Using Intel FPGA SDK for OpenCL™ on DE-Series Boards

For Quartus® Prime 18.1

1 Introduction

This tutorial provides a brief introduction to OpenCL™ and the Intel® FPGA SDK for OpenCL, and describes how to compile and execute OpenCL applications that target SoC-based DE-series boards such as the DE10-Standard, DE10-Nano, and DE1-SoC.

Contents:

- Overview of OpenCL
- Overview of Intel FPGA SDK for OpenCL
- Compiling a Sample OpenCL Application
- Executing an OpenCL Application on DE-Series Boards

Requirements:

- Familiarity with using Linux* on DE-series boards, which can be achieved by reading the tutorial *Using Linux on DE-Series Boards*

2 What is OpenCL™?

OpenCL is a C-based programming language designed for writing applications that run on heterogeneous compute platforms. These platforms generally consist of different compute devices such as CPUs, GPUs, and FPGAs. Traditionally, each of these devices has had its own methods for programming. As an example, a developer might have had to write C code for the CPU, OpenGL code for the GPU, and Verilog code for the FPGA. In contrast, a developer can use OpenCL to create a single application that executes across all of the devices. This makes it easier to extract performance out of heterogeneous systems, using each device's unique strengths to speed up corresponding portions of the application.

OpenCL has a number of key features that make it suited for programming heterogeneous systems. First, OpenCL provides an abstraction layer for programming the devices in the system. This means that device-specific details are hidden to the developer which makes writing code easier, especially if the developer does not have a deep understanding of the devices. The abstraction also means their application code is not specific to certain devices, architectures, or vendors, leaving them free to migrate their application to newer platforms. Second, OpenCL allows developers to specify parallelism in in fine detail. As a typical heterogeneous system contains parallel accelerators such as GPUs and FPGAs, it is important to be able to write parallelized applications that make effective use of these devices.

An OpenCL application is split into two parts, the **host program** and the **kernel(s)**. The host program is executed on the CPU of the system, and can perform any functions or computations as if it were a regular C program. In addition, an OpenCL host program is able to launch one or more kernels in order to speed up computation. A **kernel** is a special function written in OpenCL C that performs some user-defined computation. The kernel is executed on an accelerator device such as an FPGA. Often, the kernel is designed to perform some computation that can be executed in parallel - in order to effectively leverage the parallel nature of an accelerator like the FPGA. As an example, consider matrix addition of two m x n matrices. Here, we could design the kernel to do a single addition (between two corresponding matrix cells), allowing for any number of kernels between 1 to m x n to execute in parallel. While a CPU would take m x n cycles to do this matrix addition, an FPGA containing many of these kernels could compute the operation in parallel, thereby speeding up the application.

This tutorial is not intended to be a comprehensive guide to the OpenCL language nor the Intel FPGA SDK for OpenCL. Instead, this tutorial provides the minimal information necessary to start using OpenCL on the DE-Series boards. For more information about writing OpenCL and using the Intel FPGA SDK for OpenCL, please refer to the documents listed below:

- Intel FPGA SDK for OpenCL Getting Started Guide
- Intel FPGA SDK for OpenCL Programming Guide
- Intel FPGA SDK for OpenCL Best Practices Guide

Note: This tutorial will use the DE10-Standard board as a reference, but the procedure is almost identical for other DE-series boards.

3 Introduction to the Intel® FPGA SDK for OpenCL™

The Intel FPGA SDK for OpenCL can be used to compile OpenCL applications that target heterogeneous systems containing Intel FPGA(s). Such a system contains a CPU, such as an x86 or ARM* processor, and one or more Intel FPGAs. For x86-based systems, the FPGA typically resides on an FPGA accelerator board, which is connected to the system through the PCIe interface. For SoC-FPGA systems, the FPGA is generally connected to the processor through specialized bridges, as is the case with Intel SoC-FPGA devices found on DE-series Boards. The host program of the OpenCL application executes on the CPU, and the kernels are placed into the FPGA and launched on-demand by the host program.

The Intel FPGA SDK for OpenCL provides the tools required to allow the implementation of OpenCL applications that target Intel FPGAs. Three main components comprise the SDK:

- The *Intel Offline Compiler* (AOC) which translates the OpenCL kernel code into hardware that can be programmed onto the FPGA.
- The *host runtime* which is a collection of libraries and drivers that allow the host program to communicate with the kernel(s) in the FPGA.
- The *AOCL* utility which provides a set of commands to perform tasks such as downloading the kernel into the FPGA and running diagnostics to check that OpenCL drivers have been initialized.

The developer uses the Intel FPGA SDK for OpenCL in conjunction with a standard C++ compiler to compile the OpenCL application. Figure 1 shows the compilation and execution flow at a high level. The flow is described in more detail below:

1. The developer writes their OpenCL application code, which consists of a host program written in C/C++, and kernel(s) written in OpenCL C.
2. The developer compiles the OpenCL application:
   (a) The host program is compiled via GCC or Visual Studio, linking in the Intel OpenCL host runtime libraries. This creates the host program binary to be executed by the CPU.
   (b) The kernel(s) are compiled via the AOC. This creates the Intel OpenCL Executable (.aocx) file, which can be downloaded onto the FPGA.
3. The developer uses the AOCL utility to load the .aocx programming file onto the FPGA. The FPGA now contains the kernel(s) that will be launched by the host program.
4. The OpenCL application is executed by running the host program. Throughout its execution, the host program launches the FPGA kernels as needed.

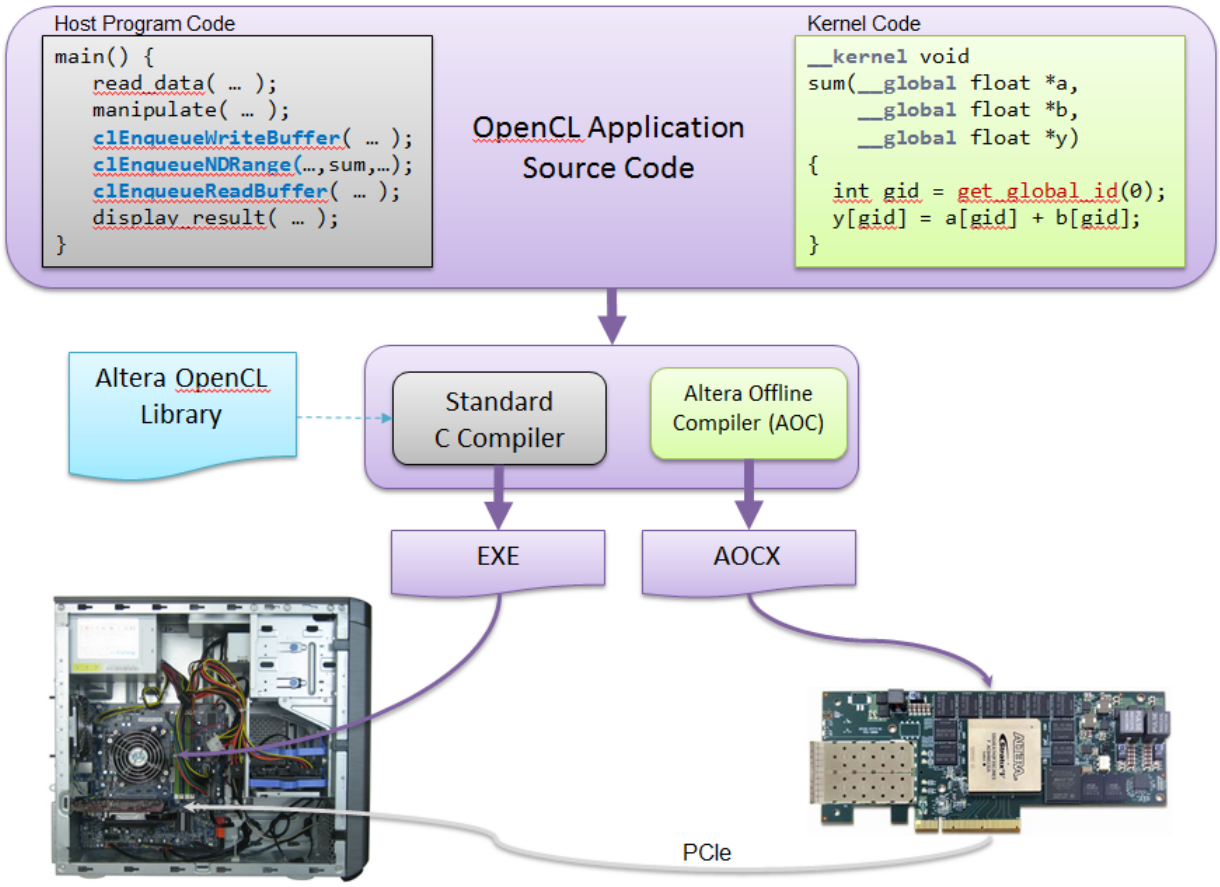

Figure 1. The Intel FPGA SDK for OpenCL Flow

3.1 Installing the Intel® FPGA SDK for OpenCL™

To install the Intel FPGA SDK for OpenCL on your host computer, follow the instructions below. The Intel FPGA SDK for OpenCL also installs the required Intel Quartus® Prime software. You do not have to install Quartus Prime software separately.

1. Go to http://dl.altera.com/opencl/.
2. Select either the PRO or STANDARD edition of the SDK depending on the device family that you wish to target. To determine which version is appropriate for your device you can consult the table at https://dl.altera.com/devices/.

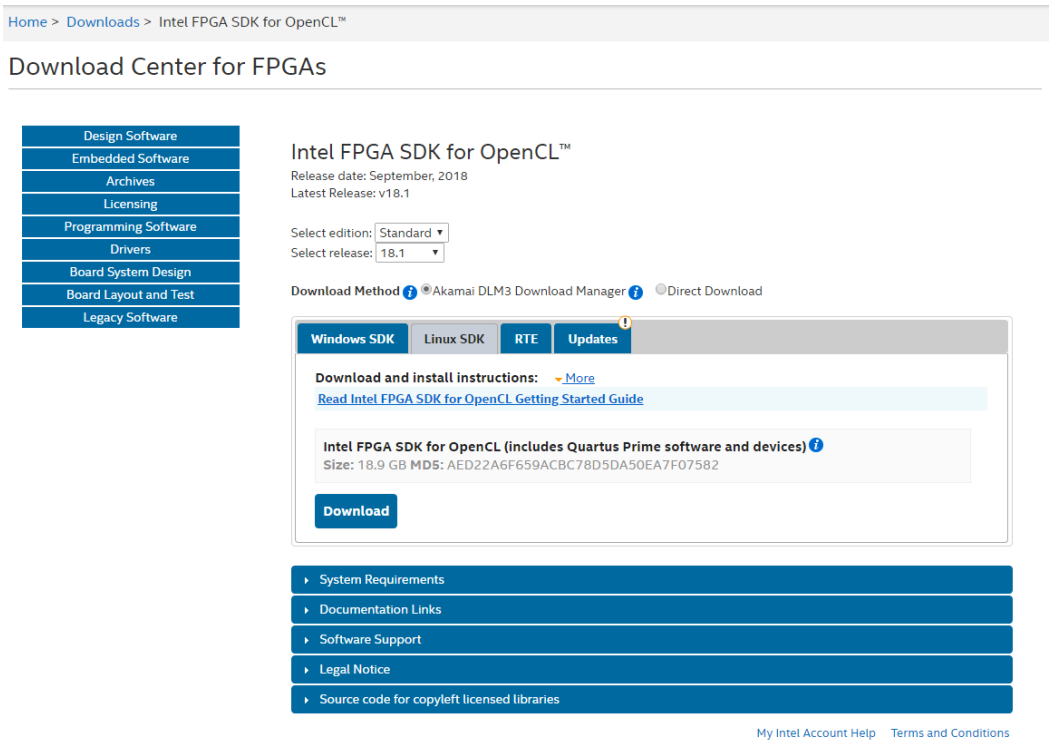

Figure 2. Intel FPGA SDK for OpenCL Download

3. Download the *Windows* SDK* or the *Linux SDK* depending on your operating system. This will download a large TAR file and may take a long time to complete.

4. Extract the contents of the TAR file.

5. Run the installer:

   On Windows, double click the *setup.bat* file.

   On Linux, open a terminal, `cd` to the extracted files, then run the command `sudo ./setup.sh`.

   The installer GUI will appear, as shown in Figure 3.

6. Follow the instructions in the installer GUI to install the SDK. At the step shown in Figure 4, take note of the directory where the SDK is being installed. The installer comes with support for a variety of FPGA device families. To save disk space, you can choose to install only the device(s) that you you need as shown in Figure 5.

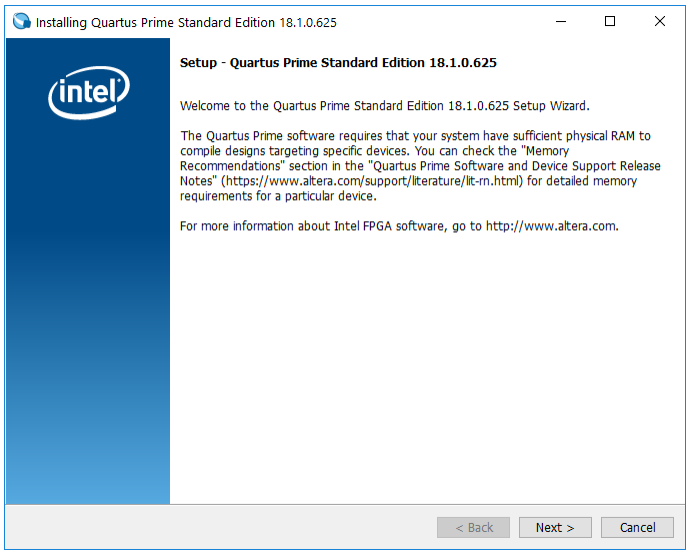

Figure 3. Quartus Installation

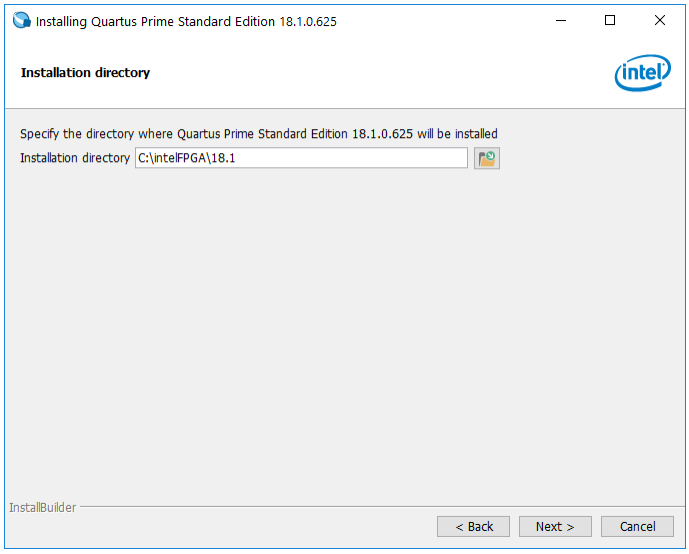

Figure 4. Quartus Installation - Selecting the Quartus Root Directory

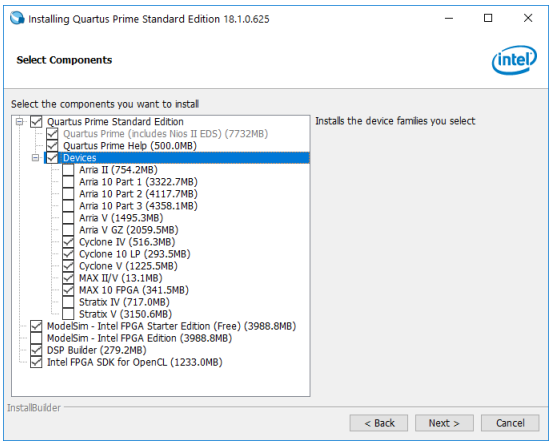

Figure 5. Quartus Installation - Selecting FPGA Devices

7. Once the installation completes, the Intel FPGA SDK for OpenCL will reside in *<quartus root>/hld/* where *<quartus root>* is the directory you chose in Figure 4.

8. Before you can call SDK commands, you must set your operating system's environment variables to point to the new installation. You can do this as follows:

   On Windows, open a CMD prompt and run the command `<quartusroot>\hld\init_opencl.bat`.

   On Linux, open a terminal and run the command `source <quartusroot>/hld/init_opencl.sh`.

   Note that the `init_opencl` script does not permanently set the environment variables, and must be run each time you open a new CMD prompt or terminal.

9. Verify the installation and environment variables by checking the output of the command `aocl version`. The command should produce a version number output similar to Figure 6.

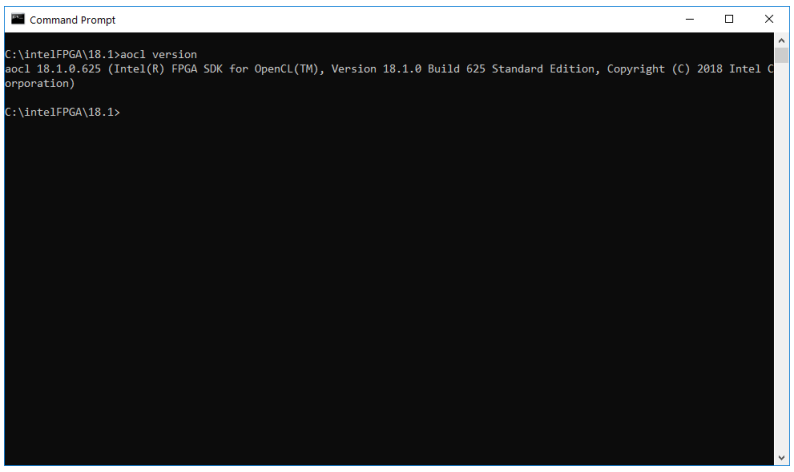

Figure 6. Verifying Intel OpenCL SDK install

For more instructions on installing the Intel FPGA SDK for OpenCL, please refer to the document Intel FPGA SDK for OpenCL Getting Started Guide.

3.2 Installing the DE-Series Board Support Package

To compile OpenCL kernels for your DE-series board, you must install and use the corresponding board support package (BSP). To install the BSP, follow these instructions:

1. Go to DE-series board section on Terasic's website (`https://www.terasic.com.tw/en/`).
2. Go to the webpage for your board.
3. Go to the *Resources* tab and scroll down to find the **BSP(Board Support Package) for Intel FPGA SDK OpenCL** for your board, as shown in Figure 7.

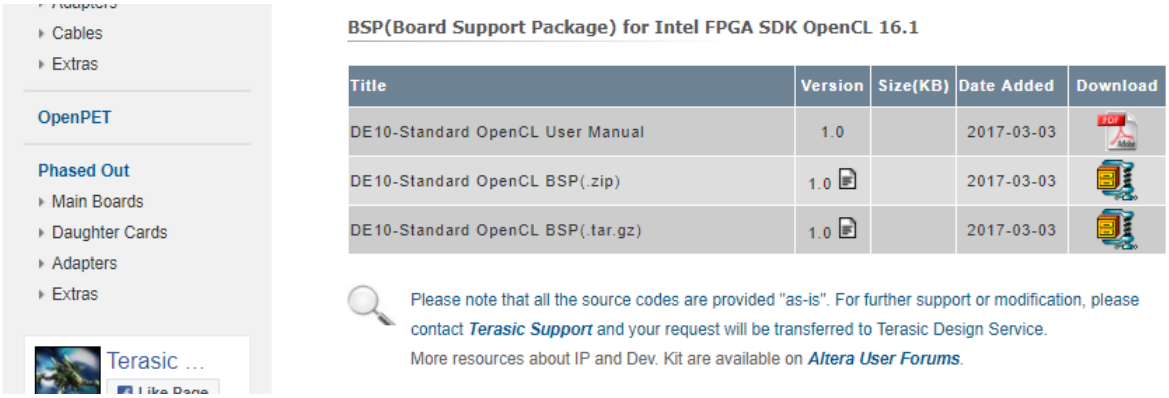

Figure 7. Terasic's Board Support Package version and download

4. Download the OpenCL BSP and extract its contents to *<quartusroot>/hld/board/*. Figure 8 shows the result of extracting the BSP contents for the DE10-Standard board to *<quartusroot>/hld/board/*.

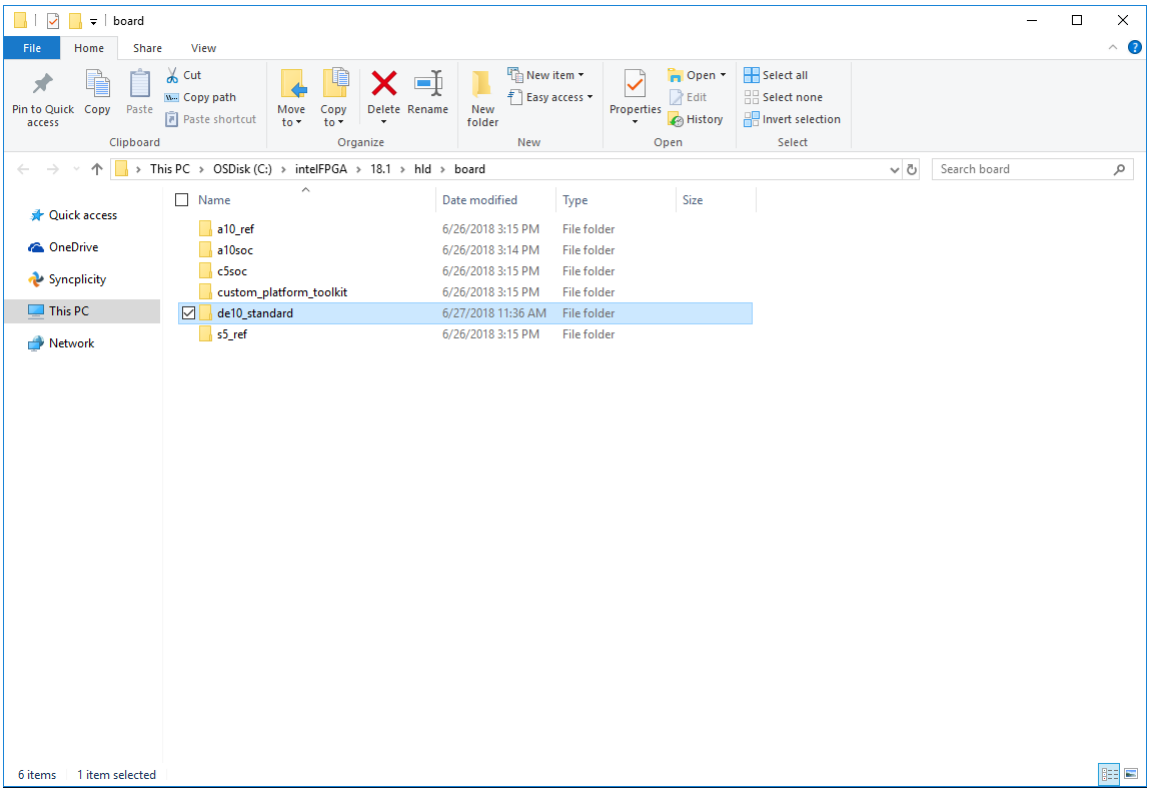

Figure 8. Extracting the Board Support Package to the *<quartusroot>/hld/board/* Directory

5. Create a new environment variable called `AOCL_BOARD_PACKAGE_ROOT` and set its value to *<quartusroot>/hld/board/<extracted-bsp-directory>* (eg. *C:/intelFPGA/18.1/hld/board/de10_standard*). In Linux you can set an environment variable by using the command `export <environment variable name>=<environment variable value>` in a terminal. In Windows you can set an environment variable by following the instructions in Section 6.

6. Ensure that the `AOCL_BOARD_PACKAGE_ROOT` points to a directory that contains a file named *board_env.xml*, as shown in Figure 9.

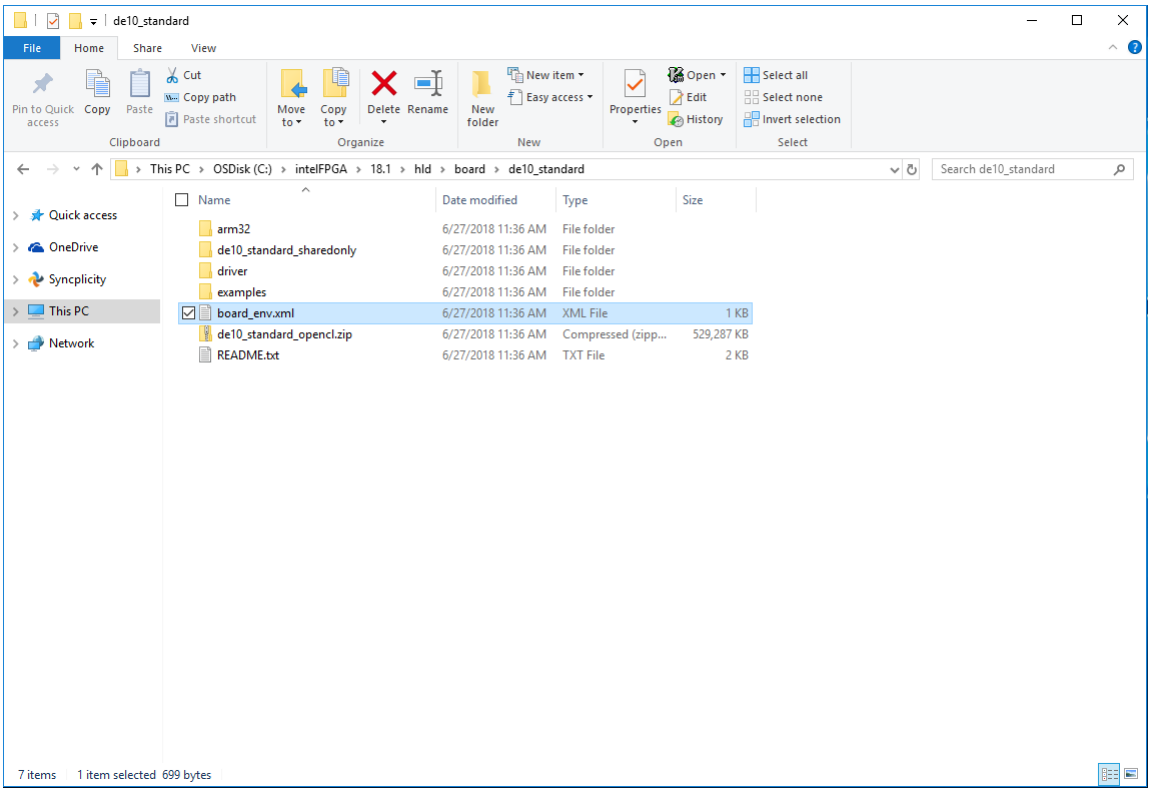

Figure 9. Downloaded Board Support Package directory

7. To test the board support package installation run the command `aoc -list-boards` in a Windows CMD prompt or a Linux terminal. The output should list the BSP that you just added, as shown in Figure 10 for the DE10-Standard.

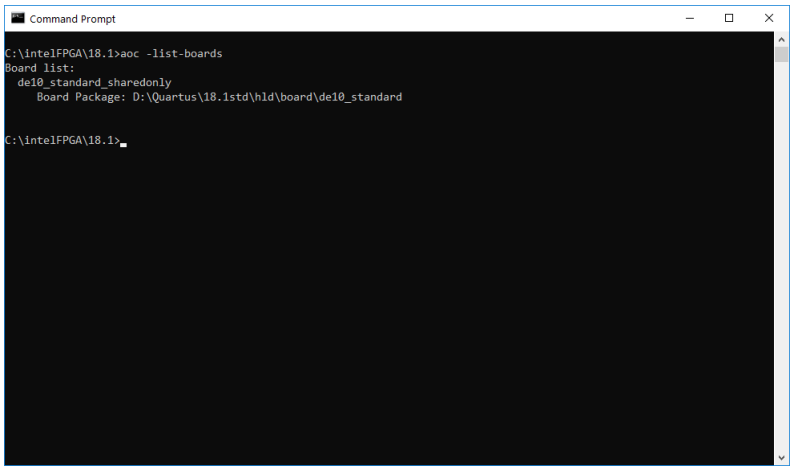

Figure 10. Verifying Board Support Package install

For more instructions on installing board support, please refer to the *Installing an FPGA Board* section of the document Intel SDK for OpenCL Getting Started Guide.

3.3 Setting up the Quartus and OpenCL SDK License File

NOTE: As of version 17.1, you no longer require a license to use the Intel FPGA SDK for OpenCL. For prior versions, please refer to the instructions below. The `INTELFPGAOCLSDKROOT` variable in the instructions below refers to the <*quartusroot*>*/hld/* directory.

If you have a **fixed** license follow these steps:

1. If you have a fixed license, copy the license (.dat) file to your local disk. A suggested location would be *INTELFPGAOCLSDKROOT/licenses/* where you may have to create the *licenses* directory in the *INTELFPGAOCLSDKROOT/* directory.
2. Add or append to the `LM_LICENSE_FILE` environment variable, the path to your license file: <*path_to_license_file*>/<*license_filename*> (eg. *INTELFPGAOCLSDKROOT/licenses/license.dat*).

If you have a **floating** license follow these steps:

1. Obtain the the port number and host name from the network or system admin. Alternatively, the information is in the license file line SERVER <hostname> <8 to 12 character host or NIC ID> <port>.
   The license for the user is <port>@<hostname>. If the port is not listed in the license file, specify @<hostname>.
2. Modify the license file to update the port number and host name.
3. Add or append to the *LM_LICENSE_FILE* environment variable, the path to your license file: `<path_to_license_file>\<license_filename>` (eg. `INTELFPGAOCLSDKROOT\licenses\license.dat`).

For more instructions on licensing the software, please refer to the *Licensing the Software* section of the document Intel FPGA SDK for OpenCL Getting Started Guide.

4 Compiling an Example OpenCL™ Application (Vector Addition)

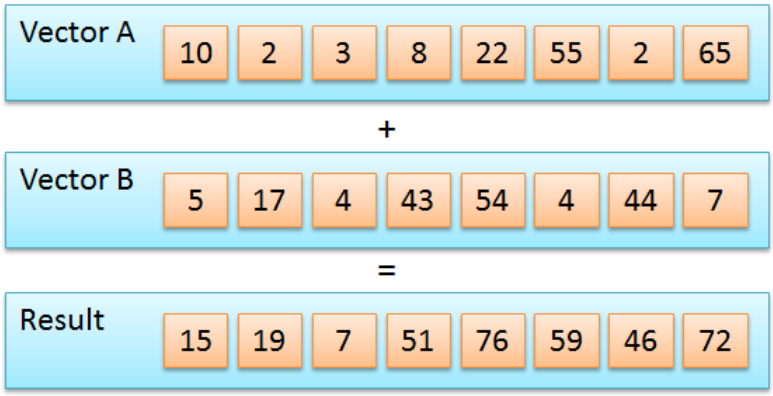

Figure 11. The Vector Addition OpenCL Application

In this section, we will compile the *OpenCL Vector Addition Design Example* application for a DE-series SoC board. The source code for this design example can be downloaded at https://www.altera.com/support/support-resources/design-examples/design-software/opencl/vector-addition.html. This application performs vector addition of two vectors as shown in Figure 11. More precisely, the application does the following:

1. The host program populates two arrays of equal length with randomized numbers.
2. The host program computes the sums in software, using the CPU.
3. The kernel is used to compute the same additions again, using the FPGA.
4. The host program compares the results from the kernel to those that resulted from using the CPU. If the results are identical, the host program outputs the text "Verification: PASS" to indicate success.

4.1 Compiling the Kernel

We will first compile the kernel whose source code is located at */vector_add/device/vector_add.cl*. Compiling the kernel requires that the Intel FPGA SDK for OpenCL has been properly installed and configured, following the instructions in Sections 3.1 and 3.2. You can compile the kernel by following the steps below:

1. Extract the contents of the *exm_opencl_vector_add_arm32_<version>.tgz* file to a directory of your choice.
2. In the commandline, navigate to the */vector_add/* directory.
3. Compile the kernel by running the command (this can take some time): **aoc device/vector_add.cl -o bin/vector_add.aocx -board=<target_board>** Where *target_board* is the board you installed in Section 3.2. The available boards can be list by running the following command: **aoc -list-boards**
4. The resulting Intel OpenCL Executable file *vector_add.aocx* is placed in the */vector_add/bin/* directory.

4.2 Compiling the Host Program

We will now compile the host program. Since the program will run on the ARM processor of the DE-series board, we must compile the host program code into an ARM binary. To do this, we will use the GNU toolchain included in the Linux distribution released for the DE-series board. This compilation process is referred to *native compilation* since we are compiling the code on the same system that will run the binary. This is in contrast to *cross-compiliation* which would be the process of compiling the binary on a non-ARM system (such as your x86 desktop or laptop) and then transferring the binary to the board to run it. To natively compile the host program, follow the instructions below:

1. Boot Linux on your board following the instructions in the tutorial *Using Linux on the DE-Series Boards*.
2. Establish a commandline interface to the board either through USB-UART or SSH.
3. In the commandline, run the command `source /home/root/OpenCL/init_opencl.sh` to configure the necessary environment variables.
4. Transfer the contents of the *exm_opencl_vector_add_arm32_<version>.tgz* file to a location of your choice on the board. For instructions on transferring files to the board's Linux file system, refer to Section *Transferring Files to/from the Host Computer* in the tutorial *Using Linux on the DE-Series Boards*.
5. In the commandline, navigate to the */vector_add/* directory.
6. Compile the host program by running **make**.
7. The resulting host program binary *vector_add* is placed in the */vector_add/bin/* directory.

5 Running OpenCL™ Applications on DE-Series boards

OpenCL applications targeting DE-series SoC boards must run on top of the Linux operating system as they rely on Linux drivers that facilitate communication between the host program and the OpenCL kernel(s) inside the FPGA. This means that before you can run an OpenCL application, you must boot Linux on the board. The tutorial *Using Linux on the DE-Series Boards* describes the process of booting Linux on DE-Series boards. Note that you must use a Linux distribution that contains support for OpenCL, such as the one used in the tutorial. Once you have booted up Linux, you can proceed to the following section.

5.1 Running the Vector Addition OpenCL™ Application

This section describes the steps for executing the *Vector Addition* sample OpenCL program included with the DE-series Linux distribution. The commands shown are to be run in a commandline interface to the board, through a USB-UART or SSH connection. For instructions on establishing a commandline interface to the board, refer to the tutorial *Using Linux on the DE-Series Boards*.

To execute an OpenCL application, we must first run a script to setup the environment and load necessary OpenCL drivers. To do this, you can use the following command:

```
source /home/root/OpenCL/init_opencl.sh
```

Once we have executed the script, we can run the vector addition OpenCL application. For your convenience, the application comes preloaded in the Linux image. You can find the files *vector_add* and *vector_add.aocx* in */home/root/OpenCL/OpenCL_Examples/vector_add/*. Alternatively, if you compiled your own host program binary and aocx in Section 4, you can use them instead. To run the application, follow the instructions below:

1. Navigate to the directory:

   ```
   cd /home/root/OpenCL/OpenCL_Examples/vector_add
   ```

2. Program the FPGA with the kernel:

   ```
   aocl program /dev/acl0 vector_add.aocx
   ```

3. Execute the host program:

   ```
   ./vector_add
   ```

After running the commands listed above, you should see output similar to what is shown in Figure 12. As shown in the figure, the host program executes, launching the kernel in the process. The host program displays the time taken in total as well as how much of the time was taken by the kernel. Finally, the host program compares the sums computed by itself to the sums computed by the kernel in the FPGA. If the sums match, the program outputs "Verification: PASS".

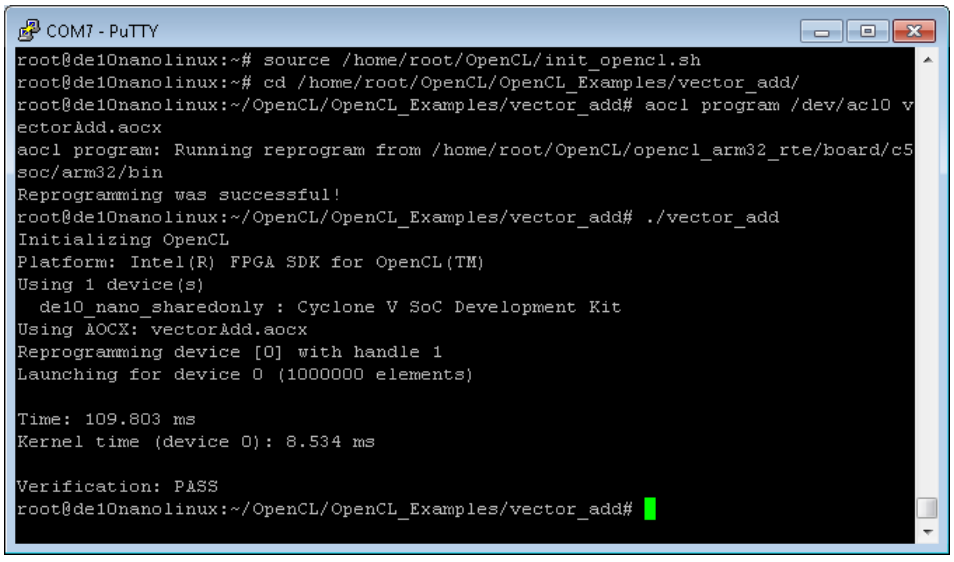

Figure 12. Executing the vector addition application on the DE10-Nano board.

6 Appendix A

6.1 Adding and Editing Environment Variables in Windows*

To add or edit environment variables first open the system control panel and navigate to System > Advanced System Settings to open the dialog window in Figure 13.

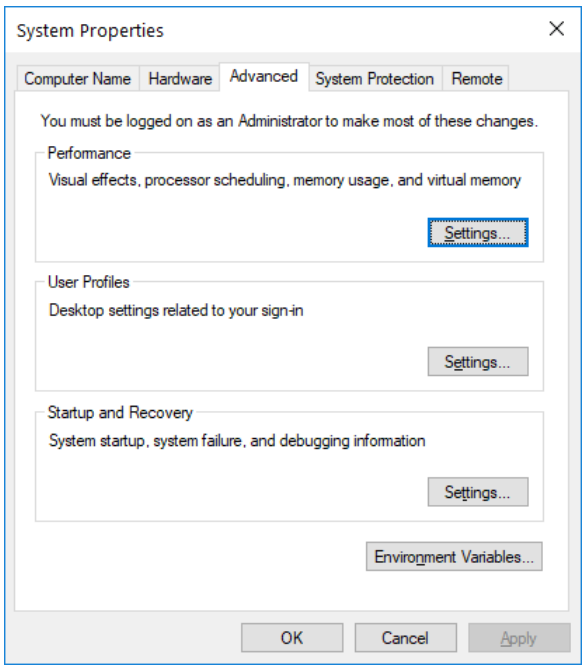

Figure 13. Advanced System Settings

From this dialog click *Environment Variables...* to open the dialog in Figure 14.

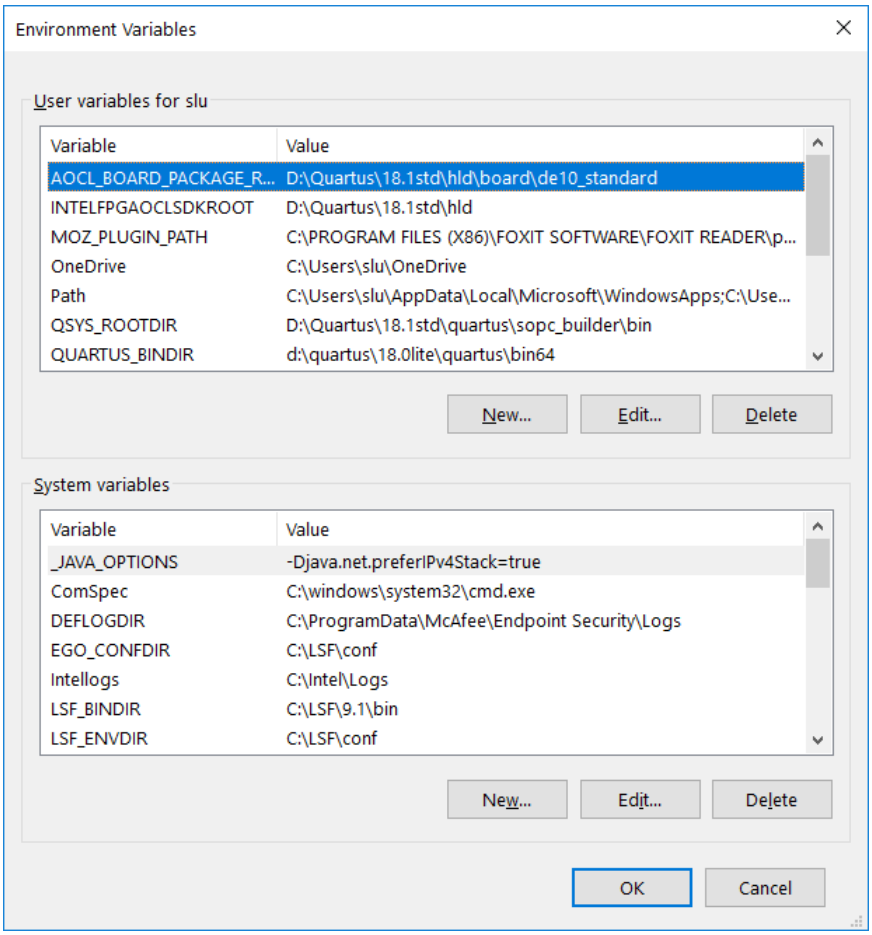

Figure 14. System Environment Variables

To edit the *PATH* environment variable, select the *Path* variable and click *Edit* to open the dialog in Figure 15. From here you can add and remove paths.

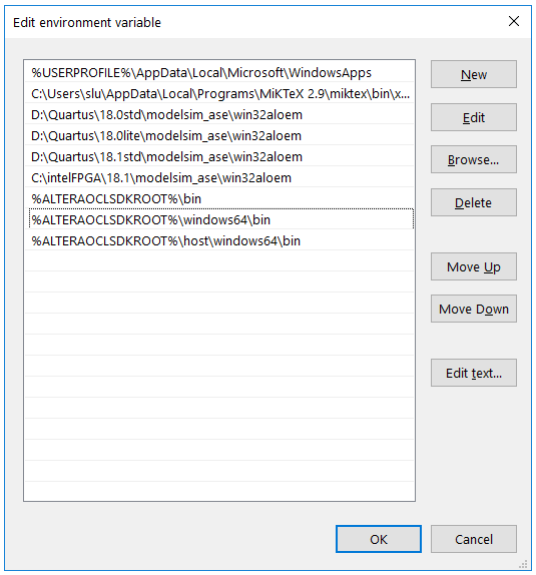

Figure 15. 'Path' System Environment Variable

To add environment variables, click *New...* in Figure 14 to open the dialog in Figure 16. From here you can define a new environment variable.

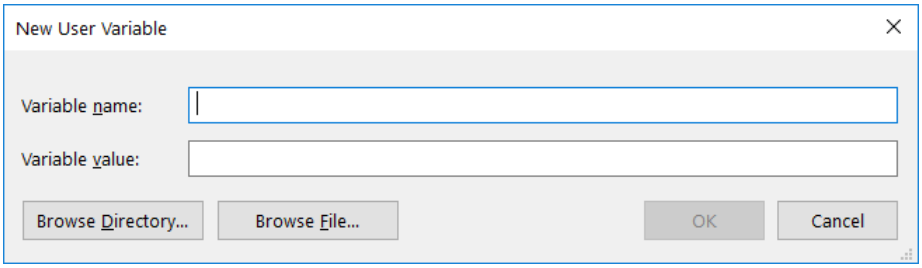

Figure 16. New System Environment Variable

OpenCL and the OpenCL logo are trademarks of Apple Inc. used by permission by Khronos.