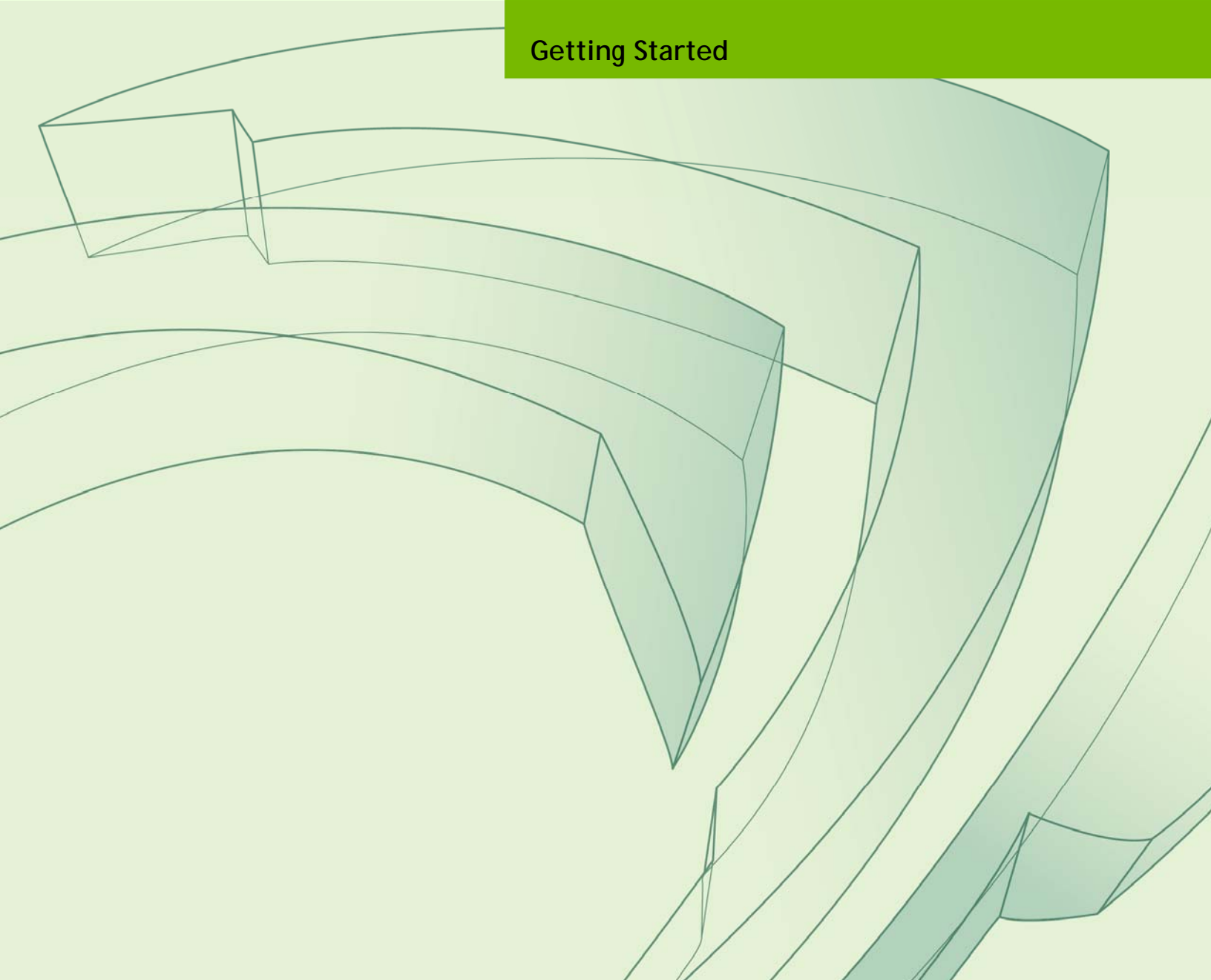




# NVIDIA CUDA C INSTALLATION AND VERIFICATION ON LINUX SYSTEMS

DU-80001-001\_v01 | April 2010

**Getting Started**



## DOCUMENT CHANGE HISTORY

DU-80001-001\_v01

| Version | Date            | Authors | Description of Change |
|---------|-----------------|---------|-----------------------|
| 01      | April 20, 2-010 | CW, TS  | Release               |
|         |                 |         |                       |
|         |                 |         |                       |

# TABLE OF CONTENTS

|  |    |
|--|----|
| <b>Introduction</b> .....                          | 5  |
| CUDA-Supercomputing on Desktop Systems .....       | 5  |
| System Requirements.....                           | 6  |
| About This Document .....                          | 6  |
| <b>Installing CUDA Development Tools</b> .....     | 7  |
| Verify You Have a CUDA-Enabled System .....        | 7  |
| Verify You Have a Supported Version of Linux ..... | 8  |
| Verify That gcc Is Installed.....                  | 8  |
| Download the NVIDIA Driver and CUDA Software.....  | 9  |
| Install the NVIDIA Driver .....                    | 10 |
| Install the CUDA Software.....                     | 11 |
| Verify the Installation .....                      | 12 |
| Compiling the Examples .....                       | 12 |
| Running the Binaries .....                         | 12 |
| <b>Additional Considerations</b> .....             | 15 |

## LIST OF FIGURES

|   |    |
|---|----|
| Figure 1. Valid Results from SDK deviceQuery Program.....   | 13 |
| Figure 2. Valid Results from SDK bandwidthTest Program..... | 14 |

# INTRODUCTION

## CUDA-SUPERCOMPUTING ON DESKTOP SYSTEMS

NVIDIA® CUDA™ is a general purpose parallel computing architecture introduced by NVIDIA. It includes the CUDA Instruction Set Architecture (ISA) and the parallel compute engine in the GPU. To program to the CUDA architecture, developers can use C, one of the most widely used high-level programming languages, which can then be run at great performance on a CUDA-enabled processor.

The CUDA architecture and its associated software were developed with several design goals in mind:

- ▶ Provide a small set of extensions to standard programming languages, like C, that enable a straightforward implementation of parallel algorithms. With CUDA and C for CUDA, programmers can focus on the task of parallelization of the algorithms rather than spending time on their implementation.
- ▶ Support heterogeneous computation where applications use both the CPU and GPU. Serial portions of applications are run on the CPU, and parallel portions are offloaded to the GPU. As such, CUDA can be incrementally applied to existing applications. The CPU and GPU are treated as separate devices that have their own memory spaces. This configuration also allows simultaneous computation on both the CPU and GPU without contention for memory resources.

CUDA-enabled GPUs have hundreds of cores that can collectively run thousands of computing threads. Each core has shared resources, including registers and memory. The on-chip shared memory allows parallel tasks running on these cores to share data without sending it over the system memory bus.

This guide will show you how to install and check the correct operation of the CUDA Development Tools.

## SYSTEM REQUIREMENTS

To use CUDA on your system, you will need the following installed:

- ▶ CUDA-enabled GPU
- ▶ Device driver
- ▶ A supported version of Linux with a gcc compiler and toolchain
- ▶ CUDA software (available at no cost from <http://www.nvidia.com/cuda>)

## ABOUT THIS DOCUMENT

This document is intended for readers familiar with the Linux environment and the compilation of C programs from the command line. You do not need previous experience with CUDA or experience with parallel computation. Note: This guide covers installation only on systems running X Windows.



Note: Many commands in this document might require super-user privileges. On most distributions of Linux, this will require you to log in as root. For systems that have enabled the sudo package, use the sudo prefix for all necessary commands. We will no longer remark on the matter of user privilege for the installation process except where critical to correct operation.

# INSTALLING CUDA DEVELOPMENT TOOLS

The installation of CUDA development tools on a system running the appropriate version of Linux consists of four simple steps:

- ▶ Verify the system has a CUDA-enabled GPU and a supported version of Linux.
- ▶ Download the NVIDIA driver and the CUDA software.
- ▶ Install the NVIDIA driver.
- ▶ Install the CUDA software.

Test your installation by compiling and running one of the sample programs in the CUDA software to validate that the hardware and software are running correctly and communicating with each other.

## VERIFY YOU HAVE A CUDA-ENABLED SYSTEM

Many NVIDIA products today contain CUDA-enabled GPUs. These include:

- ▶ NVIDIA GeForce® 8, 9, 200, and 400 series GPUs
- ▶ NVIDIA Tesla™ computing solutions
- ▶ Many of the NVIDIA Quadro® products

An up-to-date list of CUDA-enabled GPUs can be found on the NVIDIA CUDA Web site at [http://www.nvidia.com/object/cuda\\_gpus.html](http://www.nvidia.com/object/cuda_gpus.html)

The Release Notes for the CUDA Toolkit also contain a list of supported products.

To verify which video adapter your system uses, find the model number by going to your distribution's equivalent of System Properties, or, from the command line, enter:

```
lspci | grep -i nvidia
```

If you do not see any settings, update the PCI hardware database that Linux maintains by entering `update-pciids` (generally found in `/sbin`) at the command line and rerun the previous `lspci` command.

## Verify You Have a Supported Version of Linux

The CUDA Development Tools are only supported on some specific distributions of Linux. These are listed in the CUDA Toolkit release notes.

To determine which distribution and release number you're running, type the following at the command line:

```
uname -m && cat /etc/*release
```

You should see output similar to the following, modified for your particular system:

```
i386  
Red Hat Enterprise Linux WS release 4 (Nahant Update 6)
```

The `i386` line indicates you are running on a 32-bit system. On 64-bit systems running in 64-bit mode, this line will generally read: `x86_64`. The second line gives the version number of the operating system.

## Verify That gcc Is Installed

The `gcc` compiler and toolchain generally are installed as part of the Linux installation, and in most cases the version of `gcc` installed with a supported version of Linux will work correctly.

To verify the version of `gcc` installed on your system, type the following on the command line:

```
gcc --version
```

If an error message displays, you need to install the "development tools" from your Linux distribution or obtain a version of `gcc` and its accompanying toolchain from the Web.



## DOWNLOAD THE NVIDIA DRIVER AND CUDA SOFTWARE

Once you have verified that you have a supported NVIDIA processor and a supported version of Linux, you need to make sure you have a recent version of the NVIDIA driver. The CUDA Toolkit release notes specify which minimum version of the NVIDIA driver is required.

On many distributions, the driver release number can be found in the graphical interface menus under **Applications**→**System Tools**→**NVIDIA X Server Settings**. Or, from the command line, run:

```
/usr/bin/nvidia-settings.
```

The following CUDA software is required to run CUDA programs:

- ▶ The CUDA Toolkit  
The CUDA Toolkit contains the tools needed to compile and build a CUDA application in conjunction with the compilation driver. It includes tools, libraries, header files, and other resources.
- ▶ The CUDA SDK  
The CUDA SDK includes sample projects that provide source code and other resources for constructing CUDA programs.

The NVIDIA driver and CUDA software are available at no cost from the main CUDA download site at [http://www.nvidia.com/object/cuda\\_get.html](http://www.nvidia.com/object/cuda_get.html) .

Choose the Linux distribution you are using, click **Search**, and download the NVIDIA driver. Save the driver file on your local system. Likewise, download and save the SDK and Toolkit.

## INSTALL THE NVIDIA DRIVER

With the NVIDIA driver and software downloaded, you need to install the driver. Use the following procedure to install the driver:

1. Exit the GUI if you are in a GUI environment, by pressing **Ctrl-Alt-Backspace**. Some distributions require you to press this sequence twice in a row, others have disabled it altogether in favor of a command such as `sudo /etc/init.d/gdm stop`. At the command line, turn off X Windows using `/sbin/init 3`.
2. Run the driver installation package from the command line as a *superuser*.
3. Restart the GUI environment (`startx` or `init 5` or the equivalent command on your system).
4. Verify that the correct version of the driver is installed. This can be done through your System Properties (or equivalent).

More information on installing the driver is available at

<http://us.download.nvidia.com/XFree86/Linux-x86/190.53/README/index.html>.



Note: New versions of CUDA software can require later versions of Linux and of the NVIDIA driver, so always verify that you are running the correct release for the version of CUDA software you are using.

## INSTALL THE CUDA SOFTWARE

This section describes the installation and configuration of the CUDA Toolkit and SDK, which you previously downloaded.

Before installing the CUDA software packages, you should read the **Release Notes** bundled with each, as those notes provide important details on installation and software functionality.

Then, follow these few steps for a successful installation.

1. Uninstall any previous versions of the CUDA Toolkit and CUDA SDK.  
Do this by deleting the files from `/usr/local/cuda` and from `$(HOME)/NVIDIA_CUDA_SDK/`, the default installation locations. Adjust accordingly if you placed the files elsewhere. (If you wish to keep the files so you can compile for different versions of CUDA software, then rename the existing directories and modify your makefile accordingly.)
2. Install the CUDA Toolkit by running the downloaded `.run` file as a *super-user*. The CUDA Toolkit installation defaults to `/usr/local/cuda`.
3. Define the environment variables.
  - The `PATH` variable needs to include `/usr/local/cuda/bin`.
  - `LD_LIBRARY_PATH` needs to contain either `/usr/local/cuda/lib` or `/usr/local/cuda/lib64` for 32- or 64-bit operating systems, respectively.

The typical way to place these values in your environment is with the following commands:

```
export PATH=/usr/local/cuda/bin:$PATH
export LD_LIBRARY_PATH=/usr/local/cuda/lib:$LD_LIBRARY_PATH
```

for 32-bit operating systems, with `lib64` replacing `lib` for 64-bit operating systems as mentioned above. To make such settings permanent, place them in `~/.bash_profile`.

4. Install the SDK (located in the second `.run` file) as a *regular user* in the default location, `$(HOME)/NVIDIA_GPU_Computing_SDK`. Note that this location is different than the default location in previous versions:  
`$(HOME)/NVIDIA_CUDA_SDK`.  
Installing as a regular user avoids access issues.



**Important:** Best practice for a multiuser Linux system is to also install a version as root that is accessible to users on a read-only basis. This pristine copy can then be

copied to a user directory in the event users corrupt their copy of the source code.

## VERIFY THE INSTALLATION

Before continuing, it is important to verify that the CUDA programs can find and communicate correctly with the CUDA-enabled hardware. To do this, you need to compile and run some of the included sample programs.

### Compiling the Examples

The version of the CUDA Toolkit can be checked by running `nvcc -V` in a terminal window. The `nvcc` command runs the compiler driver that compiles CUDA programs. It calls the `gcc` compiler for C code and the NVIDIA PTX compiler for the CUDA code.

NVIDIA includes sample programs in source form in the CUDA SDK. You should compile them all by changing to `NVIDIA_GPU_Computing_SDK/C` in the user's home directory and typing `make`. The resulting binaries will be installed under the home directory in `NVIDIA_GPU_Computing_SDK/C/bin/linux/release`.

### Running the Binaries

The sample projects use libraries pointed to by `LD_LIBRARY_PATH`, as described earlier, so make sure it points to the right directory.

After compilation, go to `NVIDIA_GPU_Computing_SDK/C/bin/linux/release` in the user's home directory and run `deviceQuery`. If the CUDA software is installed and configured correctly, the output for `deviceQuery` should look similar to that shown in Figure 1. The exact appearance and the output lines might be different on your system. The important outcomes are that a device was found (the first highlighted line), that the device matches the one on your system (the second highlighted line), and that the test passed (the final highlighted line).

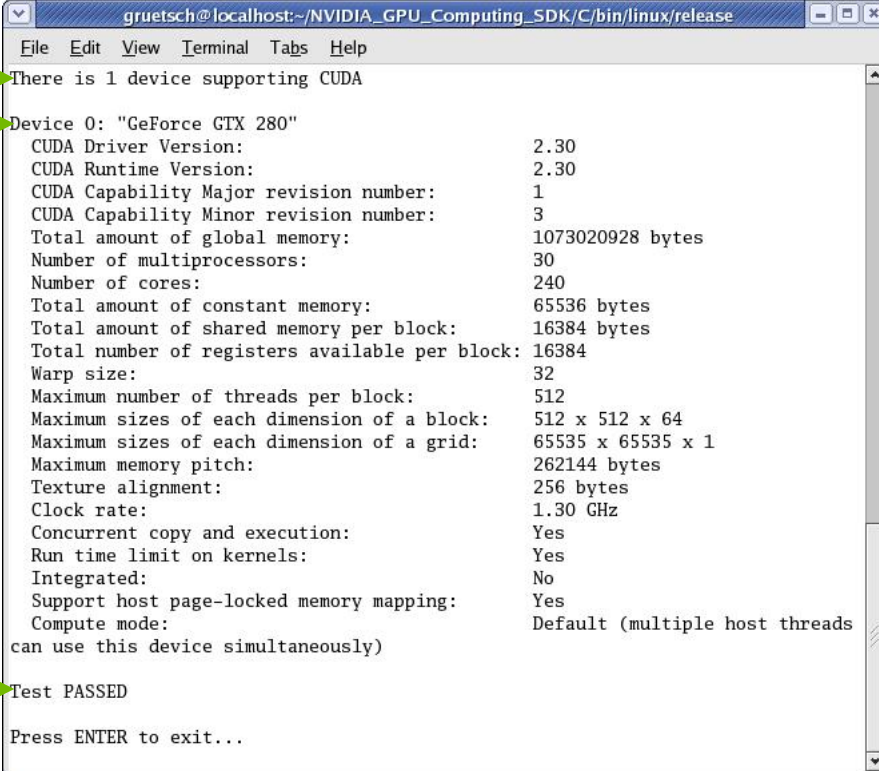
On systems where SELinux is enabled, you might need to temporarily disable this security feature to run `deviceQuery`. To do this, type:

```
#setenforce 0
```

from the command line as the *super-user*.

On multiuser systems, access to NVIDIA devices must be enabled for remote users. To do this, enable read-write privileges for all users on `/dev/nv* devices`.

Running the **bandwidthTest** program ensures that the system and the CUDA-enabled device are able to communicate correctly. Its output is shown in Figure 2.

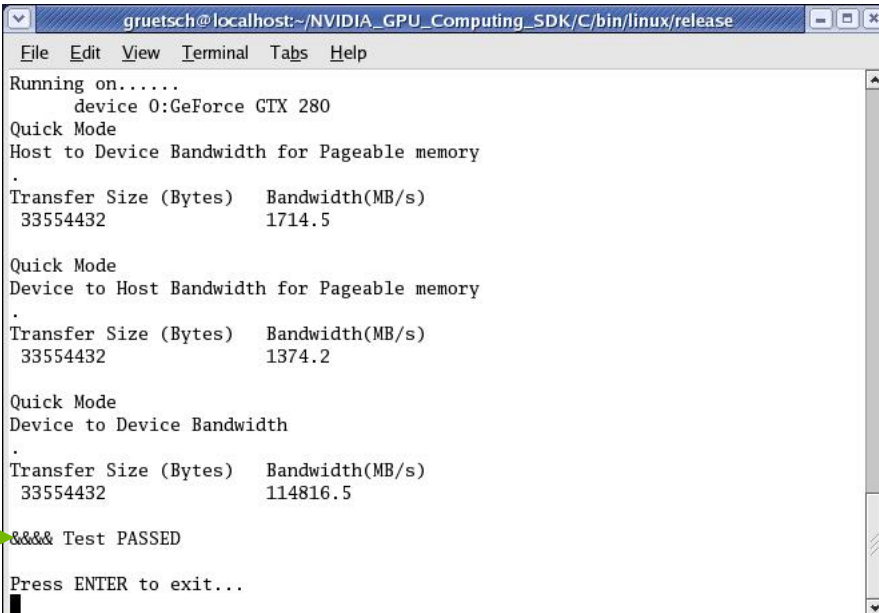


```

gruetsch@localhost:~/NVIDIA_GPU_Computing_SDK/C/bin/linux/release
File Edit View Terminal Tabs Help
There is 1 device supporting CUDA
Device 0: "GeForce GTX 280"
  CUDA Driver Version:          2.30
  CUDA Runtime Version:        2.30
  CUDA Capability Major revision number: 1
  CUDA Capability Minor revision number: 3
  Total amount of global memory: 1073020928 bytes
  Number of multiprocessors:    30
  Number of cores:             240
  Total amount of constant memory: 65536 bytes
  Total amount of shared memory per block: 16384 bytes
  Total number of registers available per block: 16384
  Warp size:                   32
  Maximum number of threads per block: 512
  Maximum sizes of each dimension of a block: 512 x 512 x 64
  Maximum sizes of each dimension of a grid: 65535 x 65535 x 1
  Maximum memory pitch:        262144 bytes
  Texture alignment:           256 bytes
  Clock rate:                  1.30 GHz
  Concurrent copy and execution: Yes
  Run time limit on kernels:   Yes
  Integrated:                  No
  Support host page-locked memory mapping: Yes
  Compute mode:                Default (multiple host threads
can use this device simultaneously)
Test PASSED
Press ENTER to exit...

```

Figure 1. Valid Results from SDK deviceQuery Program



```

gruetsch@localhost:~/NVIDIA_GPU_Computing_SDK/C/bin/linux/release
File Edit View Terminal Tabs Help
Running on.....
  device 0:GeForce GTX 280
Quick Mode
Host to Device Bandwidth for Pageable memory
.
Transfer Size (Bytes)  Bandwidth(MB/s)
33554432              1714.5

Quick Mode
Device to Host Bandwidth for Pageable memory
.
Transfer Size (Bytes)  Bandwidth(MB/s)
33554432              1374.2

Quick Mode
Device to Device Bandwidth
.
Transfer Size (Bytes)  Bandwidth(MB/s)
33554432              114816.5

Test PASSED
Press ENTER to exit...

```

## Figure 2. Valid Results from SDK bandwidthTest Program

Note that the measurements for your CUDA-enabled device description will vary from system to system. The important point is that you obtain measurements, and that the second-to-last line (in Figure 2) confirms that all necessary tests passed.

Should the tests not pass, make sure you have a CUDA-enabled NVIDIA GPU on your system and make sure it is properly installed.

If you run into difficulties with the link step (such as libraries not being found), consult the Linux Release Notes found in the doc folder in the SDK directory.

# ADDITIONAL CONSIDERATIONS

Now that you have CUDA-enabled hardware and the software installed, you can examine and enjoy the numerous included programs. To begin using CUDA to accelerate the performance of your own applications, consult the *NVIDIA CUDA Programming Guide*, located in `/usr/local/cuda/doc`.

For technical support on programming questions, consult and participate in the bulletin board and mailing list at <http://forums.nvidia.com/index.php?showforum=71>.

## Notice

ALL NVIDIA DESIGN SPECIFICATIONS, REFERENCE BOARDS, FILES, DRAWINGS, DIAGNOSTICS, LISTS, AND OTHER DOCUMENTS (TOGETHER AND SEPARATELY, "MATERIALS") ARE BEING PROVIDED "AS IS." NVIDIA MAKES NO WARRANTIES, EXPRESSED, IMPLIED, STATUTORY, OR OTHERWISE WITH RESPECT TO THE MATERIALS, AND EXPRESSLY DISCLAIMS ALL IMPLIED WARRANTIES OF NONINFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE.

Information furnished is believed to be accurate and reliable. However, NVIDIA Corporation assumes no responsibility for the consequences of use of such information or for any infringement of patents or other rights of third parties that may result from its use. No license is granted by implication of otherwise under any patent rights of NVIDIA Corporation. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all other information previously supplied. NVIDIA Corporation products are not authorized as critical components in life support devices or systems without express written approval of NVIDIA Corporation.

## HDMI

HDMI, the HDMI logo, and High-Definition Multimedia Interface are trademarks or registered trademarks of HDMI Licensing LLC.

## ROVI Compliance Statement

NVIDIA Products that are ROVI-enabled can only be sold or distributed to buyers with a valid and existing authorization from ROVI to purchase and incorporate the device into buyer's products.

This device is protected by U.S. patent numbers 6,516,132; 5,583,936; 6,836,549; 7,050,698; and 7,492,896 and other intellectual property rights. The use of ROVI Corporation's copy protection technology in the device must be authorized by ROVI Corporation and is intended for home and other limited pay-per-view uses only, unless otherwise authorized in writing by ROVI Corporation. Reverse engineering or disassembly is prohibited.

## OpenCL

OpenCL is a trademark of Apple Inc. used under license to the Khronos Group Inc.

## Trademarks

NVIDIA, the NVIDIA logo, GeForce, Tesla, and Quadro are trademarks or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

## Copyright

© 2010 NVIDIA Corporation. All rights reserved.