Bridging of Media Data Interfaces over Multimedia Serial Links

Vladimir Zapolskiy <vladimir@tuxera.com> Tuxera Inc.

Automotive Linux Summit 2019 July 18, 2019

TUXERA

Vladimir Zapolskiy

Embedded Linux Engineer at Tuxera

- Performance analysis and improvements
- Development in the Linux kernel space
- Consulting and maintenance

Open Source Contributor

- Linux kernel
- U-Boot
- OpenEmbedded

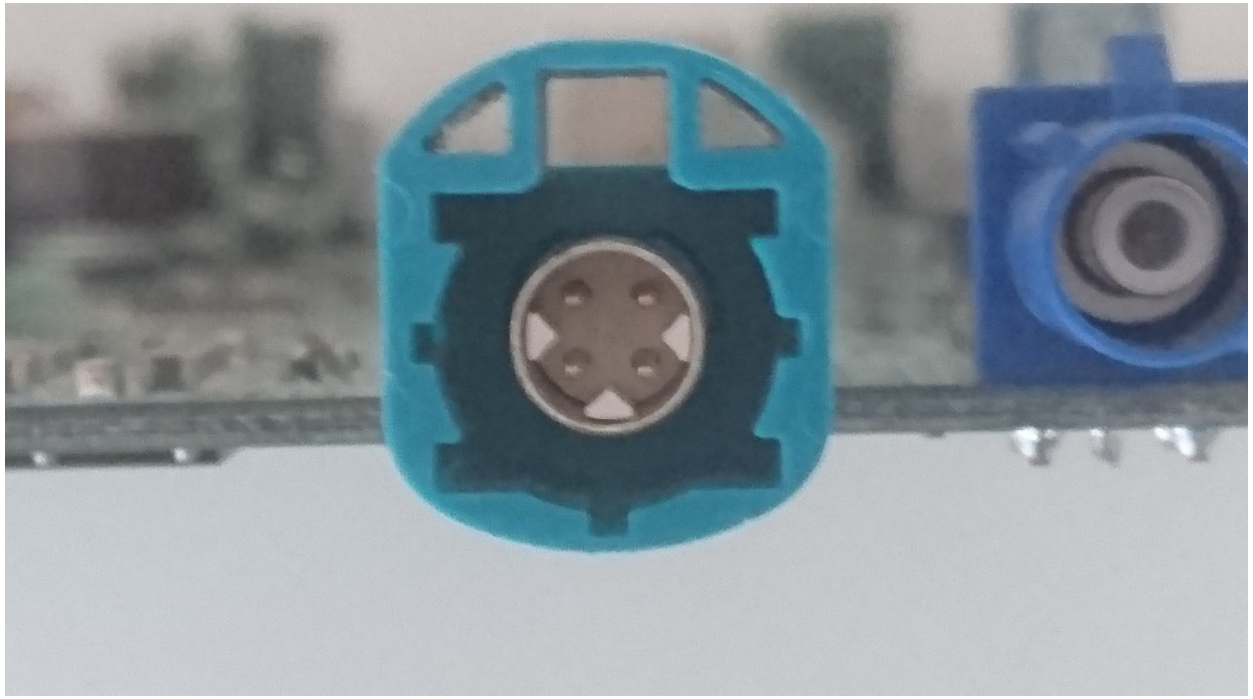
Summary of presentation

- FPD-Link III hardware description
- TI DS90Ux9xx serializers and deserializers
- Bridging functionality of DS90Ux9xx ICs
- Selection of Mutifunction device driver framework for implementation
- Display controller to FPD-Link III to display panel
- Camera sensor to FPD-Link III to image signal controller
- Audio bridging
- Current status and future work

FPD-Link III

Features

- Bidirectional channel transfer of control signals
- Up to 3Gbit/s data transfer speed
- Media contents with HDCP support
- Power Over Coax



Applications

- Automotive Infotainment Systems
- Automotive Camera Interfaces
- Security and Surveillance
- Industrial and Medical Imaging

TI DS90Ux9xx FPD-Link III ICs

Serializers of media data to FPD-Link III (hint: last digit of IC name is odd)

- Supported video interfaces as **input**:
  - Parallel RGB/YUV, 10-bit, 12-bit, 14-bit, 18-bit, 24-bit
  - LVDS
  - HDMI
  - MIPI CSI-2
  - MIPI DSI

Deserializers from FPD-Link III to media data (hint: last digit of IC name is even)

- Supported video interfaces as **output**:
  - Parallel RGB/YUV, 10-bit, 12-bit, 14-bit, 18-bit, 24-bit
  - LVDS
  - MIPI CSI-2
- May serve as hubs

TI DS90Ux9xx FPD-Link III Serializer ICs

IC	Input format	Additional Features	Supported
DS90Ux901	Parallel 14-bit		No
DS90Ux913	Parallel 12-bit		Yes
DS90Ux921	Parallel 24-bit		No
DS90Ux925	Parallel 24-bit	HDCP, backward compatible	Yes
DS90Ux927	LVDS	HDCP	Yes
DS90Ux929	HDMI	HDCP	Yes
DS90Ux933	Parallel 12-bit	PoC	No
DS90Ux935	MIPI CSI-2	PoC	No
DS90Ux941	MIPI DSI	HDCP	No
DS90Ux947	LVDS	HDCP	Yes
DS90Ux949	HDMI	FPD-Link III Dual Lane	No
DS90Ux953	MIPI CSI-2	PoC	No

TI DS90Ux9xx FPD-Link III Deserializer ICs

IC	Output format	Additional Features	Supported
DS90Ux902	Parallel 14-bit		No
DS90Ux914	Parallel 12-bit		No
DS90Ux924	LVDS	backward compatible	No
DS90Ux926	Parallel 24-bit	HDCP, backward compatible	Yes
DS90Ux928	LVDS	HDCP, backward compatible	Yes
DS90Ux934	Parallel 12-bit	PoC	No
DS90Ux936	MIPI CSI-2	PoC, FPD-Link III Dual Lane	No
DS90Ux940	MIPI CSI-2	HDCP, FPD-Link III Dual Lane	Yes
DS90Ux948	LVDS	FPD-Link III Dual Lane	No
DS90Ux954	MIPI CSI-2	FPD-Link III Dual Lane	No
DS90Ux960	MIPI CSI-2	FPD-Link III Quad Lane	No
DS90Ux962	MIPI CSI-2	PoC, FPD-Link III Quad Lane	No
DS90Ux964	MIPI CSI-2	PoC, FPD-Link III Quad Lane	Yes

TI DS90Ux9xx IC features apart of Video Bridging

- I2S audio data bridging
- Supports bidirectional bridging of control interfaces:
  - I2C
  - GPIOs: direct control and GPIO signal bridging
  - Interrupts
  - Overview of TI DS90Ux9xx control interface bridging:

https://static.sched.com/hosted_files/ossalsjp18/8a/vzapolskiy_als2018.pdf

For simplicity and flexibility the implementation of device drivers to support TI DS90Ux9xx ICs is based on MFD framework, because IC subblocks are independent and reused over the IC series.

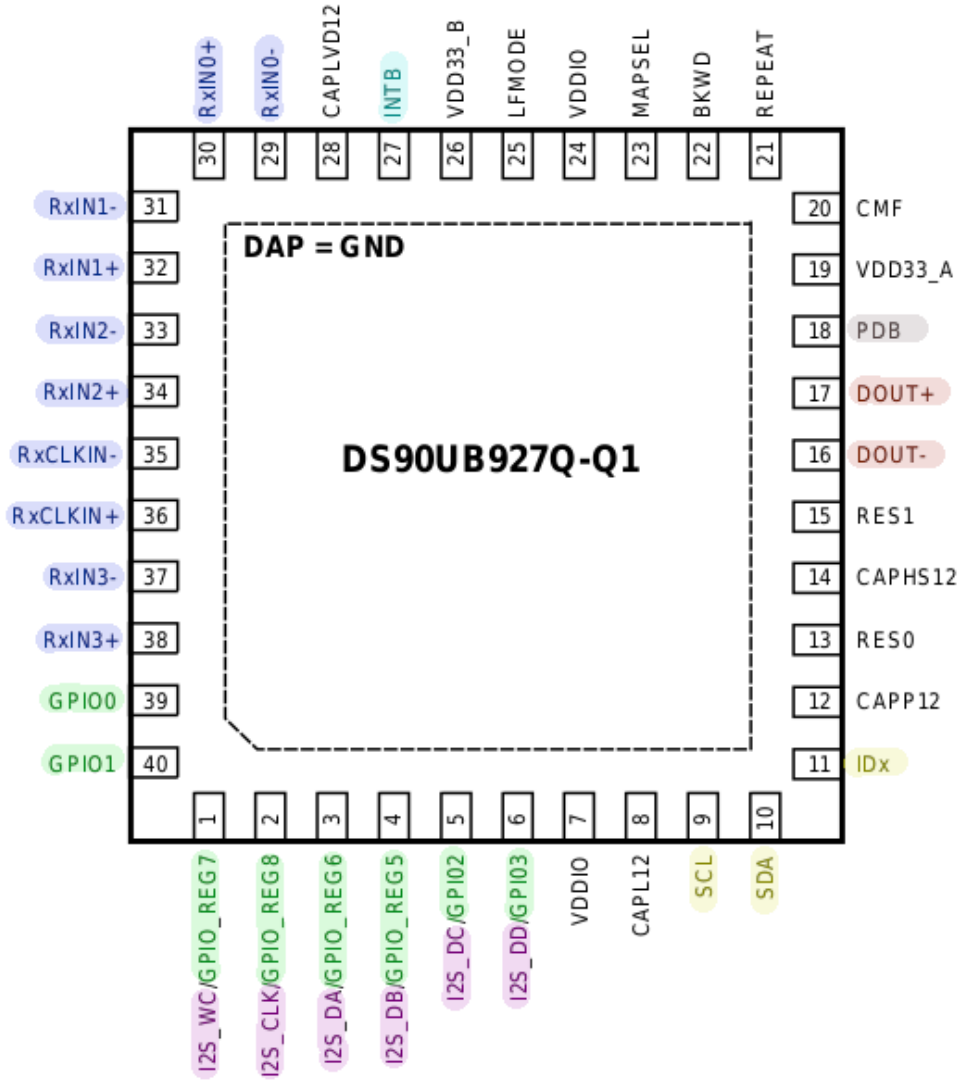
Why Multifunction Device Driver?

```
Subject: Re: [PATCH 4/7] mfd: ds90ux9xx: add TI DS90Ux9xx de-/serializer MFD driver
From: Vladimir Zapolskiy <vz@mleia.com>
To: Laurent Pinchart <laurent.pinchart@ideasonboard.com>

[snip]

The example naturally describes *two* simplistic boards in device tree
representation -- main board with an application SoC (ordinary i.MX6*)
and panel display module board. For demonstation I select a simple
FPD-Link III connection between two boards, note that significantly
more advanced configurations are also supported by the published
drivers, for example deliberately I skip audio bridging functionality.

The main board features:
* TI DS90UB927Q serializer (LVDS input) at 0xc, connected to SoC over I2C2,
  SoC GPIO5[10] signal is connected to the IC PDB pin,
* a status LED connected to DS90UB927Q GPIO2, it shall turn on,
  if FPD-Link III connection is established,
* TI DS90UB928Q GPIO0 line signal is pulled-up,
* TI DS90UB927Q GPIO3 line serves as generic GPIO, it is supposed to be
  controlled from userspace,
* TI DS90UB927Q INTB line is connected to SoC GPIO5[4], the line serves
  as an interrupt line routed from a touchscreen controller on a panel
  display module.

...
```

Why Multifunction Device Driver?

```
Subject: Re: [PATCH 4/7] mfd: ds90ux9xx: add TI DS90Ux9xx de-/serializer MFD driver
From: Vladimir Zapolskiy <vz@mleia.com>
To: Laurent Pinchart <laurent.pinchart@ideasonboard.com>

...

The panel display module board features:
* TI DS90UB928Q deserializer (LVDS output), *mapped* to have 0x3b address,
* AUO C070EAT01 panel,
* I2C EEPROM at 0x50, *mapped* to have 0x52 address on SoC I2C bus,
* Atmel MaxTouch touchscreen controller at 0x4b, *mapped* to have 0x60
  address on SoC I2C bus, power-up control signal is connected to DS90UB928Q GPIO4,
* a status LED connected to DS90UB928Q GPIO0, its on/off status shall
  repeat a user-defined status of DS90UB927Q GPIO0 on the main board,
* TI DS90UB928Q GPIO1 controls panel backlight, bridges DS90UB927Q
  GPIO1 signal level, which in turn is connected to a SoC controlled GPIO,
* TI DS90UB928Q GPIO2 line signal is pulled-up,
* TI DS90UB928Q GPIO3 line serves as generic GPIOs, it is supposed to be
  controlled from userspace.

All OF hard-coded controls like pinmuxing, I2C bridging of a remote
deserializer and I2C devices behind it, GPIO line state setting and so
forth must be applied with no interaction from a user -- and it just
works with the current / published versions of the drivers, in other
words a panel display module as a whole is truly hot-pluggable over
FPD-Link III connection.

[snip]
```

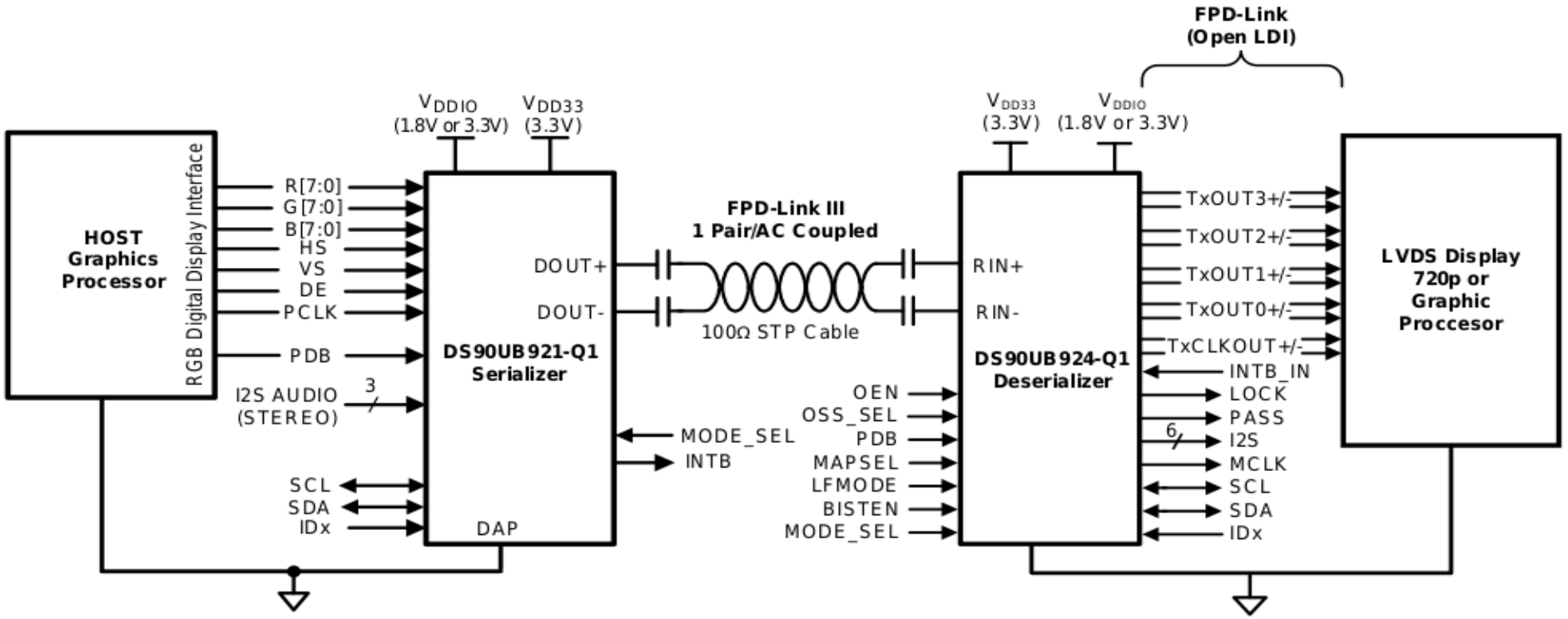
Why Multifunction Device Driver?




```
serializer: serializer@c {
        compatible = "ti,ds90ub927q", "ti,ds90ux9xx";
        reg = <0xc>;
        power-gpios = <&gpio5 10 GPIO_ACTIVE_HIGH>;

        i2c-bridge {
                compatible = "ti,ds90ux9xx-i2c-bridge";
                ti,i2c-bridges = <&deserializer 0 0x3b>;
                ti,i2c-bridge-maps = <0 0x4b 0x60>, <0 0x50 0x52>;
        };

        ds90ub927_pctrl: pin-controller {
                compatible = "ti,ds90ub927b-pinctrl", "ti,ds90ux9xx-pinctrl";
                gpio-controller;
                #gpio-cells = <2>;
                gpio-ranges = <&ds90ub927_pctrl 0 0 8>;

                led_pins: pinmux {
                        gpio-remote {
                                pins = "gpio2";
                                function = "gpio-remote";
                        };
                };
        };

        video-bridge {
                compatible = "ti,ds90ux9xx-video-bridge", "video-bridge";

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

                        port@0 {
                                reg = <0>;
                                ds90ub927_lvds: endpoint {
                                        remote-endpoint = <&lvds1_out>;
                                };
                        };

                        port@1 {
                                reg = <1>;
                                ds90ub927_fpd: endpoint {
                                        remote-endpoint = <&ds90ub928_fpd>;
                                };
                        };
                };
        };
};
```

Display Controller to FPD-Link III to Display Panel

- TI DS90Ux9xx deserializer and serializer ICs operate as **transparent** video bridges
- Video **format** conversion may happen
- Naturally maps into the Linux DRM framework as a chain of two video bridges



Display Controller to FPD-Link III to Display Panel

```
deserializer: deserializer {
        compatible = "ti,ds90ub928q", "ti,ds90ux9xx";

        i2c-bridge {
                compatible = "ti,ds90ux9xx-i2c-bridge";
                #address-cells = <1>;
                #size-cells = <0>;
        };

        video-bridge {
                compatible = "ti,ds90ux9xx-video-bridge", "video-bridge";

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

                        port@0 {
                                reg = <0>;
                                ds90ub928_fpd: endpoint {
                                        remote-endpoint = <&ds90ub927_fpd>;
                                };
                        };

                        port@1 {
                                reg = <1>;
                                ds90ub928_output: endpoint {
                                        remote-endpoint = <&panel_input>;
                                };
                        };
                };
        };
};
```

```
&ldb {
        status = "okay";

        lvds-channel@1 {
                status = "okay";
                fsl,data-mapping = "jeida";
                fsl,data-width = <24>;

                port@4 {
                        reg = <4>;

                        lvds1_out: endpoint {
                                remote-endpoint = <&ds90ub927_lvds>;
                        };
                };
        };
};

...

serializer: serializer@c {
        compatible = "ti,ds90ub927q", "ti,ds90ux9xx";
        reg = <0xc>;

        i2c-bridge {
                compatible = "ti,ds90ux9xx-i2c-bridge";
                ti,i2c-bridges = <&deserializer 0 0x3b>;
                ti,i2c-bridge-maps = <0 0x4b 0x60>, <0 0x50 0x52>;
        };

        video-bridge {
                compatible = "ti,ds90ux9xx-video-bridge", "video-bridge";

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

                        port@0 {
                                reg = <0>;
                                ds90ub927_lvds: endpoint {
                                        remote-endpoint = <&lvds1_out>;
                                };
                        };

                        port@1 {
                                reg = <1>;
                                ds90ub927_fpd: endpoint {
                                        remote-endpoint = <&ds90ub928_fpd>;
                                };
                        };
                };
        };
};
```

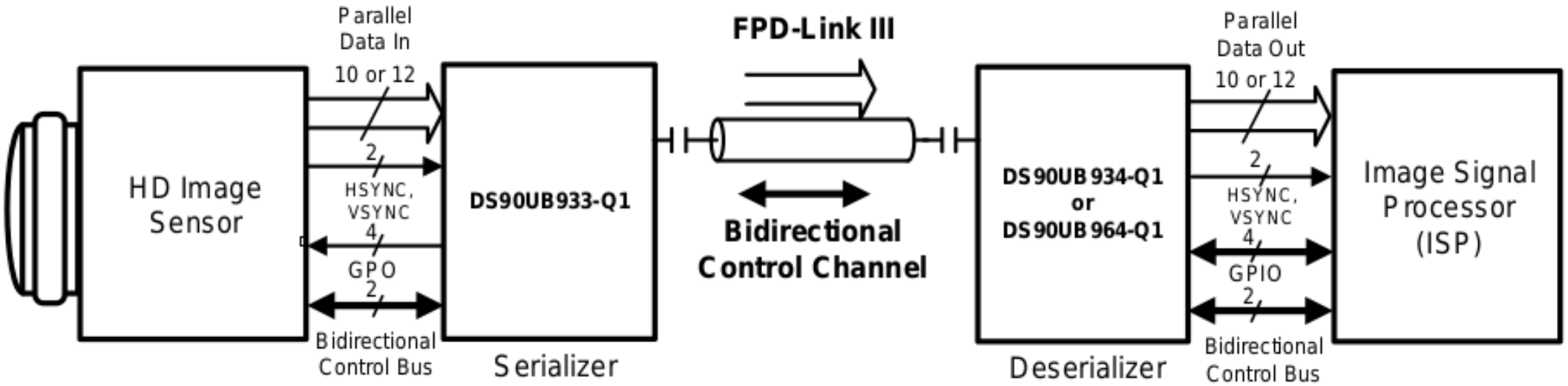
DRM driver for FPD-Link III serializers

- Nothing special to implement, because there is no controls the only necessary action is to setup a link, a trivial derivative of `drivers/gpu/drm/bridge/lvds-encoder.c` is sufficient

```
panel_node = of_graph_get_remote_port_parent(endpoint);
of_node_put(endpoint);
if (!panel_node) {
        dev_dbg(dev, "no remote endpoint for port 1\n");
        return -ENXIO;
}

panel = of_drm_find_panel(panel_node);
of_node_put(panel_node);
if (IS_ERR(panel)) {
        dev_dbg(dev, "panel not found, deferring probe\n");
        return PTR_ERR(panel);
}

lvds_encoder->panel_bridge =
        devm_drm_panel_bridge_add(dev, panel, DRM_MODE_CONNECTOR_LVDS);
if (IS_ERR(lvds_encoder->panel_bridge))
        return PTR_ERR(lvds_encoder->panel_bridge);
```

Camera Sensor to FPD-Link III to Image Signal Controller

- TI DS90Ux9xx serializer and deserializer ICs operate as **transparent** video bridges
- Video **format** conversion may happen
- Naturally maps into the Linux V4L2 framework as a chain of two video bridges



Camera Sensor to FPD-Link III to Image Signal Controller

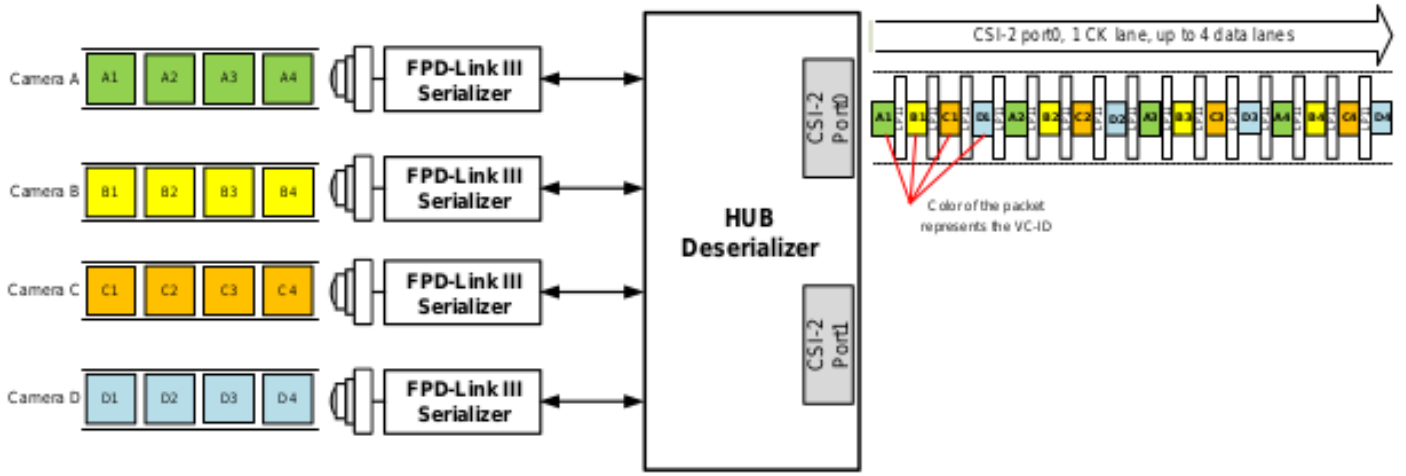


Figure 24. Four Camera Data onto CSI-2 With Virtual Channels (VC-ID)

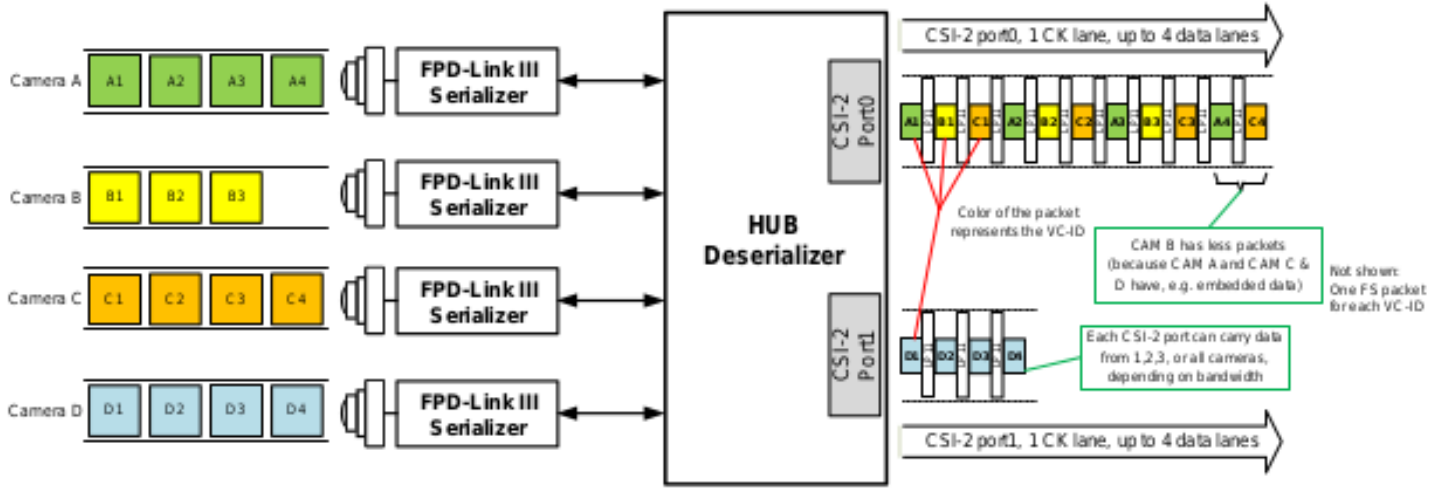


Figure 25. Four Camera Data onto CSI-2 With Virtual Channels (VC-ID) With Different Frame Size

Camera Sensor to FPD-Link III to Image Signal Controller

```
external-pcbs {
        camera_0: pcb-camera-0 {
                compatible = "ti,ds90ub913q", "ti,ds90ux9xx";

                i2c-bridge {
                        compatible = "ti,ds90ux9xx-i2c-bridge";
                        #address-cells = <1>;
                        #size-cells = <0>;

                        camera-sensor@30 {
                                compatible = "ovti,ov10635";
                                reg = <0x30>;
                        };
                };
        };

        camera_1: pcb-camera-1 {
                compatible = "ti,ds90ub913q", "ti,ds90ux9xx";

                i2c-bridge {
                        compatible = "ti,ds90ux9xx-i2c-bridge";
                        #address-cells = <1>;
                        #size-cells = <0>;

                        camera-sensor@30 {
                                compatible = "ovti,ov10635";
                                reg = <0x30>;
                        };
                };
        };

        ...
};
```

```
deserializer@30 {
        compatible = "ti,ds90ub964q", "ti,ds90ux9xx";
        reg = <0x30>;
        #address-cells = <1>;
        #size-cells = <0>;

        i2c-bridge@31 {
                compatible = "ti,ds90ux9xx-i2c-bridge";
                reg = <0x31>;
                ti,i2c-bridges = <&camera_0 0 0x70>;
                ti,i2c-bridge-maps = <0 0x30 0x71>;
        };

        i2c-bridge@32 {
                compatible = "ti,ds90ux9xx-i2c-bridge";
                reg = <0x32>;
                ti,i2c-bridges = <&camera_1 1 0x72>;
                ti,i2c-bridge-maps = <1 0x30 0x73>;
        };

        ...

        video-bridge {
                compatible = "ti,ds90ub9xx-mipi-csi2";

                port {
                        ds90ub964_1: endpoint {
                                remote-endpoint = <&csi2_40_0_in>;
                                data-lanes = <1 2 3 4>;
                                clock-lanes = <0>;
                        };
                };

                input_0 {       /* Image from 4 cameras */
                        id = <0x01>;
                        fieldtype = <1>;
                        resolution = <1280 4000>;
                        frames = <30>;
                        ofmt = <0x2006>;
                };
        };
};
```

V4L2 driver for FPD-Link III deserializers

```
static const struct v4l2_subdev_core_ops ds90ux940_core_ops = {
        .s_power = ds90ux940_s_power,
        .reset = ds90ux940_reset,
        .subscribe_event = ds90ux940_subscribe_event,
        .unsubscribe_event = v4l2_event_subdev_unsubscribe,
};

static const struct v4l2_subdev_video_ops ds90ux940_video_ops = {
        .s_parm = ds90ux940_s_parm,
        .g_parm = ds90ux940_g_parm,
        .g_input_status = ds90ux940_g_input_status,
        .enuminput = ds90ux940_enuminput,
        .g_input = ds90ux940_g_input,
        .s_input = ds90ux940_s_input,
        .s_routing = ds90ux940_s_routing,
        .querystd = ds90ux940_querystd,
        .g_mbus_config  = ds90ux940_g_mbus_config,
        .s_stream = ds90ux940_s_stream,
};

static const struct v4l2_subdev_ops ds90ux940_subdev_ops = {
        .core = &ds90ux940_core_ops,
        .video = &ds90ux940_video_ops,
        .pad = &ds90ux940_pad_ops,
};
```

```
static const struct v4l2_async_notifier_operations ds90ux940_notify_ops = {
        .bound = ds90ux940_notify_bound,
        .unbind = ds90ux940_notify_unbind,
};

static const struct v4l2_ctrl_ops ds90ux940_ctrl_ops = {
        .s_ctrl = ds90ux940_s_ctrl,
};

static const struct v4l2_ctrl_config ds90ux940_controls[] = {
        {
                .ops = &ds90ux940_ctrl_ops,
                .id = V4L2_CID_PIXEL_RATE,
                .type = V4L2_CTRL_TYPE_INTEGER64,
                .name = "Frequency control",
                .def = 0,
                .min = 0,
                .max = INT_MAX,
                .step = 1,
        },
};

static const struct media_entity_operations ds90ux940_media_ops = {
        .link_setup = ds90ux940_link_setup,
};
```

Symmetric or asymmetric?

- A pair of TI DS90Ux9xx serializer and deserializer connected over FPD-Link III can be seen as a transparent bridge
- PCBs over FPD-Link III connection can be extremely sophisticated devices, also these devices are unpluggable / replaceable, apparently it is unavoidable to describe these devices separately from main board device trees, DT overlay mechanism works perfectly
- Serializer and deserializer pair is “reversible”, from hardware perspective there is no significant difference in video bridging of a video signal from a display controller or to an image controller
- Still, looking from Linux running on an application SoC side, the difference between DRM and V4L2 frameworks shall be taken into account

Audio bridging

- Similar to video bridging there is no explicit controls of DS90Ux9xx audio bridging
- TI rightfully considers that audio briding functionality might be optional in products, thus it is pinmuxed with GPIOs, hence audio bridging support and control is implemented as pinmux configuration
- To get a more flexible control audio device tree graphs can be considered
- v4l2_audio might be a reasonable option to choose for advanced implementation, however the only known audio control given by IC is mute/unmute

```
ds90ux925_0_pctrl: pin-controller {
        compatible = "ti,ds90ux9xx-pinctrl";
        gpio-controller;
        #gpio-cells = <2>;
        gpio-ranges = <&ds90ux925_0_pctrl 0 0 9>;

        pinctrl-names = "default", "alt";
        pinctrl-0 = <&ds90ux925_0_pins>;

        ds90ux925_0_pins: pinmux {
                i2s {
                        groups = "i2s-1";
                        function = "i2s-1";
                };

                parallel {
                        groups = "parallel";
                        function = "parallel";
                };
        };
};
```

Similarities with other multimedia serial links

```
Re: [PATCH 4/7] mfd: ds90ux9xx: add TI DS90Ux9xx de-/serializer MFD driver
From: Kieran Bingham @ 2018-10-12 11:47 UTC

[snip]

The use case whether they transfer frames from a camera or to a display
are of course closely related, but ultimately covered by two separate
subsystems at the pixel level (DRM vs V4L, or other for other data)

Perhaps as they are buses - on a level with USB or I2C (except they can
of course carry I2C or Serial as well as 'bi-directional video' etc ),
they are looking for their own subsystem.

Except I don't think we don't want to add a new subsystem for just one
(or two) devices...
```

- Maxim GMSL
- TI FPD-Link III
- SMSC/Microchip MOST
- Inova APIX
- ...

Current status

- The essential parts of DS90Ux9xx device drivers is sent for review, testing and discussion
- Luca Ceresoli <luca@lucaceresoli.net> sent his own device driver for DS90UB954 support
- The cornerstone of uncertainty is how to describe I2C bridging functionality
  - Multiple points of viewson the best possible device tree representation of the ICs
- All features of ICs are supported in the drivers framework and shipped to products

Linux device driver	Status
MFD	All features are supported, the driver is published
I2C bridge	All features are supported, the driver is published and under discussion
IRQ bridge	Trivial driver, not yet published though
Pinmux	All features are supported, the driver is published
GPIO	All features are supported, the driver is published
GPIO bridge	All features are supported, the driver is published
DRM	All features are supported, terivial driver based on `gpu/drm/bridge/lvds-encoder.c`
V4L2	Non-trivial driver, DS90Ux940 and DS90Ux964 are supported in unpublished drivers
Audio bridge	No separate driver, part of pinmux

Thank you for your attention!

Questions and comments are welcome.