Demystifying Linux MIPI-DSI Subsystem

Jagan Teki - CEO | Embedded Linux Engineer, Amarula Solutions

@JaganTeki
Jagan Teki

→ Embedded Linux Engineer at Amarula Solutions
  ◆ Bootloader: BootROM, bootloaders, U-Boot, boot bsp, chip/board bring ups, devicetrees, device drivers, boottime, secure boot, atf, optee and etc.
  ◆ Embedded Linux: Linux bsp, devicetrees, device drivers, multimedia, optimizations, integrations and etc

→ Mainline contributions
  ◆ Linux
    ● Contributor of Allwinner, Rockchip, i.MX platforms, bsp, device drivers.
    ● Maintainer of few DSI LCD panels.
  ◆ U-Boot
    ● Contributor of Xilinx Zynq, Allwinner, Rockchip, i.MX platforms, bsp, device drivers.
    ● Maintainer of Allwinner sunXi SoCs
    ● Maintainer of SPI/SPI-NOR Subsystems
  ◆ Contributor of Buildroot, Yocto
This talk is about?

➔ How MIPI-DSI is different than other display interfaces.
➔ How to incorporate MIPI-DSI drivers in to Linux DRM subsystem.
➔ Identify the vendor owned DSI bridges, panels.
➔ How to write and interact DSI controller, bridges and panel.
➔ Brief overview of DRM/KMS core.
➔ Explaining the common factors required for setting up display pipeline for DSI components.
➔ Sharing my experience while bringing up several types of DSI interface panels.
➔ Open to correct me, I'm not so expert.
Agenda

Display interfaces
➔ In a Nutshell
➔ MIPI-DSI protocol

Linux DRM
➔ Subsystem overview
➔ DRM/KMS core
➔ DRM DSI core
➔ DRM Bridge core
➔ Sample DRM drivers
➔ Sample DSI panel, bridge drivers
➔ Display pipeline setup

MIPI-DSI experience
➔ Tips to develop DRM/DSI drivers
➔ How to validate them via graphics libraries
Display interface
Display interfaces, In a NutShell

PC-compatible

Embedded SoCs
Display interfaces types

Parallel RGB
Configuration usually has a full data width, but no address bus

LVDS
Low-voltage differential signaling
Diffetial, serial communication protocol

MIPI-DSI
Display Serial Interface, via MIPI standard
High performance, low power

HDMI
Uncompressed digital video and audio, with differential TMDS

eDP
Embedded Displayport, high-performance external audio/visual
MIPI-DSI: DSI panel

Lane is differential pin, so each lane has 2 differential pins
D-PHY data rate: 8Mbps 2.5Gbps
Lane is differential pin, so each lane has 2 differential pins
D-PHY data rate: 8Mbps 2.5Gbps
DSI Operating modes

➔ Command mode
  ✦ Bi-directional
  ✦ write to, and read from, the registers and frame buffer.
  ✦ simple simple command interface.

➔ Video mode
  ✦ Uni-directional
  ✦ transfers in the form of real-time pixels
  ✦ high speed mode of transfer

➔ Video mode has
  ✦ Non-Burst Mode with Sync Pulses
    ● video mode with sync pulse width.
  ✦ Non-Burst Mode with Sync Events
    ● video mode with sync events.
  ✦ Burst mode
    ● pixel packets are time-compressed
    ● multiplexing the transmission on the link
DSI Packet format

➔ Short packet

<table>
<thead>
<tr>
<th>SOT</th>
<th>DATAID</th>
<th>DATA 0</th>
<th>DATA 1</th>
<th>ECC</th>
<th>EOT</th>
</tr>
</thead>
</table>

DATAID: Data ID/Command.
0x05, DCS Short Write, no parameter
0x03, Generic Short Write, no parameter
2 bytes fixed data size.

➔ Long packet

<table>
<thead>
<tr>
<th>SOT</th>
<th>DATAID</th>
<th>DATA 0</th>
<th>DATA 1</th>
<th>ECC</th>
<th>EOT</th>
</tr>
</thead>
</table>

DATAID: Data ID/Command.
0x39, DCS Long Write
0x29, Generic Long Write
No fixed data size.
Linux DRM Subsystem

- DRM Core
  - DRM DSI Core (drm_mipi_dsi.c)
  - DRM Panel Core (drm_panel.c)
  - DRM Bridge Core (drm_bridge.c)

- DRM Core Components:
  - libdrm
  - OpenGL
  - OpenCL
  - Mesa
  - libmali
  - /dev/fbX
  - /dev/maliX
  - /dev/dri/cardX

- DRM Core Drivers:
  - Allwinner DE driver
    - sun4i_drv.c
  - Rockchip DE driver
    - rockchip_drv.c
  - Allwinner DW HDMI
    - sun8i_dw_hdmic
  - Mali-400
    - mali_kernel_linux.c
  - Allwinner MIPI-DSI
    - sun6i_mipi_dsi.c
  - Simple Panel driver
    - panel-simple.c
  - Simple lvds driver
    - panel-lvds.c
  - DSI panel
    - s070wv20-icn6211.c
  - MIPI-DSI to RGB bridge
    - chipone_icn6211.c

- DRM Core Interfaces:
  - drm_dev_register
  - drm_panel_add
  - drm_bridge_add
  - mipi_dsi_host_register

- DRM Core Integration:
  - Qt5
  - X11
  - Wayland

- DRM Core Dependencies:
  - libdrm
  - OpenGL
  - OpenCL
  - Mesa
  - libmali
  - /dev/fbX
  - /dev/maliX
  - /dev/dri/cardX

Note: DRM DSI Core, DRM Panel Core, DRM Bridge Core here are part of DRM Core itself just to mentioned here explicitly for the sake of understanding.
DRM Core: TTM/GEM

DRM Driver

KMS

Planes

HDMI Encoder

DSI Connector

HDMI Connector

DSI Encoder

DPI Encoder

LVDS Connector

LVDS Encoder

Allwinner DE driver

sun4i_drv.c

Allwinner DW HDMI

sun6i_dw_hdmi.c

Allwinner MIPI-DSI

sun6i_mipi_dsi.c

Allwinner RGB

sun4i_rgb.c

Allwinner LVDS

sun4i_lvds.c

CRTC

sun4i_crtc.c

drm framebuffer1

drm framebuffer2

KMS

DRM Driver

GEM

Display Engine/Controller

RAM

FB CMA Driver

Amarula Solutions - Embedded | Hardware | Open Source
Sample DE driver

```c
static struct drm_driver sun4i_drv_driver = {
    .driver_features = DRIVER_GEM | DRIVER_MODESET | DRIVER_PRIME | DRIVER_ATOMIC,
    .fops = &sun4i_drv_fops,
    .name = "sun4i-drm",
    .desc = "Allwinner sun4i Display Engine",
};
static int sun4i_drv_bind(struct device *dev)
{
    drm_kms_helper_poll_init(drm);
    drm_dev_register(drm, 0);
    drm_fbdev_generic_setup(drm, 32);
}
```

```c
struct sun4i_tcon_quirks {
    bool has_channel_0; /* a83t does not have channel 0 on second TCON */
    bool has_channel_1; /* a33 does not have channel 1 */
    bool needs_de_be_mux; /* sun6i needs mux to select backend */
    bool needs_edp_reset; /* a80 edp reset needed for tcon0 access */
    bool supports_lvds; /* Does the TCON support an LVDS output? */

    /* callback to handle tcon muxing options */
    int (*set_mux)(struct sun4i_tcon *, const struct drm_encoder *);
};
static int sun4i_drv_bind(struct device *dev)
{
    sun4i_tcon_find_engine(drv, dev->of_node);
    sun4i_tcon_init_clocks(dev, tcon);
    sun4i_tcon_init_regmap(dev, tcon);
}
```
Sample DE driver, CRTC

```c
struct drm_crtc_funcs {
    int (*enable_vblank)(struct drm_crtc *crtc);
    void (*disable_vblank)(struct drm_crtc *crtc);
    ...
};

struct sun4i_crtc *sun4i_crtc_init(struct drm_device *drm, struct sunxi_engine *engine,
    struct sun4i_tcon *tcon)
{
    sunxi_engine_layers_init(drm, engine);
    drm_crtc_init_with_planes(drm, &scrtc->crtc, primary, cursor, &sun4i_crtc_funcs, NULL);
}
```
Sample MIPI-DSI driver

```c
struct sun6i_dsi {
    struct drm_connector    connector;
    struct drm_encoder     encoder;
    struct mipi_dsi_host   host;
    struct mipi_dsi_device *device;
    struct drm_panel       *panel;
    struct drm_bridge      *bridge;
};
struct mipi_dsi_host_ops {
    ssize_t (*transfer)(struct mipi_dsi_host *host, const struct mipi_dsi_msg *msg);
};
static ssize_t sun6i_dsi_transfer(struct mipi_dsi_host *host, const struct mipi_dsi_msg *msg)
{
    switch (msg->type) {
        case MIPI_DSI_DCS_SHORT_WRITE:
            /* short dsi transfer */
            break;
        case MIPI_DSI_DCS_LONG_WRITE:
            /* long dsi transfer */
            break;
    }
}
static int sun6i_dsi_bind(struct device *dev, struct device *master, void *data)
{
    drm_encoder_init(drm, &dsi->encoder, &sun6i_dsi_enc_funcs, DRM_MODE_ENCODER_DSI, NULL);
    drm_connector_init(drm, &dsi->connector, &sun6i_dsi_connector_funcs, DRM_MODE_CONNECTOR_DSI);
    drm_panel_attach(dsi->panel, &dsi->connector);
    drm_bridge_attach(&dsi->encoder, dsi->bridge, NULL);
}
static int sun6i_dsi_probe(struct platform_device *pdev)
{
    mipi_dsi_host_register(&dsi->host);
}
```

Amarula Solutions - Embedded | Hardware | Open Source
struct s070wv20 {
    struct drm_panel panel;
    struct mipi_dsi_device *dsi;
};
static const struct drm_panel_funcs {};
static int s070wv20_prepare(struct drm_panel *panel) {
    __s070wv20_prepare(panel);
}
static int s070wv20_enable(struct drm_panel *panel) {
    miipi_dsi_dcs_set_display_on(ctx->dsi);
}
static int s070wv20_disable(struct drm_panel *panel) {
    miipi_dsi_dcs_set_display_off(ctx->dsi);
}
static int s070wv20_unprepare(struct drm_panel *panel) {
    miipi_dsi_dcs_enter_sleep_mode(ctx->dsi);
}
static int s070wv20_get_modes(struct drm_panel *panel) {
    /* get drm_display_mode, clock, hdisplay, vdisplay etc */
}
static int s070wv20_dsi_probe(struct miipi_dsi_device *dsi) {
    /* get power, reset gpio, backlight */
    drm_panel_add(&ctx->panel);
    dsi->mode_flags = MIPI_DSI_MODE_VIDEO_SYNC_PULSE;
    dsi->format = MIPI_DSI_FMT_RGB888;
    dsi->lanes = 4;
    miipi_dsi_attach(dsi);
}

static inline int s070wv_dsi_write(struct chipone *icn, const void *seq, size_t len) {
    struct miipi_dsi_device *dsi = to_miipi_dsi_device(icn->dev);
    return miipi_dsi_generic_write(dsi, seq, len);
}

#define S070WV20_DSI(icn, seq...)
    {
        const u8 d[] = { seq...};
        s070wv_dsi_write(icn, d, ARRAY_SIZE(d));
    }

Static void __s070wv20_prepare(struct drm_panel *panel) {
    /* lower 8 bits of hdisplay */
    S070WV20_DSI(icn, 0x20, mode->hdisplay & 0xff);
    /* lower 8 bits of vdisplay */
    S070WV20_DSI(icn, 0x21, mode->vdisplay & 0xff);
    /**
     * lsb nibble: 2nd nibble of hdisplay
     * msb nibble: 2nd nibble of vdisplay
     */
    S070WV20_DSI(icn, 0x22, (((mode->hdisplay >> 8) & 0xf) |
                             (((mode->vdisplay >> 8) & 0xf) << 4)));
    /* HFP */
    S070WV20_DSI(icn, 0x23, mode->hsync_start - mode->hdisplay);
    /* HSYNC */
    S070WV20_DSI(icn, 0x24, mode->hsync_end - mode->hsync_start);
    /* HBP */
    S070WV20_DSI(icn, 0x25, mode->htotal - mode->hsync_end);
}
DRM Bridge Core: DSI-RGB bridge

- DRM Core
  - DRM DSI Core (drm_mipi_dsi.c)
    - Allwinner MIPI-DSI
      - sun6i_mipi_dsi.c
    - Mali-400
      - mali_kernel_linux.c
  - DRM Panel Core (drm_panel.c)
    - Simple Panel driver
      - panel-simple.c
    - ICN6211 MIPI-DSI to RGB Bridge
      - chipone-icn6211.c
  - DRM Bridge Core (drm_bridge.c)
    - ICN6211
      - 40-pin FPC
      - MIPI-DSI to RGB Converter

- OpenES
  - libmali
  - libdrm
  - Mesa

- /dev/fbX
- /dev/maliX
- /dev/dri/cardX

- DRM Bridge Core: DSI-RGB bridge

- DRM Core

- DRM DSI Core (drm_mipi_dsi.c)
  - Allwinner DE driver
    - sun4i_drv.c
  - Mali GPU Controller
  - Allwinner MIPI-DSI Controller

- DRM Panel Core (drm_panel.c)
  - Simple Panel driver
    - panel-simple.c
  - S070VV20_CT16 Parallel RGB Ctrl

- DRM Bridge Core (drm_bridge.c)
  - ICN6211
    - MIPI-DSI to RGB Convtr

- Amarula Solutions - Embedded | Hardware | Open Source
Sample MIPI-DSI to RGB bridge driver

```c
struct chipone {
    struct device *dev;
    struct drm_bridge bridge;
    struct drm_connector connector;
    struct drm_panel *panel;
};
static const struct drm_bridge_funcs {};
static int chipone_attach(struct drm_bridge *bridge)
{
    drm_connector_init(drm, &icn->connector, &chipone_connector_funcs, DRM_MODE_CONNECTOR_DPI);
    drm_panel_attach(icn->panel, &icn->connector);
}
static void chipone_enable(struct drm_bridge *bridge)
{
    drm_panel_enable(icn->panel);
}
static void chipone_pre_enable(struct drm_bridge *bridge)
{
    icn6211_bridge_init(bridge);
}
static int chipone_probe(struct mipi_dsi_device *dsi)
{
    drm_bridge_add(&icn->bridge);
    dsi->mode_flags = MIPI_DSI_MODE_VIDEO_SYNC_PULSE;
    dsi->format = MIPI_DSI_FMT_RGB888;
    dsi->lanes = 4;
    miapi_dsi_attach(dsi);
}
```
Display pipeline: DSI

```c
/ {
    panel {
        compatible = "bananapi,s070wv20-ct16", "simple-panel";
        backlight = <&backlight>;
        port {
            panel_out_bridge: endpoint {
                remote-endpoint = <&bridge_out_panel>;
            ];
        ];
    ];
    }
    &dsi {
        status = "okay";
        ports {
            dsi_out: port@0 {
                reg = <0>;
                dsi_out_bridge: endpoint {
                    remote-endpoint = <&bridge_out_dsi>;
                ];
            ];
        ];
    ];
    }
    bridge@0 {
        compatible = "chipone,icn6211";
        reg = <0>;
        ports {
            bridge_in: port@0 {
                reg = <0>;
                bridge_out_dsi: endpoint {
                    remote-endpoint = <&dsi_out_bridge>;
                ];
            ];
        ];
        }
        bridge_out: port@1 {
            reg = <1>;
            bridge_out_panel: endpoint {
                remote-endpoint = <&panel_out_bridge>;
            ];
        ];
    ];
    }
    &dsi {
        status = "okay";
        ports {
            #address-cells = <1>;
            #size-cells = <0>;
            dsi_out: port@0 {
                reg = <0>;
                dsi_out_panel: endpoint {
                    remote-endpoint = <&panel_out_dsi>;
                ];
            ];
        ];
    ];
    }
    panel@0 {
        compatible = "bananapi,s070wv20-ct16-icn6211";
        reg = <0>;
        backlight = <&backlight>;
        port {
            panel_out_dsi: endpoint {
                remote-endpoint = <&dsi_out_panel>;
            ];
        ];
    ];
    }
};
```
MIPI-DSI Experience
How to develop DRM/DSI drivers

➔ Controller hacks:
  ◆ Identify controller datasheet, check the regmap, lcd mode timings.
  ◆ do reverse engineering the bsp for regmap, if no datasheets.

➔ Panel hacks:
  ◆ check the IC of the panel
    • does the IC and panel are with same vendor?
    • does the IC and panel are from different vendors?
    • does the IC is bridge controller?

➔ Sample panel drivers:
  ◆ panel-feiyang-fy07024di26a30d.c - IC and panel are from same vendor
  ◆ panel-sitronix-st7701.c - IC is from sitronix with ts8550b is DSI panel from Techstar
  ◆ chipone-icn6211.c - Bridge IC is Chipone for DSI to RGB converter.

➔ Vendor panel initialization code, can be critical if we don’t have any programming datasheet, or bsp code.
How to develop GPU drivers, testing

➔ GPU hacks:
  ◆ get the gpu userspace libraries, libmali
  ◆ get the kernel gpu drivers
    ● does it part of existing/mainline kernel?
    ● does it part of vendor libraries? do reverse-engineering and compile them as modules.

➔ Sample Allwinner Mali-400 GPU drivers and libmali
  ◆ https://github.com/mripard/sunxi-mali.git
  ◆ available in mainline buildroot, to compatible with mainline Linux.

➔ Sample Rockchip Mali-T76x/86x GPU drivers and libmali
  ◆ https://github.com/openedev/rockchip_forwardports
  ◆ libmali available in mainline buildroot.

➔ Tested hacks:
  ◆ try CONFIG_LOGO
  ◆ run sample qt5 or any simple graphic application
  ◆ try some complex graphic run, mesa, glmark2-es2
  ◆ try X11, Wayland
References

➔ Working experience with Allwinner Display controllers and vendor panels, bridges

➔ Specification for Display Serial Interface (DSI) version 1.3


➔ An introduction to Linux DRM Subsystem - Maxime Ripard
  https://www.slideshare.net/ennael/kernel-recipes-2017-an-introduction-to-the-linux-drm-subsystem-maxime-ripard
Thank You, Questions?

jagan@amarulasolutions.com