Rockchip RK3588 DP 软件开发指南

文件标识：RK-KF-YF-466

发布版本：V1.0.0

日期：2022-05-26

文件密级：□绝密 □秘密 □内部资料 ■公开

瑞芯微电子股份有限公司

Rockchip Electronics Co., Ltd.

地址： 福建省福州市铜盘路软件园A区18号

网址： www.rock-chips.com

客户服务电话： +86-4007-700-590

客户服务传真： +86-591-83951833

客户服务邮箱： fae@rock-chips.com

前言

本文主要介绍 Rockchip RK3588 平台 DP 的使用与调试方法。

产品版本

芯片名称	内核版本
RK3588	LINUX Kernel 5.10

读者对象

本文档（本指南）主要适用于以下工程师：

技术支持工程师

软件开发工程师

修订记录

版本号	作者	修改日期	修改说明
V1.0.0	张玉炳	2022-05-26	初始版本

目录

RK3588 平台 DP 简介

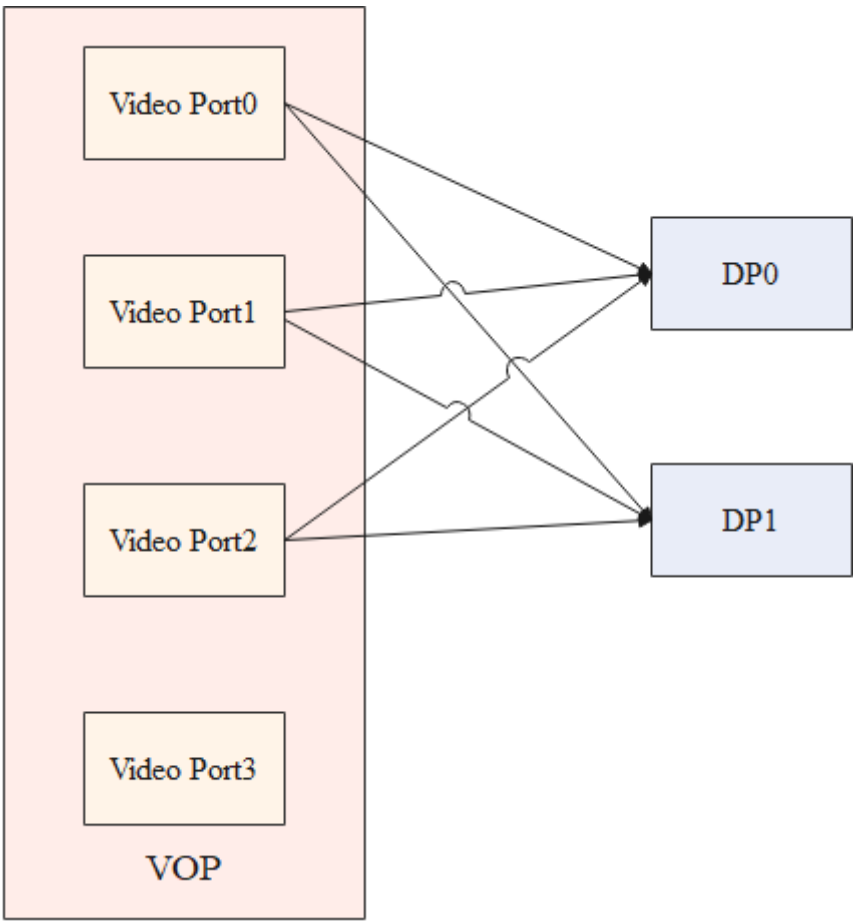
功能特性

RK3588 的 DP 支持 1.4a 版本的 DP 协议，最高的输出分辨率可达到 8K@30Hz, 最高的 PHY 链路速率可以达到 8.1Gbps/lane， 具体的特性描述如下表格

功能	RK3588
Version	1.4a
SST	Support
MST	Not support
DSC	Not support
Max resolution	8K@30Hz
Main-Link lanes	1/2/4 lanes
Main-Link rate	8.1/5.4/2.7/1.62 Gbps/lane
AUX_CH	1 M
Color Format	RGB/YUV444/YUV422/YUV420
Color Depth	8/10 bit(6bit just for RGB)
Display Split Mode	Support
HDCP	HDCP2.2/HDCP1.3
Type-C support	DP Alternate Mode
I2S	Support
SPDIF	Support

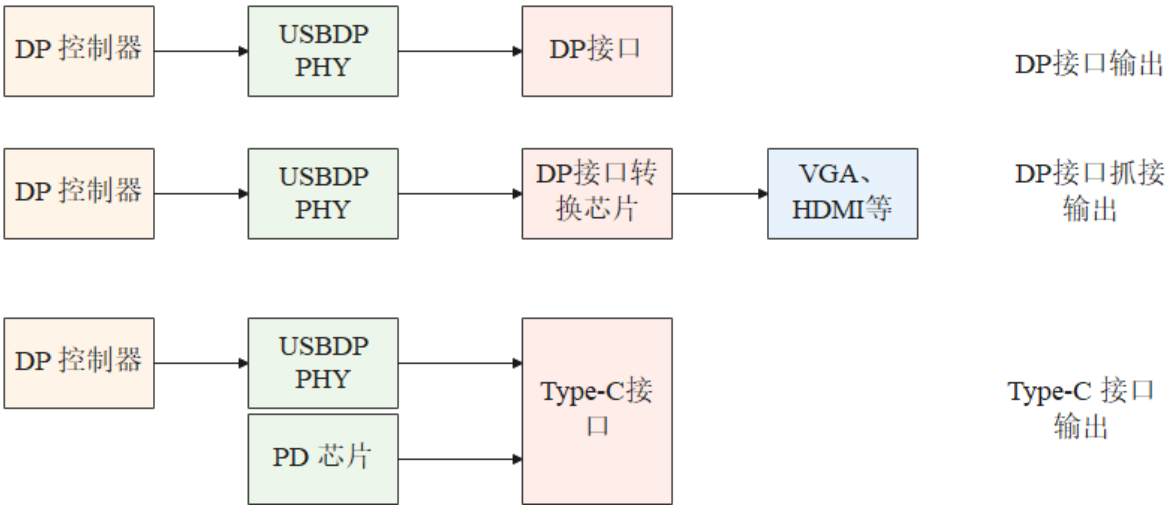
DP 输入

RK3588 的 VOP 有四个 Video Port，两个 DP 控制器，其中只有 Video Port 0/1/2 可以输出到 DP0/1，如下图。



DP 输出

根据应用场景的不同，可以设计不同的 DP 输出方式：Type-C 接口输出、DP 标准接口输出、通过其他转接芯片转接输出。



代码路径

U-Boot 驱动代码：

```
drivers/video/drm/dw-dp.c
drivers/phy/phy-rockchip-usbdp.c
```

Kernel 驱动代码：

```
drivers/gpu/drm/rockchip/dw-dp.c
drivers/phy/rockchip/phy-rockchip-usbdp.c
```

参考 DTS 配置：

```
arch/arm64/boot/dts/rockchip/rk3588-evb1-lp4.dtsi
arch/arm64/boot/dts/rockchip/rk3588-evb2-lp4.dtsi
arch/arm64/boot/dts/rockchip/rk3588-evb3-lp5.dtsi
arch/arm64/boot/dts/rockchip/rk3588-nvr-demo.dtsi
```

功能配置

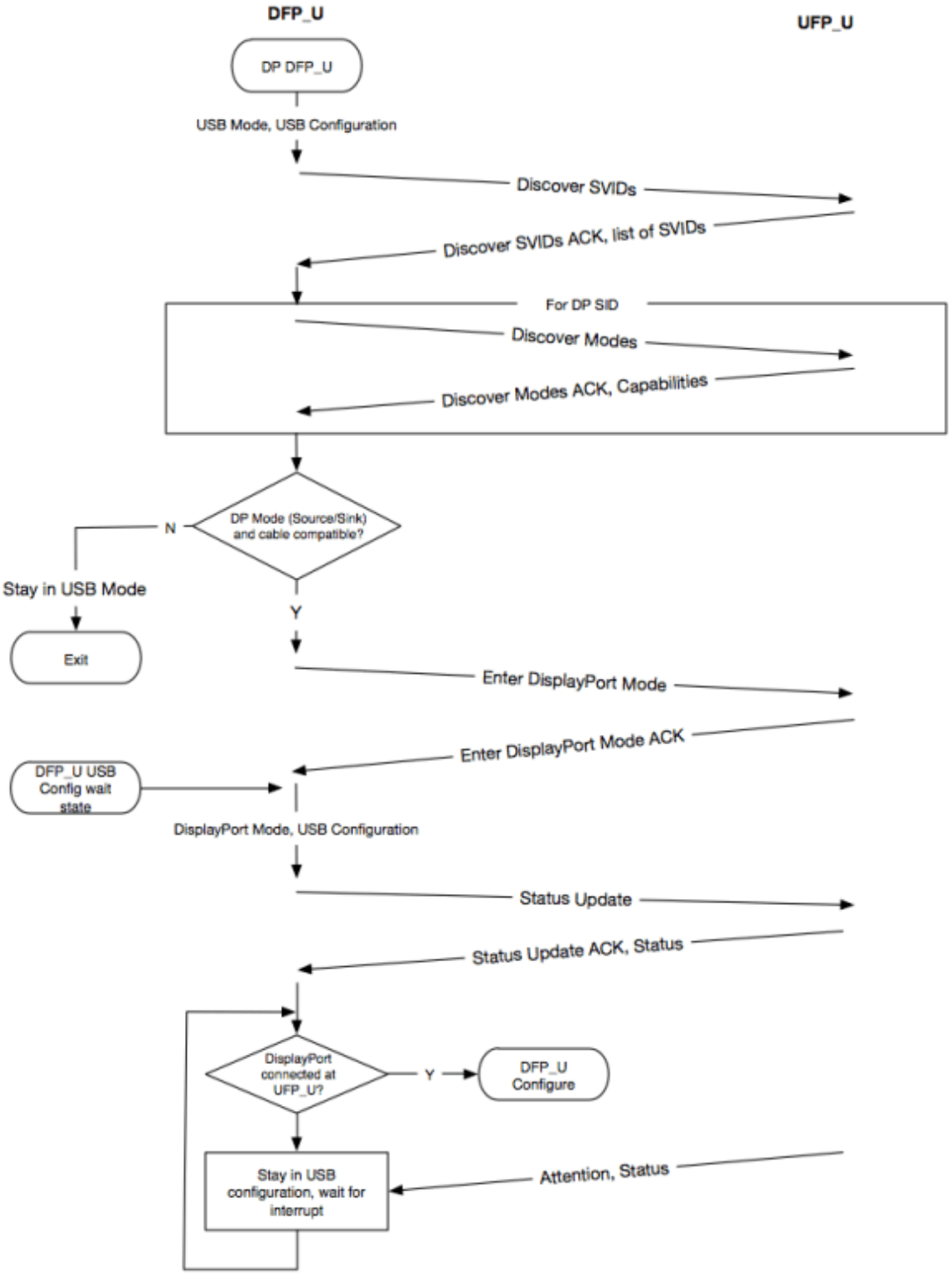
使能 DP

RK3588 DP 和 USB3.0 共用 PHY，PHY lane 的配置根据接口的不同有两种方式，Type-C 模式和非 Type-C 模式。

DP Alt Mode(Type-C)

根据 DisplayPort Alt Mode 协议，通过 PD (Power Delivery) 的状态机和显示器进行通信，进行 lane 的映射和 HPD 信息的传递。通过 PD 协议进入 DP Mode 并通过 attention 指令传递 HPD 信息的流程主要如下图所示。

Figure 5-2 DFP_U DisplayPort Alternate Mode Discovery and Entry



DP 控制器的配置如下：

```
&dp0 {
    status = "okay";
};

&dp0_in_vp2 {
    status = "okay";
};
```

在上面的配置中，使能了 DP0 接口，并把 DP0 绑定到 VOP 的 Video Port2，这只是一种参考配置，实际使用过程中，可以根据实际的需求，使能 DP0 或 DP1, 并把 DP0 或 DP1 绑定到期望的 Video Port(0/1/2) 上。

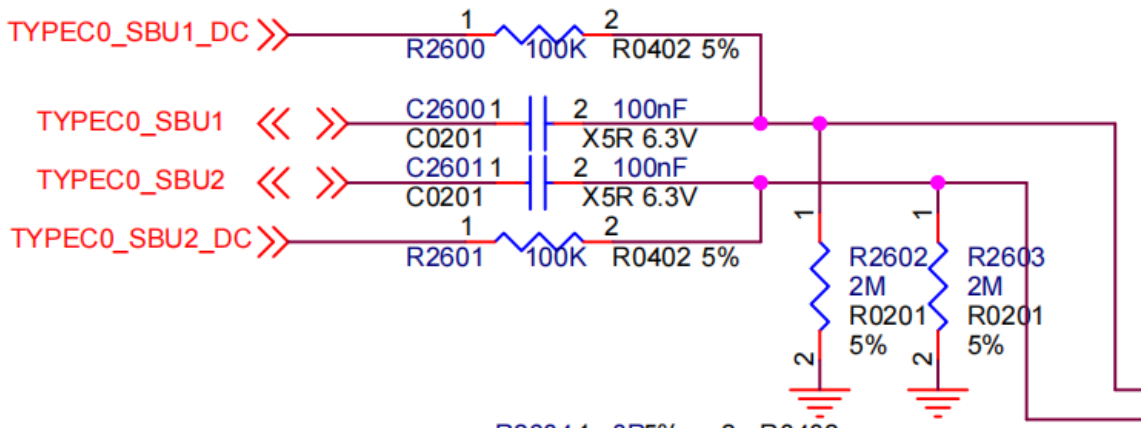
PHY 配置如下:

```
&usbdp_phy0 {
    status = "okay";
    orientation-switch;
    /* DP related config */
    svid = <0xff01>;
    sbu1-dc-gpios = <&gpio4 RK_PA6 GPIO_ACTIVE_HIGH>;
    sbu2-dc-gpios = <&gpio4 RK_PA7 GPIO_ACTIVE_HIGH>;
    /* DP related config */

    port {
        #address-cells = <1>;
        #size-cells = <0>;
        usbdp_phy0_orientation_switch: endpoint@0 {
            reg = <0>;
            remote-endpoint = <&usbc0_orien_sw>;
        };

        /* DP related config */
        usbdp_phy0_dp_altmode_mux: endpoint@1 {
            reg = <1>;
            remote-endpoint = <&dp_altmode_mux>;
        };
        /* DP related config */
    };
};
```

Type-C 的 SBU1 和 SBU2 引脚是和 DP 的 AUX_CH 复用的，在 Type-C 正插时， AUX_CH_P 复用 SBU1，AUX_CH_N 复用 SUB2。在 Type-C 反插时，AUX_CH_P 复用 SBU2, AUX_CH_N 复用 SBU1。根据 DP 协议要求，AUX_CH_P 需要配置为下拉状态，AUX_CH_N 需要配置成上拉状态。Type-C 不同的插入状态(正插和反插) AUX_CH_N 和 AUX_CH_P 的复用配置是不一样的，需要通过配置 GPIO 动态配置 SUB1 和 SUB2 的上下拉状态。因此，在配置 PHY 时，需要配置`sbu1-dc-gpios`和`sbu2-dc-gpios`(实际配置这两个 GPIO 的时候要参照硬件设计的原理图，例如下图的 TYPEC0_SBU1_DC 和 TYPEC0_SBU2_DC)，PHY 驱动会根据当前的 Type-C 正反插状态去调整 GPIO 输出的电平。`svid`对 DP 来说是固定值 0xff01。



Type-C 接口需要通过 Type-C 的 CC 检测和 PD 协商来配置 lane 和 HPD 的状态， 所以还需要配置 PD 芯片(当前支持的 PD 芯片有 fusb302, husb311):

```
&i2c2 {
    status = "okay";

    usbc0: fusb302@22 {
        compatible = "fcs,fusb302";
        reg = <0x22>;
        interrupt-parent = <&gpio3>;
        interrupts = <RK_PB4 IRQ_TYPE_LEVEL_LOW>;
        pinctrl-names = "default";
        pinctrl-0 = <&usbc0_int>;
        vbus-supply = <&vbus5v0_typec>;
        status = "okay";

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

            port@0 {
                reg = <0>;
                usbc0_role_sw: endpoint@0 {
                    remote-endpoint = <&dwc3_0_role_switch>;
                };
            };
        };

        usb_con: connector {
            compatible = "usb-c-connector";
            label = "USB-C";
            data-role = "dual";
            power-role = "dual";
            try-power-role = "sink";
            op-sink-microwatt = <1000000>;
            sink-pdos =
                <PDO_FIXED(5000, 1000, PDO_FIXED_USB_COMM)>;
            source-pdos =
                <PDO_FIXED(5000, 3000, PDO_FIXED_USB_COMM)>;

            /* DP related config */
            altmodes {
                #address-cells = <1>;
                #size-cells = <0>;

                altmode@0 {
                    reg = <0>;
                    svid = <0xff01>;
                    vdo = <0xffffffff>;
                };
            };
            /* DP related config */

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

                port@0 {
                    reg = <0>;
                    usbc0_orien_sw: endpoint {
```

```
                        remote-endpoint = <&usbdp_phy0_orientation_switch>;
                    };
                };

                /* DP related config */
                port@1 {
                    reg = <1>;
                    dp_altmode_mux: endpoint {
                        remote-endpoint = <&usbdp_phy0_dp_altmode_mux>;
                    };
                };
                /* DP related config */
            };
        };
    };
};
```

DP Legacy Mode

非 Type-C 接口输出，无论是 DP 接口，还是通过其他的转接芯片输出，配置流程基本一致，并且都需要配置 HPD Pin。 在实际分配 IO 引脚的时候，可以使用DP_HPD 专用引脚, 这种情况按 IOMUX 进行配置，还可以使用普通的 GPIO 进行检测。

使用 DP_HPD Pin 的时候配置如下：

```
&dp1 {
    pinctrl-0 = <&dp1m2_pins>;
    pinctrl-names = "default";
    status = "okay";
};

&dp1_in_vp2 {
    status = "okay";
};
```

使用普通 GPIO 作 HPD 检测的时候配置如下：

```
&dp1 {
    pinctrl-names = "default";
    pinctrl-0 = <&dp1_hpd>;
    hpd-gpios = <&gpio1 RK_PB5 GPIO_ACTIVE_HIGH>;
    status = "okay";
};

&dp1_in_vp2 {
    status = "okay";
};

&pinctrl {
    dp {
        dp1_hpd: dp1-hpd {
            rockchip,pins = <1 RK_PB5 RK_FUNC_GPIO &pcfg_pull_none>;
        };
    };
};
```

DP 和 USB 3.0 共用 PHY，当 DP 为 非 Type-C 接口输出时，就需要指定 lane 配置给 DP 使用以及对应的 lane 序号，这部分内容在 DTS 中指定。对于 DP PHY lane 的配置，可以配置成 2 lane 模式或 4 lane 模式。

PHY lane 接口的物理编号和 Pin 脚的关系如下：

Pin Name	SSRX1	SSTX1	SSRX2	SSTX2
Phy Lane	0	1	2	3

对于 DP 配置 4 lane, dtsi 配置属性如下：

```
rockchip,dp-lane-mux = <x x x x>;
```

对于 DP 配置 2 lane, dtsi 配置属性如下：

```
rockchip,dp-lane-mux = <x x>;
```

其中，索引为 DP 的 lane, 值为 PHY 的 lane。

如下为 DP 4 lane 的配置：

```
&usbdp_phy1 {
    rockchip,dp-lane-mux = <0 1 2 3>;
    status = "okay";
};
```

对于如上的配置， DP lane 的映射关系如下

DP lane	Phy lane	Pin Name
0	0	SSRX1
1	1	SSTX1
2	2	SSRX2
3	3	SSTX2

其中 DP lane 为 DP 的 lane 的序号。

DP 2 lane 的可以配置如下：

```
&usbdp_phy1 {
    rockchip,dp-lane-mux = <2 3>;
    status = "okay";
};
```

其映射关系如下：

DP lane	Phy lane	Pin Name
0	2	SSRX2
1	3	SSTX2

DP 开机 logo

配置开机 logo 后， 如果在开机前就插入 DP 显示器，即可在 U-Boot 阶段就开始显示 logo， 否则，只能等到系统启动后才能看到应用显示的图像。添加 DP 开机 logo 支持的配置如下：

```
&route_dp0 {
    status = "okay";
    connect = <&vp2_out_dp0>;
};
```

需要注意的是，这里的 connect 属性配置 DP 在 U-Boot 阶段绑定 VOP Port2, 所以 dtsi 中的配置要允许 DP 绑定 VOP Port2：

```
&dp0_in_vp2 {
    status = "okay";
};
```

Note: 目前不支持 Type-C 接口的 DP 开机 logo!

DP connector-split mode

DP connector-split mode 如图所示，一幅图像被平分成左右两部分，并分别通过 DP0/DP1 接口传输给显示器，下图中 DP0 作为左半屏，DP1 作为右半屏。

配置如下：

```
&dp0 {
    split-mode;
    status = "okay";
};

&dp0_in_vp2 {
    status = "okay";
};

&dp1 {
    status = "okay";
};
```

在作为左半屏的 DP 节点加入`split-mode`属性，并绑定要输出的 Video Port，在如上的配置中，即 DP0 作为左边屏，DP1 作为右半屏。在 Split Mode 模式下，两个 DP 当作一个 connector, 只有 DP0 和 DP1 同时连接时，这个 connector 才处于连接状态，才会开始显示，只要有一个 DP 接口处于断开状态，connector 即处于断开状态，不会输出显示。 在该模式下，两个 DP 接口输出的时序是一样的，建议使用两个一样的显示器。

在用户空间下，通过 modetest 或者 cat dri 的 state 节点(`cat /sys/kernel/debug/dri/0/state`)，只会看到一个 DP connector。

常用 DEBUG 方法

查看 connector 状态

在 /sys/class/drm 目录下可以看到驱动注册的各个 card，其中 card0-DP-1 和 card0-DP-2 是 DP 显示设备

```
rk3588_s:/ # ls /sys/class/drm/
card0       card0-DP-2   card0-HDMI-A-1  card0-Writeback-1  renderD128  version
card0-DP-1  card0-DSI-1  card0-HDMI-A-2  card1              renderD129
```

以 card0-DP-1 为例，其目录下有如下内容：

```
rk3588_s:/ # ls /sys/class/drm/card0-DP-1/
device  dpms  edid  enabled  modes  power  status  subsystem  uevent
```

enable 查看使能状态：

```
rk3588_s:/ # cat /sys/class/drm/card0-DP-1/enabled
disabled
```

status 查看连接状态：

```
rk3588_s:/ # cat /sys/class/drm/card0-DP-1/status
disconnected
```

modes 设备支持的分辨率列表：

```
rk3588_s:/ # cat /sys/class/drm/card0-DP-1/modes
```

```
1440x900
1280x1024
1280x1024
1280x960
1152x864
1024x768
1024x768
832x624
800x600
800x600
640x480
640x480
720x400
```

edid 设备的 EDID, 通过如下命令保存：

```
rk3588_s:/ # cat /sys/class/drm/card0-DP-1/edid > /data/edid.bin
```

强制使能/禁用 DP

```
#强制禁用 DP
rk3588_s:/ # echo off > /sys/class/drm/card0-DP-1/status
#强制使能 DP
rk3588_s:/ # echo on > /sys/class/drm/card0-DP-1/status
#恢复热插拔检测
rk3588_s:/ # echo detect > /sys/class/drm/card0-DP-1/status
```

DPCP 读写

DPCP 通过 AUX_CH 读写，读写节点的实现在

```
/drivers/gpu/drm/drm_dp_aux_dev.c
```

使用此功能前，先确认相关的编译选项是否已经配置：

```
CONFIG_DRM_DP_AUX_CHARDEV=y
```

读取 DPCD 如下：

```
#if 后面为 aux 节点，当注册两个 DP 接口时，会有 /dev/drm_dp_aux0 和 /dev/drm_dp_aux1
#skip 值为起始的 DPCD 寄存器地址
#count 值为要读取的 DPCD 寄存器的数量
dd if=/dev/drm_dp_aux0 bs=1 skip=$((0x00200)) count=2 status=none | od -tx1
#如下为读取地址为 0x00200 开始的 2 个 DPCD 寄存器的内容
rk3588_s:/ # dd if=/dev/drm_dp_aux1 bs=1 skip=$((0x00200)) count=2 status=none |
od -tx1
0000000 01 00
0000002
```

写入 DPCD 寄存器：

```
#echo 后为要写入的值，如下为需要写入两个 16 进制的值，分别为 0x0a，0x80
#of 后面为 aux 节点，当注册两个 DP 接口时，会有 /dev/drm_dp_aux0 和 /dev/drm_dp_aux1
#seek 后为起始的 DPCD 寄存器地址
#count 值为要写入的 DPCD 寄存器的数量
#如下指令为把 0x0a 和 0x80 两个值写入 0x100 起始的两个 DPCD 寄存器处
echo -e -n "\x0a\x80" | dd of=/dev/drm_dp_aux0 bs=1 seek=$((0x100)) count=2
status=none
```

Type-C 接口 Debug

Type-C 接口的 HPD 检测部分由 PD 芯片完成，这部分的软件流程主要由 TCPM 的框架完成，TCPM 检测这部分 log 可以由以下方式获取：

```
rk3588_s:/ # ls -l /sys/kernel/debug/usb/
total 0
-r--r--r--  1 root root 0 1970-01-01 00:00 devices
drwxr-xr-x 18 root root 0 1970-01-01 00:00 fc000000.usb
drwxr-xr-x  2 root root 0 1970-01-01 00:00 fc400000.usb
-r--r--r--  1 root root 0 1970-01-01 00:00 fusb302-2-0022
drwxr-xr-x  4 root root 0 1970-01-01 00:00 ohci
-r--r--r--  1 root root 0 1970-01-01 00:00 tcpm-2-0022
drwxr-xr-x  2 root root 0 1970-01-01 00:00 usbmon
drwxr-xr-x  2 root root 0 2021-01-01 12:00 uvcvideo
drwxr-xr-x  3 root root 0 1970-01-01 00:00 xhci
```

在 /sys/kernel/debug/usb/ 目录中，可以看到 fusb302-2-0022 和 tcpm-2-0022， 其中 fusb302-2-0022 为 PD 芯片的节点， tcpm-2-0022 为 TCPM 框架的节点， 获取 TCPM 框架的 log 命令如下：

```
cat /sys/kernel/debug/usb/tcpm-2-0022
```

获取 PD 芯片的 log 如下(不同的 PD 芯片节点名称不一样)：

```
cat /sys/kernel/debug/usb/fusb302-2-0022
```

除了 log 外，在 Type-C 节点下还可以获取其他的一些信息，Type-C 节点路径如下：

```
console:/ # ls /sys/class/typec
port0  port0-partner
```

port0 表示 SoC 这端的 Type-C 接口， port0-partner 表示通过 Type-C 连接设备后设备端的节点目录。

Type-C 连接的正反面信息：

```
cat /sys/class/typec/port0/orientation
reverse
```

port0-partner 下可能有多个 目录，对于 DP Alt Mode 对应的目录，其对应的目录先会有 displayport 子目录，并且 svid 的值为 0xff01。

```
ls -l /sys/class/typec/port0-partner/port0-partner.0/
total 0
-r--r--r-- 1 root root 4096 2022-04-14 14:50 active
```

```
-r--r--r-- 1 root root 4096 2022-04-14 14:50 description
drwxr-xr-x 2 root root    0 2022-04-14 14:50 displayport
lrwxrwxrwx 1 root root    0 2022-04-14 14:50 driver ->
../../../../../../../../../bus/typec/drivers/typec_displayport
-r--r--r-- 1 root root 4096 2022-04-14 14:50 mode
drwxr-xr-x 2 root root    0 2022-04-14 14:50 mode1
lrwxrwxrwx 1 root root    0 2022-04-14 14:50 port -> ../../port0.0
drwxr-xr-x 2 root root    0 2022-04-14 14:50 power
lrwxrwxrwx 1 root root    0 2022-04-14 14:50 subsystem ->
../../../../../../../../../bus/typec
-r--r--r-- 1 root root 4096 2022-04-14 14:50 svid
-rw-r--r-- 1 root root 4096 2022-04-14 14:50 uevent
-r--r--r-- 1 root root 4096 2022-04-14 14:50 vdo
```

```
cat /sys/class/typec/port0-partner/port0-partner.0/svid
ff01
```

获取当前的 pin assignment 信息：

```
cat /sys/class/typec/port0-partner/port0-partner.0/displayport/pin_assignment
C [D]
#当前连接的设备支出 C assignment 和 D assignment, 目前配置的是 D assignment
```

查看 DP 寄存器

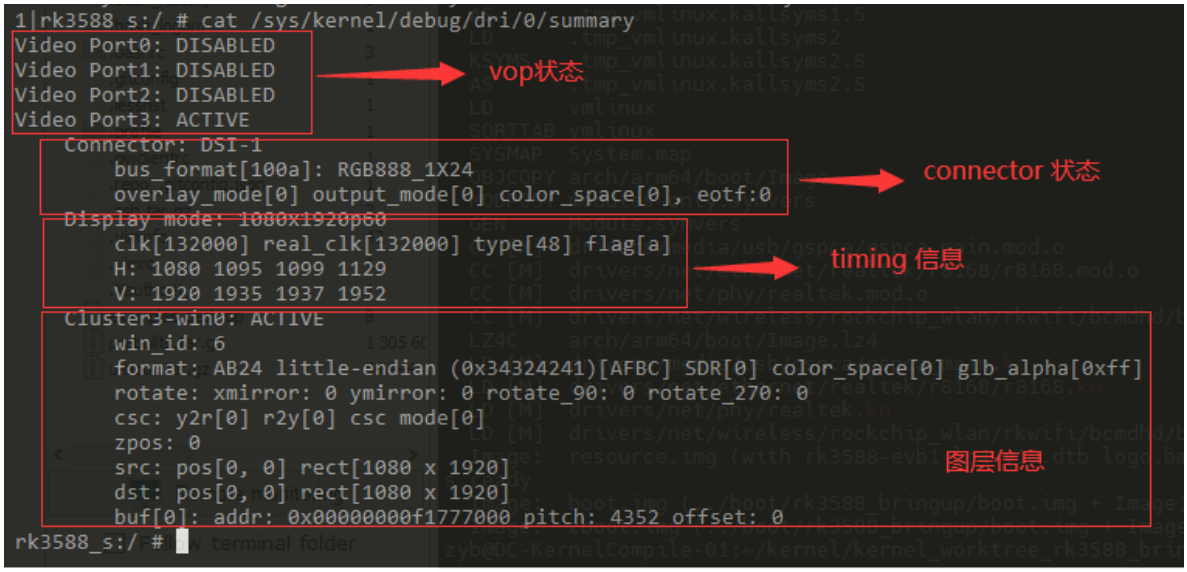
```
#dp0 控制器
cat /sys/kernel/debug/regmap/fde50000.dp/registers
#dp1 控制器
cat /sys/kernel/debug/regmap/fde60000.dp/registers
# vo0_grf
cat /sys/kernel/debug/regmap/dummy-syscon@fd5a6000/registers
```

查看 VOP 状态

通过如下指令即可查询 VOP 的状态：

```
cat /sys/kernel/debug/dri/0/summary
```

获取的 VOP 状态如下图：



Video Portx: 表示当前的 Video Port 的状态

Connector: Video Port 当前连接的输出接口

Display mode: Video Port 当前输出时序

Clusterx-winx(Esmartx-winx): 图层信息

查看当前显示时钟

获取整个时钟树：

```
cat /sys/kernel/debug/clk/clk_summary
```

获取 dp aux 16M clk:

```
cat /sys/kernel/debug/clk/clk_summary | grep -e "clk_aux16m_"
```

获取 vop dclk：

```
cat /sys/kernel/debug/clk/clk_summary | grep -e "dclk"
```

调整 DRM log 等级

DRM 有如下的打印等级定义，可以根据需要，动态的打开对应的 log 打印：

```
enum drm_debug_category {
        /**
         * @DRM_UT_CORE: Used in the generic drm code: drm_ioctl.c, drm_mm.c,
         * drm_memory.c, ...
         */
        DRM_UT_CORE             = 0x01,
        /**
         * @DRM_UT_DRIVER: Used in the vendor specific part of the driver: i915,
         * radeon, ... macro.
         */
        DRM_UT_DRIVER           = 0x02,
        /**
         * @DRM_UT_KMS: Used in the modesetting code.
         */
```

```
        DRM_UT_KMS              = 0x04,
        /**
         * @DRM_UT_PRIME: Used in the prime code.
         */
        DRM_UT_PRIME            = 0x08,
        /**
         * @DRM_UT_ATOMIC: Used in the atomic code.
         */
        DRM_UT_ATOMIC           = 0x10,
        /**
         * @DRM_UT_VBL: Used for verbose debug message in the vblank code.
         */
        DRM_UT_VBL              = 0x20,
        /**
         * @DRM_UT_STATE: Used for verbose atomic state debugging.
         */
        DRM_UT_STATE            = 0x40,
        /**
         * @DRM_UT_LEASE: Used in the lease code.
         */
        DRM_UT_LEASE            = 0x80,
        /**
         * @DRM_UT_DP: Used in the DP code.
         */
        DRM_UT_DP               = 0x100,
        /**
         * @DRM_UT_DRMRES: Used in the drm managed resources code.
         */
        DRM_UT_DRMRES           = 0x200,
};
```

DP 排查问题时，目前比较多的是打开 ATOMIC，如下：

```
echo 0x10 > /sys/module/drm/parameters/debug
```

FAQ

插入 DP 无显示或显示异常

首先查看是否有如下 log：

```
[   14.857002] rockchip-vop2 fdd90000.vop: [drm:vop2_crtc_atomic_enable] Update
mode to 1920x1080p60, type: 10(if:200) for vp2 dclk: 148500000
[   14.857149] rockchip-vop2 fdd90000.vop: [drm:vop2_crtc_atomic_enable]
dclk_out2 div: 2 dclk_core2 div: 2
[   14.857868] rockchip-vop2 fdd90000.vop: [drm:vop2_crtc_atomic_enable] set
dclk_vop2 to 148500000, get 148500000
[   14.872406] dw-dp fde50000.dp: full-training link: 2 lanes at 5400 MHz
[   14.893269] dw-dp fde50000.dp: clock recovery succeeded
[   14.899797] dw-dp fde50000.dp: channel equalization succeeded
```

DP Link Training 成功

出现如上 log 时，说明已经检测到 DP 连接，并且 DP 已经成功 link training 并输出图像，出现无显示或显示异常的原因可能如下：

1. dclk 分的不准

可以看如下的 log 如下，请求的 dclk 为 25.175MHz，实际分到的为 20MHz, 出现这种 clk 分配问题，抓取完整的 log 并提供时钟树 log 供进一步分析。

```
[ 268.733803] rockchip-vop2 fdd90000.vop: [drm:vop2_crtc_atomic_disable] Crtc
atomic disable vp2
[ 268.759178] rockchip-vop2 fdd90000.vop: [drm:vop2_crtc_atomic_enable] Update
mode to 640x480p60, type: 10(if:200) for vp2 dclk: 25175000
[ 268.759447] rockchip-vop2 fdd90000.vop: [drm:vop2_crtc_atomic_enable]
dclk_out2 div: 2 dclk_core2 div: 2
[ 268.759665] rockchip_rk3588_pll_set_rate: Invalid rate : 25175000 for pll clk
pll_v0pll
[ 268.759715] rockchip-vop2 fdd90000.vop: [drm:vop2_crtc_atomic_enable] set
dclk_vop2 to 25175000, get 20000000
[ 268.775591] dw-dp fde50000.dp: full-training link: 4 lanes at 2700 MHz
[ 268.790059] dw-dp fde50000.dp: clock recovery succeeded
[ 268.795376] dw-dp fde50000.dp: channel equalization succeeded
```

2. 未分配图层

userspace 未分配图层，执行`cat /sys/kernel/debug/dri/0/summary`，如果获取的信息如下所示，即没有图层信息，需要从 userspace 部分进一步分析。

```
rk3588_s:/ # cat /sys/kernel/debug/dri/0/summary
Video Port0: DISABLED
Video Port1: DISABLED
Video Port2: DISABLED
Video Port3: ACTIVE
    Connector: DSI-1
        bus_format[100a]: RGB888_1X24
        overlay_mode[0] output_mode[0] color_space[0], eotf:0
    Display mode: 1080x1920p60
        clk[132000] real_clk[132000] type[48] flag[a]
        H: 1080 1095 1099 1129
        V: 1920 1935 1937 1952
```

DP connected

如果未出现本小节开头出现的 log，先获取 DP 的连接状态如下：

```
cat /sys/class/drm/card0-DP-1/status
```

如果 DP 是 connected 状态， 先分析 log 是否有异常报错，有异常报错从异常处分析，如果 log 无异常，打开 DRM 的 ATOMIC log 等级复现，确认是否在 drm atomic commit 中途异常返回。

DP disconnected

对于 DP 标准口输出，确认 HPD 配置是否正确以及硬件连接是否正常, 对于 Type-C 接口，参考后文的 Type-C 接口连接异常分析。

Type-C 接口连接异常

这里的 Type-C 接口连接异常指的是 CC 阶段和 PD 阶段即出现异常，首先获取 DP 的连接状态：

```
cat /sys/class/drm/card0-DP-1/status
```

连接异常时这里获取到状态都是 disconnected。

通过 tcpm 的调试节点获取 tcpm 的 log：

```
cat /sys/kernel/debug/usb/tcpm-2-0022
```

正常连接的 log 如下：

```
[   25.026952] AMS DISCOVER_IDENTITY start
[   25.026967] PD TX, header: 0x176f
[   25.035314] PD TX complete, status: 0
[   25.042866] PD RX, header: 0x524f [1]
[   25.042880] Rx VDM cmd 0xff00a041 type 1 cmd 1 len 5
[   25.042894] AMS DISCOVER_IDENTITY finished
[   25.042898] cc:=4
[   25.052343] Identity: 04e8:a020.0212
[   25.052364] AMS DISCOVER_SVIDS start
[   25.052372] PD TX, header: 0x196f
[   25.061314] PD TX complete, status: 0
[   25.067667] PD RX, header: 0x344f [1]
[   25.067680] Rx VDM cmd 0xff00a042 type 1 cmd 2 len 3
[   25.067695] AMS DISCOVER_SVIDS finished
[   25.067705] cc:=4
[   25.077097] SVID 1: 0xff01
[   25.077114] SVID 2: 0x4e8
[   25.077129] AMS DISCOVER_MODES start
[   25.077135] PD TX, header: 0x1b6f
[   25.086092] PD TX complete, status: 0
[   25.092224] PD RX, header: 0x264f [1]
[   25.092237] Rx VDM cmd 0xff01a043 type 1 cmd 3 len 2
[   25.092252] AMS DISCOVER_MODES finished
[   25.092256] cc:=4
[   25.101432]  Alternate mode 0: SVID 0xff01, VDO 1: 0x00000c05
[   25.101517] AMS DISCOVER_MODES start
[   25.101526] PD TX, header: 0x1d6f
[   25.109717] PD TX complete, status: 0
[   25.114919] PD RX, header: 0x284f [1]
[   25.114937] Rx VDM cmd 0x4e8a043 type 1 cmd 3 len 2
[   25.114951] AMS DISCOVER_MODES finished
[   25.114956] cc:=4
[   25.124604]  Alternate mode 1: SVID 0x04e8, VDO 1: 0x00000001
[   25.125676] AMS DFP_TO_UFP_ENTER_MODE start
[   25.125686] PD TX, header: 0x1f6f
[   25.134560] PD TX complete, status: 0
[   25.137903] PD RX, header: 0x1a4f [1]
[   25.137917] Rx VDM cmd 0xff01a144 type 1 cmd 4 len 1
[   25.137930] AMS DFP_TO_UFP_ENTER_MODE finished
[   25.137936] cc:=4
[   25.145828] AMS STRUCTURED_VDMS start
[   25.145836] PD TX, header: 0x216f
[   25.154942] PD TX complete, status: 0
[   25.161111] PD RX, header: 0x2c4f [1]
[   25.161125] Rx VDM cmd 0xff01a150 type 1 cmd 16 len 2 //STATUS UPDATE
[   25.161138] AMS STRUCTURED_VDMS finished
```

```
[   25.161142] cc:=4
[   25.171888] AMS STRUCTURED_VDMS start
[   25.171911] PD TX, header: 0x236f
[   25.182016] PD TX complete, status: 0
[   25.185550] PD RX, header: 0x1e4f [1]
[   25.185563] Rx VDM cmd 0xff01a151 type 1 cmd 17 len 1 //CONFIGURATION
[   25.185577] AMS STRUCTURED_VDMS finished
[   25.185581] cc:=4
[   26.392673] PD RX, header: 0x204f [1]
[   26.392687] Rx VDM cmd 0xff018106 type 0 cmd 6 len 2 //ATTENTION
```

从 log 看，正常的完整流程会有 DISCOVER_IDENTITY，DISCOVER_MODES、DFP_TO_UFP_ENTER_MODE、STATUS UPDATE、CONFIGURATION、ATTENTION等命令的交互，如果没有以上的交互流程，即说明 PD 的交互出现了异常。

出现异常这种异常时，需要提供完整的 tcpm 的 log 和 PD 芯片的 log 进一步分析。

AUX_CH 异常

AUX_CH 异常时，会导致读写 DPCD 和读 EDID 出现异常，log 中可能会出现如下报错：

```
[ 1368.952182] dw-dp fde50000.dp: failed to probe DP link: -110
```

可能的原因如下：

1. aux16m clk 值异常

   aux16m clk偏差太大，这种情况只有在 aux16m clk 的 parent rate 的被重新设置时才出现，实际应用中比较少，获取的默认值如下：

```
root@RK3588:/# cat /sys/kernel/debug/clk/clk_summary | grep "aux16m"
          clk_aux16m_1                1        2        0    15840000          0
    0  50000
          clk_aux16m_0                1        2        0    15840000          0
    0  50000
```

2. phy power on/off 流程异常

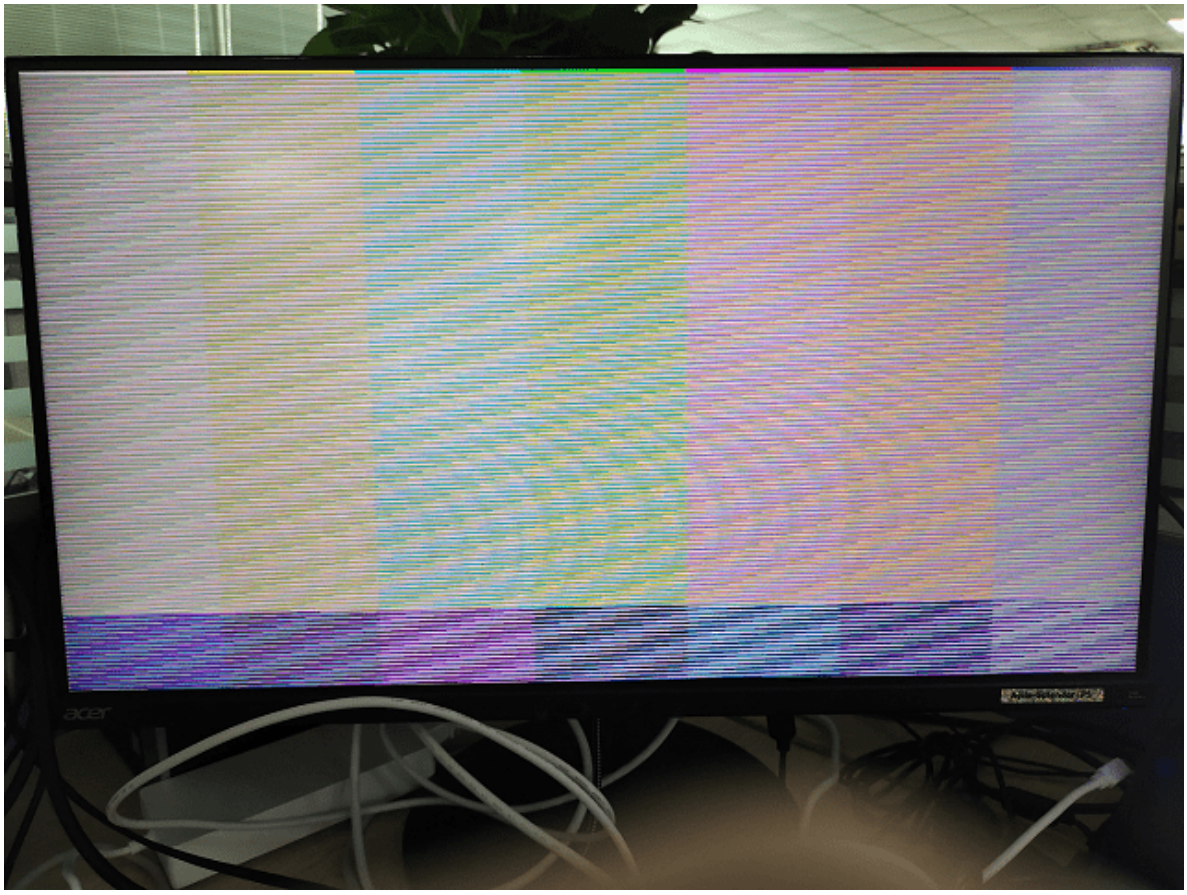
   这种一般出现在 Type-C 接口， USB 和 Type-C 共用 phy 的场景, 如果出现 Type-C 一面可以正常工作，换另一面插入报错，即流程异常，重新插入时 PHY 未重新初始化， 可以添加 log 先确认 USB 插拔时是否有执行 phy power on/off。

3. 硬件异常

   需要确认硬件连接通路是否正常，AUX_CH 通路的外围电路是否参照 DP 协议进行设计。

4K 120Hz 输出配置

RK3588 默认的 VOP ACLK 是 500M，对于输出的 4K 120Hz 这种高 pixel clk 的配置， 会由于性能问题导致出现如下的显示异常：



对于这种问题，需要把 VOP ACLK 提高到 800M：

```
&vop {
    assigned-clocks = <&cru ACLK_VOP>;
    assigned-clock-rates = <800000000>;
};
```

获取 VOP ACLK 如下：

```
cat /sys/kernel/debug/clk/clk_summary | grep "aclk_vop"
```

DP 带宽计算

获取 DP 每条 lane 支持的带宽，公式如下：

$$bandwidth\_per\_lane = pixel\_clk * bit\_per\_pixel * 1.25 / lane\_count$$

其中，`bit_per_pixel` 是每个 pixel 的 bit 数， `1.25` 是 phy lane 的编码转换效率， `lane_count` 是可用的 lane 的数量，最终的计算结果 `bandwith_per_lane` 即每条 lane 需要提供的最小带宽，如果当前的 lane rate 比需要的最小带宽小，对应的 pixel clk 的 display mode 就会被 DP 的驱动程序过滤掉。

对于使用转接线或拓展坞时，需要确定转接线和拓展坞支持的 lane rate 和 lane count 是否满足当前的带宽要求，如果无法满足，需要更换支持更高 lane rate 和更到 lane count 的转接线和拓展坞。

例如，对于一个 lane 数量为 2， 最大的 lane rate 为 5.4 Gbps/lane 的拓展坞，如果要输出的 4K@60Hz, pixel clock 为 594MHz, RGB888 格式的图像数据时，需要的每条 lane 的带宽为:

$$bandwidth\_per\_lane = 594 * 24 * 1.25 / 2 = 8.91 Gbps/lane > 5.4 Gbps/lane$$

可以看到，当前的拓展坞不支持输出 4K@60Hz, pixel clock 为 594MHz, RGB888 格式的数据，需要使用 4 lane 输出的拓展坞， 增加 PHY lane 的带宽，或输出 YUV420 格式的数据，减少需要使用 PHY lane 的带宽。