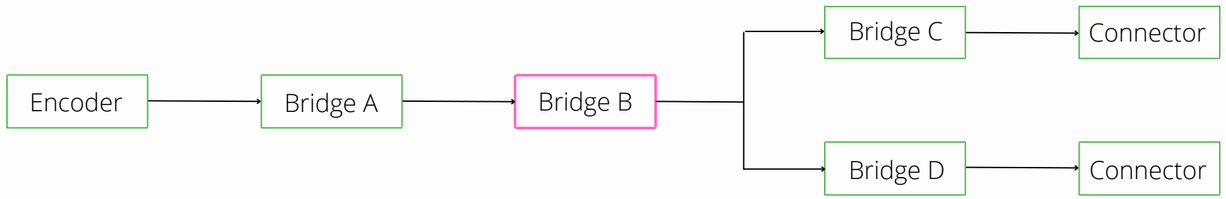
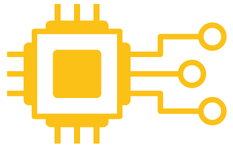


Runtime display switching in a Linux DRM bridge subsystem



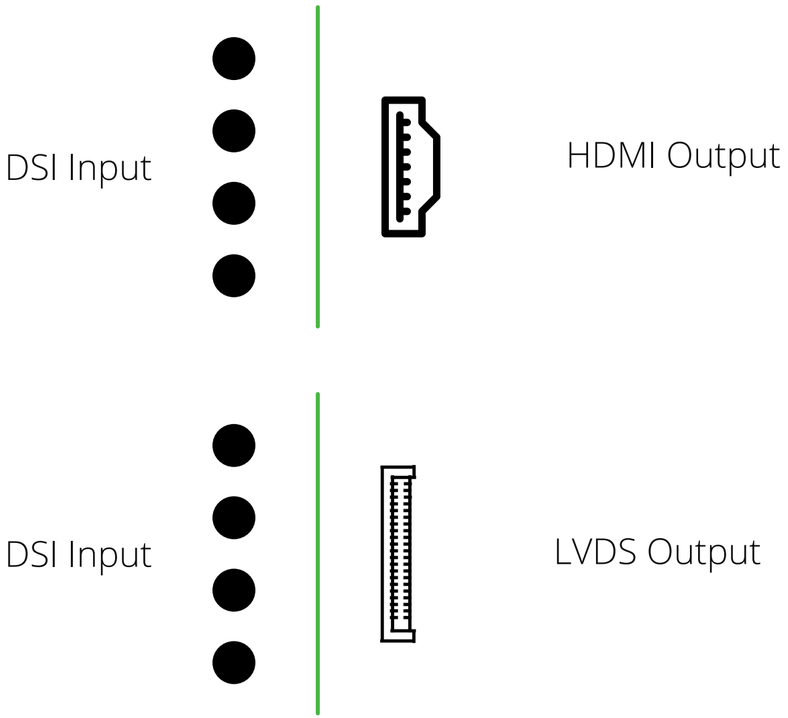
Embedded Linux Engineer

Co-Founder, Amarula Solutions(India)

Technical Conference Speaker

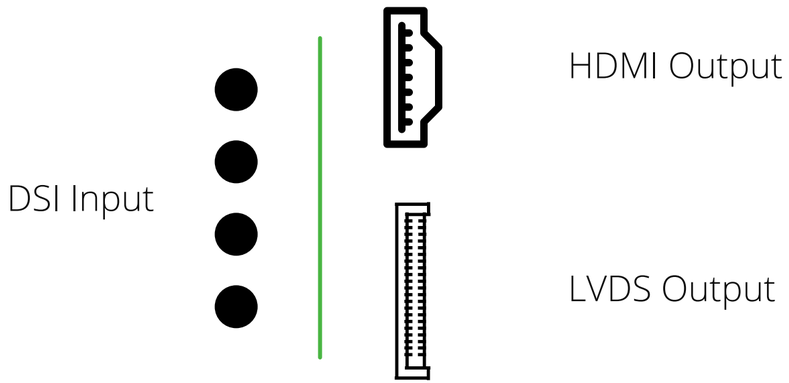
	U-Boot		BuildRoot Making Embedded Linux Easy / yocto PROJECT
Contributions (patches)	1000+	280+	50+
Maintainer (Subsystems)	SPI/SPI Flash Allwinner sunXi SoC	MIPI DSI Bridge/Panel drivers NXP PF8X00 PMIC driver	Hardware platforms based on i.MX6/8, Rockchip, Allwinner

Jagan Teki



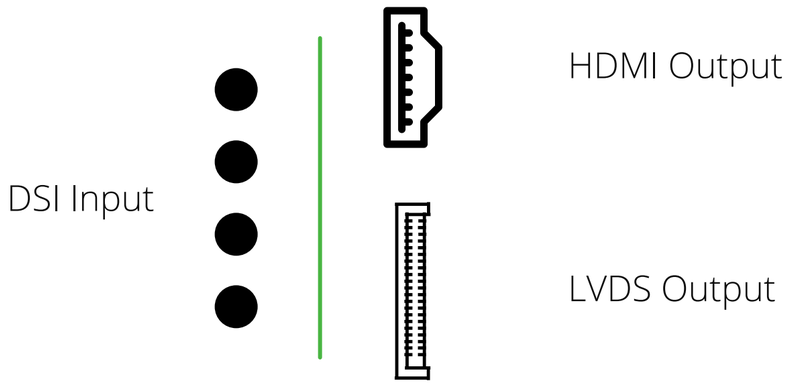
1x1 - bridge conversion

Display bridge



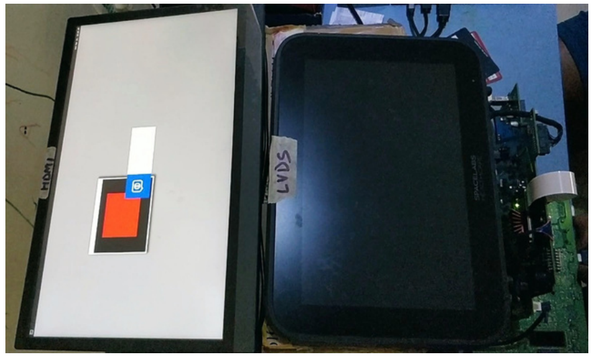
1xn - access one output at a time
One of the outputs must have HP

Display bridge switch

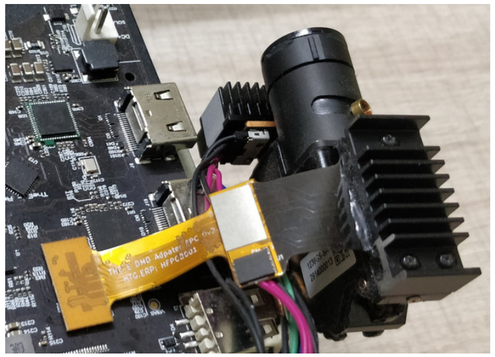


1xn - access one output at a time
One of the outputs must have HP

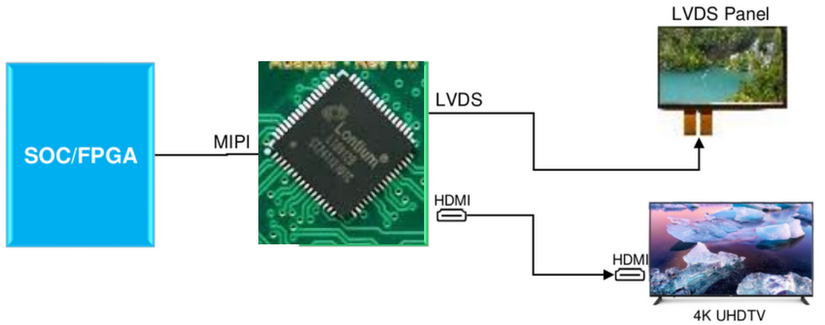
Display bridge switch



PI3WVR626



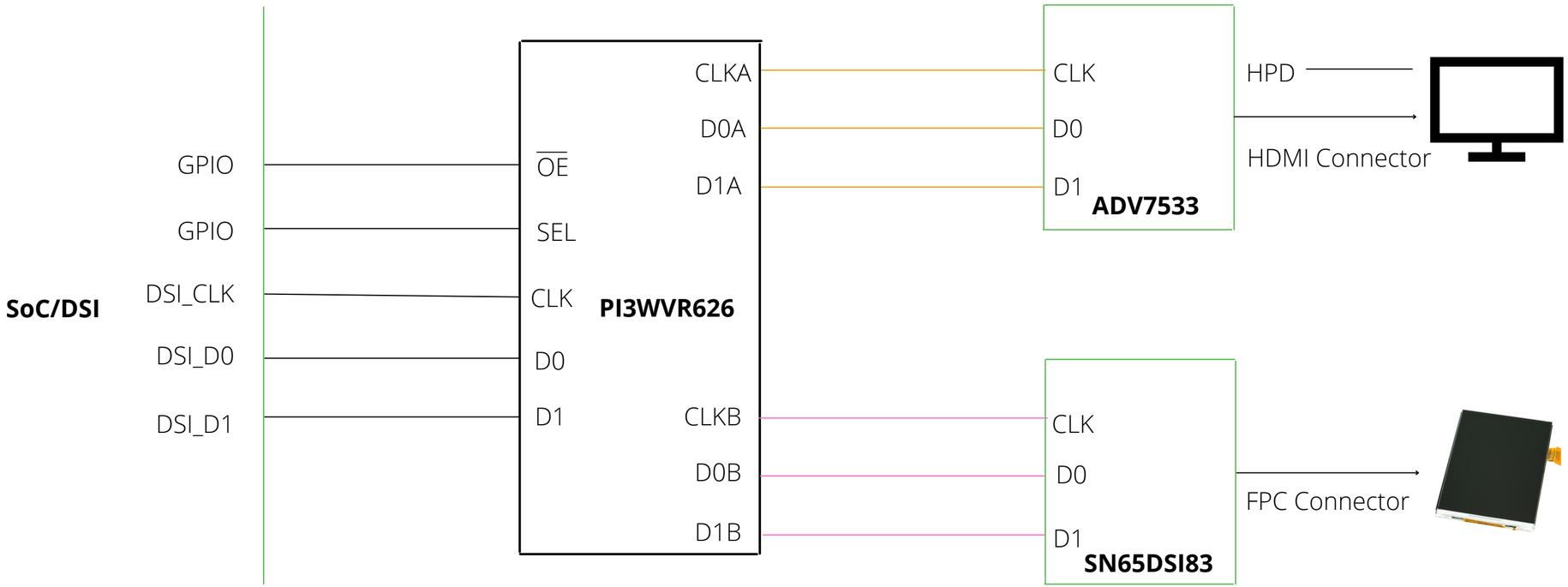
TS3L4892



LT8912B

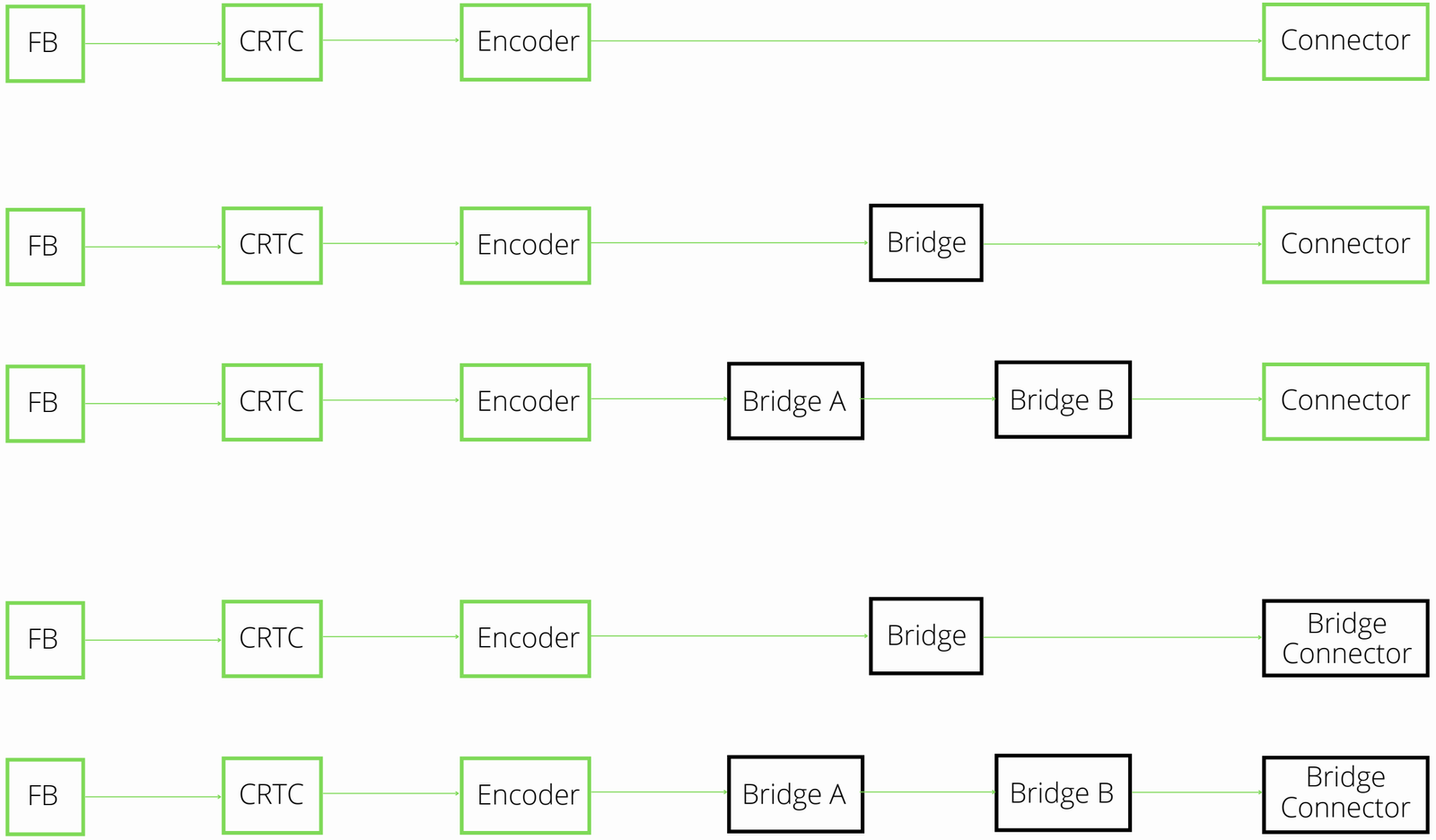
Display switch possibilities

Display switch - MIPI Switch, Switch Bridge

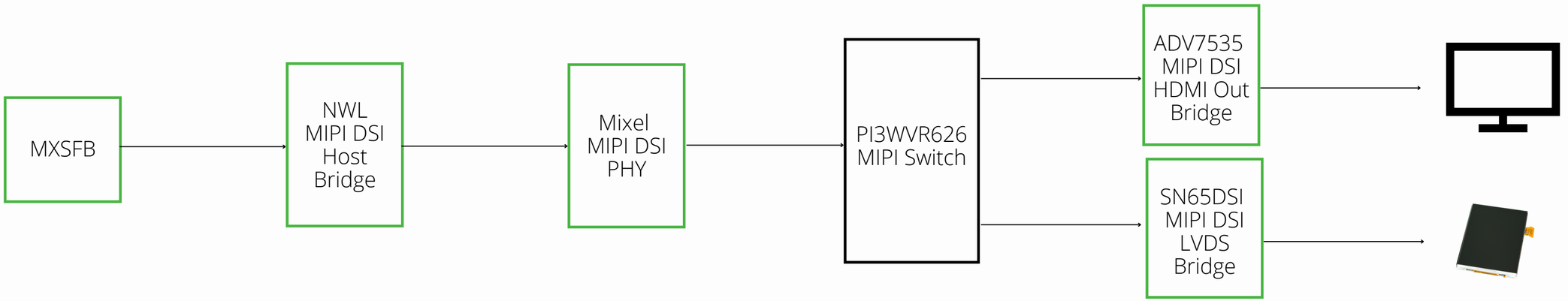


SEL	$\overline{OE}$	Out
0	0	HDMI
1	0	LVDS

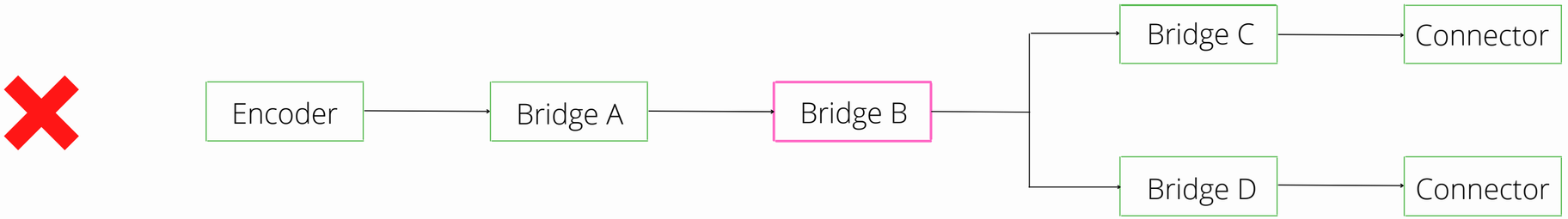
MIPI Switch - *PI3WVR626*



Linux DRM Bridge

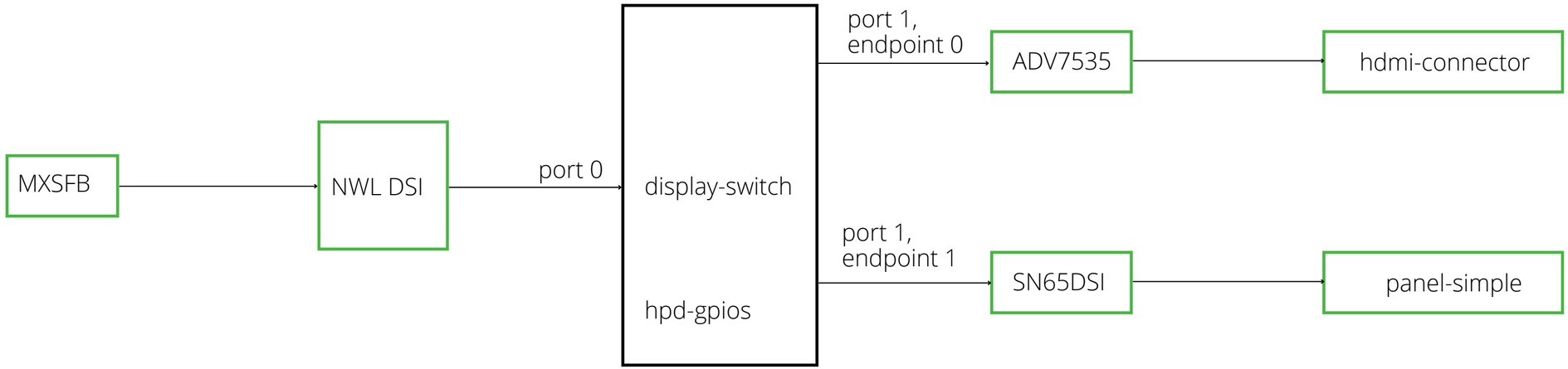


Linux DRM Bridge Switch

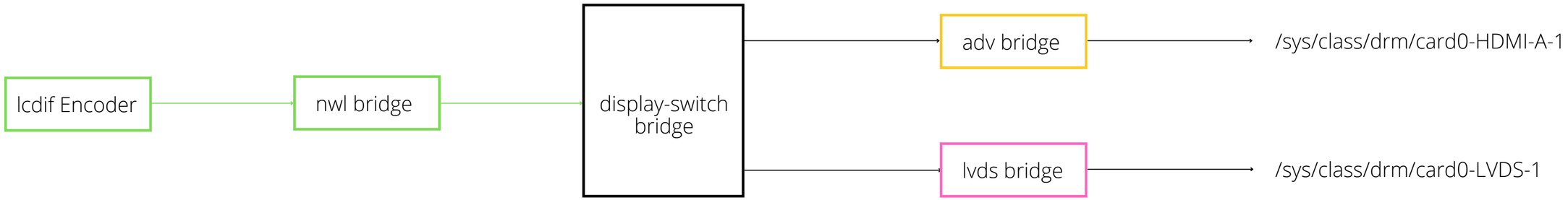


Linux DRM Bridges are lists ~~not tree~~

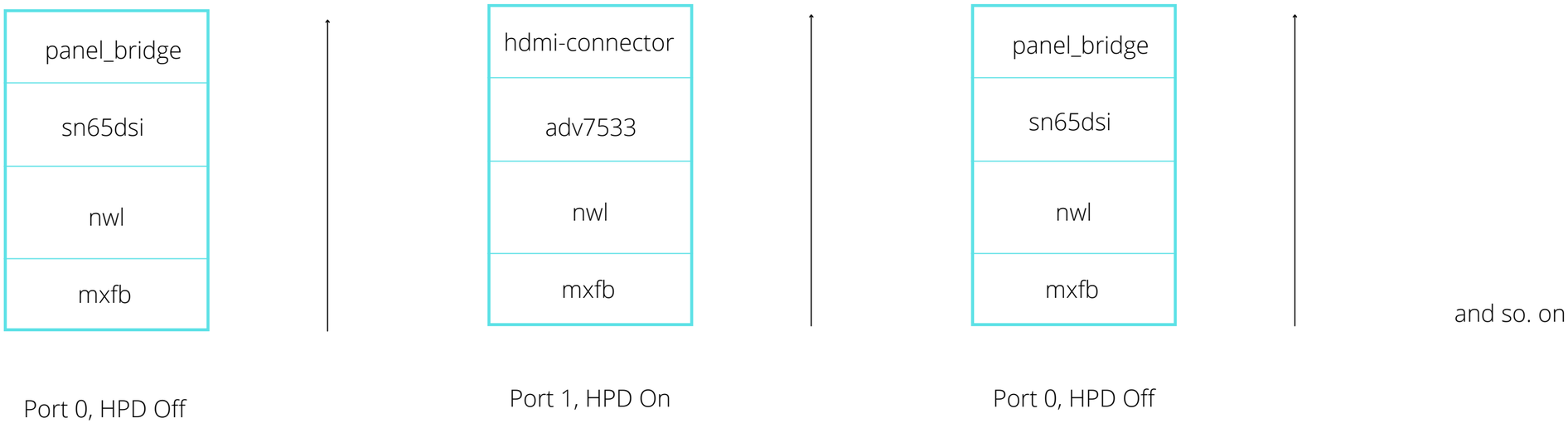
Implementation 1 - Create one connector at a time



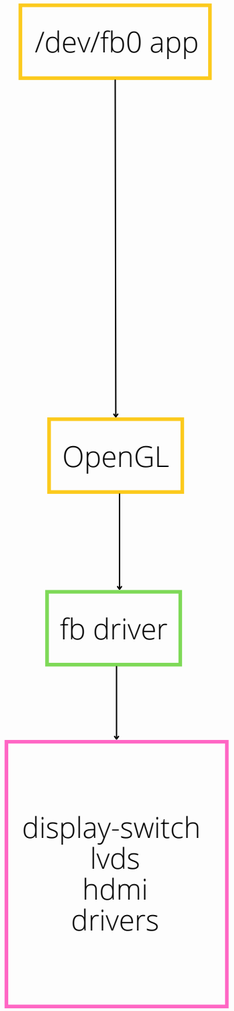
Channel	Port Change	Out	HPD	Bridge State
0	0	HDMI	On	Detach SN65DSI, Disable SN65DSI, Attach ADV7535, Enable ADV7533
1	1	LVDS	Off	Detach ADV7535, Disable ADV7533,Attach SN65DSI, Enable SN65DSI



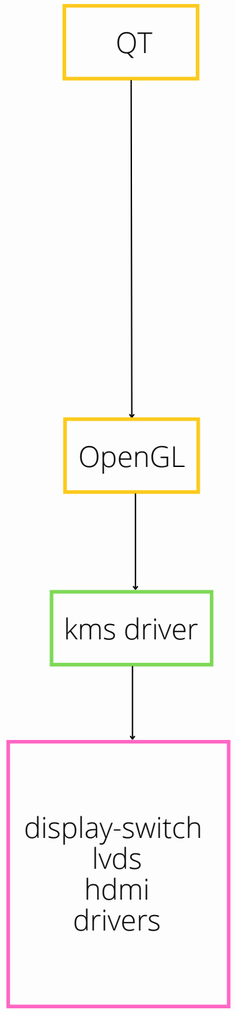
Bridge chain from display-switch establish one at a time as DRM bridges are linear - So we can see only enabled connector in /sys not both



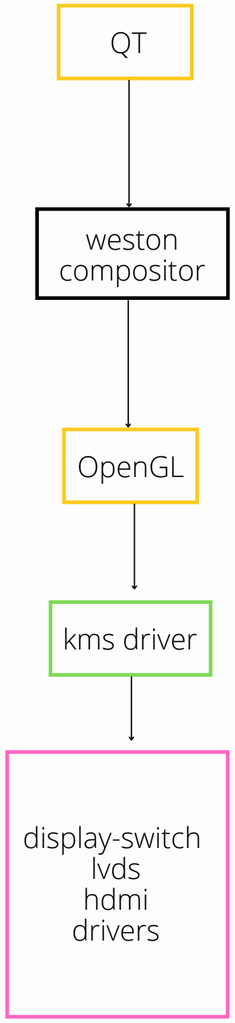
DRM Bridge Chain (Stack)



fbcon flow



kms flow



kms + weston flow

Demo Test

You should not handle this at all from a hotplug.

The way kms works is roughly as follows:

1. hw output state changes
2. driver detects this (either through hpd interrupt or polling)
3. driver sends out hotplug uevent

That's it. Nothing else, no bridge rebinding, no link retaining is
required.

Then either userspace or fbcon emulation reacts to this hotplug event by
doing an atomic modeset, where it hopefully disables the old output and
re-enables the new output. Your atomic_check needs to validate that
everything is all right (i.e. not enabling both at the same time).

Note that if you change stuff underneath, then that tends to seriously
upset atomic users. Also you should try to continue supporting at least
page flips with the wrong config, compositors otherwise tend to crash.

This also means that if userspace doesn't handle hotplug events, then you
might end up with a black screen. That's ok. We try to avoid that when
it's practical (e.g. dp sst link retraining), but not when it's too hard
(dp mst hot-replug relies on userspace restoring everything).

Finally exchanging the bridge chain isn't supported, there's no locking
for that since it's assumed to be invariant over the lifetim of the
drm_device instance. The simplest way to make that happen right now is to
have 2 drm_encoder instances, one with the lvds bridge chain, the other
with the hdmi bridge chain, and select the right encoder/bridge chain
depending upon which output userspace picks.

Also ofc your atomic_check needs to make sure that they're not both
enabled at the same time :-)

I wouldn't try to make bridge chains exchangeable instead, that's
headaches - e.g. with dp mst we've also opted for a bunch of fake
drm_encoders to model that kind of switching.
-Daniel
--
Daniel Vetter
Software Engineer, Intel Corporation

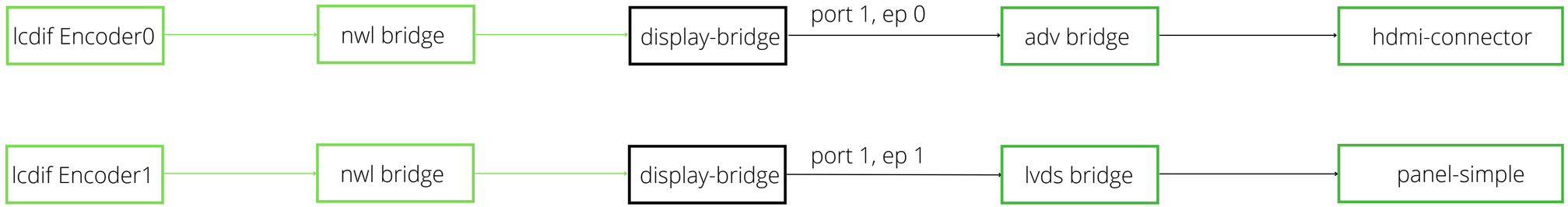
I think we could simply create two connectors, one for LVDS, one for HDMI, with atomic_check making sure only one of them is enabled at the same time?

The one thing that would make it difficult is that we're changing the bridge list to a tree. For example, in such a case, what should drm_bridge_get_next_bridge return? This will obviously depend on the state, but it's used in context were we don't have a state (such as drm_bridge_connector_init).

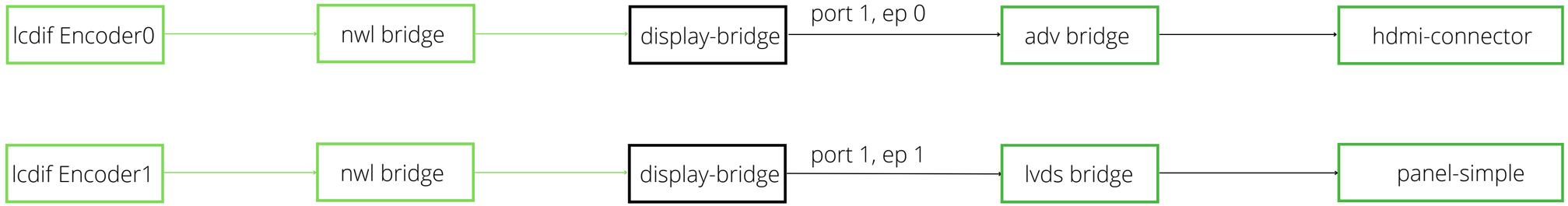
Maxime

~~DRM Bridge rebinding at runtime~~

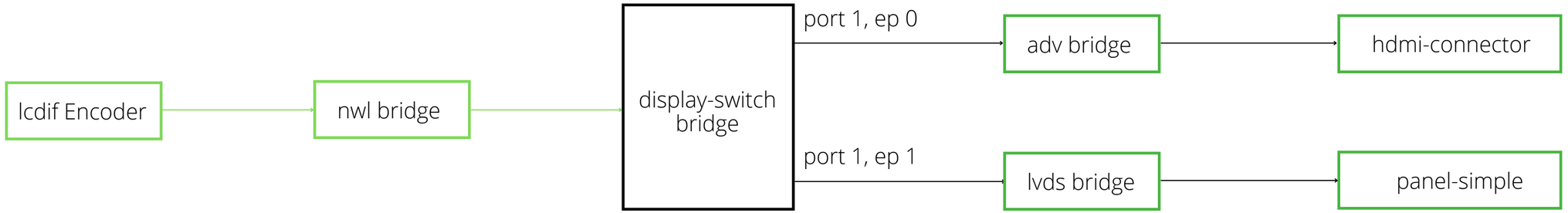
Implementation 2 - Create two connectors, enable one at a time



Two Encoder and create two connectors but enable one at a time: display-bridge chain return -EBUSY for endpoint 1



Two Encoder and create two connectors but enable one at a time: display-bridge chain return -EBUSY for endpoint 1



One Encoder and create two connectors but enable one at a time: bridge list to tree logic make last bridge appends to first bridge list

- Deep drive to DRM bridge
- Mainline solutions for Bridge switch

TODO