Rockchip Developer Guide DVR&DMS Product

文件标识：RK-SM-YF-398

发布版本：V1.2.0

日期：2023-03-27

文件密级：□绝密 □秘密 □内部资料 ■公开

免责声明

本文档按“现状”提供，瑞芯微电子股份有限公司（“本公司”，下同）不对本文档的任何陈述、信息和内容的准确性、可靠性、完整性、适销性、特定目的性和非侵权性提供任何明示或暗示的声明或保证。本文档仅作为使用指导的参考。

由于产品版本升级或其他原因，本文档将可能在未经任何通知的情况下，不定期进行更新或修改。

商标声明

“Rockchip”、“瑞芯微”、“瑞芯”均为本公司的注册商标，归本公司所有。

本文档可能提及的其他所有注册商标或商标，由其各自拥有者所有。

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

概述

DVR&DMS 产品方案使用说明。

产品版本

芯片名称	内核版本
RV1126, RV1109	Linux 4.19

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

日期	版本	作者	修改说明
2021-01-31	V1.0.0	Vicent Chi, Zhihua Wang, Zhichao Yu	初始版本
2021-03-01	V1.0.1	Vicent Chi	添加MIPI+DVP方案描述
2021-03-15	V1.0.2	Ruby Zhang	完善产品版本信息
2021-04-30	V1.1.0	Zhichao Yu	修改AD芯片接入方案描述；增加VP介绍
2021-06-03	V1.1.1	Vicent Chi	添加FAQ
2021-07-24	V1.1.2	Zhichao Yu	添加音频相关说明
2021-10-11	V1.1.3	Ruby Zhang	修正一些语言表达
2021-10-26	V1.1.4	Zhichao Yu	增加CVBS奇偶场合成说明
2021-10-27	V1.1.5	Xing Zheng	补充多路音频采集方法说明
2022-03-08	V1.1.6	Zhichao Yu	添加一些注意事项
2022-06-08	V1.1.7	Zhichao Yu	补充不同分辨率摄像头混接说明
2022-07-20	V1.1.8	Yiqing Zeng	修改CVBS奇偶场支持的说明
2022-11-29	V1.1.9	Zhichao Yu	添加产品性能优化相关说明
2023-03-27	V1.2.0	Yiqing Zeng	完善混接以及复位参数的相关说明

目录

1. 瑞芯微DVR/DMS产品方案说明

RV1126芯片有两路MIPI接口以及一路DVP接口，另外提供强大的编码性能最高支持8路1080@15fps同时编码，内置2T算力NPU，因此非常适合开发DVR/DMS产品。

1.1 RV1126芯片平台开发DVR/DMS的产品优势

- 支持最高8路1080P模拟高清视频输入；
- 强大的AI处理能力，能够支持DMS+ADAS算法同时运行；
- 强大的编码能力，最高支持8路1080P@15fps同时编码；
- 支持8路视频OSD叠加；
- 支持8路视频分屏显示Demo；

1.2 模拟高清RX芯片选型列表

目前RV1126平台已经适配了比较多的模拟高清RX芯片，并已经在SDK中集成了这些芯片的驱动，可以通过下表选择：

型号	厂家	接口	通道数	最大支持分辨率
NVP6188	Nextchip	MIPI	4	4K
N4	Nextchip	MIPI	4	1080P
TP2815	Techpoint	MIPI	4	1080P
TP2855	Techpoint	MIPI	4	1080P
TP9930	Techpoint	DVP	4	2K
TP9950	Techpoint	MIPI/DVP	1	1080P
RN6854	Richnex	MIPI	4	1080P

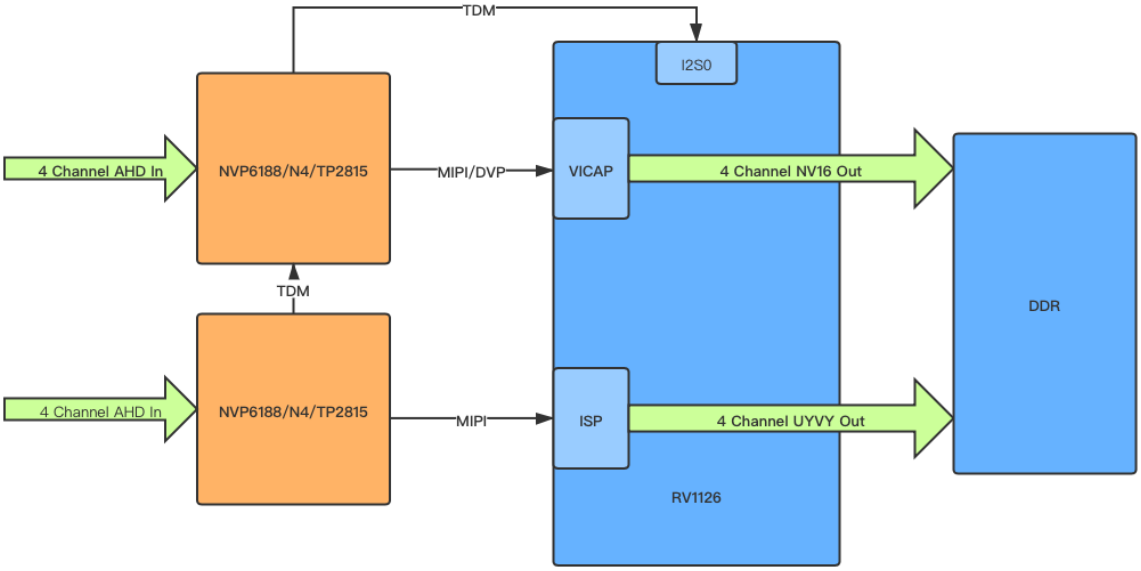
1.3 RV1126 DVR/DMS产品应用框图

RV1126支持不同AD芯片的接入方式，MIPI + MIPI 以及 MIPI+DVP。

由于RV1126芯片限制，需要限制不同通路图像输出格式：

- **VICAP通路：要统一采用NV16格式；**
- **ISP通路：要统一采用UYVY格式；**

Camera输入的参考方案框图如下：



注意：1、DVP接口输入时，VICAP不支持每个通道设置分辨率，故不支持不同分辨率混接；

2、MIPI接口输入时，VICAP支持每个通道单独设置分辨率，故支持不同分辨率混接；

3、MIPI接口输入时，ISP不支持每个通道设置分辨率，故不支持不同分辨率混接；

综上所述，当接入8路AHD时，仅MIPI接口输入接在VICAP的4路AHD支持不同分辨率混接；

2. 模拟高清RX芯片驱动开发说明

2.1 内核config配置

根据需求打开RX芯片相关config配置：

```
CONFIG_VIDEO_NVP6188=y
```

2.2 内核dts配置

以NVP6188 RX芯片为例：

```
    nvp6188_0: nvp6188_0@30 {
        compatible = "nvp6188";
        reg = <0x30>;
        clocks = <&cru CLK_MIPICSI_OUT>;
        clock-names = "xvclk";
        power-domains = <&power RV1126_PD_VI>;
        pinctrl-names = "rockchip,camera_default";
        pinctrl-0 = <&mipicsi_clk0>;
        reset-gpios = <&gpio4 RK_PA0 GPIO_ACTIVE_HIGH>;
        power-gpios = <&gpio1 RK_PD4 GPIO_ACTIVE_HIGH>;
```

```
        vi-gpios = <&gpio3 RK_PC0 GPIO_ACTIVE_HIGH>;
        rockchip,camera-module-index = <0>;
        rockchip,camera-module-facing = "front";
        rockchip,camera-module-name = "nvp6188";
        rockchip,camera-module-lens-name = "nvp6188";
        port {
            ucam_out0: endpoint {
                remote-endpoint = <&mipi_in_ucam0>;
                data-lanes = <1 2 3 4>;
            };
        };
    };

    nvp6188_1: nvp6188_1@32 {
        compatible = "nvp6188";
        reg = <0x32>;
        clocks = <&cru CLK_MIPICSI_OUT>;
        clock-names = "xvclk";
        power-domains = <&power RV1126_PD_VI>;
        pinctrl-names = "rockchip,camera_default";
        pinctrl-0 = <&mipicsi_clk1>;
        reset-gpios = <&gpio4 RK_PA1 GPIO_ACTIVE_HIGH>;
        vi-gpios = <&gpio3 RK_PC1 GPIO_ACTIVE_HIGH>;
        rockchip,camera-module-index = <1>;
        rockchip,camera-module-facing = "back";
        rockchip,camera-module-name = "nvp6188";
        rockchip,camera-module-lens-name = "nvp6188";
        port {
            ucam_out1: endpoint {
                remote-endpoint = <&csi_dphy1_input>;
                data-lanes = <1 2 3 4>;
            };
        };
    };
```

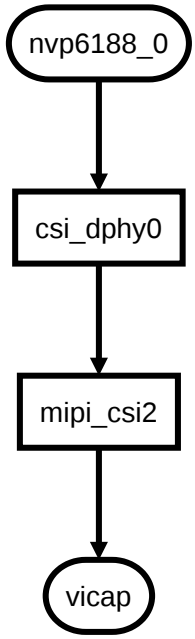
3. 数据流通路说明

3.1 双路方案通路

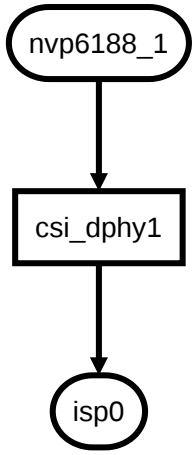
双MIPI方案

以双片NVP6188为例:

- VICAP通路0



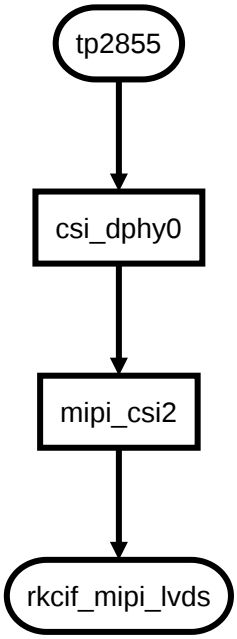
- ISP通路1



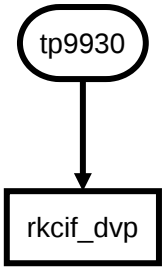
MIPI+DVP方案

以TP9930+TP2855为例：

- VICAP通路0



- VICAP通路1



3.2 通道对应的video格式限制

3.2.1 VICAP通路

- 建议使用NV16格式采集，也可支持NV12格式采集

  注意：MIPI+DVP方案时，统一使用NV16格式采集

3.2.2 ISP通路

- 要统一使用UYVY格式采集

3.3 通道对应的video节点枚举

3.3.1 VICAP通路 - MIPI接口

- media-ctl -p -d /dev/mediaX 获取 stream_cif_mipi_id0/1/2/3 的 device node name

```
[root@RV1126_RV1109:/]# media-ctl -p -d /dev/media0
```

```
Media controller API version 4.19.111

Media device information
------------------------
driver          rkcif
model           rkcif_mipi_lvds
serial
bus info
hw revision     0x0
driver version  4.19.111

Device topology
- entity 1: stream_cif_mipi_id0 (1 pad, 4 links)
            type Node subtype V4L flags 0
            device node name /dev/video0
        pad0: Sink
                <- "rockchip-mipi-csi2":1 [ENABLED]
                <- "rockchip-mipi-csi2":2 []
                <- "rockchip-mipi-csi2":3 []
                <- "rockchip-mipi-csi2":4 []

- entity 5: stream_cif_mipi_id1 (1 pad, 4 links)
            type Node subtype V4L flags 0
            device node name /dev/video1
        pad0: Sink
                <- "rockchip-mipi-csi2":1 []
                <- "rockchip-mipi-csi2":2 [ENABLED]
                <- "rockchip-mipi-csi2":3 []
                <- "rockchip-mipi-csi2":4 []

- entity 9: stream_cif_mipi_id2 (1 pad, 4 links)
            type Node subtype V4L flags 0
            device node name /dev/video2
        pad0: Sink
                <- "rockchip-mipi-csi2":1 []
                <- "rockchip-mipi-csi2":2 []
                <- "rockchip-mipi-csi2":3 [ENABLED]
                <- "rockchip-mipi-csi2":4 []

- entity 13: stream_cif_mipi_id3 (1 pad, 4 links)
             type Node subtype V4L flags 0
             device node name /dev/video3
        pad0: Sink
                <- "rockchip-mipi-csi2":1 []
                <- "rockchip-mipi-csi2":2 []
                <- "rockchip-mipi-csi2":3 []
                <- "rockchip-mipi-csi2":4 [ENABLED]
```

注意：四个通道支持不同分辨率接入，即转换芯片可以接入四路不同分辨率的摄像头。

3.4 VICAP通路 - DVP接口

media-ctl -p -d /dev/mediaX 获取 stream_cif_dvp_id0/1/2/3 的 device node name

```
[root@RV1126_RV1109:/]# media-ctl -p -d /dev/media0
Media controller API version 4.19.206

Media device information
------------------------
driver          rkcif
model           rkcif_dvp
serial
bus info
hw revision     0x0
driver version  4.19.206

Device topology
- entity 1: stream_cif_dvp_id0 (1 pad, 1 link)
            type Node subtype V4L flags 0
            device node name /dev/video0
        pad0: Sink
                <- "m00_b_techpoint 2-0046":0 [ENABLED]

- entity 5: stream_cif_dvp_id1 (1 pad, 1 link)
            type Node subtype V4L flags 0
            device node name /dev/video1
        pad0: Sink
                <- "m00_b_techpoint 2-0046":1 [ENABLED]

- entity 9: stream_cif_dvp_id2 (1 pad, 1 link)
            type Node subtype V4L flags 0
            device node name /dev/video2
        pad0: Sink
                <- "m00_b_techpoint 2-0046":2 [ENABLED]

- entity 13: stream_cif_dvp_id3 (1 pad, 1 link)
             type Node subtype V4L flags 0
             device node name /dev/video3
        pad0: Sink
                <- "m00_b_techpoint 2-0046":3 [ENABLED]
```

注意：DVP接口不支持输入不同分辨率的摄像头，即DVP接口的转换芯片需要接入四个相同分辨率的摄像头。

3.4.1 ISP通路

- media-ctl -p -d /dev/mediaX 获取 rkisp_mainpath、rkisp_rawwr0/1/2 的 device node name

```

media-ctl -p -d /dev/media1
Media controller API version 4.19.111

Media device information
------------------------
```

```
driver          rkisp
model           rkisp0
serial
bus info
hw revision     0x0
driver version  4.19.111

Device topology
- entity 17: rkisp_mainpath (1 pad, 1 link)
             type Node subtype V4L flags 0
             device node name /dev/video5
        pad0: Sink
                <- "rkisp-isp-subdev":2 [ENABLED]

- entity 29: rkisp_rawwr0 (1 pad, 1 link)
             type Node subtype V4L flags 0
             device node name /dev/video7
        pad0: Sink
                <- "rkisp-csi-subdev":2 [ENABLED]

- entity 35: rkisp_rawwr1 (1 pad, 1 link)
             type Node subtype V4L flags 0
             device node name /dev/video8
        pad0: Sink
                <- "rkisp-csi-subdev":3 [ENABLED]

- entity 41: rkisp_rawwr2 (1 pad, 1 link)
             type Node subtype V4L flags 0
             device node name /dev/video9
        pad0: Sink
                <- "rkisp-csi-subdev":4 [ENABLED]
```

注意：ISP通路不支持输入不同分辨率的摄像头，即转换芯片需要接入四个相同分辨率的摄像头。

3.5 通道对应的video采集限制

3.5.1 ISP通路

- pipeline 切换

```
    开机启动脚本需要添加  (/dev/media1根据实际isp注册情况决定)

    media-ctl -d /dev/media1 -l '"rkisp-isp-subdev":2->"rkisp-bridge-ispp":0[0]'
    media-ctl -d /dev/media1 -l '"rkisp-isp-subdev":2->"rkisp_mainpath":0[1]'
```

- stream on 开关

```
    因为rkisp_mainpath、rkisp_rawwr0/1/2 四个通道的stream on开关没有单独的开关，
    因此如果要采集rkisp_rawwr0/1/2 三路通道任一一路，都要需要保证rkisp_mainpath通道已经在stream on状态之后，该三路才会出流。
```

- rkisp_mainpath 格式切换

```
    默认输出1080p，如果要格式切换720p，需要先执行：

    media-ctl -d /dev/media1 --set-v4l2 '"m01_b_nvp6188 1-0032":0[fmt:UYVY8_2X8/1280x720]'
    media-ctl -d /dev/media1 --set-v4l2 '"rkisp-csi-subdev":1[fmt:UYVY8_2X8/1280x720]'
    media-ctl -d /dev/media1 --set-v4l2 '"rkisp-isp-subdev":0[fmt:YUYV8_2X8/1280x720]'
    media-ctl -d /dev/media1 --set-v4l2 '"rkisp-isp-subdev":0[crop:(0,0)/1280x720]'
    media-ctl -d /dev/media1 --set-v4l2 '"rkisp-isp-subdev":2[fmt:YUYV8_2X8/1280x720]'
    media-ctl -d /dev/media1 --set-v4l2 '"rkisp-isp-subdev":2[crop:(0,0)/1280x720]'
```

3.6 通道对应的分辨率查询、视频信号查询

- media-ctl -p -d /dev/mediaX 获取 sensor 的 subdev node name

```
media-ctl -p -d /dev/media1
Media controller API version 4.19.111

Media device information
------------------------
driver          rkisp
model           rkisp0
serial
bus info
hw revision     0x0
driver version  4.19.111

Device topology

....

- entity 92: m01_b_nvp6188 1-0032 (1 pad, 1 link)
             type V4L2 subdev subtype Sensor flags 0
             device node name /dev/v4l-subdev6
        pad0: Source
                [fmt:UYVY8_2X8/1920x1080 field:none]
                -> "rockchip-mipi-dphy-rx":0 [ENABLED]
```

- open通道之前获取分辨率

```
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <sys/time.h>
#include <sys/mman.h>
#include <sys/ioctl.h>
#include <linux/videodev2.h>

#define RKMODULE_MAX_VC_CH      4

struct rkmodule_vc_fmt_info {
    __u32 width[RKMODULE_MAX_VC_CH];
    __u32 height[RKMODULE_MAX_VC_CH];
    __u32 fps[RKMODULE_MAX_VC_CH];
} __attribute__ ((packed));

struct rkmodule_vc_hotplug_info {
    __u8 detect_status;
} __attribute__ ((packed));

#define RKMODULE_GET_VC_FMT_INFO \
    _IOR('V', BASE_VIDIOC_PRIVATE + 12, struct rkmodule_vc_fmt_info)

#define RKMODULE_GET_VC_HOTPLUG_INFO \
    _IOR('V', BASE_VIDIOC_PRIVATE + 13, struct rkmodule_vc_hotplug_info)

int main(int argc, char *argv[]) {
    int ch = 0;
    struct rkmodule_vc_hotplug_info status;
    struct rkmodule_vc_fmt_info fmt;
    int fd = open("/dev/v4l-subdev2", O_RDWR, 0);
    ioctl(fd, RKMODULE_GET_VC_FMT_INFO, &fmt);
    ioctl(fd, RKMODULE_GET_VC_HOTPLUG_INFO, &status);
    for(ch = 0; ch < 4; ch++) {
        printf("# ch: %d\n", ch);
        printf("\t width: %d\n", fmt.width[ch]);
        printf("\t height: %d\n", fmt.height[ch]);
        printf("\t fps: %d\n", fmt.fps[ch]);
        printf("\t plug in: %d\n", (status.detect_status & (1 << ch)) ? 1 : 0);
    }
    close(fd);
    return 0;
}
```

3.7 实时查询热拔插接口

- 提供sysfs节点给用户层进行读查询。

```
/sys/devices/platform/ff510000.i2c/i2c-1/1-0032/hotplug_status
/sys/devices/platform/ff510000.i2c/i2c-1/1-0030/hotplug_status
```

4. 多路音频采集

DVR产品音频一般通过TDM的方式采集到RV1126，可以用如下命令来同时采集8个通道的音频数据：

```
arecord -Dhw:0,0 -c 8 -f S16_LE -r 8000 /tmp/record.wav -vv
```

另外我们可以通过修改/etc/asound.conf来利用dsnooper抽出某一通道数据，方法如下：

假设当前有6路音频通过TDM输入，配置asound.conf如下：

```
# Copyright © 2021 Rockchip Electronics Co. Ltd.
# Author: Xing Zheng <zhengxing@rock-chips.com>

pcm.!default
{
    type asym
    playback.pcm "hw:0,0"
    capture.pcm "hw:0,0"
}

pcm.dsnooper_6 {
    type dsnoop
    ipc_key 5978291 # must be unique for all dmix plugins!!!!
    ipc_key_add_uid yes
    slave {
        pcm "hw:0,0"
        channels 6
    }
    bindings {
        0 0
        1 1
        2 2
        3 3
        4 4
        5 5
    }
}

pcm.cap_ch0_rt {
    type route
    slave {
        pcm dsnooper_6
```

```
        channels 6
    }
    ttable {
        0.0 1.0
    }
}

pcm.cap_ch1_rt {
    type route
    slave {
        pcm dsnooper_6
        channels 6
    }
    ttable {
        0.1 1.0
    }
}

pcm.cap_ch2_rt {
    type route
    slave {
        pcm dsnooper_6
        channels 6
    }
    ttable {
        0.2 1.0
    }
}

pcm.cap_ch3_rt {
    type route
    slave {
        pcm dsnooper_6
        channels 6
    }
    ttable {
        0.3 1.0
    }
}

pcm.cap_ch4_rt {
    type route
    slave {
        pcm dsnooper_6
        channels 6
    }
    ttable {
        0.4 1.0
    }
}

pcm.cap_ch5_rt {
    type route
    slave {
        pcm dsnooper_6
        channels 6
    }
    ttable {
```

```
        0.5 1.0
    }
}

pcm.cap_ch0 {
    type plug
    slave.pcm "cap_ch0_rt"
}

pcm.cap_ch1 {
    type plug
    slave.pcm "cap_ch1_rt"
}

pcm.cap_ch2 {
    type plug
    slave.pcm "cap_ch2_rt"
}

pcm.cap_ch3 {
    type plug
    slave.pcm "cap_ch3_rt"
}

pcm.cap_ch4 {
    type plug
    slave.pcm "cap_ch4_rt"
}

pcm.cap_ch5 {
    type plug
    slave.pcm "cap_ch5_rt"
}
```

修改上述配置后，可以用以下命令尝试单独录制并抽取出第6通道的音频数据（这里标定ch0为第一通道顺序）：

```
arecord -Dcap_ch5 -c 1 -r 16000 -f S16_LE /tmp/record_ch5.wav
```

如果arecord命令可以抽取某一路音频进行录制，可以尝试在APP调用rkmedia的时候，pcm_name用cap_ch5打开，提取第6个channel。同理，其他5个channel的pcm_name用“cap_chX” 打开（这里例子为6ch，X取值为0-5）。

注意：RV1126/RV1109芯片只有I2S0才支持TDM模式。

5. rkmedia_vmix_vo_dvr_test应用说明

rkmedia_vmix_vo_dvr_test主要实现8路视频采集、编码，8路视频合成显示。源代码位于SDK/external/rkmedia/examples。

5.1 支持8路视频采集、H264编码

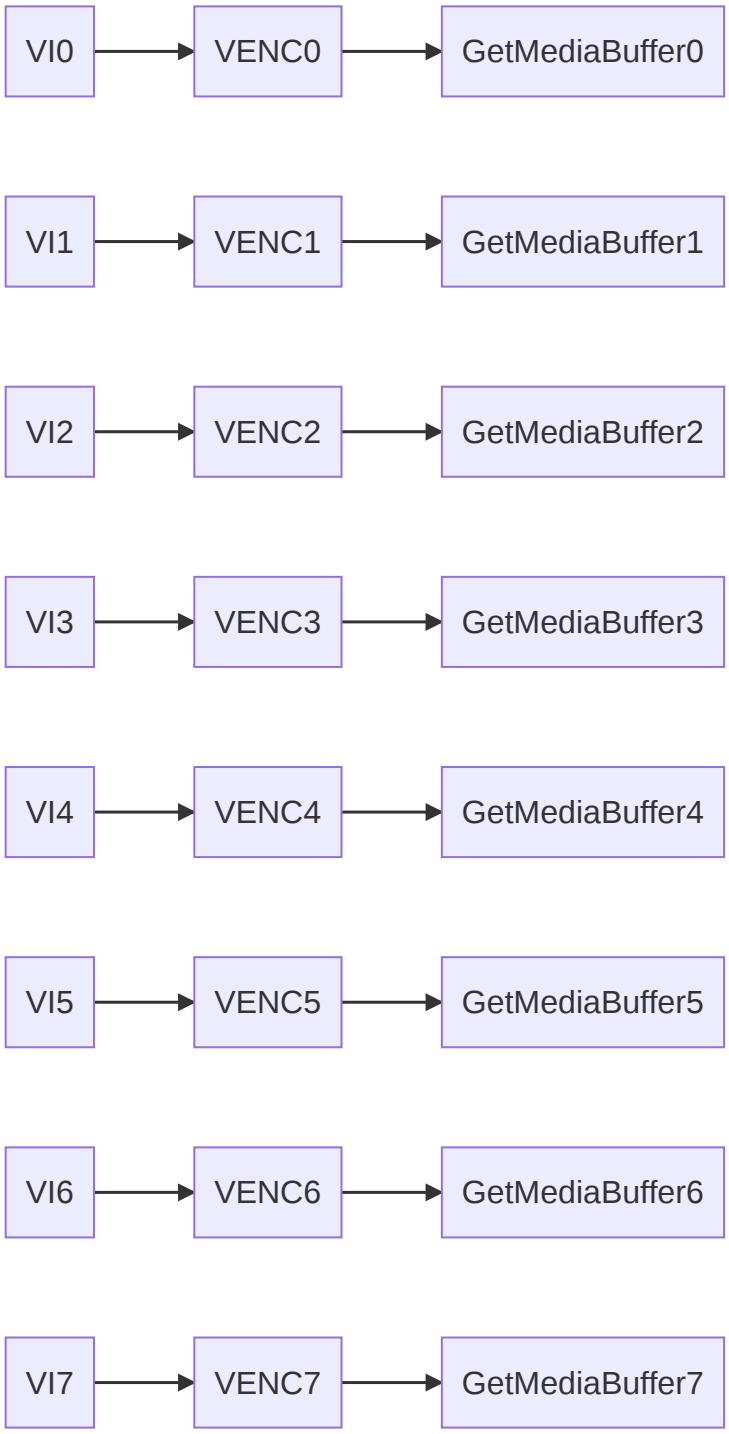
8路视频采集节点、分辨率、格式通过数组配置，方便用户修改调试：

```
stDvr dvr8[8] = {
    {0, "/dev/video30", 1920, 1080, IMAGE_TYPE_NV12, 0},
    {1, "/dev/video31", 1920, 1080, IMAGE_TYPE_NV12, 0},
    {2, "/dev/video32", 1920, 1080, IMAGE_TYPE_NV12, 0},
    {3, "/dev/video33", 1920, 1080, IMAGE_TYPE_NV12, 0},
    {4, "/dev/video37", 1920, 1080, IMAGE_TYPE_NV12, 0},
    {5, "/dev/video38", 1920, 1080, IMAGE_TYPE_NV12, 0},
    {6, "/dev/video39", 1920, 1080, IMAGE_TYPE_NV12, 0},
    {7, "/dev/video40", 1920, 1080, IMAGE_TYPE_NV12, 0},
};
```

根据双mipi方案的推荐，需要修改为：

```
stDvr dvr8[8] = {
    {0, "/dev/video0", 1920, 1080, IMAGE_TYPE_NV16, 0},
    {1, "/dev/video1", 1920, 1080, IMAGE_TYPE_NV16, 0},
    {2, "/dev/video2", 1920, 1080, IMAGE_TYPE_NV16, 0},
    {3, "/dev/video3", 1920, 1080, IMAGE_TYPE_NV16, 0},
    {4, "/dev/video5", 1920, 1080, IMAGE_TYPE_UYVY422, 0},
    {5, "/dev/video7", 1920, 1080, IMAGE_TYPE_UYVY422, 0},
    {6, "/dev/video8", 1920, 1080, IMAGE_TYPE_UYVY422, 0},
    {7, "/dev/video9", 1920, 1080, IMAGE_TYPE_UYVY422, 0},
};
```

8路视频VI通过bind VENC实现8路H264编码，通过GetMediaBuffer线程可以获取到8路VENC编码后的数据，用户可以在这个基础上实现视频传输需求。

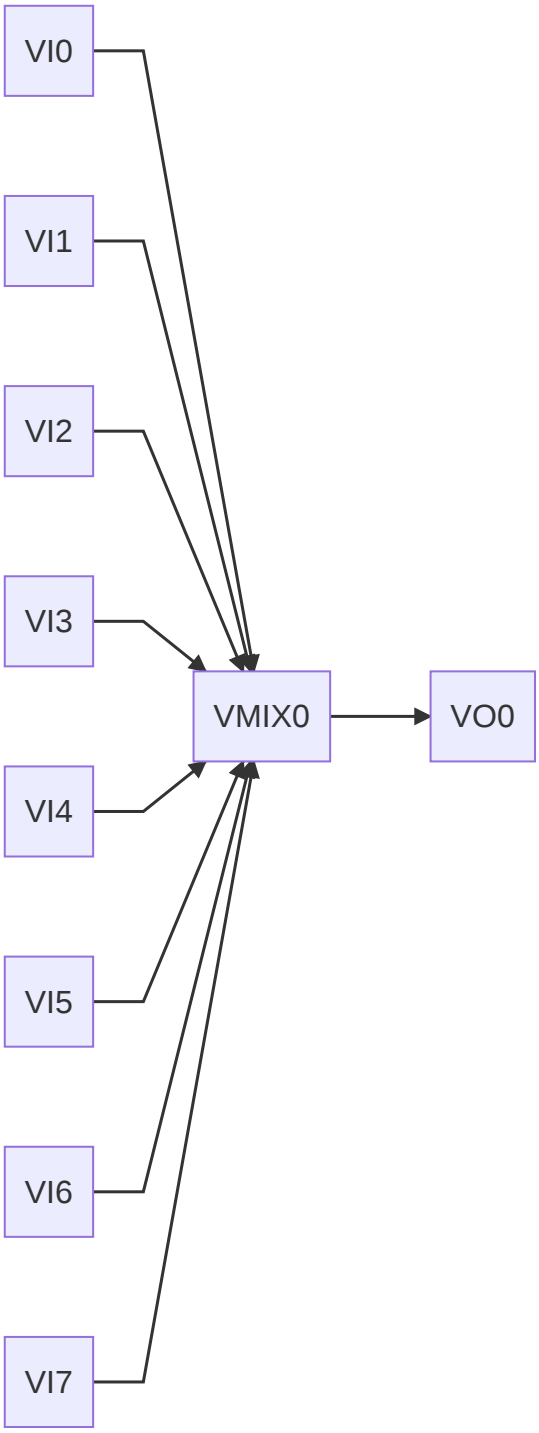


5.2 支持8路视频合成显示

8路视频通过数组指定屏幕显示矩形区域，方便用户修改调试：

```
RECT_S area_2x4[8] = {
    {0, 0, WIDTH / 2, HEIGHT / 4},
    {WIDTH / 2, 0, WIDTH / 2, HEIGHT / 4},
    {0, HEIGHT / 4, WIDTH / 2, HEIGHT / 4},
    {WIDTH / 2, HEIGHT / 4, WIDTH / 2, HEIGHT / 4},
    {0, HEIGHT / 2, WIDTH / 2, HEIGHT / 4},
    {WIDTH / 2, HEIGHT / 2, WIDTH / 2, HEIGHT / 4},
    {0, HEIGHT * 3 / 4, WIDTH / 2, HEIGHT / 4},
    {WIDTH / 2, HEIGHT * 3 / 4, WIDTH / 2, HEIGHT / 4},
};
```

8路视频合成显示通过VMIX+VO模块实现：



5.3 支持8路视频切换为前4路、后4路显示

通过dvr_bind、dvr_unbind实现8路视频切换为前4路、后4路显示，用户只需要定义前4路和后4路的矩形显示区域即可。

5.4 支持区域画框

通过对整个屏幕画线实现区域画框，增加区域边界，通过数组指定画线区域，线宽最小为2，要求偶数：

```
RECT_S line_2x4[4] = {
    {0, HEIGHT / 4, WIDTH, 2},
    {0, HEIGHT / 2, WIDTH, 2},
    {0, HEIGHT * 3 / 4, WIDTH, 2},
    {WIDTH / 2, 0, 2, HEIGHT},
};
```

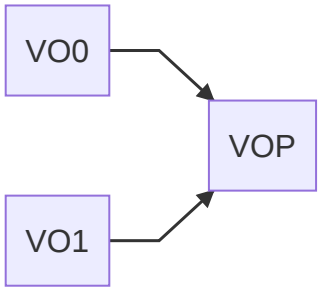
通过RK_MPI_VMIX_SetLineInfo设置画线区域。

5.5 支持RGN Cover

支持对每个通道设置敏感区域，通过RK_MPI_VMIX_RGN_SetCover实现。

5.6 支持屏幕OSD

通过VO1实现屏幕OSD，用户可以把OSD绘制在buffer（格式为ABGR）里面，送VO1后通过Alpha即可以实现OSD叠加在VO0视频上面显示的效果。应用里面在osd_thread线程里面每隔500ms绘制一块两条色块的OSD切换显示。



5.7 支持通道显示、隐藏

通过RK_MPI_VMIX_ShowChn、RK_MPI_VMIX_HideChn实现通道显示、隐藏。

5.8 支持通道的区域亮度获取

通过RK_MPI_VMIX_GetChnRegionLuma实现通道的区域亮度获取，每次最多可以获取64个区域亮度，每个通道的区域的坐标都是相对通道的区域起始坐标，不是相对屏幕的起始坐标。可以通过区域亮度实现屏幕OSD反色效果。

6. VP模块介绍

在DVR/DMS产品中，RGA的使用非常频繁，为了缓解RGA的压力，我们将RV1126芯片中的ISPP模块缩放功能利用起来。因此我们在rkmedia提供了一个VP模块，通过VP的接口可以使用ISPP的缩放功能。

VP详细使用文档请参考文档：
docs/RV1126_RV1109/Multimedia/Rockchip_Developer_Guide_Linux_RKMedia_CN.pdf。

需要注意的是：ISPP的缩放功能对Buffer的宽度有限制，需要16Byte对齐。

7. 产品系统性能优化说明

由于DVR产品业务逻辑较为复杂，可能同时存在多路主子码流同时编码、拍照、OSD、算法处理等需求，系统性能常常会遇到瓶颈。因此，本章节总结了目前常用的一些系统性能优化手段，客户可以根据自己产品的实际情况导入。

7.1 CPU性能优化方法

CPU性能优化之前，可以用top、perf等常用工具来分析CPU的占用情况，把可以优化的地方先进行优化。另外，应用上也要尽量限制无效Log打印，Log输出也是需要占用CPU资源。

7.1.1 关闭高精度定时器

关闭高精度定时器配置，可以降低CPU使用率。关闭高精度定时器需要修改kernel的配置，修改方法如下：

```
-CONFIG_HIGH_RES_TIMERS
```

修改配置后需要重新编译内核。

注意：高精度定时器关闭后，可能会影响NPU驱动加载，如果NPU驱动加载失败报错，需要在Redmine申请一份兼容高精度定时器关闭的NPU驱动。

7.1.2 ISPP关闭IOMMU

IOMMU可以将不连续的物理内存映射成连续的物理内存给硬件使用。ISPP默认会使能IOMMU，使用IOMMU，在进行内存映射时会有额外的CPU开销。可以通过配置一块连续的物理内存给ISPP使用，来达到关闭IOMMU的目的。以下补丁是修改范例：

```
diff --git a/arch/arm/boot/dts/xxx.dts b/arch/arm/boot/dts/xxx.dts
index e2501ed7f996..360169b7fdd1 100644
--- a/arch/arm/boot/dts/xxx.dts
+++ b/arch/arm/boot/dts/xxx.dts
@@ -318,7 +318,15 @@
 };

 &isp_reserved {
-    size = <0x06400000>;
+    size = <0x0b000000>;
+};
+
+&rkispp_mmu {
+       status = "disabled";
+};
+
+&rkispp {
+       memory-region = <&isp_reserved>;
 };

 &mipi_csi2 {
```

注意：isp_reserved中预留的连续物理内存大小，不同的产品使用大小不同，需要根据产品需求进行合理配置。

7.2 RGA性能优化方法

7.2.1 DMA Buffer Cache使能

DVR产品需要用RGA进行大量的缩放、裁剪等操作。经过RGA处理的Buffer在每次操作之前会进行Buffer映射，消耗CPU资源。本章节提供一种方法可以缓存RGA DMA Buffer的映射信息，从而减少CPU资源消耗。

DMA Buffer Cache在V3.0之后的SDK是默认使能的，如果是更早的SDK，可以通过应用以下补丁来达到使能DMA Buffer Cache的目的（说明：以下3个补丁都需要，并且要按照序号依次、逐个打上）。

1. 补丁1：0001-dma-buf-dma-buf-cache-fix-error-case-for-attach-deta.patch

```
From 0423f7d60199d4aaf015924d6dd5983ff90b5998 Mon Sep 17 00:00:00 2001
From: Jianqun Xu <jay.xu@rock-chips.com>
Date: Tue, 28 Dec 2021 14:42:56 +0800
Subject: [PATCH] dma-buf: dma-buf-cache: fix error case for attach / detach

Signed-off-by: Jianqun Xu <jay.xu@rock-chips.com>
Change-Id: I84ff3ac7c1357416bb12ca61aa7134fc652538d6
---
 drivers/dma-buf/dma-buf-cache.c | 13 ++++++++-----
```

```
 1 file changed, 8 insertions(+), 5 deletions(-)

diff --git a/drivers/dma-buf/dma-buf-cache.c b/drivers/dma-buf/dma-buf-cache.c
index 79f35d359241..7bfa6a5e5fc4 100644
--- a/drivers/dma-buf/dma-buf-cache.c
+++ b/drivers/dma-buf/dma-buf-cache.c
@@ -29,7 +29,7 @@ static int dma_buf_cache_destructor(struct dma_buf *dmabuf,
void *dtor_data)

    mutex_lock(&data->lock);
    list_for_each_entry_safe(cache, tmp, &data->head, list) {
-       if (cache->sg_table)
+       if (!IS_ERR_OR_NULL(cache->sg_table))
            dma_buf_unmap_attachment(cache->attach,
                         cache->sg_table,
                         cache->direction);
@@ -83,6 +83,7 @@ EXPORT_SYMBOL(dma_buf_cache_detach);
 struct dma_buf_attachment *dma_buf_cache_attach(struct dma_buf *dmabuf,
                        struct device *dev)
 {
+   struct dma_buf_attachment *attach;
    struct dma_buf_cache_list *data;
    struct dma_buf_cache *cache;

@@ -117,8 +118,13 @@ struct dma_buf_attachment *dma_buf_cache_attach(struct
dma_buf *dmabuf,
        return ERR_PTR(-ENOMEM);

    /* Cache attachment */
-   cache->attach = dma_buf_attach(dmabuf, dev);
+   attach = dma_buf_attach(dmabuf, dev);
+   if (IS_ERR_OR_NULL(attach)) {
+       kfree(cache);
+       return attach;
+   }

+   cache->attach = attach;
    mutex_lock(&data->lock);
    list_add(&cache->list, &data->head);
    mutex_unlock(&data->lock);
@@ -163,9 +169,6 @@ struct sg_table *dma_buf_cache_map_attachment(struct
dma_buf_attachment *attach,
    cache->sg_table = dma_buf_map_attachment(attach, direction);
    cache->direction = direction;

-   if (!cache->sg_table)
-       return ERR_PTR(-ENOMEM);
-
    return cache->sg_table;
 }
 EXPORT_SYMBOL(dma_buf_cache_map_attachment);
```

2. 补丁2：0001-dma-buf-support-to-cache-dma-buf-attachment.patch

```
From 04273ec4b3523ed3420e16828b7731c20ec3a167 Mon Sep 17 00:00:00 2001
From: Jianqun Xu <jay.xu@rock-chips.com>
Date: Fri, 9 Jul 2021 10:00:21 +0800
```

```
Subject: [PATCH] dma-buf: support to cache dma-buf-attachment

This patch try to fix this issue by caching the dma-buf attachments and
stores the cache list to dtor_data of dma-buf structor. The dma-buf
attach with cache will try to find cached attachment first and return
the valid instance.

This patch also store the deattch operation to dtor of dma-buf structor
by dma_buf_set_destructor.

Change-Id: I4778c3328825f6c04f5d2608994e62fe3478bf1b
Signed-off-by: Jianqun Xu <jay.xu@rock-chips.com>
(cherry picked from commit 36514da674cb71553472a66704eae5981035c44c)
---
 drivers/dma-buf/Kconfig         |   7 ++
 drivers/dma-buf/Makefile        |   1 +
 drivers/dma-buf/dma-buf-cache.c | 171 ++++++++++++++++++++++++++++++++++++++++
 include/linux/dma-buf-cache.h   |  32 ++++++
 4 files changed, 211 insertions(+)
 create mode 100644 drivers/dma-buf/dma-buf-cache.c
 create mode 100644 include/linux/dma-buf-cache.h

diff --git a/drivers/dma-buf/Kconfig b/drivers/dma-buf/Kconfig
index cc70e8bee874..414a05a8b7aa 100644
--- a/drivers/dma-buf/Kconfig
+++ b/drivers/dma-buf/Kconfig
@@ -1,5 +1,12 @@
 menu "DMABUF options"

+config DMABUF_CACHE
+   bool "DMABUF cache attachment"
+   default n
+   help
+     This option support to store attachments in a list and destroy them by
+     set to a callback list in the dtor of dma-buf.
+
 config SYNC_FILE
    bool "Explicit Synchronization Framework"
    default n
diff --git a/drivers/dma-buf/Makefile b/drivers/dma-buf/Makefile
index c33bf8863147..87048e5d5aba 100644
--- a/drivers/dma-buf/Makefile
+++ b/drivers/dma-buf/Makefile
@@ -1,3 +1,4 @@
 obj-y := dma-buf.o dma-fence.o dma-fence-array.o reservation.o seqno-fence.o
+obj-$(CONFIG_DMABUF_CACHE) += dma-buf-cache.o
 obj-$(CONFIG_SYNC_FILE)        += sync_file.o
 obj-$(CONFIG_SW_SYNC)      += sw_sync.o sync_debug.o
diff --git a/drivers/dma-buf/dma-buf-cache.c b/drivers/dma-buf/dma-buf-cache.c
new file mode 100644
index 000000000000..79f35d359241
--- /dev/null
+++ b/drivers/dma-buf/dma-buf-cache.c
@@ -0,0 +1,171 @@
+// SPDX-License-Identifier: GPL-2.0
+/*
+ * Copyright (c) 2021 Rockchip Electronics Co. Ltd.
+ */
```

```
+
+#include <linux/slab.h>
+#include <linux/dma-buf.h>
+#undef CONFIG_DMABUF_CACHE
+#include <linux/dma-buf-cache.h>
+
+struct dma_buf_cache_list {
+   struct list_head head;
+   struct mutex lock;
+};
+
+struct dma_buf_cache {
+   struct list_head list;
+   struct dma_buf_attachment *attach;
+   enum dma_data_direction direction;
+   struct sg_table *sg_table;
+};
+
+static int dma_buf_cache_destructor(struct dma_buf *dmabuf, void *dtor_data)
+{
+   struct dma_buf_cache_list *data;
+   struct dma_buf_cache *cache, *tmp;
+
+   data = dmabuf->dtor_data;
+
+   mutex_lock(&data->lock);
+   list_for_each_entry_safe(cache, tmp, &data->head, list) {
+       if (cache->sg_table)
+           dma_buf_unmap_attachment(cache->attach,
+                        cache->sg_table,
+                        cache->direction);
+
+       dma_buf_detach(dmabuf, cache->attach);
+       list_del(&cache->list);
+       kfree(cache);
+
+   }
+   mutex_unlock(&data->lock);
+
+   kfree(data);
+   return 0;
+}
+
+static struct dma_buf_cache *
+dma_buf_cache_get_cache(struct dma_buf_attachment *attach)
+{
+   struct dma_buf_cache_list *data;
+   struct dma_buf_cache *cache;
+   struct dma_buf *dmabuf = attach->dmabuf;
+
+   if (dmabuf->dtor != dma_buf_cache_destructor)
+       return NULL;
+
+   data = dmabuf->dtor_data;
+
+   mutex_lock(&data->lock);
+   list_for_each_entry(cache, &data->head, list) {
+       if (cache->attach == attach) {
```

```
+           mutex_unlock(&data->lock);
+           return cache;
+       }
+   }
+   mutex_unlock(&data->lock);
+
+   return NULL;
+}
+
+void dma_buf_cache_detach(struct dma_buf *dmabuf,
+             struct dma_buf_attachment *attach)
+{
+   struct dma_buf_cache *cache;
+
+   cache = dma_buf_cache_get_cache(attach);
+   if (!cache)
+       dma_buf_detach(dmabuf, attach);
+}
+EXPORT_SYMBOL(dma_buf_cache_detach);
+
+struct dma_buf_attachment *dma_buf_cache_attach(struct dma_buf *dmabuf,
+                       struct device *dev)
+{
+   struct dma_buf_cache_list *data;
+   struct dma_buf_cache *cache;
+
+   if (!dmabuf->dtor) {
+       data = kzalloc(sizeof(*data), GFP_KERNEL);
+       if (!data)
+           return ERR_PTR(-ENOMEM);
+
+       mutex_init(&data->lock);
+       INIT_LIST_HEAD(&data->head);
+
+       dma_buf_set_destructor(dmabuf, dma_buf_cache_destructor, data);
+   }
+
+   if (dmabuf->dtor && dmabuf->dtor != dma_buf_cache_destructor)
+       return dma_buf_attach(dmabuf, dev);
+
+   data = dmabuf->dtor_data;
+
+   mutex_lock(&data->lock);
+   list_for_each_entry(cache, &data->head, list) {
+       if (cache->attach->dev == dev) {
+           /* Already attached */
+           mutex_unlock(&data->lock);
+           return cache->attach;
+       }
+   }
+   mutex_unlock(&data->lock);
+
+   cache = kzalloc(sizeof(*cache), GFP_KERNEL);
+   if (!cache)
+       return ERR_PTR(-ENOMEM);
+
+   /* Cache attachment */
+   cache->attach = dma_buf_attach(dmabuf, dev);
```

```
+
+   mutex_lock(&data->lock);
+   list_add(&cache->list, &data->head);
+   mutex_unlock(&data->lock);
+
+   return cache->attach;
+}
+EXPORT_SYMBOL(dma_buf_cache_attach);
+
+void dma_buf_cache_unmap_attachment(struct dma_buf_attachment *attach,
+                   struct sg_table *sg_table,
+                   enum dma_data_direction direction)
+{
+   struct dma_buf_cache *cache;
+
+   cache = dma_buf_cache_get_cache(attach);
+   if (!cache)
+       dma_buf_unmap_attachment(attach, sg_table, direction);
+}
+EXPORT_SYMBOL(dma_buf_cache_unmap_attachment);
+
+struct sg_table *dma_buf_cache_map_attachment(struct dma_buf_attachment *attach,
+                         enum dma_data_direction direction)
+{
+   struct dma_buf_cache *cache;
+
+   cache = dma_buf_cache_get_cache(attach);
+   if (!cache)
+       return dma_buf_map_attachment(attach, direction);
+
+   if (cache->sg_table) {
+       /* Already mapped */
+       if (cache->direction == direction)
+           return cache->sg_table;
+
+       /* Different directions */
+       dma_buf_unmap_attachment(attach, cache->sg_table,
+                    cache->direction);
+
+   }
+
+   /* Cache map */
+   cache->sg_table = dma_buf_map_attachment(attach, direction);
+   cache->direction = direction;
+
+   if (!cache->sg_table)
+       return ERR_PTR(-ENOMEM);
+
+   return cache->sg_table;
+}
+EXPORT_SYMBOL(dma_buf_cache_map_attachment);
diff --git a/include/linux/dma-buf-cache.h b/include/linux/dma-buf-cache.h
new file mode 100644
index 000000000000..d97545560990
--- /dev/null
+++ b/include/linux/dma-buf-cache.h
@@ -0,0 +1,32 @@
+/* SPDX-License-Identifier: GPL-2.0 */
```

```
+/*
+ * Copyright (c) 2021 Rockchip Electronics Co. Ltd.
+ */
+#ifndef _LINUX_DMA_BUF_CACHE_H
+#define _LINUX_DMA_BUF_CACHE_H
+
+#include <linux/dma-buf.h>
+
+extern void dma_buf_cache_detach(struct dma_buf *dmabuf,
+                struct dma_buf_attachment *attach);
+
+extern void dma_buf_cache_unmap_attachment(struct dma_buf_attachment *attach,
+                      struct sg_table *sg_table,
+                      enum dma_data_direction direction);
+
+extern struct dma_buf_attachment *
+dma_buf_cache_attach(struct dma_buf *dmabuf, struct device *dev);
+
+extern struct sg_table *
+dma_buf_cache_map_attachment(struct dma_buf_attachment *attach,
+                enum dma_data_direction direction);
+
+#ifdef CONFIG_DMABUF_CACHE
+/* Replace dma-buf apis to cached apis */
+#define dma_buf_attach dma_buf_cache_attach
+#define dma_buf_detach dma_buf_cache_detach
+#define dma_buf_map_attachment dma_buf_cache_map_attachment
+#define dma_buf_unmap_attachment dma_buf_cache_unmap_attachment
+#endif
+
+#endif /* _LINUX_DMA_BUF_CACHE_H */
```

3. 补丁3：

```
— a/drivers/video/rockchip/rga2/rga2_drv.c
+++ b/drivers/video/rockchip/rga2/rga2_drv.c
@@ -45,7 +45,7 @@

 #if (LINUX_VERSION_CODE >= KERNEL_VERSION(4, 4, 0))
 #include <linux/pm_runtime.h>
-#include <linux/dma-buf.h>
+#include <linux/dma-buf-cache.h>
 #endif
```

内核打开配置：

```
CONFIG_DMABUF_CACHE=y
```

可以加Log打印确认RGA DMA-Cache是否生效：

```
--- a/drivers/dma-buf/dma-buf-cache.c
+++ b/drivers/dma-buf/dma-buf-cache.c
@@ -61,6 +61,7 @@ dma_buf_cache_get_cache(struct dma_buf_attachment *attach)
        list_for_each_entry(cache, &data->head, list) {
                if (cache->attach == attach) {
                        mutex_unlock(&data->lock);
+                       printk("%s %d\n", __func__, __LINE__);
                        return cache;
                }
        }
```

注意：确认RGA的DMA-Cache生效后，需要将调试Log删除。

8. FAQ

8.1 热拔插出现画面错开

8.1.1 问题分析

此问题出现的主要原因是当摄像头重新插入后AD转换芯片无法保证数据流从Frame Start(FS)开始输出，而主控VICAP要求输入帧的完整性，因此导致了分层的现象；

8.1.2 解决方法

解决此类问题，只能通过复位主控VICAP来实现，需要在dts添加如下配置：

```
&rkcif {
    ....
    rockchip,cif-monitor = <3 200 1000 5 0>;
    ....
};
```

参数从左至右依次：

index0：复位模式，有以下几种：

0：idle，即不启动复位模式；

1：continue，用于实时连续监测mipi出错及断流，如有出错或断流时复位vicap；

2：trigger，用于监测是否有csi报错，如有出错或断流时复位vicap；

3：hotplug，用于监测AD芯片热拔插，如有热拔插时复位vicap；

index1：定时器监测复位周期，以毫秒为单位；

index2：在发现vicap相关报错后，在该定义时间内，持续对监测，不立即复位，不再新增报错或超时再进行复位，时间单位毫秒

index3：用于设定mipi csi err的出现次数，在达到该次数后，触发复位；

index4：用户设置复位标志，如设置为0，则vicap检测到复位标志后，自主复位，不受应用控制；如使能为1，则需要应用开启一个线程，检测驱动的复位标志is_need_reset，当is_need_reset为1时，表明此时vicap需要复位，调用RKCIF_CMD_SET_RESET这个IOCTL来控制vicap复位；如果拔插后接入的AHD摄像头分辨率有变化，那么此参数必须置为1，由应用控制vicap复位，具体流程如下：1）先关闭该通道的数据流；2）重新设置新接入AHD摄像头的分辨率；3）应用控制vicap复位；4）重新开启该通道数据流；

8.2 CVBS奇偶场合成功能说明

RV1126平台仅VICAP支持奇偶场合成（即标清摄像头D1输入），RKISP只有selfpath支持奇偶场输出，不适用于多路D1摄像头输入，因此不推荐RKISP接入多路D1摄像头。针对MIPI接口，VICAP支持D1和AHD混接；针对DVP接口，VICAP只能支持四路D1或者四路AHD，不支持D1和AHD混接。

当需要支持D1和AHD混接时，需要加上如下补丁：

```
diff --git a/drivers/media/media-entity.c b/drivers/media/media-entity.c
index 3498551e6..e19545041 100644
--- a/drivers/media/media-entity.c
+++ b/drivers/media/media-entity.c
@@ -484,7 +484,7 @@ __must_check int __media_pipeline_start(struct media_entity
*entity,
                    link->source->entity->name,
                    link->source->index,
                    entity->name, link->sink->index, ret);
-               goto error;
+               //goto error;
            }
        }
```