19.1 Potential Performance Improvement Using Graphics GPU

With certain newer, more powerful graphics boards and newer device drivers, there is the potential for enhanced “decode” and “encode” performance. *Decode* refers to loading and playing video in Cinelerra. The GPU, Graphics Processing Unit, on the graphics board is accessed via one of the following libraries: vdpau or vaapi. The hardware acceleration done by the graphics card increases performance by activating certain functions in connection with a few of the FFmpeg decoders. This use makes it possible for the graphics card to decode video, thus offloading the CPU. Decode operations are described here next. *Encode* refers to rendering video and is described at the end of this section under “Rendering with Hardware Acceleration / Encoding using VA-API and NVENC.”

VDPAU, Video Decode and Presentation API for Unix, is an open source library to offload portions of the video decoding process and video post-processing to the GPU of graphics boards, such as Nvidia. It may also apply to Nouveau and Amdgpu boards, but that has not been verified.

VA-API, Video Acceleration API, is an open source library which provides both hardware accelerated video encoding and decoding for use mostly with Intel graphics boards.

Currently only the most common codecs, such as MPEG-1, MPEG-2, MPEG-4, and H.264/MPEG-4, are accelerated/optimized by the graphics card to play these particular video formats efficiently. The other formats are not optimized so you will see no performance improvement since the CPU will handle them as before, just as if no hardware acceleration was activated. There are many different graphics cards and computer systems setup, so you will have to test which specific settings work best for you. So far this has been tested at least with Nvidia, Radeon, and Broadwell graphics boards on some AMD and Intel computers; depending on the graphics card, two to ten times higher processing speeds can be achieved. However, most graphic operations are single-threaded and may be slower as opposed to multiple CPUs, which frequently multi-thread many operations simultaneously,

<u>To use your graphics card GPU for decode with Cinelerra:</u>

1) Verify that you have installed libva-dev or libva on your operating system.
2) Verify that you also have libvdpau-dev or libvdpau installed.
3) Verify Settings->Preferences, Playback tab, Video Driver is set to X11 -- or X11-OpenGL if that produces better results for your configuration.
4) Before starting CinelerraGG, you can set an environment variable that can be easily reversed and then, to run from the Cinelerra installed directory, key in:

```
CIN_HW_DEV=vdpau ./cin        # for computers with Nvidia and some other graphics cards
CIN_HW_DEV=vaapi ./cin        # mostly for computers with Intel specific graphics hardware
```

If you find that the environment variable setting is advantageous for your CinGG usage and you want to always use it, you can add it to your $HOME directory .profile file which takes effect every time you log in. The line you would add would look something like this:

```
        export CIN_HW_DEV=vdpau
  Or    export CIN_HW_DEV=vaapi
```

It might be more difficult to analyze problems as a result of using the GPU because of the wide variation in hardware. When you do not set the CIN_HW_DEV environment variable, the code will work exactly as before since this feature is self-contained.

In addition to the environment variable, there is a Settings->Preferences, Performance tab, "Use HW device" flag with a pulldown to set up "none, vdpau, vaapi, or cuda". To ensure it takes effect, it is best to set it the way you want, quit out of Cinelerra and then restart. Its current purpose is for flexibility, but there is a possibility that it might eventually take the place of CIN_HW_DEV – both are not needed.

Precedence of the decode hardware acceleration settings are:

yourfile.opts is checked first so is of the highest priority; special .opts usage is described below
environment variable CIN_HW_DEV is checked next
preferences “Use HW device” settings is of the lowest priority

<u>How hardware acceleration for decode is handled in Cinelerra</u>
There are 4 phases during Cinelerra’s handling of hardware acceleration. These first 2 steps occur just *before* the first read.

1) Check to see if Hardware Acceleration is enabled, usually indicated by CIN_HW_DEV being set to vaapi or vdpau. If enabled, try to activate the decode device, and if that fails revert to software.

2) The next step is to send some data to decode to see if that works. If this does not work, you will see an error message of “HW device init failed, using SW decode”.

These next 2 steps occur *during* any read. Now there is no turning back to software so if the hardware gets an error, that video will not be handled correctly.

3) Read the media and send the raw stream data to the device for processing.

4) Read the device to receive the decoded data, and convert it to the session color model. If the GPU can not convert the data, you will see the error message of “Error retrieving data from GPU to CPU”.

Due to variations in user’s computer hardware configuration, it is often suggested that you refer to your startup window to check for error messages. Since your situation is unique, the error may not have been seen by anyone else and is probably unknown/undocumented.

<u>Possible improvements or differences you will see using hardware acceleration for decode in Cinelerra</u>
1) The Frames Per Second (FPS) in playback might be mostly at the maximum rate. You can check this in Settings→Preferences, Playback A, looking at “Framerate achieved”; the higher, the better.
2) Percent of the CPU used should be less, thus saving more CPU for other operations.
3) Some users get the the impression that playing seems smoother.
4) The CPU fan noise may go down as the CPU is used less.
5) The GPU graphics fan noise may go up as the GPU is used more.

Using the GPU is going to react differently depending on your hardware, software, and the number of files loaded. A good way to determine how well it is performing, is to watch the CPU load from another window running the command line routine “top”. Consider the following possibilities:

- If you only use smaller videos occasionally and then use other codecs than the ones mentioned previously as being the optimized set, it is usually a better idea to stay with the default settings without all the hardware tests.
- If you have 4 cores or less, but a really good "gaming card", using vaapi/vdpau could be a big help.
- If you load only a couple of files, the GPU vaapi/vdpau should be faster depending on your graphics board and how capable it is.
- If you load 10 camera Mixers of H.264 format, it always seems to work a lot better playing them.
- If you have an epyc AMD chip with 128 CPUs and load 50 files, without vaapi/vdpau may be better.

<u>Special .opts file usage for recalcitrant files</u>
The situation may arise where you have enabled hardware acceleration and after loading several files for a project, you find that a file had some kind of error resulting in a black video instead of an image or you see an error message pop up which states something like “Error retrieving data from GPU to CPU” or “err: Unknown error occurred”. Because the CIN_HW_DEV environment variable is either all or none, ordinarily in order to correct the non-working video you would have to turn off hardware acceleration for the entire project/session. However, there is a way to continue working on your project without having to reload all of your files. You still use the environment variable and it will be effective for all of the formats it is able to handle, but you make an exception for any of the files that erred out. To do this you simply create a file in the same directory with the same name as the erring file with the different extension of .opts. The contents of this .opts file would just be the one line of:

```
cin_hw_dev=none
```

Conversely, if you have a bunch of files in your project, like dnxhd format, that are not hardware accelerated, but you have an accompanying large file of type .mp4 for which you would like the hardware acceleration, you can leave the CIN_HW_DEV variable unset (that is, do not use it) and just create an .opts file containing the line:

```
cin_hw_dev=vdpau
```

For example your file, test.mp4, would have a side-kick called test.opts that will use the GPU for decoding/playing and the other files will just use the software. This is of some advantage because the ones that can not use the GPU if the environment variable is enabled, will not have to even check which saves a minuscule bit of time.

It is important to note that if using the .opts file to override the default ffmpeg/decode.opts file, you will most likely see more warnings (not really errors) in the Cinelerra startup window because the standard decode.opts file has "loglevel=fatal" whereas the default is "loglevel=error". To avoid seeing all of the extra warnings, you can simply add the line   loglevel=fatal   to your .opts file.

<u>How to verify that the hardware acceleration for decode is actually taking effect</u>
Probably the easiest way to tell if hardware acceleration is working, is just to look at the messages in the window from where you started Cin (not available if start using the application icon). For example load a png, dnxhd, or some other non-supported format file and you will see messages similar to those below. The line “HW device init failed, using SW decode” indicates that the vdpau/vaapi HW (hardware) decode is not available so will use SW (software) decode instead.

```
Failed to get HW surface format.
HW device init failed, using SW decode.
```

```
file:/tmp/media/aer_zypr.mp4
  err: Success

Decoder dnxhd does not support device type vdpau.
HW device init failed, using SW decode.
File:/tmp/media/test.gxf
  err: Success

HEVC with NVIDIA, VDPAU driver is buggy, skipping
```

If you would like to see more information on what is occurring, you can modify in the Cinelerra ffmpeg subdirectory, the file: decode.opts by temporarily changing the line from loglevel=fatal to loglevel=verbose and restarting Cinelerra. Then you will see messages in the startup window like:

```
[AVHWDeviceContext @ 0x7fc9540be940] Successfully created a VDPAU device
(NVIDIA VDPAU Driver Shared Library 390.116 Sun Jan 27 06:28:58 PST 2019) on X11 display :0
[h264 @ 0x7fc950159380] Reinit context to 1920x1088, pix_fmt: vdpau
[h264 @ 0x7fc92c3dd4c0] Reinit context to 1920x1088, pix_fmt: yuv420p
```

Again, to measure the performance run "top" from another window to check the CPU usage which will go lower as more work is offloaded to the GPU graphics card instead (it may go down by 2 to 10 times) or check the “Framerate achieved” while playing.

Some mixed preliminary results that have been reported are provided below.

Case #1:

X11 Video Driver set in Settings→Preferences, Playback A tab

```
CIN_HW_DEV=off ./cin      -> CPU 58%
CIN_HW_DEV=vdpau ./cin -> CPU 32%     # Note that in this case, using X11-OpenGL is best
CIN_HW_DEV=vaapi ./cin  -> CPU 82%
```

X11-OpenGL Video Driver

```
CIN_HW_DEV=off ./cin      -> CPU 48%
CIN_HW_DEV=vdpau ./cin -> CPU 12%     # Best shows the least amount of CPU usage
CIN_HW_DEV=vaapi ./cin  -> CPU 80%
```

Case #2:

X11 Video Driver set in Settings→Preferences, Playback A tab

```
CIN_HW_DEV=off ./cin      -> CPU 60%
CIN_HW_DEV=vdpau ./cin  -> CPU 11%     # Best shows the least amount of CPU usage
CIN_HW_DEV=vaapi ./cin   -> CPU 60%
```

X11-openGL Video Driver

```
CIN_HW_DEV=off ./cin       -> CPU 67%
CIN_HW_DEV=vdpau ./cin  -> CPU 60%    # Note that in this case, using X11 is better
CIN_HW_DEV=vaapi ./cin   -> CPU 67%
```

Older graphics cards or non-performing graphics cards will probably bring only a small amount of improvement or no speed advantage at all. You can check to see if vdpau is implemented for your specific Nvidia board at:

https://download.nvidia.com/XFree86/Linux-x86_64/304.137/README/supportedchips.html

And, you can see what your specific hardware and software might support by running either vainfo or vdpauinfo from the command line. Partial examples of each are shown below.

```
 # vainfo

vainfo: VA-API version: 1.4 (libva 2.4.0)
vainfo: Driver version: Intel i965 driver for Intel(R) Broadwell - 2.4.0.pre1 (2.3.0-11-g881e67a)
vainfo: Supported profile and entrypoints
      VAProfileMPEG2Simple
       ...
      VAProfileH264Main
      VAProfileH264High
      ...
      VAProfileH264MultiviewHigh
      VAProfileH264StereoHigh
      ...
      VAProfileVC1Simple
      ...
      VAProfileVP8Version0_3
```

```
# vdpauinfo

display: :0   screen: 0
API version: 1
Information string: G3DVL VDPAU Driver Shared Library version 1.0
...
Decoder capabilities:

name                    level macbs width height
----------------------------------------------------
MPEG1                      --- not supported ---
MPEG2_SIMPLE                3 65536  4096  4096
MPEG2_MAIN                  3 65536  4096  4096
H264_BASELINE              52 65536  4096  4096
H264_MAIN                  52 65536  4096  4096
H264_HIGH                  52 65536  4096  4096
VC1_SIMPLE                  1 65536  4096  4096
VC1_MAIN                    2 65536  4096  4096
VC1_ADVANCED                4   65536  4096  4096
```

One last item of note, *nvdec* is also enabled in the ffmpeg build, but at this time it is not known how this decode option on Nvidia graphics boards works or does not work in conjunction with vdpau.

Rendering with Hardware Acceleration / Encoding using VA-API and NVENC

Encoding using hardware acceleration of your graphics board GPU is included in CinelerraGG but it is of limited availability and works only with a specific set of hardware graphics boards, a certain level of graphics driver versions and only with certain ffmpeg formats. The encoding is done via vaapi (libva installed), which is known to work with Intel HD graphics boards and some others or via nvenc as developed by Nvidia for Nvidia graphics boards.

Broadcom, Intel HD, Radeon, and other graphics boards
To use hardware acceleration for rendering (that is, encoding) you do not have to set a preference or an environment variable, as was required for decoding. To use this feature you use an ffmpeg render options file which specifies a vaapi codec, such as h264_vaapi. You must include this line in that options file to trigger the hardware probe: cin_hw_dev=vaapi .

There are currently 4 options files available in the Render menu already set up for you that you see when you select the Video wrench and use the down arrow on the first line in the menu. These are:

```
h264_vaapi.mp4     # known to work on Intel computer with Intel Broadwell graphics driver
mpeg2_vaapi.mp4    # known to work on Intel computer with Intel Broadwell graphics driver
mjpeg_vaapi.mp4    # error message of “open failed with mjpeg_vaapi...” on above computer
hevc_vaapi.mp4     # error message of “open failed with hevc_vaapi...” on above computer
```

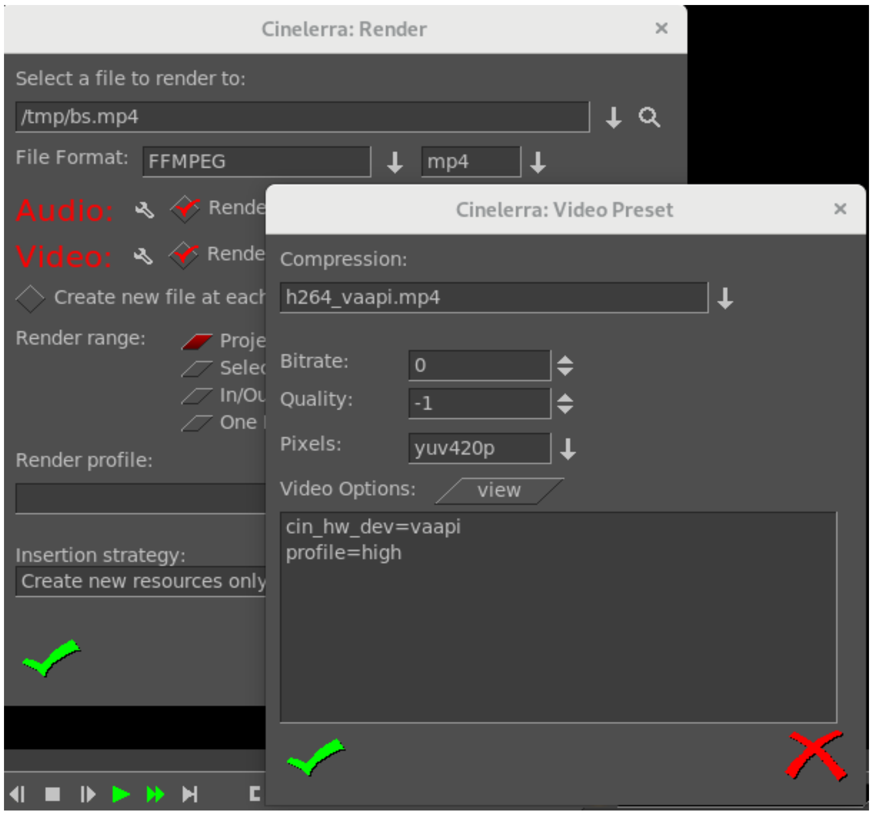
Other option files can be added as needed for your specific hardware if it is known to work for you, such as VP8 and VP9. An example of the included Cinelerra’s ffmpeg/video/h264_vaapi.mp4 file:

```
mp4 h264_vaapi
cin_hw_dev=vaapi
profile=high
```

According to an online wiki, hardware encoders usually create output of lower quality than some software encoders like x264, but are much faster and use less CPU. Keep this in mind as you might want to set a higher bitrate to get output of similar visual quality.

Results of a particular test case performed on a Intel, 4-core computer, with Broadwell Graphics using an mp4 input video/audio file with dimensions of 1440x1080 / 29.97fps is shown next (note, filename is tutorial.mp4). This may very well be a “best case” scenario! But clearly, at least on this computer with only 4 cores, the hardware acceleration seems to be quite advantageous. A comparison of the 2 output files using ydiff (as described in Appendix C in the manual) shows no obvious defects.

	CPU usage	Render Time	Output File Size	Options File
Without Hardware Acceleration	388%	100 secs.	36,862,542	h264.mp4
		versus		
With vaapi Hardware Acceleration	150%	19 secs.	74,522,736	h264_vaapi.mp4


Cinelerra: Render

Select a file to render to: /tmp/bs.mp4

File Format: FFMPEG mp4

Cinelerra: Video Preset

Compression: h264_vaapi.mp4

Bitrate: 0

Quality: -1

Pixels: yuv420p

Video Options:

```
cin_hw_dev=vaapi
profile=high
```

Insertion strategy: Create new resources only

Figure 1: Render menu setup to encode using GPU with vaapi

<u>Nvidia graphics boards</u>
To use hardware acceleration for rendering (that is, encoding) you do not have to set a preference or an environment variable, as was required for decoding. To use this feature you use an ffmpeg render options file which specifies the nvenc codec, either h264_nvenc.mp4 or nvenc.mp4. There are several requirements in order for this to work on your computer as listed here:

1) Nvidia graphics board at or above a certain hardware level. For h265, newer boards are required.
2) Software drivers for your graphics board must be installed on your computer.
3) The driver must support at least API version 9.0 – minimum required Nvidia driver for nvenc is 390.25 or newer. You will see error messages on the startup window if you are on lower versions.

If you try to render using h264/h265_nvenc.mp4 formats and do not have an Nvidia graphics card or this feature was not built in, you will see in the window from where you started Cinelerra, the error message:

```
Cannot load libcuda.so.1
```

A small test using 2 minutes from the 4k version of Big Buck Bunny shows using nvenc can be about 4 times faster. The test was done on a 4 core Intel laptop with an Nvidia 950M graphics board.

	CPU usage	Render Time	Output File Size	Options File
Without Hardware Acceleration	388%	20 mins 18 secs	156,517,069	h264.mp4

versus

With nvenc Hardware Acceleration	252%	5 mins 44 secs.	42,052,920	h264_nvenc.mp4

Of note in this test, 388% CPU usage with only 4 cores shows that there is probably slow down because there is no more CPU power available. Therefore, using the GPU hardware acceleration with nvenc provides a significant speed-up. Also, note the larger file size without making use of the GPU – this probably indicates that there is a big difference in bitrate or quality parameter settings used in the options file and this should be taken into consideration.

Important Tip:

There is one last potentially significant graphics speedup when using the X11-OpenGL driver for users with Nvidia graphics boards who are seeing frames/sec achieved lower than what the video format is set to. You may want to disable "sync to vblank" (an option for OpenGL) in NVIDIA X Server Settings for the proprietary drivers. This could increase your frames per second on playback.

And then there was Cuda

CUDA® is a parallel computing platform / programming model developed by Nvidia that provides big increases in computing performance through use of the GPU. It was first introduced in about 2006 for applications in computationally intense fields such as astronomy, biology, chemistry, and physics.

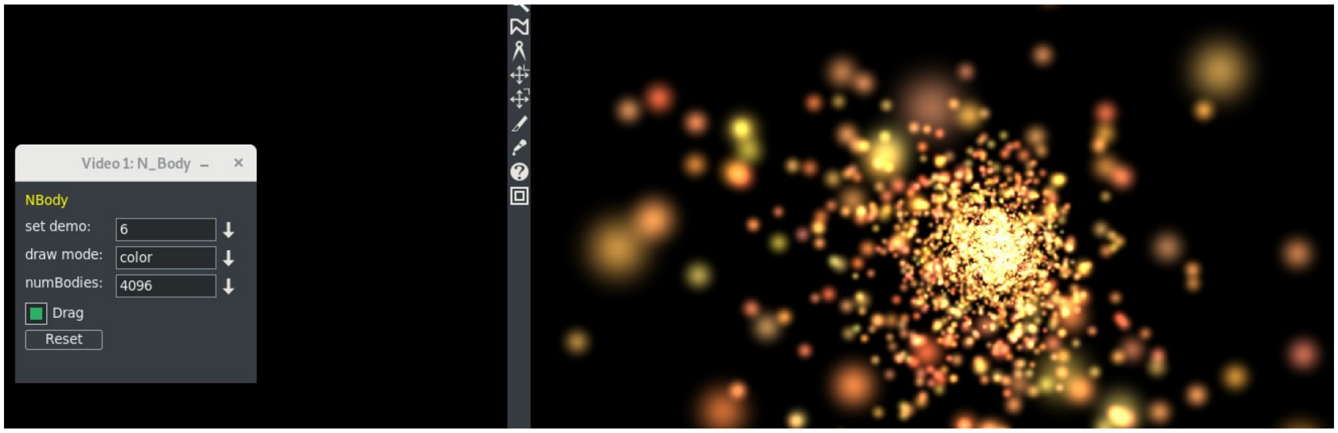
At the time this was written, the use of Cuda is not going to improve the playing and rendering of video in Cinelerra except in the case where you use a specific Cuda-enabled plugin that is computationally intense - sadly, most of what Cin does, Cuda will not help. Cuda is mostly a "block oriented algorithm" which works well for such things as "a flock of birds all flying next to each other".

The same as for vaapi and vdpau, you can enable Cuda in the Settings→Preferences, Performance tab, *Use HW Device* but it will not affect anything unless you have Cuda installed on your system and have built Cinelerra yourself with Cuda build enabled. To install it on your computer, you will need to do the following:

1) Make sure you have the Nvidia proprietary library drivers for your graphics board already installed.
2) Go to the Nvidia Cuda development website and choose one of the available operating system's such as Fedora, OpenSuse, CentOS, Ubuntu, … at https://developer.nvidia.com/ .
3) You will be installing repositories by package – this will be around 3 GB.
4) Also, install the Fusion repo, although it is unknown if necessary or not.

There is a very good set of directions on the website to just follow. Once you have installed the Cuda software on your computer, you must build Cinelerra yourself – the default build in the configure script for cuda is "auto". For Arch, be sure to first key in: setenv CUDA_PATH=/opt/cuda .

There are currently 2 available plugins for “show and tell” that take advantage of the hardware acceleration of Cuda – N_Body and Mandelbrot as shown next.


Video 1: N_Body

NBody

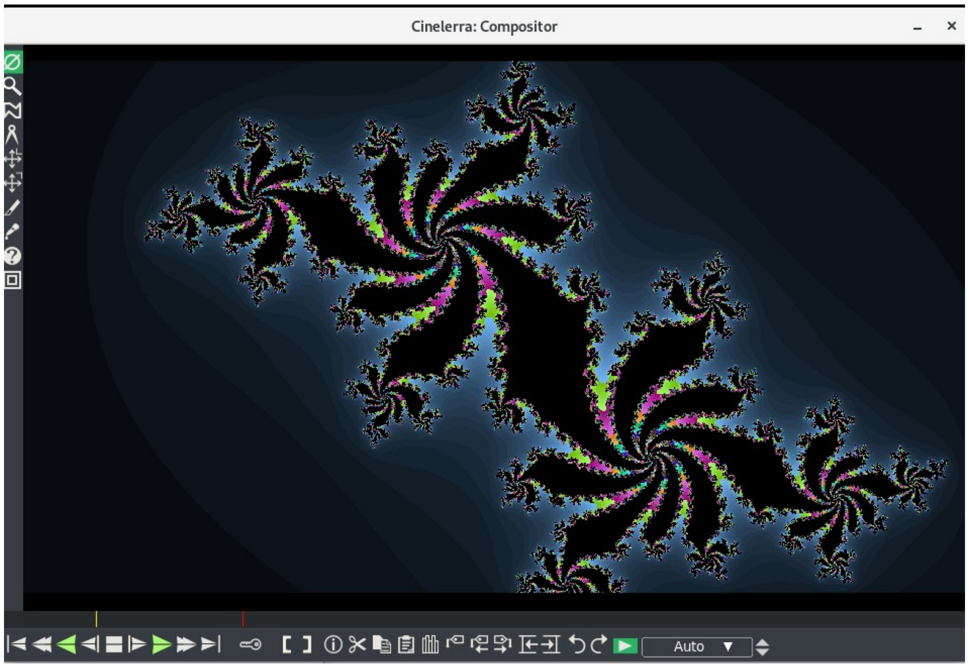
set demo: 6

draw mode: color

numBodies: 4096

🟩 Drag

Reset


Cinelerra: Compositor

An error you may see on your Cinelerra startup window when you have Cuda installed and try to run one of the 2 plugins is “cudaErrorInsufficientDriver”. This indicates CUDA 10 (the current version at the time of this writing) is not compatible with the driver version on your computer. You can either:

1) Upgrade the driver if your board supports newer nvidia builds.
2) Or downgrade the cuda development package to a version that works for your board.

One final note on Hardware Acceleration

In wrapping up this Hardware Acceleration section, you may want to refer to the following to determine the current supported formats:

https://wiki.archlinux.org/index.php/Hardware_video_acceleration .