

H3C XG310 GPU Module

User Guide

Copyright © 2021, New H3C Technologies Co., Ltd. and its licensors. All rights reserved.

No part of this manual may be reproduced or transmitted in any form or by any means without prior written consent of New H3C Technologies Co., Ltd.

Except for the trademarks of New H3C Technologies Co., Ltd., any trademarks that may be mentioned in this document are the property of their respective owners.

The information in this document is subject to change without notice. All contents in this document, including statements, information, and recommendations, are believed to be accurate, but they are presented without warranty of any kind, express or implied. H3C shall not be liable for technical or editorial errors or omissions contained herein.

1 About the GPU module

Overview

ⓘ **IMPORTANT:**

The model name of a hardware option in this document might differ slightly from its model name label. A model name label might add a prefix or suffix to the hardware-coded model name for purposes such as identifying the matching server brand or applicable region. For example, the GPU-XG310-32GB-FHFL GPU module model represents GPU module labels including UN-GPU-XG310-32GB-FHFL and UN-GPU-XG310-32GB-FHFL-F, which have different prefixes and suffixes.

H3C GPU-XG310-32GB-FHFL GPU module (hereinafter referred to as GPU module) uses Intel SG1 GPUs on the Gen12 graphics architecture, delivering increased performance and high energy efficiency. The XG310 has four independent GPUs, allowing the server to call different number of GPUs as needed for image processing and data operations. Each GPU supports a maximum of 96 execution units (EUs). The XG310 is applicable to cloud gaming, virtual desktop, image and video processing, and other scenarios.

In this document, "GPU module" refers to the GPU-XG310-32GB-FHFL module and "GPU" refers to a GPU chip on the GPU-XG310-32GB-FHFL module.

Figure1-1 GPU module



Connectors

Figure1-2 Connectors



(1) Power connector

(2) PCIe3.0 x16 connector

Specifications

Table1-1 Hardware specifications

Item	Specifications
GPU package type	FCBGA
Number of GPUs	4
EUs	96 EUs per GPU
Video memory type	LPDDR4x
Video memory size	<ul style="list-style-type: none"> GPU: 8 GB GPU module: 32 GB
GPU video memory bitwidth	128 bits
GPU video memory bandwidth	68.25 GB/s
Video memory operating rate	4267 MT/s
GPU module operating frequency	<ul style="list-style-type: none"> Boost: 1.1 GHz Base: 0.9 GHz
Power consumption	<ul style="list-style-type: none"> GPU: 23 W GPU module: 150 W
Bus type	PCIe3.0 x16

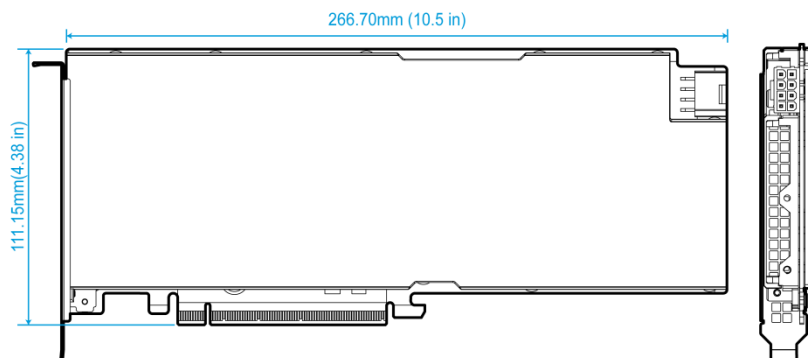
Item	Specifications
Power connector type	PCIe 8-pin
Thermal solution	Passive
Maximum operating temperature	95°C (203°F) The GPU module operates at reduced frequency if the temperature exceeds 95°C (203°F).
Forced power-off temperature	130°C (266°F)

Table1-2 Software specifications

Item	Specifications
Fixed function decode	Supported (AVC/HEVC/VP9/MPEG-2/JPEG)
Fixed function encode	Supported (AVC/HEVC/VP9/JPEG)
Programmable encode	Supported (AVC/HEVC/MPEG-2)
AV1 decode	Supported
GFX/APIs	Supported (OpenGL/OpenGL ES)
Virtualization mode	Only pass-through mode supported
ECC	Not supported
Compatible operating system	<ul style="list-style-type: none"> Linux: CentOS and Debian. VMware: Not supported. Windows: Not supported.
Hypervisor	KVM
Host OS	CentOS
Guest OS	CentOS

Dimensions

Figure1-3 Dimensions



Environment requirements

Table1-3 Environment requirements

Item	requirements
Temperature	<ul style="list-style-type: none"> Operating temperature: 0°C to 55°C (32°F to 131°F) Storage temperature: -40°C to +70°C (-40°F to +158°F)
Humidity	<ul style="list-style-type: none"> Operating humidity: 8%RH to 90%RH, noncondensing Storage humidity: 5%RH to 95%RH, noncondensing

Cooling requirements

Figure1-4 Airflow through the GPU module



Table1-4 Required air volumes for different inlet temperatures

GPU module inlet temperature	Required air volume
55°C (131°F)	12 CFM
50°C (122°F)	10 CFM
45°C (113°F)	8 CFM
40°C (104°F)	6 CFM

2 Compatibility

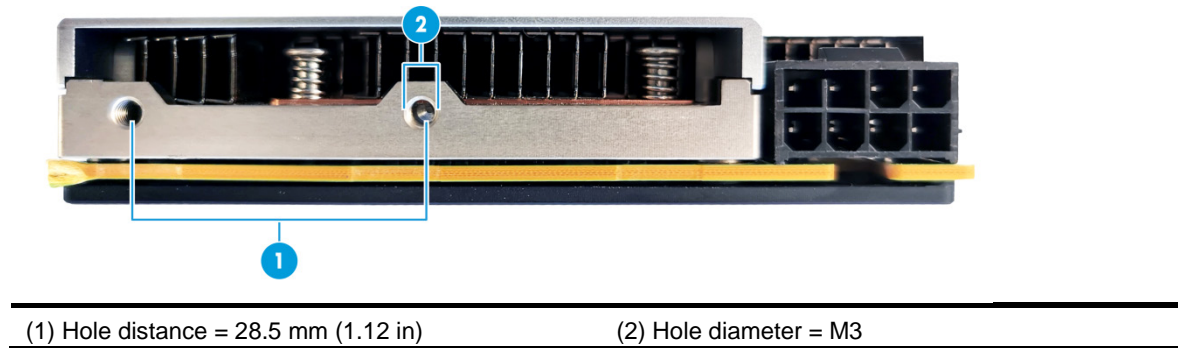
The XG310 GPU module is developed based on the Intel SG1 chip and can only be used on servers that use Intel CPUs.

3 Installing a GPU module

Attaching a support bracket to the GPU module

To install the GPU module securely on a server, attach a support bracket to the GPU module as required. See [Figure3-1](#) for the support bracket securing hole specifications.

Figure3-1 Support bracket securing hole specifications



Installing the GPU module on a server

For information about installing the GPU module on a server, see the user guide for the server.

4 Installing drivers and the firmware

Downloading the firmware and drivers

Access https://www.h3c.com/en/Support/Resource_Center/Software_Download/Servers/ to obtain the firmware and drivers for the GPU module and the related installation guides.

Installing drivers

A GPU module can be identified and used only after drivers are installed correctly on it. The following procedure installs drivers for the GPU module on a CentOS 7.4 operating system.

Installing drivers

1. Copy the downloaded driver program to the server and decompress it.
2. Install the kernel and drivers:
 - a. Execute the `./install-sg1.sh` command to start the driver installation.

Figure4-1 Starting the driver installation

```
[root@localhost SG1_20044_c7.4_k4.14.105]# ./install-sg1.sh
warning: /home/SG1_20044_c7.4_k4.14.105/mini_env_depend_packages/xkeyboard-config-1.3.0-1.el7.x86_64.rpm:
3 RSA/SHA256 Signature, key ID f4a80eb5: NOKEY
Preparing... ##### [100%]
```

- b. Enter `y` to install the kernel.

Figure4-2 Installing the kernel

```
Do you want to update kernel 4.14.105? 'y/n' default is y:y
install kernel 4.14.105
Preparing... ##### [100%]
Updating / installing...
 1:kernel-4.14.105-1 ##### [100%]
Preparing... ##### [100%]
Updating / installing...
 1:kernel-headers-4.14.105-1 ##### [100%]
Preparing... ##### [100%]
Updating / installing...
 1:kernel-devel-4.14.105-1 ##### [100%]
install GPU driver for kernel 4.14.105
Preparing... ##### [100%]
Updating / installing...
 1:kmod-ukmd-4.14.105-20044.el7.centos ##### [100%]
Install working. This may take some time ...
install mss
```

- c. Enter `y` to install the media driver.

Figure4-3 Installing the media driver

```
Do you want to install Media? 'y/n' default is n:y
install mss
MediaServerStudioEssentialsSG12020R2_20044/
MediaServerStudioEssentialsSG12020R2_20044/intel-linux-media-sg1-20044-c7.4.tar.gz
/home/SG1_20044_c7.4_k4.14.105/MediaServerStudioEssentialsSG12020R2_20044 /home/SG1_20044_c7.4_k4.14.105
intel-linux-media-sg1-20044/
intel-linux-media-sg1-20044/opt/
intel-linux-media-sg1-20044/opt/intel/
intel-linux-media-sg1-20044/opt/intel/common/
intel-linux-media-sg1-20044/opt/intel/common/mdf/
intel-linux-media-sg1-20044/opt/intel/common/mdf/lib64/
```

- d. Enter `y` to install the Mesa driver.

Figure4-4 Installing the Mesa driver

```
Do you want to install Mesa? 'y/n' default is n:y
install mesa
mesa-c7.4/
mesa-c7.4/llvm-8.0.1-1.x86_64.rpm
mesa-c7.4/mesa-20.3.0-20044.c7.4.x86_64.rpm
mesa-c7.4/install_mesa.sh
/home/SG1_20044_c7.4_k4.14.105/mesa-c7.4 /home/SG1_20044_c7.4_k4.14.105
Preparing... ##### [100%]
Updating / installing...
```

3. Modify the system configuration file.
 - a. Modify the GRUB setting. As shown in [Figure4-5](#), open the `/boot/efi/EFI/centos/grubenv` file and replace the `saved_entry` line with `saved_entry=CentOS Linux (4.14.105) 7 (Core)`.

Figure4-5 Modifying the GRUB setting

```
# GRUB Environment Block
saved_entry=CentOS Linux (4.14.105) 7 (Core)
```

- b. Add the kernel option. Open the `/boot/efi/EFI/centos/grub.cfg` file, add space and `modprobe.blacklist=ast` to the end of `en_US.UTF-8`, as shown in [Figure4-6](#).

Figure4-6 Adding the kernel option

```
linuxefi /vmlinuz-4.14.105 root=/dev/mapper/centos-root ro crashkernel=auto rd.lvm.lv=centos/root rd.lvm.lv=centos/swap rhgb quiet LANG=en_US.UTF-8 modprobe.blacklist=ast
```

4. Execute the `systemctl disable gdm` command to disable the GDM service.
5. Execute the `reboot` command to restart the server.
6. Configure the following settings on the server:

```
export
LD_LIBRARY_PATH=/usr/local/lib:/usr/local/lib64:/usr/local/lib64/dri:/usr/l
ib64:/usr/lib64/dri:${LD_LIBRARY_PATH}
export DISPLAY=:0.0
export MESA_LOADER_DRIVER_OVERRIDE=iris
export LD_LIBRARY_PATH=./opt/intel/mediasdk/lib64:${LD_LIBRARY_PATH}
export LIBVA_DRIVER_NAME=iHD
export LIBVA_DRIVERS_PATH=/opt/intel/mediasdk/lib64
```
7. Execute the `/usr/bin/X -sharevts &` command to connect the GPUs to the X service.

Verifying the installation

1. In the root directory of the driver, execute the `./gpu_sanity_test_sh` command to run a sanity test on the four GPUs.
As shown in [Figure4-7](#), all GPUs pass the sanity test.

Figure4-7 Sanity test on GPUs

```
[root@localhost SG1_20044_c7.4_k4.14.105]# ./gpu_sanity_test_sh
.....Sanity test begin.....
Intel GPU card 0:
PCI:0000:1b:00.0, ID:0x4907
card:/dev/dri/card0 render:/dev/dri/renderD128
transcode sanity PASS

Intel GPU card 1:
PCI:0000:20:00.0, ID:0x4907
card:/dev/dri/card1 render:/dev/dri/renderD129
transcode sanity PASS

Intel GPU card 2:
PCI:0000:25:00.0, ID:0x4907
card:/dev/dri/card2 render:/dev/dri/renderD130
transcode sanity PASS

Intel GPU card 3:
PCI:0000:2a:00.0, ID:0x4907
card:/dev/dri/card3 render:/dev/dri/renderD131
transcode sanity PASS

.....end.....
```

2. Execute the `glxinfo | grep Mesa` command to view Mesa driver version information.

Figure4-8 Viewing Mesa driver version information

```
[root@localhost SG1_20044_c7.4_k4.14.105]# glxinfo |grep Mesa
client glx vendor string: Mesa Project and SGI
OpenGL renderer string: Mesa Intel(R) Server Graphics (SG1 GT2)
OpenGL core profile version string: 4.6 (Core Profile) Mesa 20.3.0-devel (git-b00c3a03e4)
OpenGL version string: 4.6 (Compatibility Profile) Mesa 20.3.0-devel (git-b00c3a03e4)
OpenGL ES profile version string: OpenGL ES 3.2 Mesa 20.3.0-devel (git-b00c3a03e4)
```

3. Execute the `vainfo` command to view media driver version information.

Figure4-9 Viewing media driver version information

```
[root@localhost home]# vainfo
libva info: VA-API version 1.10.0
libva info: User environment variable requested driver 'iHD'
libva info: Trying to open /opt/intel/mediasdk/lib64/iHD_drv_video.so
libva info: Found init function __vaDriverInit_1_10
libva info: va_openDriver() returns 0
vainfo: VA-API version: 1.10 (libva 2.10.0.pre1)
vainfo: Driver version: Intel iHD driver for Intel(R) Gen Graphics - 20.3.pre-20044 (89c667f)
vainfo: Supported profile and entrypoints
  VAProfileNone          : VAEntrypointVideoProc
  VAProfileNone          : VAEntrypointStats
  VAProfileMPEG2Simple   : VAEntrypointVLD
  VAProfileMPEG2Simple   : VAEntrypointEncSlice
  VAProfileMPEG2Main     : VAEntrypointVLD
  VAProfileMPEG2Main     : VAEntrypointEncSlice
  VAProfileH264Main      : VAEntrypointVLD
  VAProfileH264Main      : VAEntrypointEncSlice
  VAProfileH264Main      : VAEntrypointFEI
```

Viewing GPU module information in the operating system

1. Execute the `lspci | grep VGA` command to view GPU information.

A GPU module typically has four GPUs. As shown in [Figure4-10](#), four PCI devices, each represented by Device 4907, are identified for a GPU module.

The first PCIe device with bus number 05:00.0 in [Figure4-10](#) is the VGA device provided on the system board.

Figure4-10 Viewing GPUs

```
[root@localhost ~]# lspci |grep VGA
05:00.0 VGA compatible controller: ASPEED Technology, Inc. ASPEED Graphics Family (rev 41)
1b:00.0 VGA compatible controller: Intel Corporation Device 4907 (rev 01)
20:00.0 VGA compatible controller: Intel Corporation Device 4907 (rev 01)
25:00.0 VGA compatible controller: Intel Corporation Device 4907 (rev 01)
2a:00.0 VGA compatible controller: Intel Corporation Device 4907 (rev 01)
```

- Execute the `lspci | grep 4910` command to view PCI bridges of the GPU module. Typically, four PCI bridges (Device 4910) are displayed for a GPU module as shown in [Figure4-11](#).

Figure4-11 Viewing PCI bridges

```
19:00.0 PCI bridge: Intel Corporation Device 4910
1e:00.0 PCI bridge: Intel Corporation Device 4910
23:00.0 PCI bridge: Intel Corporation Device 4910
28:00.0 PCI bridge: Intel Corporation Device 4910
```

- Execute the `lspci -vvvnn -s bus` command to view the detailed PCI information of a GPU. To obtain the bus number of a GPU, see [1](#).

Figure4-12 Viewing detailed PCI information about a GPU

```
root@9deff0cc9c:~/WS# lspci -vvvnn -s 1b:00.0
1b:00.0 VGA compatible controller [0300]: Intel Corporation Device [8086:4907] (rev 01) (prog-if 00 [VGA controller])
Control: I/O- Mem+ BusMaster+ SpecCycle- MemWINV- VGASnoop- ParErr- Stepping- SERR+ FastB2B- DisINTx-
Status: Cap+ 66MHz- UDF- FastB2B- ParErr- DEVSEL=fast >TAbort- <TAbort- <MAbort- >SERR- <PERR- INTx-
Latency: 0, Cache Line Size: 32 bytes
Interrupt: pin ? routed to IRQ 40
NUMA node: 0
Region 0: Memory at c2000000 (64-bit, non-prefetchable) [size=16M]
Region 2: Memory at 301f0000000 (64-bit, prefetchable) [size=4G]
Expansion ROM at c3000000 [disabled] [size=2M]
Capabilities: [40] Vendor Specific Information: Len=0c <?>
Capabilities: [70] Express (v2) Endpoint, MSI 00
DevCap: MaxPayload 128 bytes, PhantFunc 0, Latency L0s <64ns, L1 <1us
ExtTag+ AttnBtm- AttnInd- PwrInd- RBE+ FLReset- SlotPowerLimit 0.000W
DevCtl: Report errors: Correctable+ Non-Fatal+ Fatal+ Unsupported-
RlxDrd+ ExtTag+ PhantFunc- AuxPwr- NoSnoop+ FLReset-
MaxPayload 128 bytes, MaxReadReq 128 bytes
DevSta: CorrErr- UncorrErr- FatalErr- UnsuppReq+ AuxPwr- TransPend+
LnkCap: Port #0, Speed 2.5GT/s, Width x1, ASPM L0s L1, Exit Latency L0s <64ns, L1 <1us
ClockPM- Surprise- LLActRep- BwNot- ASPMOptComp-
LnkCtl: ASPM Disabled; RCB 64 bytes Disabled- CommClk-
ExtSynch- ClockPM- AutwidDis- BWInt- AutBWInt-
LnkSta: Speed 2.5GT/s, Width x1, TrErr- Train- SlotClk- DLActive- BWMgmt- ABWMgmt-
DevCap2: Completion Timeout: Range B, TimeoutDis+, LTR+, OBFF Not Supported
DevCtl2: Completion Timeout: 50us to 50ms, TimeoutDis-, LTR-, OBFF Disabled
LnkCtl2: Target Link Speed: 2.5GT/s, EnterCompliance- SpeedDis-
Transmit Margin: Normal Operating Range, EnterModifiedCompliance- ComplianceSOS-
Compliance De-emphasis: -6dB
LnkSta2: Current De-emphasis Level: -6dB, EqualizationComplete-, EqualizationPhase1-
EqualizationPhase2-, EqualizationPhase3-, LinkEqualizationRequest-
Capabilities: [ac] MSI: Enable+ Count=1/1 Maskable+ 64bit+
Address: 00000000fee000b8 Data: 0000
Masking: 00000000 Pending: 00000000
Capabilities: [d0] Power Management version 3
Flags: PMEClk- DSI- D1- D2- AuxCurrent=0mA PME(D0+,D1-,D2-,D3hot+,D3cold-)
Status: D0 NoSoftRst+ PME-Enable- Dsel=0 Dscale=0 PME-
Capabilities: [100 v1] Latency Tolerance Reporting
Max snoop latency: 0ns
Max no snoop latency: 0ns
Kernel driver in use: i915
lspci: Unable to load libkmod resources: error -12
```

- Execute the `lspci -s bus -vv | grep "LnkSta"` command to view the link status of a PCI bridge. To obtain the bus number of a PCI bridge, see [2](#).

Figure4-13 Viewing the link status of a PCI bridge

```
lroot@localhost ~# lspci -s 19:00.0 -vv |grep "LnkSta"
LnkSta: Speed 8GT/s, Width x8, TrErr- Train- SlotClk+ DLActive- BWMgmt- ABWMgmt-
LnkSta2: Current De-emphasis Level: -6dB, EqualizationComplete+, EqualizationPhase1+
```

Upgrading the firmware

For better use of the GPU module, upgrade the firmware to the latest version as a best practice. For more information, see the firmware installation guide. To obtain the guide, see "[Downloading the firmware and driver](#)."

5 Safety standards and EMC standards

Safety standards

UL 60950-1

UL 62368-1

IEC 60950-1

IEC 62368-1

EN 60950-1

EN 62368-1

GB 4943.1

EMC standards

EN 55024

EN 55032

EN 55035

CISPR 24

CISPR 32

CISPR 35

AS/NZS CISPR 32

FCC Part 15 Subpart B

ICES-003 Issue 7

ANSI C63.4

VCCI-CISPR 32

6 Acronyms

Acronym	Full name
A	
AVC	Advanced Video Coding
AV1	AOMedia Video 1
C	
CFM	Cubic Feet per Minute
E	
ECC	Error Correcting Code
EU	Execution Unit in the Graphics Processor
G	
GPU	Graphics Processing Unit
H	
HEVC	High Efficiency Video Coding
M	
MPEG2	Moving Picture Experts Group
S	
SoC	System-on-chip
SDK	Software Development Kit
T	
TDP	Thermal Design Power