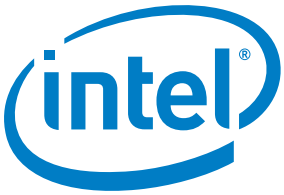


# Enabling the GuC/HuC Firmware for Linux* on New Intel GPU Platforms

## Advanced Media Feature Enabling Application Note

*February 2019*

Document Number: 609249-1.0

# Contents



## Figures



## Tables

# Revision History

| Date | Revision | Description |
|---|---|---|
| February 2019 | 1.0 | Initial release. |

§

# 1.0 Introduction

## 1.1 Terminology

### Table 1. Terminology

| Term | Description |
|---|---|
| MSDK | Intel® Media SDK |
| Gen9 | The 9th Generation GPU Architecture |
| EU | Execution Unit Compute/Render Engine in Intel GPU |

## 1.2 Background

The new generations of the Intel graphics hardware use firmware that have power, performance benefits, and functionalities, such as scheduling and media offloading. Some advanced GPU features (e.g., low power/EU- less than H.264 encoding in Gen9 and higher GPU platforms) cannot be achieved if the GuC and HuC lack support.

It is important for users to understand how to enable and check the firmware status before using it. The 9th platform of the Intel® Core™ i5-6600K Processor will be used in this paper to enable the GuC HuC on an Ubuntu* 16.04 system.



# 2.0 GuC/HuC Status Check and Enablement

## 2.1 Preliminaries

### 2.1.1 What is GuC/HuC

GuC is designed to perform graphics workload scheduling on the various graphics parallel engines. In this scheduling model, the host software submits work through one of the 256 graphics doorbells. This invokes the scheduling operation on the appropriate graphics engine. Scheduling operations include determining which workload to run next, submitting a workload to a command streamer, pre-empting existing workloads running on an engine, monitoring progress, and notifying the host software when work is done.

HuC is designed to offload some of the media functions from the CPU to GPU. These include bitrate control and header parsing. For example, in the case of bitrate control, the driver invokes the HuC in the beginning of each frame encoding pass. The encode bitrate is adjusted by the calculation from HuC. Both the HuC hardware and the encode hardcode reside in the GPU. Using the HuC will save unnecessary CPU-GPU synchronization.

### 2.1.2 Where to Download

The firmware for the Intel Linux* Graphics is available on www.git.kernel.org. It is sorted by the three letter product code of the processor (e.g., for the Kaby Lake GuC, it may be kbl_guc_ver9_14.bin). The i915 firmware download site for the Linux* Graphics can be found on:

https://git.kernel.org/pub/scm/linux/kernel/git/firmware/linux-firmware.git/tree/i915

## 2.2 Check the GuC/HuC Load Status

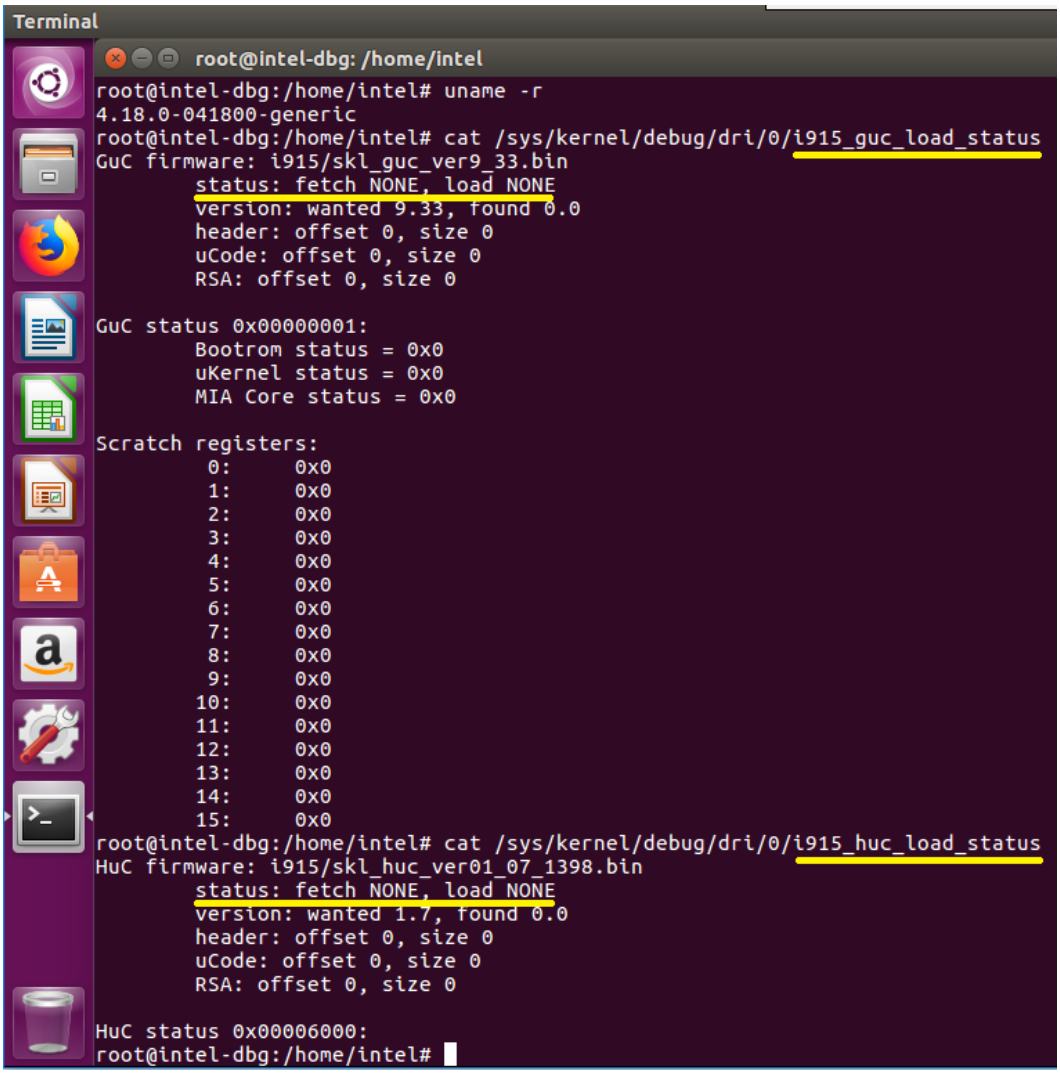
Run these commands to check the load status of the GuC/HuC firmware:

GuC: sudo cat /sys/kernel/debug/dri/0/i915_guc_load_status

HuC: sudo cat /sys/kernel/debug/dri/0/i915_huc_load_status

As shown in Figure 1, both the GuC and HuC are not loaded in this system (with 4.18 kernel).

**Figure 1. GuC and HuC Status**

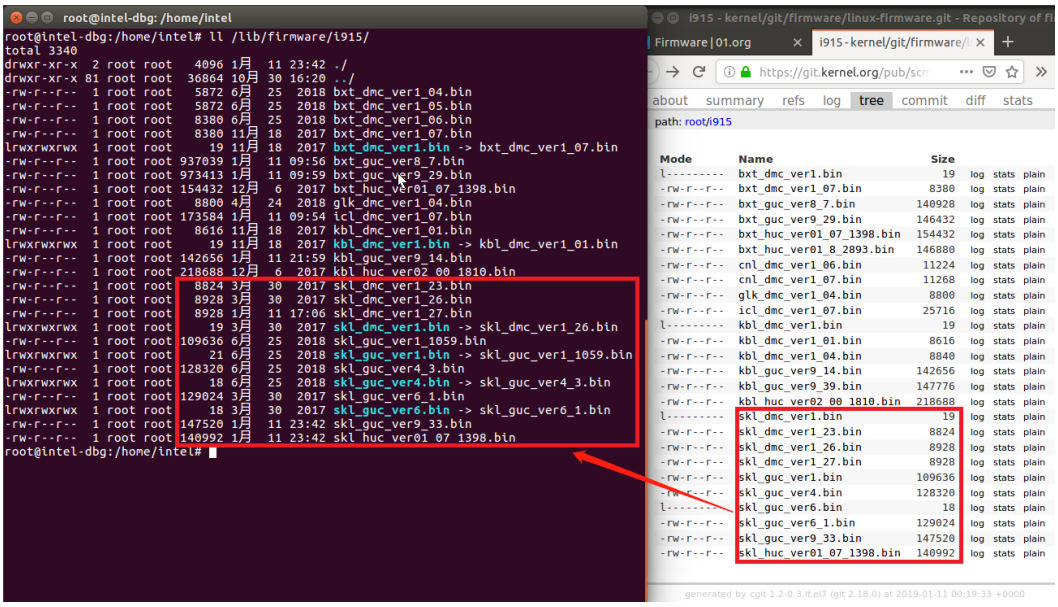



## 2.3 How to Enable the GuC/HuC

### 2.3.1 Download Platform Firmware Files from www.git.kernel.org

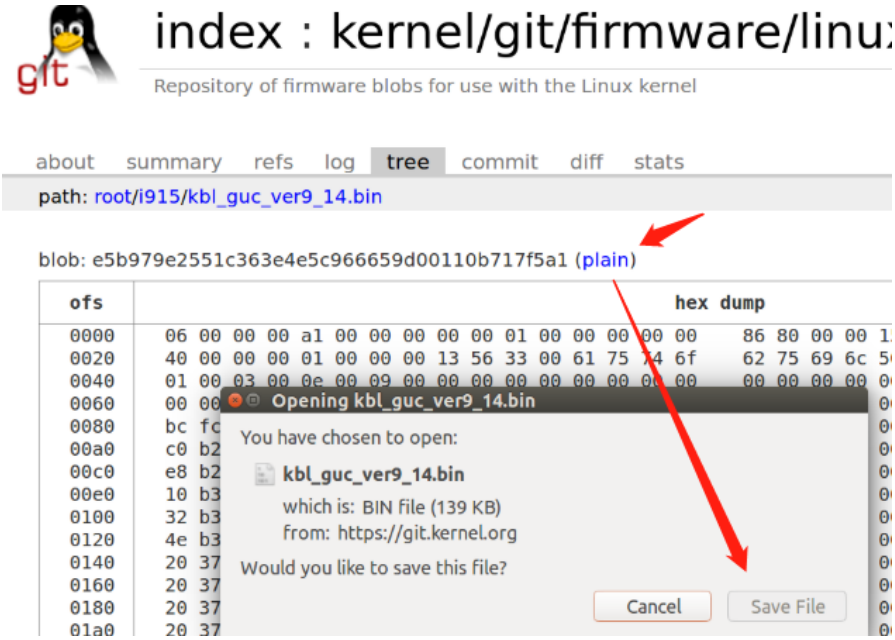
The GPU firmware files should be stored in /lib/firmware/i915. Ensure that all platform-related files have been downloaded and placed in the directory shown in Figure 2.

**Figure 2. Local Firmware Path and Download Site**




***Note:*** Take note of the binary file size that was downloaded; Certain methods (e.g., using Wget*) may obtain a smaller file size that may cause an unexpected failure. Intel recommends using the “save file” option by opening the file link and clicking “plain.”

**Figure 3. Download the Firmware File by Clicking “plain”**



***Note:*** There is a possibility that the current Linux* distribution is packaged with the firmware. However, this depends on whether a user has installed a newer graphics driver or has updated to a new Linux* kernel. Visit www.git.kernel.org to get the complete list.

**Table 2. Platforms**

| Prefix Name | Referred Platforms |
|---|---|
| BXT | Apollo Lake (a.k.a Broxton - previous code name for Apollo Lake, BXT was canceled in 2016.) |
| GLK | Gemini Lake |
| SKL | Sky Lake - 6th Platform of the Intel® Core™ Processor |
| KBL | Kaby Lake - 7th Platform of the Intel® Core™ Processor<br>Coffee Lake - 8th Platform of the Intel® Core™ Processor |

### 2.3.2 Enable GuC/HuC Firmware Loading

Currently, the Guc/HuC is not enabled by default (as of 4.16). Users should add specific kernel parameters to enable it during the system boot. Note that different Linux* kernels have different parameters. Table 3 shows the related kernel parameters.

**Table 3. Kernel Parameters**

| Kernel Version | Parameter |
|---|---|
| 4.15 | i915.enable_guc_loading=1 i915.enable_guc_submission=1 |
| 4.18, 4.19 | I915.enable_guc=2 |

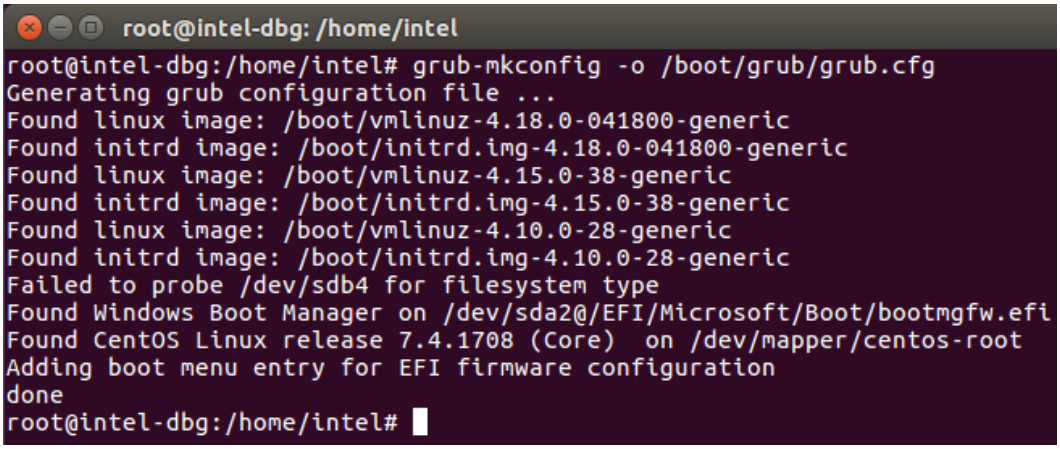
- Edit "/etc/default/grub" and extend GRUB_CMDLINE_LINUX_DEFAULT with the corresponding parameter. The result is shown in Figure 4.

**Figure 4. Modify the Kernel Parameters**

```
root@intel-dbg: /home/intel

root@intel-dbg:/home/intel# cat /etc/default/grub
# If you change this file, run 'update-grub' afterwards to update
# /boot/grub/grub.cfg.
# For full documentation of the options in this file, see:
#   info -f grub -n 'Simple configuration'

GRUB_DEFAULT=0
#GRUB_HIDDEN_TIMEOUT=0
GRUB_HIDDEN_TIMEOUT_QUIET=true
GRUB_TIMEOUT=10
GRUB_DISTRIBUTOR=`lsb_release -i -s 2> /dev/null || echo Debian`
GRUB_CMDLINE_LINUX_DEFAULT="quiet splash i915.enable_guc=2"

# Uncomment to enable BadRAM filtering, modify to suit your needs
# This works with Linux (no patch required) and with any kernel that obtains
# the memory map information from GRUB (GNU Mach, kernel of FreeBSD ...)
#GRUB_BADRAM="0x01234567,0xfefefefe,0x89abcdef,0xefefefef"
```

- Regenerate the grub configuration file by running “grub-mkconfig -o /boot/grub/grub.cfg.”

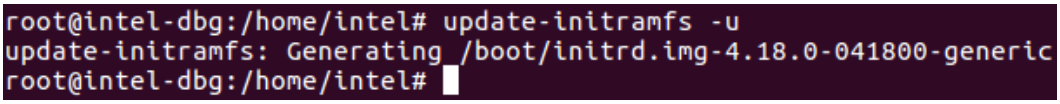
**Figure 5. Generate a New Grub Configuration File**




- Update “initramfs” to ensure that the kernel parameters are fully updated during the boot stage.

```
sudo update-initramfs –u
```

**Figure 6. Update “initramfs”**



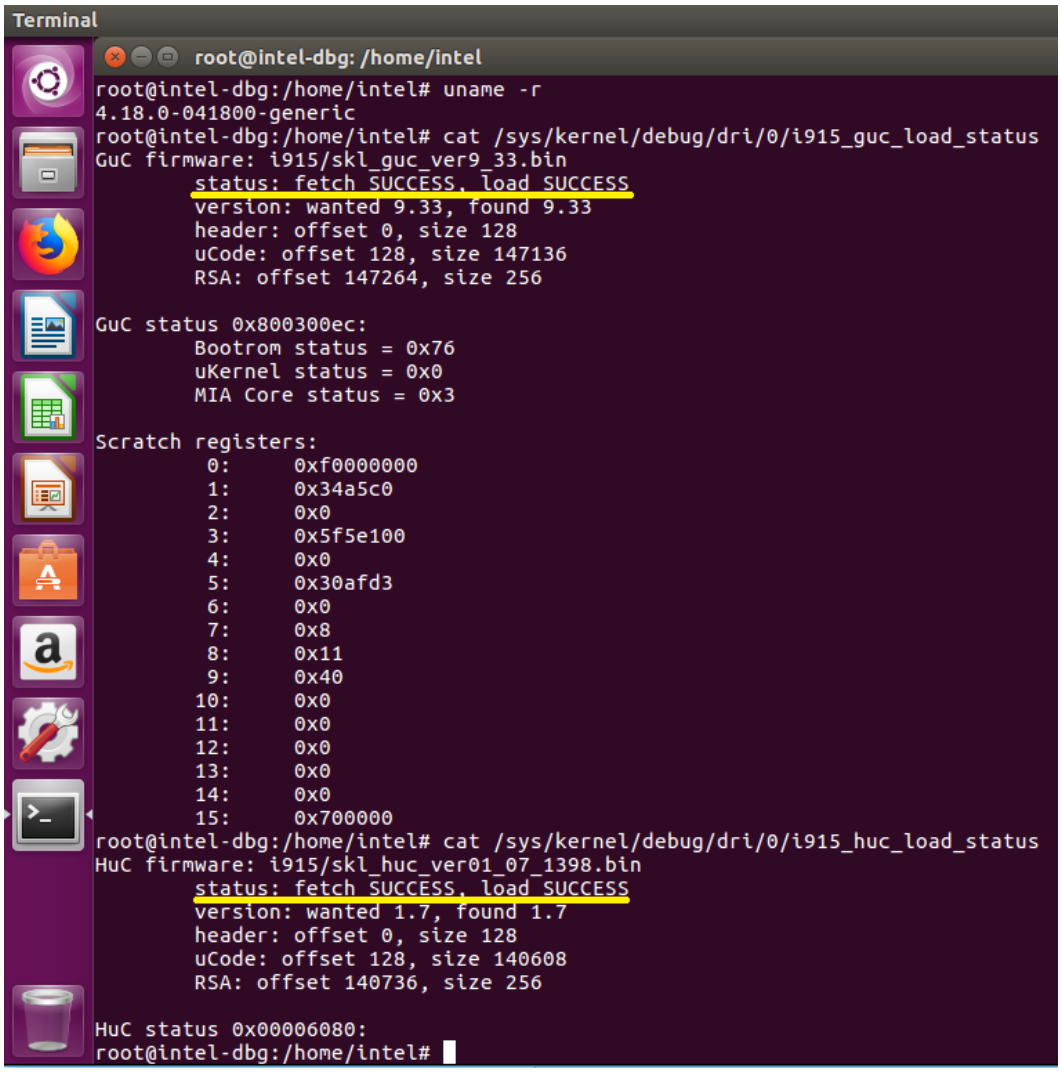

- After rebooting, the GuC/HuC should be loaded successfully.

# 3.0 Final Results

As shown in Figure 7, the system status is displayed when the GuC/HuC is correctly loaded. The “found” version and “wanted” version will correspond similarly.

***Note:*** A different Linux* kernel may display a different version number.

**Figure 7. GuC/HuC Successful Loading Status**



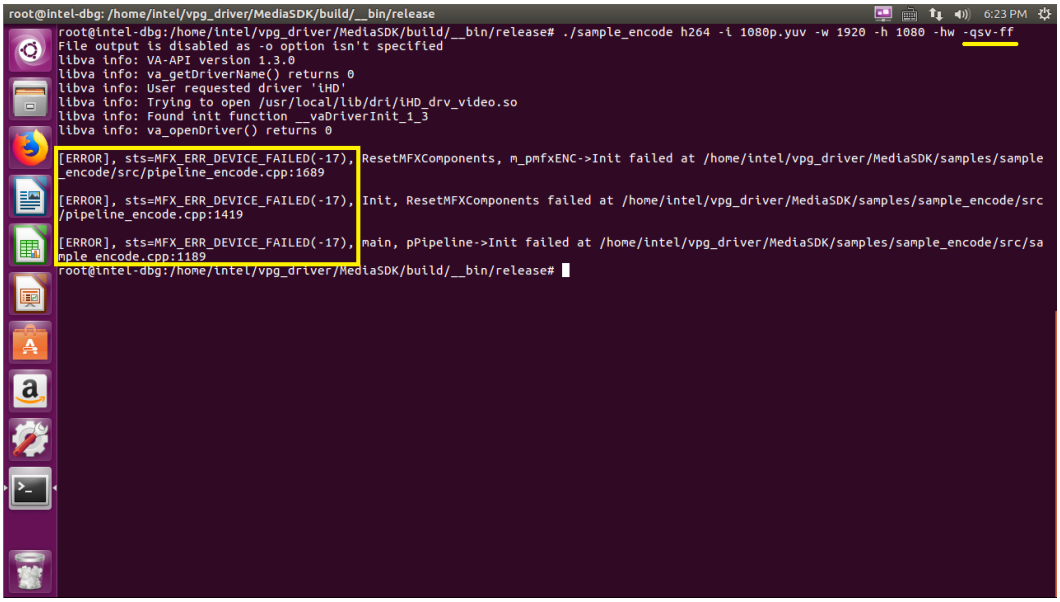
# *Appendix – Enable Low Power (EU-less) Encoding on Gen9+ Intel HD Graphics*

Intel has introduced a brand new fix-function IP (a.k.a VDENC) in Generation 9 GPU architecture. This new IP has the potential to realize low power H.264 video encoding without involving the Execution Unit (EU - the most important GPU engine for computing and rendering).

The Intel® Media SDK contains samples (sample_encode and sample_multi_transcode) to demonstrate how users can enable low power encoding from the application level by using the "-qsv-ff" parameter. However, this function strongly depends on the GuC/HuC enablement. The different possible scenarios for this feature are shown in Figure 8, Figure 9, and Figure 10.

## A.1 Low-Power Encoding Will Receive "DEVICE_FAILED" if There is No GuC/HuC Loaded

**Figure 8. Failure Caused by Lack of GuC/HuC**



## A.2 Normal Encoding vs. Low-Power Encoding (GuC/HuC Enabled) Comparison

The H264 low-power encoding can be implemented after enabling the GuC/HuC. The differences in using this feature are shown in Figure 9 and Figure 10. The EU (shown as "render busy" engine) usage is dropped from 65+% to 0. Thus, the computer program source and silicon power are now saved.

**Figure 9. GPU Usage of Normal AVC Encoding**

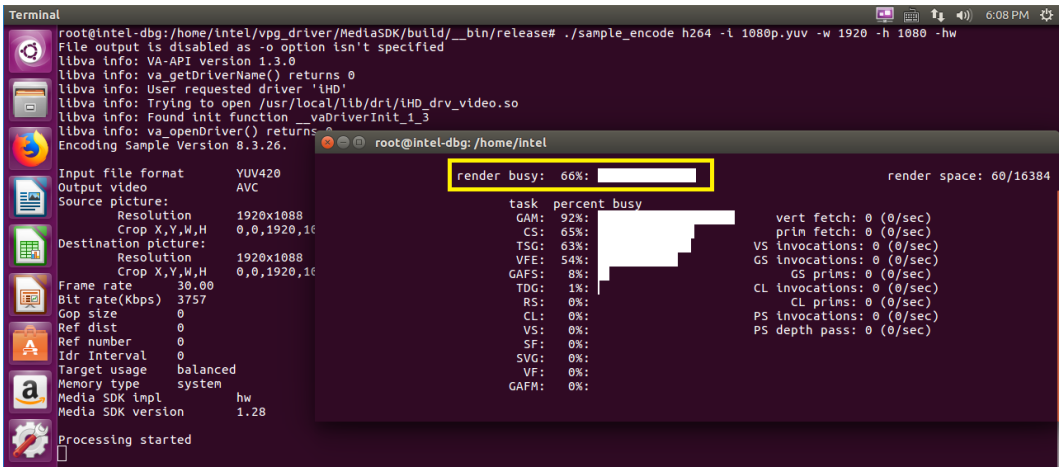
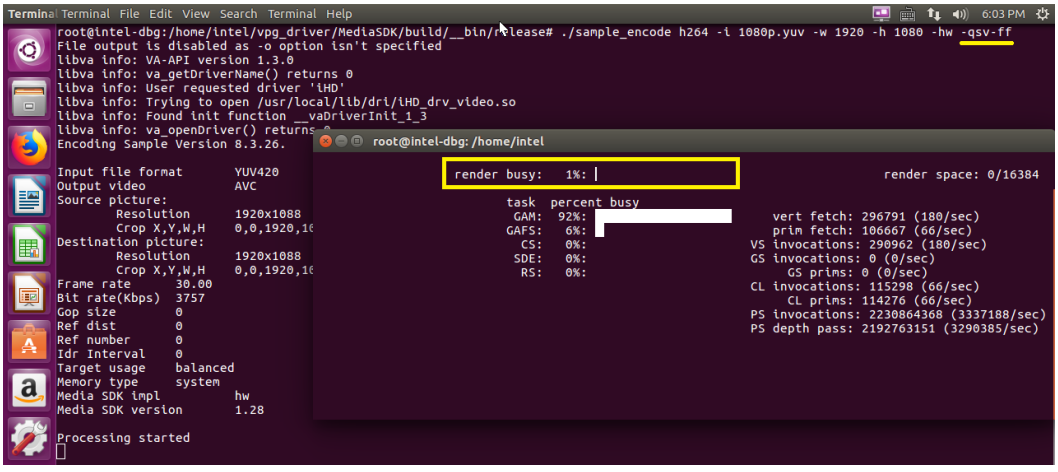



**Figure 10. GPU Usage of Low Power AVC Encoding**




§