Rockchip Gstreamer用户指南

文件标识：RK-YH-YF-921

发布版本：V1.1.1

日期：2022-07-26

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

概述

本文档主要介绍 Gstreamer及相关插件的编译和测试方法。

产品版本

芯片名称	版本
RK356X	1.14.x
RK3588	1.18.x

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
2022-01-06	V1.0.0	Jair Wu	初始版本
2022-02-24	V1.0.1	Jair Wu	修复错误的命令选项
2022-05-10	V1.1.0	Jair Wu	新增MPP插件和环境变量说明，增加命令示例
2022-07-26	V1.1.1	LGZ	新增Gstreamer简介及AFBC dump解码数据

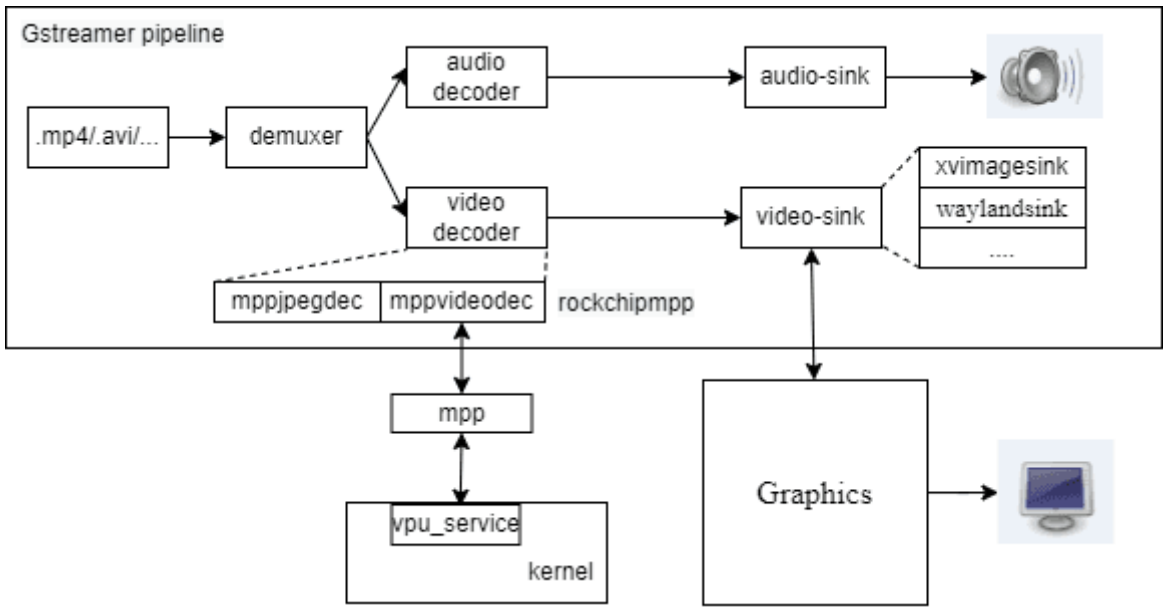
目录

1. GStreamer 简介

GStreamer是一个开源多媒体框架，目前Linux SDK（除了IPC外）的多媒体都主要用GStreamer来对接 app 和 编解码组件。利用GStreamer插件的强大特性，通过编写的GStreamer插件适配Rockchip硬件，使得app能够使用硬件编解码进行加速。

1.1 GStreamer 视频编解码适配方案

如下图所示，以视频播放为例，说明下Rockchip平台视频编解码和显示基本流程：



视频文件（如mp4）先经过demuxer解封装为视频(如h264,h265编码)和音频流，视频流经过解码器video decoder（如mppvideodec和mppjpegdec）解码；音频流经过音频解码器audio decoder解码，最后通过显示插件（如xvimagesink，waylandsink等）将解码后的视频数据送显，通过音频播放插件（alsasink等）将解码后的音频数据送入声卡播放声音。

Rockchip平台针对视频编解码实现硬件加速，通过插件rockchipmpp实现，其中包括解码插件：mppvideodec和mppjpegdec；编码插件：mpph264enc，mppvp8enc，mppjpegenc等。GStreamer在视频解码阶段会优先调用rockchipmpp插件，大致流程如下：mppvideodec等插件调用MPP提供的接口，MPP是Rockchip平台的视频编解码中间件会调用vpu驱动（vpu_service）。硬件编解码功能也可直接通过MPP提供测试接口进行测试 (比如mpi_dec_test\mpi_enc_test...)。

解码后的视频数据经过显示插件（如xvimagesink，waylandsink等）送入显示设备进行显示，不同显示插件调用不同显示架构接口以对接不同的显示架构，如xvimagesink会调用X11接口对接X11显示架构，waylandsink调用Wayland接口对接Wayland显示架构等。

显示相关具体参考`SDK文档Rockchip_Developer_Guide_Linux_Graphics_CN.pdf`

MPP源码参考 `<SDK>/external/mpp/`，MPP相关说明文档参考 `<SDK>/docs/Linux/Multimedia/Rockchip_Developer_Guide_MPP_CN.pdf`

测试demo参考: `<SDK>/external/mpp/test`

2. 源码及编译

2.1 源码路径

Buildroot：

Gstreamer及相关插件的源码均通过网络下载，再打上我们提供的补丁的方式生成，具体可以查看`<SDK>/buildroot/package/gstreamer1/`。

Debian：

Debian版本源码可通过Debian仓库查找下载，并在对应的版本打上补丁，补丁路径见`<SDK>/buildroot/package/gstreamer1/<submodule>/<version>/*.patch`，目前主要提供1.18.5和1.20.0两个版本。

Gstreamer-rockchip：

MPP编解码插件及rkximagesink显示插件源码在`<SDK>/external/gstreamer-rockchip`，Buildroot与Debian共用同一仓库。

2.2 编译

Buildroot：

开启相关宏（默认开启），直接在SDK根目录编译即可，相关宏均统一整理至`<SDK>/buildroot/configs/rockchip/*_gst.config`，在目标config里直接包含即可。支持选择编译版本，如`BR2_PACKAGE_GSTREAMER1_18`和`BR2_PACKAGE_GSTREAMER1_20`。

```
BR2_PACKAGE_MPP=y
BR2_PACKAGE_MPP_ALLOCATOR_DRM=y
BR2_PACKAGE_GSTREAMER1_ROCKCHIP=y
BR2_PACKAGE_LINUX_RGA=y
BR2_PACKAGE_CA_CERTIFICATES=y
BR2_PACKAGE_LIBSOUP_SSL=y
BR2_PACKAGE_GSTREAMER1=y
BR2_PACKAGE_GST1_PLUGINS_BASE=y
BR2_PACKAGE_GST1_PLUGINS_BASE_PLUGIN_ALSA=y
BR2_PACKAGE_GST1_PLUGINS_BASE_PLUGIN_VIDEOCONVERT=y
BR2_PACKAGE_GST1_PLUGINS_BASE_PLUGIN_VIDEOTESTSRC=y
BR2_PACKAGE_GST1_PLUGINS_GOOD=y
BR2_PACKAGE_GST1_PLUGINS_GOOD_PLUGIN_AUDIOPARSERS=y
BR2_PACKAGE_GST1_PLUGINS_GOOD_PLUGIN_AUTODETECT=y
BR2_PACKAGE_GST1_PLUGINS_GOOD_PLUGIN_DEINTERLACE=y
BR2_PACKAGE_GST1_PLUGINS_GOOD_PLUGIN_FLV=y
BR2_PACKAGE_GST1_PLUGINS_GOOD_PLUGIN_GDKPIXBUF=y
BR2_PACKAGE_GST1_PLUGINS_GOOD_PLUGIN_MATROSKA=y
BR2_PACKAGE_GST1_PLUGINS_GOOD_PLUGIN_MPG123=y
BR2_PACKAGE_GST1_PLUGINS_GOOD_PLUGIN_SOUPHTTPSRC=y
BR2_PACKAGE_GST1_PLUGINS_BAD=y
BR2_PACKAGE_GST1_PLUGINS_BAD_PLUGIN_DVBSUBOVERLAY=y
BR2_PACKAGE_GST1_PLUGINS_BAD_PLUGIN_DVDSPU=y
BR2_PACKAGE_GST1_PLUGINS_BAD_PLUGIN_JPEGFORMAT=y
BR2_PACKAGE_GST1_PLUGINS_BAD_PLUGIN_KMS=y
BR2_PACKAGE_GST1_PLUGINS_BAD_PLUGIN_MPEGDEMUX=y
```

```
BR2_PACKAGE_GST1_PLUGINS_BAD_PLUGIN_MPEG2ENC=y
BR2_PACKAGE_GST1_PLUGINS_BAD_PLUGIN_VIDEOPARSERS=y
BR2_PACKAGE_GST1_PLUGINS_BAD_PLUGIN_ADPCMDEC=y
BR2_PACKAGE_GST1_PLUGINS_BAD_PLUGIN_ADPCMENC=y
BR2_PACKAGE_GST1_PLUGINS_BAD_PLUGIN_FAAD=y
BR2_PACKAGE_GST1_PLUGINS_UGLY=y
BR2_PACKAGE_GST1_PLUGINS_UGLY_PLUGIN_ASFDEMUX=y
BR2_PACKAGE_GST1_PLUGINS_UGLY_PLUGIN_DVDLPCMDEC=y
BR2_PACKAGE_GST1_PLUGINS_UGLY_PLUGIN_DVDSUB=y
BR2_PACKAGE_GST1_PLUGINS_UGLY_PLUGIN_MPEG2DEC=y
...
```

完整插件列表可进入menuconfig->Target packages->Audio and video applications->gstreamer 1.x查看。

Debian：

需要将源码放至板端，并确认源码根目录下存在`debian`目录。进入源码根目录，执行：

```
# 1 更新软件源
apt update
# 2 安装依赖库
apt build-dep .
# 3 可选：开始编译deb安装包
dpkg-buildpackage -b -d -uc -us
# 编译完成后会在上一级目录生成deb安装包，使用dpkg -i xxx.deb即可安装。
# 3 可选：编译并安装
meson build && ninja -C build install
```

通常建议使用第一种即编译deb安装包的方式，可以保证编译，安装等选项统一。

注意：某些编译选项依赖于`video-format.h`等头文件内的宏定义，因此需要先安装`libgstreamer-plugins-base1.0-dev`包，保证`video-format.h`等头文件最新，从而保证某些功能开启。部分插件的编译依赖于系统环境，如发现缺少插件，可检查编译脚本和日志，安装依赖库后重新编译，并确保在debian/*.install文件中有包含目标库。

3. 常用命令

- gst-launch-1.0

  Gstreamer启动器，用于快速构建pipeline，示例如下：

```
# 使用videotestsrc生成一段视频，并使用xvimagesink显示
gst-launch-1.0 videotestsrc ! xvimagesink
```

- gst-play-1.0

  Gstreamer播放器，用于播放各种流媒体，示例如下：

```
# 播放test.mp4，并通过xvimagesink显示
gst-play-1.0 test.mp4 --videosink=xvimagesink
# 常用命令选项
--flags          # bit0:视频，bit1:音频,bit2:字幕，如--flags=1表示只播放视频
--videosink      # 指定videosink
--audiosink      # 指定audiosink
--use-playbin3   # 使用playbin3，否则使用playbin2
```

- gst-inspect-1.0

  查找器，用于列出所有插件或某一插件的具体信息，示例如下：

```
# 不带任何参数，列出所有插件
gst-inspect-1.0
# 列出xvimagesink插件的所有信息
gst-inspect-1.0 xvimagesink
```

- 开启日志功能

```
#设置环境变量
export GST_DEBUG=2
#或在命令前指定，命令结束即失效
GST_DEBUG=2 gst-play-1.0 ...

#指定不同模块不同日志等级，支持通配符，fpsdisplaysink指定为DEBUG（5），xvimage*指定为FIXME（3），其他指定为WARNING（2）
GST_DEBUG=2,fpsdisplaysink:5,xvimage*:3
```

日志等级分为ERROR(1)，WARNING(2)，FIXME(3)，INFO(4)，DEBUG(5)，LOG(6)，TRACE(7)等。

4. 常用插件

4.1 Source

指可以产生数据但不能接收数据的插件。

- filesrc

  从文件读取数据，示例如下：

```
gst-launch-1.0 filesrc location=/tmp/test ! filesink location=/tmp/test2
```

- videotestsrc

  生成视频数据，示例如下：

```
# 使用默认格式输出视频
gst-launch-1.0 videotestsrc ! xvimagesink
# 使用指定格式输出视频
gst-launch-1.0 videotestsrc ! "video/x-raw,width=1920,height=1080,format=(string)NV12" ! xvimagesink
```

- v4l2src

  从摄像头获取视频数据，示例如下：

```
gst-launch-1.0 v4l2src ! video/x-raw,width=1920,height=1080,format=NV12 !
waylandsink
```

- rtspsrc

  从RTSP服务器中获取视频流，示例如下：

```
gst-launch-1.0 rtspsrc location=rtsp://192.168.1.105:8554/ ! rtph264depay !
h264parse ! mppvideodec ! waylandsink
```

4.2 Sink

指可以接受数据但不会发送数据的插件。

- filesink

  将收到的数据保存为文件，示例如下：

```
  gst-launch-1.0 filesrc location=/tmp/test ! filesink location=/tmp/test2
```

- fakesink

  将收到的数据全部丢弃，示例如下：

```
  gst-launch-1.0 filesrc location=/tmp/test ! fakesink
```

- xvimagesink

  视频Sink，接收视频并显示，使用X11接口实现，示例如下：

```
  gst-launch-1.0 videotestsrc ! xvimagesink
```

- kmssink

  视频Sink，接收视频并显示，使用kms接口实现，需要独占硬解图层，示例如下：

```
  gst-launch-1.0 videotestsrc ! kmssink
  # 常用命令
  connector-id        #指定屏幕
  plane-id            #指定硬件图层
  render-rectangle    #指定渲染范围
```

- waylandsink

  视频Sink，接收视频并显示，使用wayland接口实现，示例如下：

```
  gst-launch-1.0 videotestsrc ! waylandsink
```

- rkximagesink

视频Sink，接收视频并显示，使用drm接口实现零拷贝等功能，性能较好，但需要独占硬解图层。示例如下：

```
gst-launch-1.0 videotestsrc ! rkximagesink
```

- fpsdisplaysink

  视频Sink，接收视频并统计帧率，同时会将视频中转至下一级Sink显示，示例如下：

```
# 日志等级为TRACE(7)即可查看实时帧率，设置为DEBUG(5)则只显示最大/最小帧率
GST_DEBUG=fpsdisplaysink:7 gst-play-1.0 --flags=3 --videosink="fpsdisplaysink
video-sink=xvimagesink signal-fps-measurements=true text-overlay=false
sync=false"
```

5. Rockchip MPP插件

基于MPP的硬件编解码插件。基于Gstreamer原有GstVideoDecoder类和GstVideoEncoder类开发。源码地址`<SDK>/external/gstreamer-rockchip/gst/rockchipmpp`。

解码支持的格式有JPEG，MPEG，VP8，VP9，H264，H265[1]。

编码支持的格式有JPEG，H264，H265，VP8。

5.1 gstmppdec

源码地址为gstreamer-rockchip/gst/rockchipmpp/，包含插件mppvideodec，mppjpegdec，以下以mppvideodec为例进行说明。

```
gstreamer-rockchip/gst/rockchipmpp/
├── gstmppdec.c
├── gstmppdec.h
├── gstmppjpegdec.c
├── gstmppjpegdec.h
├── gstmppvideodec.c
├── gstmppvideodec.h
……
```

5.1.1 主要函数说明

gst_mpp_dec_start： 创建MPP实例，内存分配器等。

gst_mpp_dec_set_format： 对MPP实例进行初始化，设置编解码类型和格式，设置Fast Mode，Ignore Error等属性。

gst_mpp_dec_handle_frame： 通过get_mpp_packet获取mpp_packet，填充数据后通过send_mpp_packet发至MPP解码。

gst_mpp_dec_loop： 通过poll_mpp_frame获取解码帧，并推送至下一级插件。

gst_mpp_dec_rga_convert： 如在输出buffer前需要进行格式转换，旋转，缩放，裁剪等操作，则会通过RGA[2]完成，再推送至下一级插件。

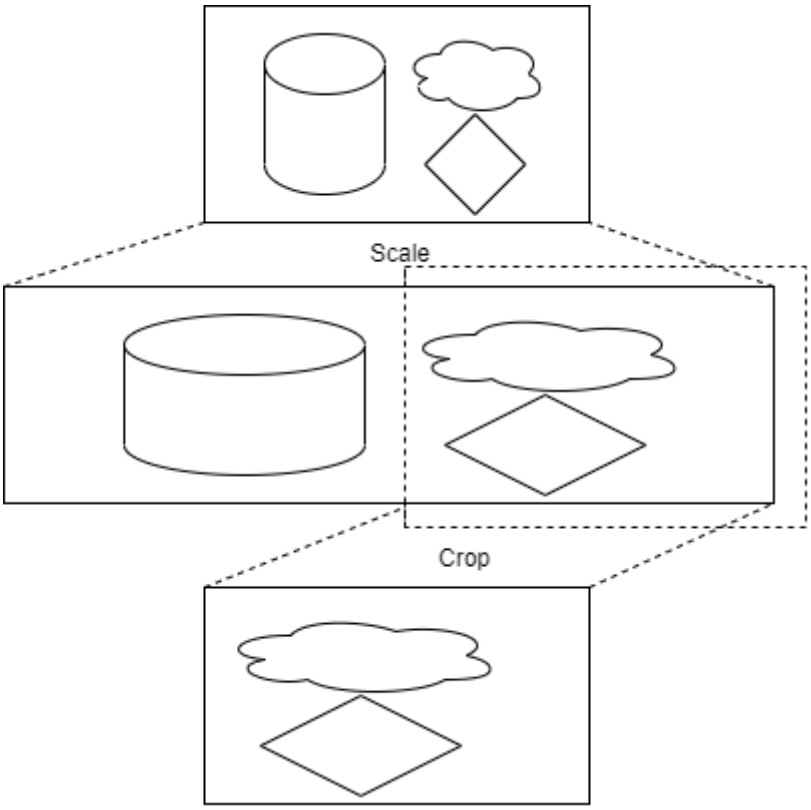
5.1.2 主要属性说明

rotation： 旋转角度，默认为0°，可选0°，90°，180°，270°。

width： 宽度，默认为0，不进行缩放。

height： 高度，默认为0，不进行缩放。

crop-rectangle： 裁剪，使用方式为<x, y, w, h>，即裁剪源<x, y>为起点，宽高为w * h的图像送至下级。需要注意的是，缩放的优先级比裁剪高，因此裁剪参数应以缩放后宽高为标准进行计算，如图所示为指定`crop-rectangle='<1920,0,1920,1080>' width=3840 height=1080`时的处理逻辑：



arm-afbc： AFBC压缩格式，默认不开启，部分平台如RK3399不支持。开启后可以降低DDR带宽占用，部分芯片解码效率会有明显提高。

format： 输出格式，默认为0 "auto"，不进行格式转换。

fast-mode： 开启MPP Fast Mode，如在RK3588平台上可以使部分解码流程并行，提升解码效率。默认开启。

ignore-error： 忽略MPP解码错误，强制输出解码帧。默认开启。

5.2 gstmppenc

源码地址为gstreamer-rockchip/gst/rockchipmpp/，包含插件mpph264enc，mppvp8enc，mppjpegenc等，以下以mpph264enc为例进行说明。

```
gstreamer-rockchip/gst/rockchipmpp/
├── gstmppenc.c
├── gstmppenc.h
├── gstmppjpegenc.c
├── gstmppjpegenc.h
├── gstmpph264enc.c
├── gstmpph264enc.h
……
```

5.2.1 主要函数说明

gst_mpp_enc_start： 创建和初始化MPP实例，设置实例类型和格式。

gst_mpp_enc_apply_properties： 设置编码参数，如gop，bps等。

gst_mpp_enc_handle_frame： 传入上一级插件的输出buffer，并存入编码器缓存中。

gst_mpp_rga_convert： 如需要对输入的buffer进行旋转，缩放等操作，则会先使用RGA[3]完成操作，再存入缓存。

gst_mpp_enc_loop： 按时间顺序取出编码器缓存内的buffer，使用encode_put_frame送至MPP编码，再使用encode_get_packet获取编码后码流，并送至下一级插件。

5.2.2 主要属性说明

width： 宽度，默认为0，不进行缩放。

height： 高度，默认为0，不进行缩放。

rc-mode： 码率控制模式，可选VBR，CBR和Fixed QP。

bps： 目标码率，在Fixed QP模式下忽略。

bps-max： 最高码率，在Fixed QP模式下忽略。

bps-min：最低码率，在Fixed QP模式下忽略。

gop： Group Of Picture，即两I帧的间隔。如0表示仅有一个I帧，其余为P帧，1表示全为I帧，2表示每两帧为I帧，即I P I P I P ...形式。默认为-1，按帧率设置，即每秒有一个I帧。

level： 表示 SPS 中的 level_idc 参数。

profile： 表示 SPS 中的 profile_idc 参数。

rotation： 旋转输入buffer，可选0°，90°，180°，270°。

6. 环境变量

常用环境变量均整理至/etc/profile.d/gst.sh，相关详细说明可以直接查看脚本内注释。

```
export GST_MPP_VIDEODEC_DEFAULT_ARM_AFBC=1：开启AFBC压缩格式，效果等同于设置mppvideodec arm-afbc=true，适用于gst-play-1.0等无法直接操作mppvideodec的情况。
export GST_MPP_VIDEODEC_DEFAULT_FORMAT=NV12：开启格式转换，mppvideodec始终输出NV12格式。
export GST_V4L2_PREFERRED_FOURCC=NV12:YU12:NV16:YUY2：设置v4l2输出格式。
export GST_VIDEO_CONVERT_PREFERRED_FORMAT=NV12:NV16:I420:YUY2：设置videoconvert输出格式。
export GST_VIDEO_CONVERT_USE_RGA=1：设置videoconvert使用RGA进行格式转换等操作。
export GST_VIDEO_FLIP_USE_RGA=1：设置videoflip使用RGA进行翻转等操作。
export GST_MPP_DEC_DEFAULT_IGNORE_ERROR=0：取消忽略MPP解码错误。
export GST_MPP_DEC_DEFAULT_FAST_MODE=0：关闭Fast Mode。
...
```

7. 命令示例

7.1 播放视频

```
gst-play-1.0 --flags=3 --videosink="fpsdisplaysink video-sink=xvimagesink signal-fps-measurements=true text-overlay=false sync=false" --audiosink="alsasink device=hw:0,0" test.mp4
```

7.2 多路视频播放

```
# 使用waylandsink的render-rectangle指定不同的渲染位置
gst-launch-1.0 filesrc location=/usr/local/test.mp4 ! parsebin ! mppvideodec ! waylandsink render-rectangle='<0,0,400,400>' &
gst-launch-1.0 filesrc location=/usr/local/test.mp4 ! parsebin ! mppvideodec ! waylandsink render-rectangle='<0,500,400,400>' &
gst-launch-1.0 filesrc location=/usr/local/test.mp4 ! parsebin ! mppvideodec ! waylandsink render-rectangle='<0,1000,400,400>' &
```

7.3 编码预览

使用tee插件，将摄像头采集的数据拷贝为两路，其中一路送至mpph264enc进行编码，而后送至filesink保存文件。另一路送至autovideosink显示。注意在tee插件后需要加上queue插件，会对数据进行缓存，防止出现卡死的情况。

```
gst-launch-1.0 v4l2src ! 'video/x-raw,format=NV12' ! tee name=tv ! queue ! mpph264enc ! 'video/x-h264' ! h264parse ! 'video/x-h264' ! filesink location=/data/out.h264 tv. ! queue ! autovideosink
```

7.4 拆分码流

部分插件如qtdemux，会出现多个Source Pad的情况，如音频流、视频流、字幕流等，则可以将该插件命名，并提取出需要的码流。如将qtdemux命名为qt，则qt.audio_0就是第一个音频流，qt.video_0就是第一个视频流，可提取后分别做处理。同样建议在分流后加上queue插件。不同插件码流命名方式不同，可以通过gst-inspect命令查看命名方式，或直接使用类似`qt. ! queue ! mppvidedec`的形式进行构建，gstreamer会与后级插件协商格式。

```
gst-launch-1.0 filesrc location=test.mp4 ! qtdemux name=qt qt.audio_0 ! queue !
filesink location=audio.bin qt.video_0 ! queue ! filesink location=video.bin
```

8. AFBC

AFBC全称ARM Frame Buffer Compression，是一种压缩格式，用于节省带宽。目前mppvideodec插件支持AFBC的编码格式有：H264，H265，VP9，支持的色彩格式有NV12，NV12 10bit，NV16。开启方法如下：

```
# 开启全局AFBC，适用于使用gst-play-1.0等无法直接操作mppvideodec的情况
export GST_MPP_VIDEODEC_DEFAULT_ARM_AFBC=1
# 单独开启AFBC
gst-launch-1.0 filesrc location=/test.mp4 ! parsebin ! mppvideodec arm-afbc=true
! waylandsink
```

waylandsink和xvimagesink支持AFBC格式合成，或使用kmssink/rkximagesink指定Cluster图层播放，该方式需要独占图层，如：

```
# GST_DEBUG=*mpp*:4开启mpp插件DEBUG开关，可以通过rkmpp打印的日志判断是否成功开启AFBC，如未
打印AFBC可能是未成功开启或格式不支持压缩
GST_DEBUG=*mpp*:4 gst-play-1.0 --flags=3 --videosink=waylandsink test.mp4
GST_DEBUG=*mpp*:4 gst-play-1.0 --flags=3 --videosink="kmssink plane-id=101"
...
0:00:00.256819945 29143   0x7f70008700 INFO                 mppdec
gstmppdec.c:465:gst_mpp_dec_apply_info_change:<mppvideodec0> applying NV12(AFBC)
1920x1080 (1920x1104)
...
```

8.1 AFBC dump解码数据

GStreamer想要查看硬件解码的数据是否正确，可以通过下面的方式dump解码数据（一般为NV12等格式图像）：

1. 打开MPP 日志功能

```
export mpp_debug=0x400
```

然后/data目录下就会有MPP自己dump的解码后数据。这是MPP自带的dump调试功能，开启AFBC时dump不支持；未开启AFBC时dump的数据一般为NV12格式，可以使用rawplayer（或其他裸视频播放器）查看。

2. 使用GStreamer的插件filesink dump 解码数据,这种方式无论开不开启AFBC都支持dump，使用方式如下：

```
gst-launch-1.0 uridecodebin uri=file://xxx ! filesink location=xxx.yuv
```

解码的AFBC数据就在xxx.yuv文件中，因为开启了AFBC还要将dump出的图像再讲过解压缩才能够使用rawplayer（或其他裸视频播放器）查看，解压缩命令（解压缩软件afbcDec要找相关负责人获取）：

```
./afbcDec filename w h format afbcmode
#eg: ./afbcDec 178_Surfa_id-26_1088x1824_z-0.bin 1088 1824 0 1
# 0=RGBA,1=NV12,2=RGB888, afbcmode 0=afbc, 1=afbc|YTR
```

afbcDec输出的图像格式为ARGB。

若想要查看视频的每一帧是否正确，还要将filesink dump的文件分帧：因为上面的解压缩软件只能转一帧数据，而dump出的视频所有帧都在同一个文件。分帧的示例如下：

```
# GST_DEBUG查看每帧大小
GST_DEBUG=filesink:6 gst-launch-1.0 uridecodebin uri=file://xxx ! filesink
location=xxx.yuv

0:00:01.224149631 14266   0x7f7c00ab00 DEBUG  filesink
gstfilesink.c:769:gst_file_sink_flush_buffer:<filesink0> Flushing out buffer of
size 1390080
# 使用split命令分帧
split -b 1390080 -a 5 -d  xxx.yuv dump_frame
```

9. 字幕

开启字幕会出现卡顿，通常字幕合成需要从视频中截取部分图像并转为RGB，再合成字幕后再转回源格式，才能进行送显，即解码的耗时还需考虑字幕合成的耗时，导致整体帧率下降。使用gst-play-1.0命令测试可以通过`--flags=3`关闭字幕。字幕需要自行使用QT等框架独立于视频层实现。

10. 图层指定

使用rkximagesink或kmssink时，需要独占一个硬件图层，并且插件会自动寻找图层播放，但自动寻找的图层可能无法满足需求，因此需要手动指定图层，方法如下：

```
gst-play-1.0 --flags=3 test.mp4 --videosink="kmssink plane-id=117"
```

其中117即目标图层的ID，可通过`/sys/kernel/debug/dri/0/state`节点确认，可以使用如下命令列出所有图层：

```
root@linaro-alip:/# cat /sys/kernel/debug/dri/0/state | grep "plane\["
plane[57]: Smart1-win0
plane[71]: Cluster1-win0
plane[87]: Smart0-win0
plane[101]: Cluster0-win0
plane[117]: Esmart1-win0
plane[131]: Esmart0-win0
# 也可以直接使用cat /sys/kernel/debug/dri/0/state列出完整信息
```

其中plane[xx]即为plane-id。通常不同图层支持的格式不同，如Cluster支持AFBC，但Esmart不支持AFBC，具体可查阅datasheet或TRM了解。若不存在该节点，则可通过modetest -p查看。

11. FAQ

1. 播放4K 30FPS不会卡顿，播放4K 60FPS出现卡顿

   由于系统负载、DDR带宽等问题，有可能导致无法达到4K 60FPS，可以尝试开启AFBC，参考AFBC章节。另外可以关闭字幕和sink的同步功能，如`gst-play-1.0 test.mp4 --flags=3 --videosink="waylandsink sync=false"`，在帧率无法达到60FPS时，开启sync会由于视频帧时间戳无法对齐时钟从而出现明显丢帧。

2. 播放某些片源比较卡顿，CPU占用率很高

   目前硬解支持H264，H265，VP8，VP9，MPEG。可以通过`echo 0x100 > /sys/module/rk_vcodec/parameters/mpp_dev_debug`开启DEBUG，看串口或dmesg有没有出现解码打印。如果没有可能是硬解不支持的格式。

3. 某些片源无法播放，LOG卡住未打印进度或进度始终为0

   可以尝试使用playbin3，如`gst-play-1.0 --flags=3 --use-playbin3 test.mp4`。

4. 开启AFBC后播放4K视频时出现闪烁

   首先确认开启性能模式，`echo performance | tee $(find /sys/ -name *governor)`。另外确认在纵向上是否有明显缩放，如使用竖屏播放横屏画面，在这种情况下AFBC性能没有非AFBC性能好。

5. 播放有画面但没有声音

   可以手动指定下audiosink，如`gst-play-1.0 --flags=3 test.mp4 --audiosink="alsasink device=hw:0,0"`。建议先使用aplay等基础测试工具测试可用再使用gstreamer测试。

6. 运行解压缩命令afbcDec时遇到缺少库libgraphic_lsf.so报错

   找相关负责人获取libgraphic_lsf.so，将缺少的libgraphic_lsf.so库拷贝到/usr/lib/目录即可。

1. 该处仅列出插件支持的格式，具体芯片是否支持请查询相关datasheet。 ↩
2. 目前部分平台如RK3588 RGA功能异常，不建议使用。 ↩
3. 目前部分平台如RK3588 RGA功能异常，不建议使用。 ↩