FFmpeg –
RTK Hardware Transcode Tutorial

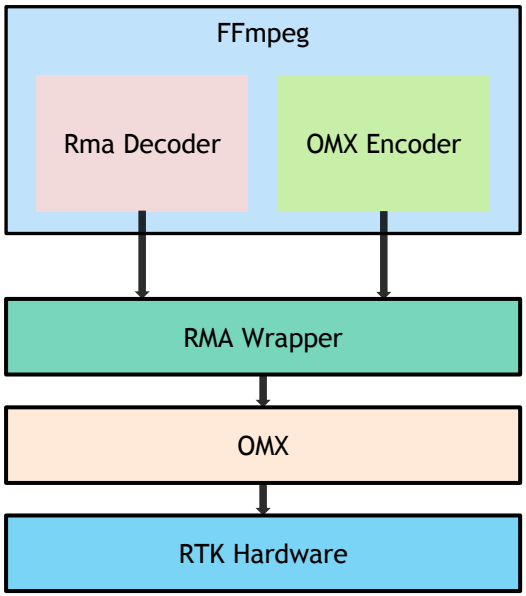
Driving IC Innovation

Agenda

- Introduction
- Build FFmpeg
- Transcoding command line
- Issue

Introduction

- Realtek uses RMA (RTK Media Accelerator) as wrapper layer between FFmpeg and Openmax Integration Layer
  - Decoder: libavcodec/rma_dec.c
  - Encoder: libavcodec/omx.c
  - Wrapper library: libRMA.so



Build FFmpeg

- Get FFmpeg 3.3.7 source code.
  - https://ffmpeg.org/releases/ffmpeg-3.3.7.tar.xz
- Patch RTK FFmpeg patch. Run
  - patch –p0 < 0001-FFmpeg3.3.7-RTK-HW-ACCEL.patch
- Copy RTK-NAS-Transcode-Libs to target. Run
  - cp -dR RTK-NAS-Transcode-Libs/usr /usr
- Configure FFmpeg. You can reference build.sh to enable RMA(RTK Media Accelerator). Run
  - ./configure --enable-omx --enable-rma --enable-decoder=h264_rma --enable-decoder=mpeg4_rma --enable-decoder=hevc_rma --enable-decoder=mpeg1_rma --enable-decoder=mpeg2_rma --enable-decoder=vp8_rma --enable-decoder=vp9_rma --enable-decoder=vc1_rma --enable-decoder=wmv3_rma --enable-decoder=mjpeg_rma --enable-decoder=h263_rma --enable-decoder=avs_rma --enable-encoder=h264_omx
- Build and install FFmpeg. Run
  - make ; make install

Transcoding Command line

- ffmpeg -dec_o_width 720 -dec_o_height 480 -i ./input.mp4 -c:a copy -c:v h264_omx -b:v 5500k -i_frame_interval 1 -f mpegts -copyts -start_at_zero ./output.ts

[Decoder]:
dec_o_width: scaling down width (optional, default: original width)
dec_o_height: scaling down height (optional, default: original height)
auto_resize: keeping the original width/height ratio (default: 0)
dec_o_fps: set the output fps to RMA decoder (default: 0)
turbo_mode: Speedup decode performance. Suggest enabling on 4k2k case (default: 0)
rtk_version: Show verion information about RTK patch and libs (default: 0)
search_I_frm: Start to decode from first I frame (default: 1)
search_I_err_tolerance: The percentage of error MBs that an I frame can display(only valid when search_I_frm is 1, default is 3)

[Encoder]:
i_frame_interval:H264 encode I-Frame Interval, specified in seconds (default: 0, the same with decoded frame interval)
rotation: H264 encode rotation, you can choose 0, 90, 180 or 270 (default: 0)
enc_select [1]: Select which encoder(0 or 1) that you want (default: 0)

[1] RTD1619 only

Scaling Down and Aspect Ratio

- RTK hardware transcode can support scaling down feature
  - dec_o_width/dec_o_height in **decoder** side
  - You can input a 4k resolution video but RTK HW can only support 1920x1080 output. So you need to set dec_o_width/dec_o_height to a value less than or equal to 1920x1080.
  - E.g. Transcoding a 4k input.mp4 to h264 1080p output.ts:

```
ffmpeg -dec_o_width 1920 -dec_o_height 1080 -i ./input.mp4 -c:a copy -c:v h264_omx ./output.ts
```

- RTK hardware transcode can keep the original width/height ratio
  - auto_resize in **decoder** side
  - E.g. Transcoding a 4k input.mp4 to h264 480p output.ts but keeping original aspect ratio:

```
ffmpeg -dec_o_width 852 -dec_o_height 480 -auto_resize 1 -i ./input.mp4 -c:a copy -c:v h264_omx ./output.ts
```

Output FPS

- RTK hardware transcode can set output fps to a small value
  - dec_o_fps in **decoder** side
  - E.g. Transcoding a input.mp4 at 30 fps to h264 1080p output.ts at 15 fps :

```
ffmpeg -dec_o_fps 15 -i ./input.mp4 -c:a copy -c:v h264_omx ./output.ts
```

I Frame

- Decode I frame only
  - turbo_mode in **decoder** side
  - It will drop all non-I frames. If GOP is very big, the transcode video will be non-smooth.
  - E.g. Decoding only I frames and encoding to output.ts:

```
ffmpeg -turbo_mode 1 -i ./input.mp4 -c:a copy -c:v h264_omx ./output.ts
```

- Find the first I frame and start to decode
  - search_I_frame in **decoder** side
  - It will drop all non-I frames until first I frame.
  - You can also set search_I_err_tolerance to set the percentage of error MBs that an I frame can decode.

```
ffmpeg -search_I_frame 1 -search_I_err_tolerance 5 -i ./input.mp4 -c:a copy -c:v
h264_omx ./output.ts
```

Encoder

- Output I frame interval
  - i_frame_interval in **encoder** side
  - The setting of GOP
  - E.g. Transcoding with output file that I frame appears once per second:

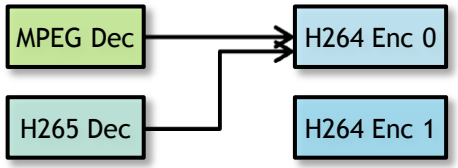
```
ffmpeg -i ./input.mp4 -c:a copy -c:v h264_omx -i_frame_interval 1 ./output.ts
```

- Output video rotation
  - rotation in **encoder** side
  - It will rotate the angle of encoded video.
  - E.g. Rotate the output video to 180°

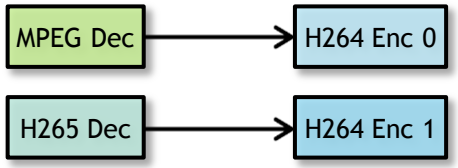
```
ffmpeg -i ./input.mp4 -c:a copy -c:v h264_omx -rotate 180 ./output.ts
```

Encoder Selection

- There are two video encode units in RTD1619. In order to get better performance, you can select different encoder when transcoding two files at the same time.
  - enc_select in **encoder** side (Default is 0)
    - 0 : Use encoder 0
    - 1 : Use encoder 1
  - E.g. Transcode MPEGII to H264 and H265 to H264 at the same time
    - Bad :



    - Good:




```
ffmpeg -i ./mpeg2.mpeg -c:a copy -c:v h264_omx -enc_select 0 ./output1.ts
ffmpeg -i ./h265.mp4 -c:a copy -c:v h264_omx -enc_select 1 ./output2.ts
```

Issue

- Conversion failed!
  - Error log: Too many packets buffered for output stream 0:1.
    - Add option: "-max_muxing_queue_size 1024“
    - Relative FFmpeg Discussion in https://trac.ffmpeg.org/ticket/6375