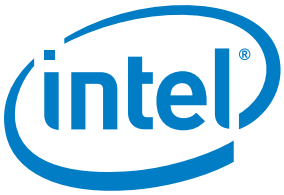


XenGT* for Intel® In-Vehicle Solution Development Kit

Getting Started Guide

March 2015

Document Number: 332058-001

Legal Disclaimer

You may not use or facilitate the use of this document in connection with any infringement or other legal analysis concerning Intel products described herein. You agree to grant Intel a non-exclusive, royalty-free license to any patent claim thereafter drafted which includes subject matter disclosed herein

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest Intel product specifications and roadmaps.

The products described may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting: http://www.intel.com/design/literature.htm

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Learn more at http://www.intel.com/ or from the OEM or retailer.

No computer system can be absolutely secure.

Basis, BlueMoon, BunnyPeople, Celeron, Centrino, Cilk, Flexpipe, Intel, the Intel logo, the Intel Anti-Theft technology logo, Intel AppUp, the Intel AppUp logo, Intel Atom, Intel CoFluent, Intel Core, Intel Inside, the Intel Inside logo, Intel Insider, Intel NetMerge, Intel NetStructure, Intel RealSense, Intel SingleDriver, Intel SpeedStep, Intel vPro, Intel Xeon Phi, Intel XScale, InTru, the InTru logo, the InTru Inside logo, InTru soundmark, Iris, Itanium, Kno, Look Inside., the Look Inside. logo, Mashery, MCS, MMX, Pentium, picoArray, Picochip, picoXcell, Puma, Quark, SMARTi, smartSignaling, Sound Mark, Stay With It, the Engineering Stay With It logo, The Creators Project, The Journey Inside, Thunderbolt, the Thunderbolt logo, Transcede, Transrf, Ultrabook, VTune, Xeon, X-GOLD, XMM, X-PMU and XPOSYS are trademarks of Intel Corporation in the U.S. and/or other countries

*Other names and brands may be claimed as the property of others.

Contents

Tables

Revision History

Date	Revision	Description
March 2015	001	Initial release

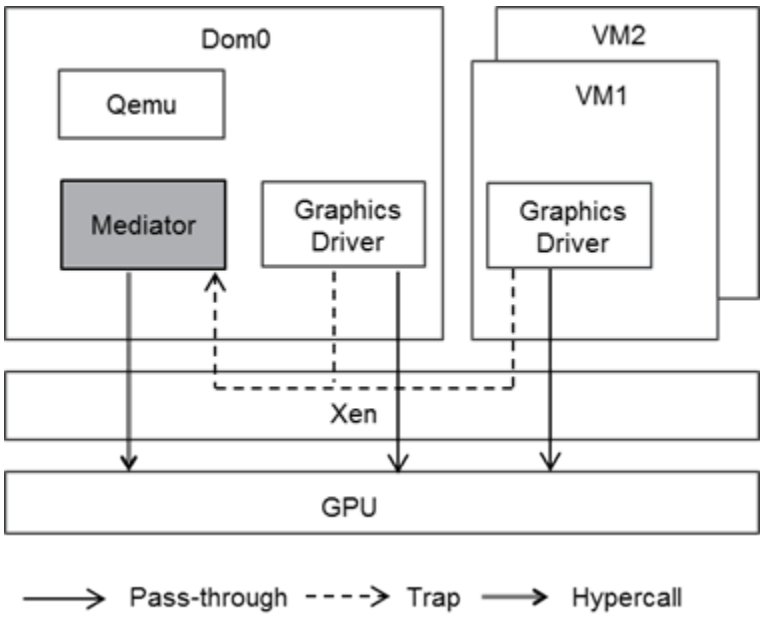
§

1 Introduction

The graphics processing unit (GPU) has become a fundamental building block in today's computing environment, accelerating tasks ranging from entertainment (gaming, video playback, etc.), GUI acceleration, automotive applications (such as navigation, digital instrument cluster, etc.) to high-performance computing.

A recent trend has been to add GPU acceleration to virtual machines provided by popular desktop virtualization. And, there has also been a demand for buying GPU computing resources from the cloud. GPU virtualization has become a challenging industry feature request!

Intel has answered this challenge with XenGT*, which is a mediated pass-through solution based on Intel Gen graphics hardware using the well-known Xen hypervisor. As illustrated below, XenGT allows a native graphics driver running in VMs to achieve high performance. Each VM is allowed to access a partial performance critical resource without hypervisor intervention. Privileged operations are trapped and forwarded to the mediator for emulation. The mediator creates a virtual GPU context for each VM and schedules one of them to run on physical Gen graphics hardware. In the implementation shown below, the mediator is a separate driver residing in the Dom0 kernel, called the "vgt" driver.



Each VM is allowed to access a partial performance critical resource without hypervisor intervention. Privileged operations are trapped by Xen and forwarded to the mediator for emulation. Context switches are conducted by the mediator when switching the GPU between VMs. XenGT implements the mediator in dom0. This avoids adding complex device knowledge to Xen, and also permits a more flexible release model. Meanwhile, a unified architecture to mediate all the VMs, including dom0, itself is

desired. So, the mediator is implemented as a separate module from dom0's graphics driver. This brings a new challenge: Xen must selectively trap the accesses from dom0's driver while granting permission to the mediator. This is called a "deprivileged" dom0 mode.

Performance-critical resources are passed through to a VM as:

- Part of the global virtual memory space
- VM's own per-process virtual memory spaces
- VM's own allocated command buffers (actually in graphics memory)

This minimizes hypervisor intervention in the critical rendering path. Even when a VM is not scheduled to use the render engine, that VM can continuously queue commands in parallel.

Other operations are privileged, and must be trapped and emulated by the mediator, including:

- MMIO/PIO
- PCI configuration registers
- GTT tables
- Submission of queued GPU commands

The mediator maintains the virtual GPU context based on the traps mentioned above, and schedules use of the render engine among VMs to ensure secure sharing of the single physical GPU.

Note: Intel tested XenGT's functionality using the 64-bit version of Ubuntu 12.04 and 14.04.

1.1 Terminology

Table 1. Terminology

Term	Description
apt	Advanced Packing Tool
EFI	Extensible Firmware Interface
GPU	Graphics Processing Unit
GUI	Graphical User Interface
HD	Hard Disk
HSW	Haswell
IVB	Ivy Bridge

Term	Description
OS	Operating System
SNB	Sandy Bridge
SSD	Solid-State Drive
SSH	Secure Shell
Vgt / VGT	Virtual Guest Tagging
VM	Virtual Machine

1.2 Reference Documents

Table 2. Reference Documents

Document	Link
XenGT_Setup_Guide.pdf	https://github.com/01org/XenGT-Preview-kernel/blob/byt-experiment/XenGT_Setup_Guide.pdf
XenGT Architecture	https://01.org/xen/blogs/srclarkx/2013/graphics-virtualization-xengt

§

2 System Requirements

2.1 Operating System Requirements

The build and install environment has been validated using x86_64 Ubuntu 12.04 and 14.04 as the host.

2.2 Hardware Requirements

- SSD
- 4th Generation Intel® Core™ processor graphics
- Sandy Bridge/Ivy Bridge/Haswell platform
- Intel In-Vehicle FFRD

2.3 Software Requirements

2.3.1 Intel In-Vehicle FFRD

The following binary file is required to flash the system BIOS:

CVHSCRB_X64_R_SPI_0092_30_SeC_Enable.bin

3 Build and Host Installation Instructions

3.1 Ubuntu Installation

Standard Ubuntu* 14.04 installation with 'Download & Third Party' option.

Note:

- Intel tested XenGT's functionality using the 64-bit version of Ubuntu 12.04 and
- Used SNB/IVB/HSW for content provided in Section 3.2 – 3.3

3.1.1 Proxy Configuration Setup

If building a package behind a firewall, set up the following proxies in order to download needed libraries:

```
# export http_proxy=<proxy_server>:<proxy_port>

# export https_proxy=<proxy_server>:<proxy_port>

# export ftp_proxy=<proxy_server>:<proxy_port>
```

To configure the proxy for apt, you need add following lines into /etc/apt/apt.conf

```
Acquire::http::proxy "<proxy_server>:<proxy_port>";

Acquire::https::proxy "<proxy_server>:<proxy_port>";

Acquire::ftp::proxy "<proxy_server>:<proxy_port>";
```

3.2 Install Basic Packages in Ubuntu

```
# apt-get update
# apt-get -y install openssh-server
```

SSH remotely to the target and complete the following:

```
# echo "user ALL=(ALL) NOPASSWD:ALL" >> /etc/sudoers

# echo "xhost + local:" >> /etc/X11/Xsession.d/40x11-common_xsessionrc

# echo "xset s off -dpms" >> /etc/X11/Xsession.d/40x11-common_xsessionrc
# apt-get -y install build-essential libarchive-dev \
libghc-bzlib-dev zlib1g-dev mercurial gettext bcc \
iasl uuid-dev libncurses5-dev kpartx  libperl-dev \
```

```
libgtk2.0-dev libaio-dev libsdl1.2-dev  nfs-common\
libyajl-dev libx11-dev autoconf libtool xsltproc bison \
flex  xutils-dev xserver-xorg-dev x11proto-gl-dev \
libx11-xcb-dev vncviewer  libxcb-glx0 libxcb-glx0-dev \
libxcb-dri2-0-dev libxcb-xfixes0-dev bridge-utils  \
python-dev bin86 git vim libssl-dev pciutils-dev \
tightvncserver ssh texinfo gnome-tweak-tool unity-2d \
libudev-dev mesa-utils ncurses-dev autoconf libtool \
freeglut3-dev swig2.0 libc6-dev-i386 libaio-dev \
gnome-session-fallback openssh-server aptitude \
libegl1-mesa-dev libxml2-dev yasm  libgles2-mesa-dev \
libgbm-dev glmark2 glmark2-es2 libva-intel-vaapi-driver \
libva-glx1 libva-x11-1 libva1
```

Note: To avoid clock skew issues, set the date and time before proceeding to the kernel setup.

3.3 Kernel Setup

1. Create work directory

   ```
   # mkdir /work
   ```

2. Edit grub and comment '#' GRUB_HIDDEN_TIMEOUT=0

   ```
   # vi /etc/default/grub
   comment '#' GRUB_HIDDEN_TIMEOUT=0
   # update-grub
   ```

3. Backup grub

   ```
   # cd /work
   # cp /boot/grub/grub.cfg grub_cfg
   # cp /etc/default/grub grub_etc
   ```

4. Kernel rebuild

   ```
   # cd /work
   # git clone https://github.com/01org/XenGT-Preview-kernel.git
   # cd XenGT-Preview-kernel/
   # git checkout byt-experiment
   # git clone git://kernel.ubuntu.com/ubuntu-archive/ubuntu-
   saucy.git linux-vgt
   # cd linux-vgt
   # git checkout 549fad2377f797d330565a7a7669b478ba474091 -b v3.11.6
   # patch -p1 < ../linux-vgt.patch
   # patch -p1 < ../byt-support-experiment.patch
   # cp config-3.11.6-dom0 .config
   # make –j4 && make modules_install
   # mkinitramfs -o /boot/initrd-vgt-3.11.6-vgt.img 3.11.6-vgt+
   # cp arch/x86/boot/bzImage /boot/vmlinuz-vgt-3.11.6-vgt
   # cp vgt.rules /etc/udev/rules.d
   # chmod a+x vgt_mgr
   # cp vgt_mgr /usr/bin
   ```

5. Kernel rebuild with qemu

```
# cd /work
# git clone https://github.com/01org/XenGT-Preview-xen.git
# cd XenGT-Preview-xen/
# git checkout byt-experiment
# git clone https://github.com/01org/XenGT-Preview-qemu.git
# git clone git://xenbits.xen.org/xen.git xen-vgt
# git clone git://git.qemu-project.org/qemu.git qemu-xen
# cd qemu-xen
# git checkout -b v1.3.0 v1.3.0
# patch -p1 < ../XenGT-Preview-qemu/qemu-vgt.patch
# cd ../xen-vgt
# git checkout -b RELEASE-4.3.1 RELEASE-4.3.1
# patch -p1 < ../xen-vgt.patch
# patch -p1  < ../byt-support-experiment.patch
# cp –r ../qemu-xen tools/
# sed -i '/QEMU_UPSTREAM_URL/s:http\://xenbits.xen.org/git-
http/qemu-upstream-4.3-testing.git:$(XEN_ROOT)/tools/qemu-xen:'
Config.mk
# sed -i '/QEMU_UPSTREAM_URL/s:git\://xenbits.xen.org/qemu-
upstream-4.3-testing.git:$(XEN_ROOT)/tools/qemu-xen:' Config.mk
# ./autogen.sh
# ./configure --prefix=/usr
# make –j8 xen tools
# cp xen/xen.gz /boot/xen-vgt.gz
# make install-tools PYTHON_PREFIX_ARG=
```

6. Starting xen service by default

```
# update-rc.d xencommons defaults
```

7. Edit grub to include new kernel configuration for Xen

```
# vi /boot/grub/grub.cfg

menuentry 'Xen-VGT 3.11.6' --class ubuntu --class gnu-linux --
class gnu --class os {
insmod part_msdos
insmod ext2
set root='(hd0,msdos1)'
search --no-floppy --fs-uuid --set=root ce5a00dd-68d0-4cdb-a1d8-
2204f46f3845
multiboot    /boot/xen-vgt.gz dom0_mem=2048M loglvl=all
guest_loglvl=all conring_size=4M noreboot
module   /boot/vmlinuz-vgt-3.11.6-vgt  root=UUID=ce5a00dd-68d0-
4cdb-a1d8-2204f46f3845 rw rd_NO_LUKS rd_NO_LVM LANG=en_US.UTF-8
rd_NO_MD SYSFONT=latarcyrheb-sun16 rhgb crashkernel=auto
KEYBOARDTYPE=pc KEYTABLE=us rd_NO_DM ignore_loglevel console=tty0
console=hvc0 consoleblank=0 log_buf_len=4M
xen_vgt.hvm_boot_foreground=1
module   /boot/initrd-vgt-3.11.6-vgt.img
}
```

8. To enable serial/Graphics in grub

```
# vi /boot/grub/grub.cfg
```

Replace terminal_output gfxterm with:

```
#set have_gfxconsole=true
if [ -n "${have_gfxconsole}" ]; then
 terminal_output gfxterm
else
 serial --speed=115200 --unit=0 --word=8 --parity=no --stop=1
 terminal_input serial
 terminal_output serial
fi
```

9. Reboot the system

Note: From this point in this document onwards, XenGT is only for use with Intel® Atom™ Processor E3845 or Haswell.

3.4 Setup on Intel In-Vehicle FFRD

Connect an Ubuntu-installed SSD to the Intel In-Vehicle FFRD.

3.4.1 Firmware Setup

1. Flash SPI on the platform with EFI firmware
2. Ensure the correct configuration is used:
   i. Uncore | apparture → 256MB
   ii. Boot → enable legacy HD

3.4.2 Network Configuration

The Intel In-Vehicle FFRD uses the Intel I210 Ethernet module. For the interface to work, the igb_avb driver needs to be installed and loaded (please note that the ibg_avb driver does not come as part of Ubuntu pre-installed drivers)..

Open source link: https://github.com/AVnu/Open-AVB/tree/master/kmod/igb

```
# cd /work
# tar xvf igb_openavb-20130814.tgz
# cd igb
# make -j 4 install
# modprobe igb_avb (only for the first time)
```

3.4.3 Xen Bridge Configuration

To get network access over a guest OS, network bridging should be enabled.

```
# brctl addbr xenbr0
# ifconfig eth0 0.0.0.0 down
# brctl addif xenbr0 eth0
# ifconfig eth0 up
# dhclient xenbr0
```

```
# ifconfig
```

3.4.4 Graphics Performance Tuning

The GPU generally operates at an optimal frequency, depending on the workload. It can be tweaked for higher performance by toggling to max frequency.

```
# vi maxperf.sh
```

Add:

```
/usr/sbin/xenpm  set-sched-smt  disable
/usr/sbin/xenpm  set-scaling-maxfreq  1909000
/usr/sbin/xenpm  set-scaling-minfreq  1909000
/usr/sbin/xenpm  set-scaling-governor  userspace
/usr/sbin/xenpm  get-cpufreq-para
echo 797 > /sys/class/drm/card0/gt_min_freq_mhz
echo 797 > /sys/class/drm/card0/gt_max_freq_mhz

# chmod +x maxperf.sh

# service xencommons start

# service xend start

# ./maxperf.sh
```

4 Guest OS Installation and Configuration

4.1 General Setup

1. Archive the VGT kernel from host configuration for easier VM integration

```
# tar cvfz 311_vgt.tgz /boot/initrd-vgt-3.11.6-vgt.img
/boot/vmlinuz-vgt-3.11.6-vgt /lib/modules/3.11.6-vgt+/
```

2. For easier file copying and remote connection:

```
# vi /etc/ssh/sshd_config
Comment → #PermitRootLogin without-password
Add → PermitRootLogin yes
```

3. Copy the Ubuntu installer

```
# mkdir /DEMO
# cd /DEMO
# cp ubuntu-14.04.1-desktop-amd64.iso .
```

4. Create an empty image with at least 10GB for the guest

```
# dd if=/dev/zero of=system-10G.img bs=1M seek=10000 count=0
```

4.2 Guest Configuration

Copy Xen configuration file from dom0 (Ubuntu-host) and modify it for VM (Guest)

```
# cp /etc/xen/xmexample.hvm vgt_ub.hvm
# vi vgt_ub.hvm

memory =  1024
name = "vgtHVMDomain"
disk = [ 'file:/DEMO/system-10G.img,hda,w', 'file:/DEMO/ubuntu-
14.04.1-desktop-amd64.iso,hdc:cdrom,r' ]
vif = [ 'type=ioemu,
mac=00:16:3e:2f:3a:28,bridge=xenbr0,model=e1000' ]
#device_model = 'qemu-dm'
device_model_version='qemu-xen'
device_model_override='/usr/lib/xen/bin/qemu-system-i386'
boot="d"
vnc=0
usb=1
usbdevice='tablet'
```

```
keymap='en-us'
vgt=0
vgt_low_gm_sz=128
vgt_high_gm_sz=192
vgt_fence_sz=4
```

XenGT parameters description:

Configuration Options	Description
vgt	Enable virtual graphics acceleration
vgt_low_gm_sz	The low gm size which is CPU-visible
vgt_high_gm_sz	The high gm size which is CPU-invisible
vgt_fence_sz	The number of the fence registers; default is 4

4.3 Ubuntu Guest Setup

This section provides step-by-step instructions for installing and configuring Ubuntu 14.04. The OS can be installed and configured as a guest with graphics virtualization.

1. Create guest OS; it may take a while for the Ubuntu OS installation to complete

   ```
   # cd /DEMO
   # xl create vgt_ub.hvm
   ```

2. After Ubuntu OS installation, stop the guest OS

   ```
   # xl destroy vgtHVMDomain
   ```

3. After installation, change the boot partition to start installing the guest OS by modifying the config file

   ```
   # vi vgt_ub.hvm
   ```

   Change boot=”c”

4. Start the guest OS again

   ```
   # xl create vgt_ub.hvm
   ```

5. Configure and set up the Ubuntu OS

   ```
   # apt-get update
   # apt-get -y install openssh-server
   ```

6. From SSH:

   ```
   apt-get -y install build-essential libarchive-dev \
   libghc-bzlib-dev zlib1g-dev mercurial gettext bcc \
   iasl uuid-dev libncurses5-dev kpartx  libperl-dev \ libgtk2.0-dev
   libaio-dev libsdl1.2-dev  nfs-common \
   libyajl-dev libx11-dev autoconf libtool xsltproc bison \
   flex  xutils-dev xserver-xorg-dev x11proto-gl-dev \
   ```

```
libx11-xcb-dev vncviewer  libxcb-glx0 libxcb-glx0-dev \ libxcb-
dri2-0-dev libxcb-xfixes0-dev bridge-utils  \ python-dev bin86 git
vim libssl-dev pciutils-dev \ tightvncserver ssh texinfo  gnome-
tweak-tool unity-2d \ libudev-dev mesa-utils ncurses-dev autoconf
libtool \ freeglut3-dev swig2.0 libc6-dev-i386 libaio-dev \
gnome-session-fallback openssh-server aptitude \
libegl1-mesa-dev libxml2-dev yasm  libgles2-mesa-dev \ libgbm-dev
glmark2 glmark2-es2 libva-intel-vaapi-driver \ libva-glx1 libva-
x11-1 libva1
```

7. Logout of the guest OS

8. Perform grub updates

```
vi /etc/default/grub
#GRUB_HIDDEN_TIMEOUT=0
update-grub
```

9. Copy kernel images with xen & qemu enabled

```
cd /
ssh <dom0 ip>
scp root@<dom0 ip>:/311_vgt.tgz /
tar xvf 311_vgt.tgz
```

10. Edit grub to enable a new kernel image

```
# vi /boot/grub/grub.cfg

menuentry 'Ubuntu-VGT' --class ubuntu --class gnu-linux --class
gnu --class os $menuentry_id_option 'gnulinux-simple-f564a5c7-
f030-41c5-8c63-32290c387402' {
       recordfail
       load_video
       gfxmode $linux_gfx_mode
       insmod gzio
       insmod part_msdos
       insmod ext2
       if [ x$feature_platform_search_hint = xy ]; then
         search --no-floppy --fs-uuid --set=root  f564a5c7-f030-
41c5-8c63-32290c387402
       else
         search --no-floppy --fs-uuid --set=root f564a5c7-f030-
41c5-8c63-32290c387402
       fi
       linux   /boot/vmlinuz-vgt-3.11.6-vgt root=/dev/xvda1 ro
quiet splash $vt_handoff
       initrd  /boot/initrd-vgt-3.11.6-vgt.img
}
```

11. Shut down the guest OS and finalize the configuration to enable graphics acceleration on the guest OS

```
# vi vgt_ub.hvm

vgt=1
vgt_low_gm_sz=128
vgt_high_gm_sz=192
```

```
vgt_fence_sz=4

# modprobe loop
# kpartx -a -v /DEMO/system-10G.img
# mount /dev/mapper/loop0p1 /mnt/
```

4.4 Testing Guest OS

Start the guest OS

```
# xl create vgt_ub.hvm

# glxinfo

# vblank_mode=0 glxgears
```

Note: 3D Test: Install glmark2-es2

§