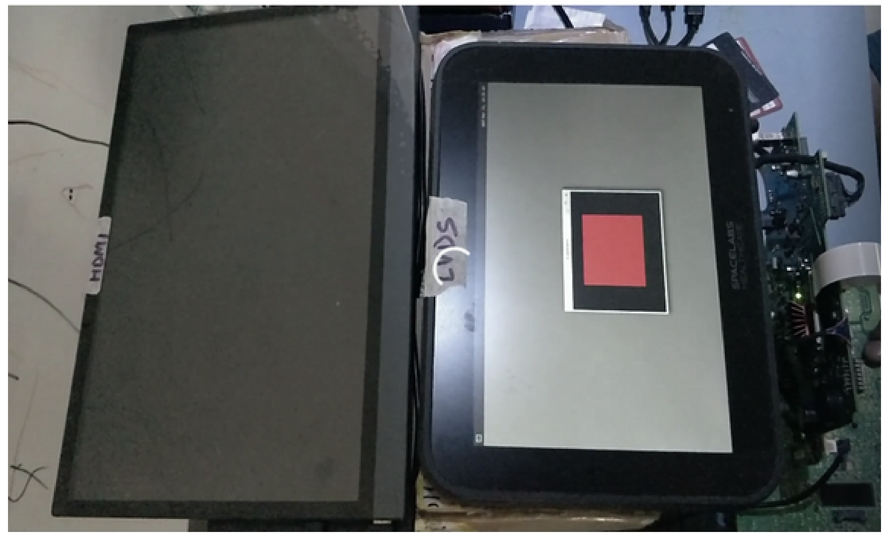


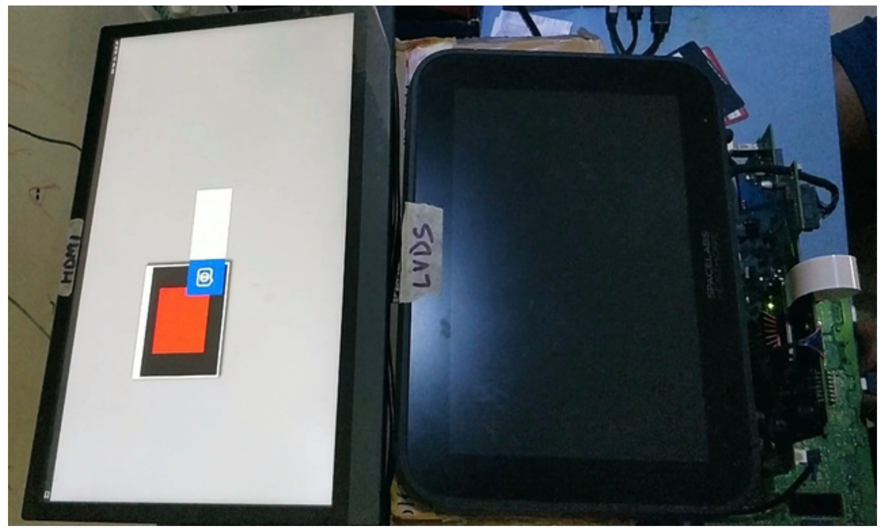
1xn - access one output at a time
One of the outputs must have HP

Bridge Switch (not official)

LVDS Output

DSI Input

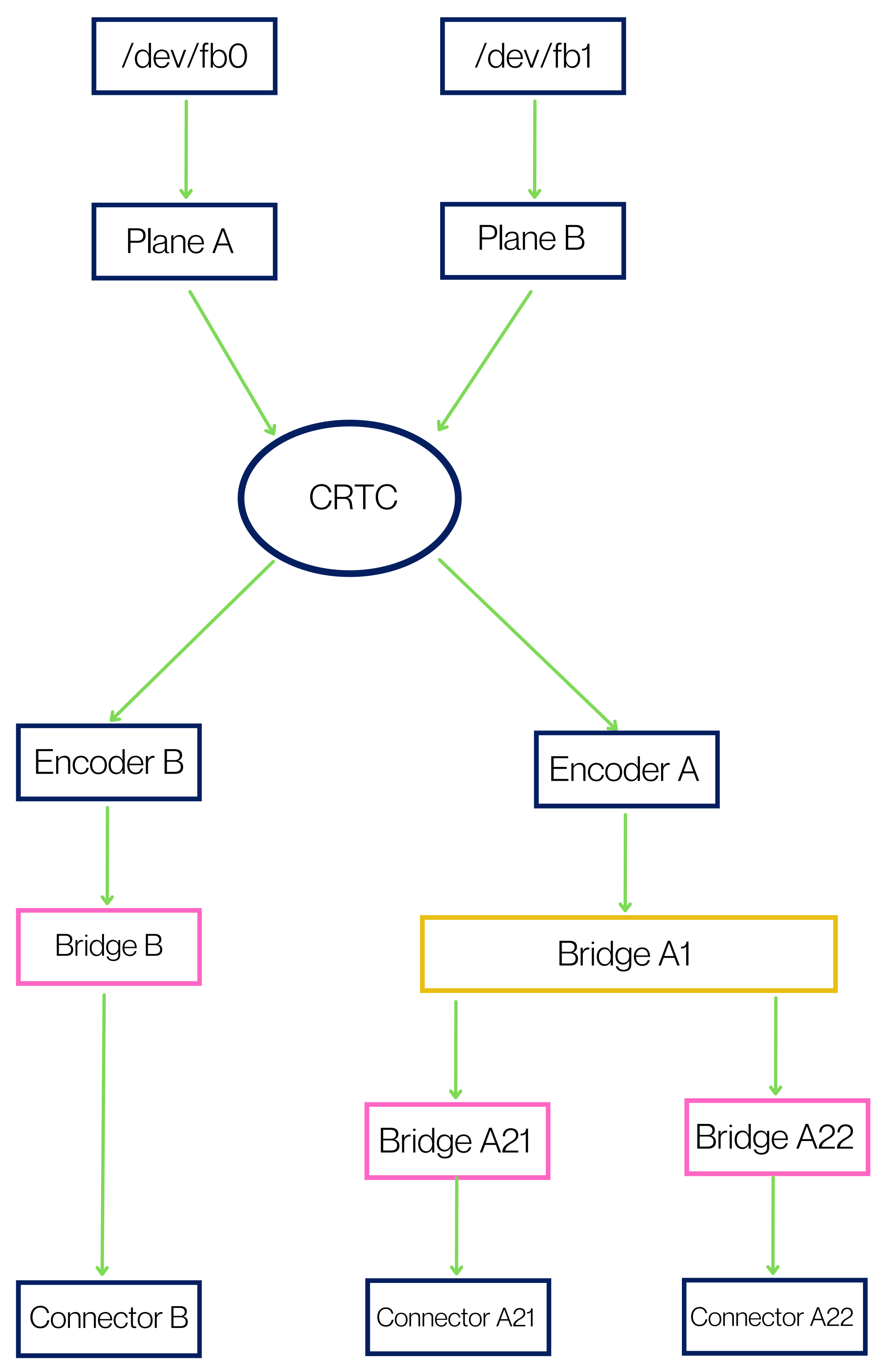


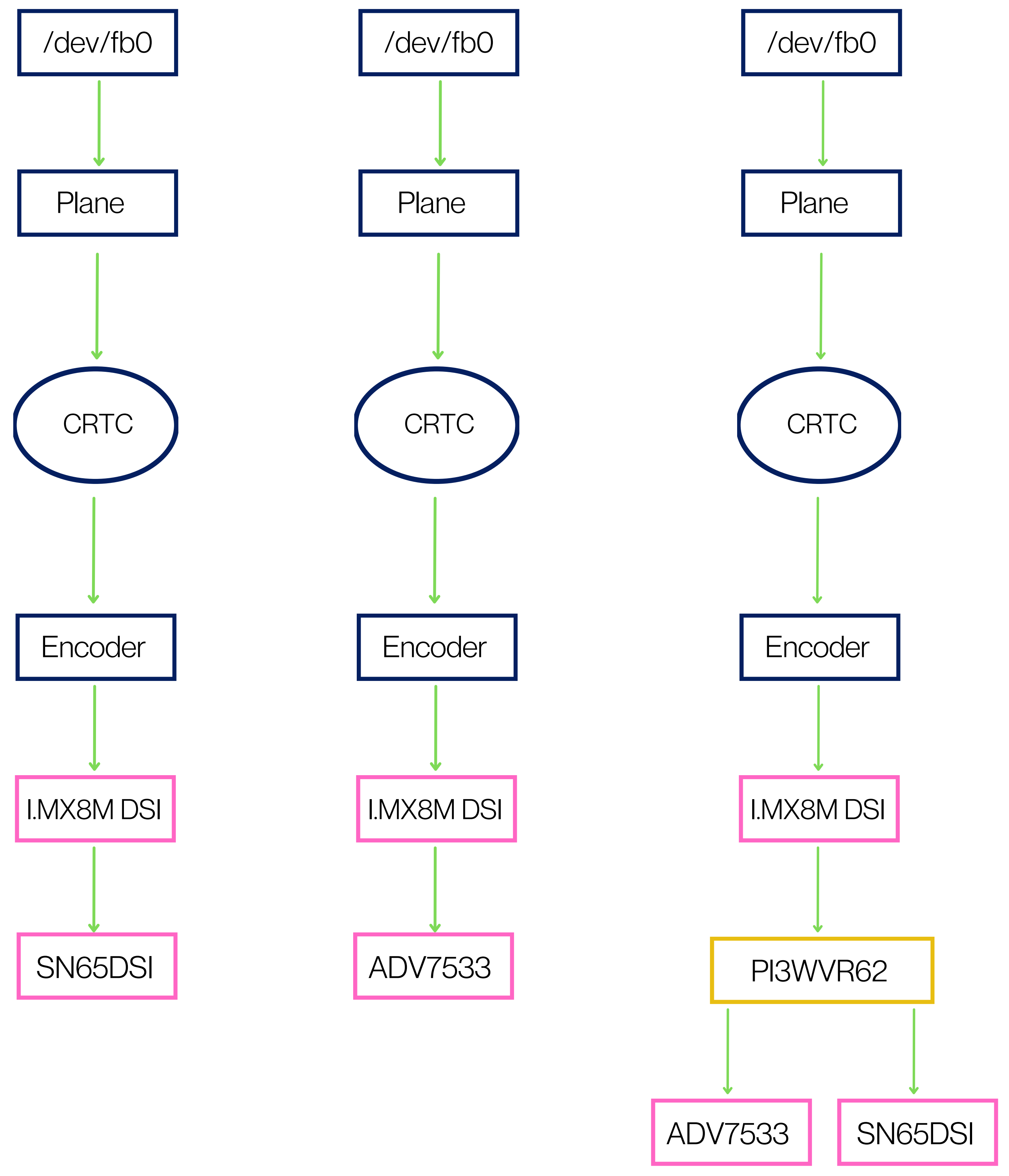


HDMI Output

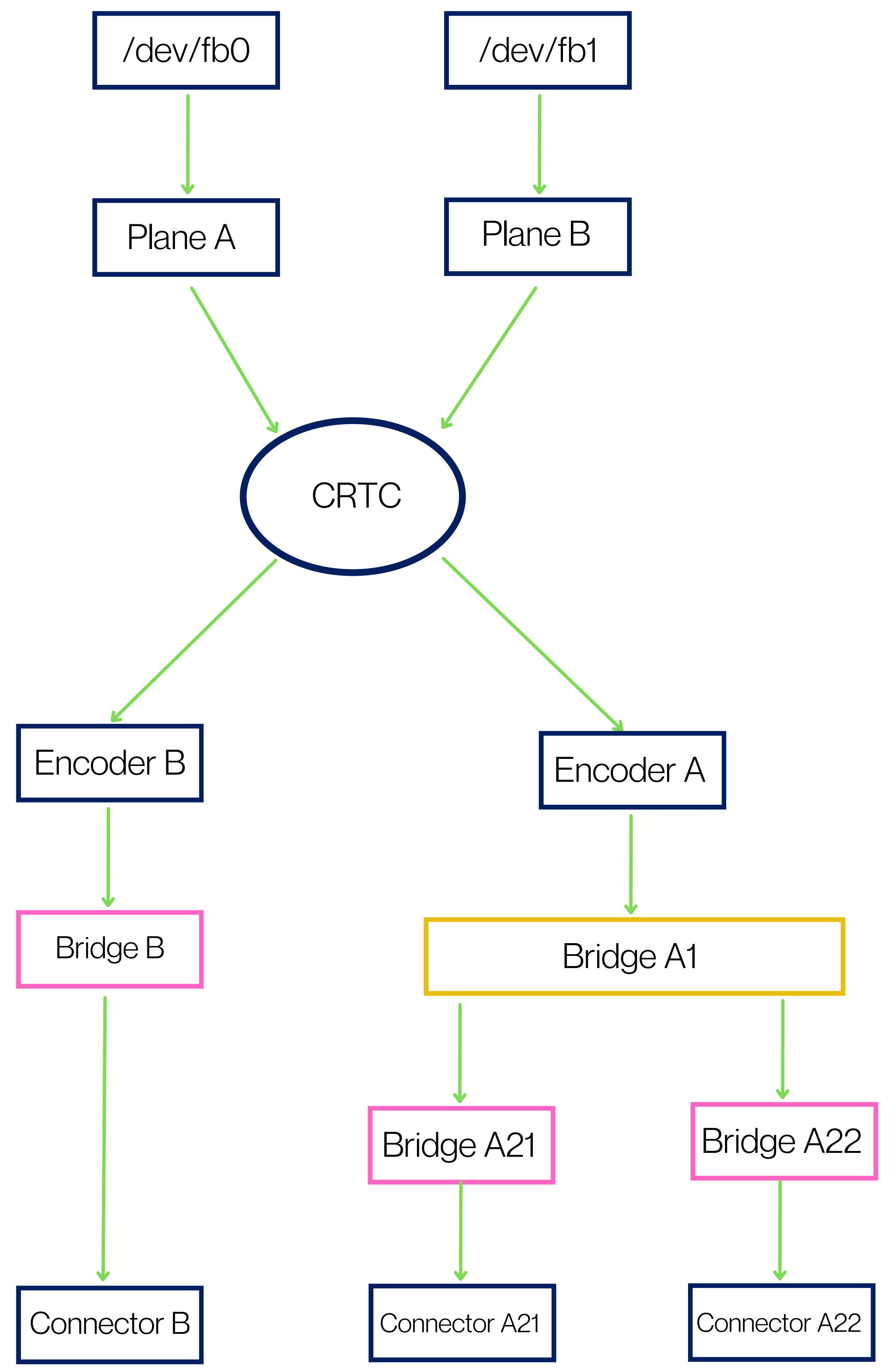
1xn - access one output at a time
One of the outputs must have HP

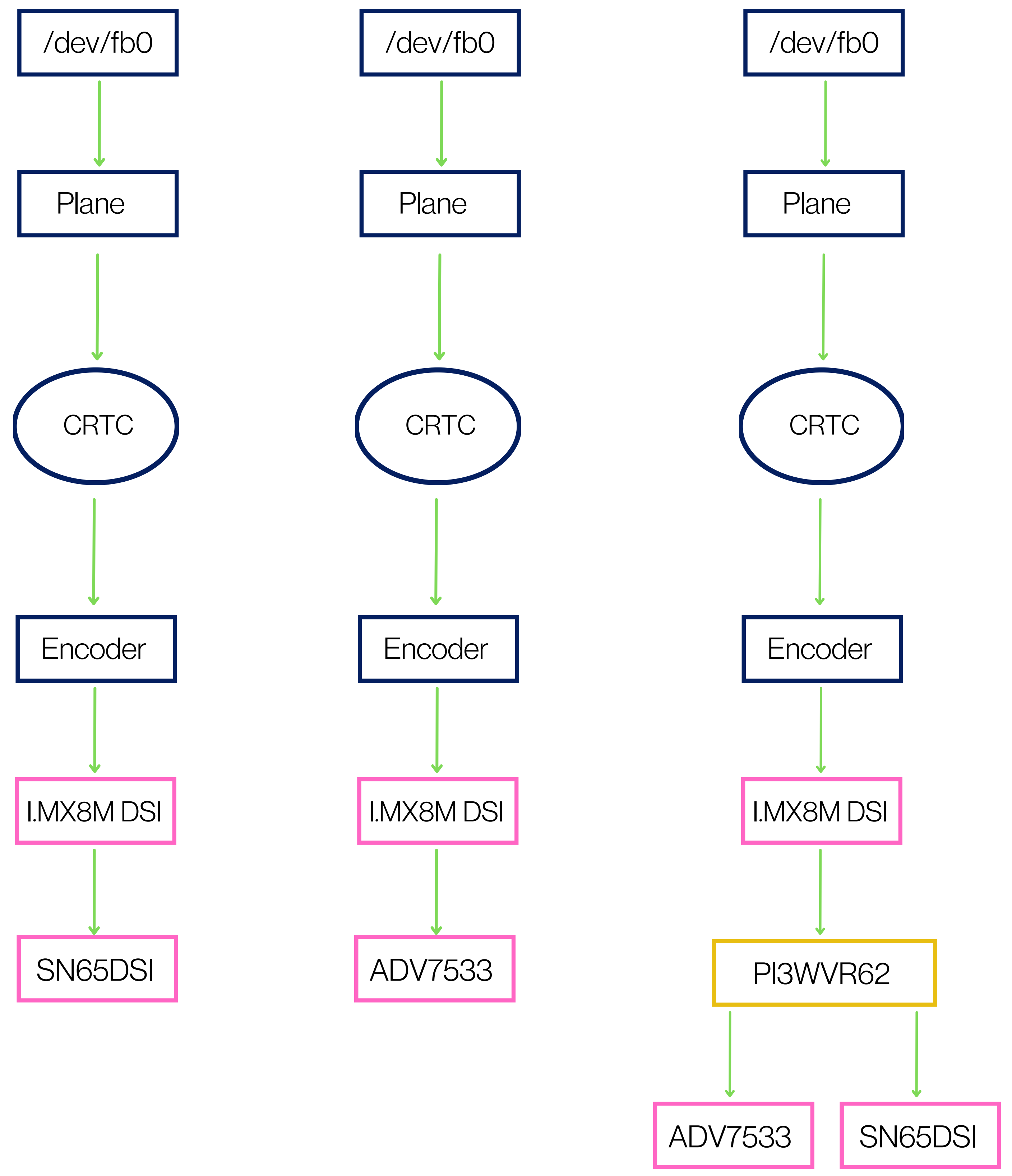
Bridge Switch (not official)





Bridge Switch - KMS Pipeline





Bridge Switch - Implementation

Linux DRM Bridges are lists ~~not tree~~

- [PATCH v3 0/5] Add generic-display-mux driver and bindings
    - **Pin-yen Lin** <treapking@chromium.org>
- DW-MIPI-DSI as common Bridge
- Mainline Solutions for Bridge Switch
- New Bridge drivers DSI to HDMI Out and HDMI In
- DLP Project to handle DSI to DMD Out, HDMI Out, and HDMI In
- *Any questions - jagan@amarulasolutions.com*
- *IRC: jagan_ at #dri-devel*

Potential Common Bridge conversions, TODO