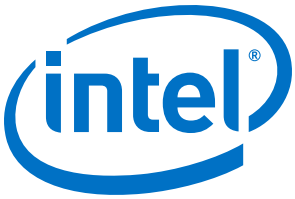


Intel® FPGA SDK for OpenCL™ Pro Edition

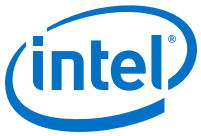
Getting Started Guide

Updated for Intel® Quartus® Prime Design Suite: **18.1**

Subscribe
Send Feedback

UG-OCL001 | 2019.01.14
Latest document on the web: PDF | HTML

Contents



UG-OCL001 | 2019.01.14

Send Feedback

1. Intel® FPGA SDK for OpenCL™ Pro Edition Getting Started Guide

The *Intel® FPGA SDK for OpenCL™ Pro Edition Getting Started Guide* describes the procedures to install the Intel FPGA Software Development Kit (SDK) for OpenCL [(1)]Pro Edition. This document also contains instructions on how to compile an example OpenCL[(2)] application with the Intel FPGA SDK for OpenCL Pro Edition.

OpenCL is a C-based open standard for the parallel programming of heterogeneous devices. For more information about the OpenCL Specification version 1.0, refer to the OpenCL 1.0 Reference Pages. For detailed information on the OpenCL application programming interface (API) and programming language, refer to the OpenCL Specification version 1.0.

The Intel FPGA SDK for OpenCL Pro Edition provides a compiler and tools for you to build and run OpenCL applications that target Intel FPGA products. The Intel FPGA SDK for OpenCL Pro Edition supports the embedded profile of the OpenCL Specification version 1.0.

Attention:

- If you only require the Intel FPGA SDK for OpenCL's kernel deployment functionality, download and install the Intel FPGA Runtime Environment (RTE) for OpenCL. Refer to the *Intel FPGA RTE for OpenCL Pro Edition Getting Started Guide* for more information.

  Do not install the SDK and the RTE on the same host system. The SDK already contains the RTE.
- If you want to use the Intel FPGA SDK for OpenCL Pro Edition with the Intel Arria 10 GX FPGA Development Kit, refer to the Application Note *Configuring the Intel Arria 10 GX FPGA Development Kit for the Intel FPGA SDK for OpenCL* for more information.

Related Information

- OpenCL Reference Pages
- OpenCL Specification version 1.0
- Intel FPGA RTE for OpenCL Pro Edition Getting Started Guide
- Configuring the Intel Arria 10 GX FPGA Development Kit for the Intel FPGA SDK for OpenCL

---

(1) The Intel FPGA SDK for OpenCL is based on a published Khronos Specification, and has passed the Khronos Conformance Testing Process. Current conformance status can be found at www.khronos.org/conformance.

(2) OpenCL and the OpenCL logo are trademarks of Apple Inc. used by permission of the Khronos Group™.

1.1. Prerequisites for the Intel FPGA SDK for OpenCL Pro Edition

To install the Intel FPGA SDK for OpenCL Pro Edition and create an OpenCL application for an Intel FPGA Preferred Board for OpenCL, your system must meet certain hardware, target platform, and software requirements.

Hardware Requirements

Accelerator boards requirements:

- Acquire a Reference Platform from Intel, or a Custom Platform from an Intel preferred board vendor.

  For more information, refer to the Intel FPGA SDK for OpenCL FPGA Platforms page on the Intel FPGA website.

Development system requirements:

- You must have administrator, root, or sudo privileges on the development system to install the necessary packages and drivers.
- The development system has at least 85 gigabytes (GB) of free disk space for software installation.
- The development system has at least 24 GB of RAM.

  Tip: Refer to board vendor's documentation on the recommended system storage size.

- For PCI Express* (PCIe*) accelerator boards, the host machine motherboard must have an available PCIe port slot that is at least the same width (that is, the same number of PCIe lanes) as the board.

The host system must be running one of the following supported operating systems:

- For a list of supported Windows and Linux operating systems, refer to the Operating System Support page on the Intel FPGA website.

Software Prerequisites

Develop your host application using one of the following Intel FPGA SDK for OpenCL- and Intel Quartus® Prime software-compatible C compiler or software development environment:

- For Windows systems, use Microsoft Visual Studio Professional version 2015 or later.
- For Linux systems, use the C compiler included with the GCC.

Linux systems require the Perl command version 5 or later. Include the path to the Perl command in your *PATH* system environment variable setting.

For Intel FPGA SDK for OpenCL packages that include Intel Code Builder, Intel Code Builder requires Java SE version 1.8.71 or later to run.

Related Information

Intel FPGA SDK for OpenCL FPGA Platforms

1.1.1. Intel FPGA SDK for OpenCL Pro Edition and BSP Backwards Compatibility

The Intel FPGA SDK for OpenCL Pro Edition Version 18.1 compiler is compatible with board support packages (BSPs) from Version 18.1, Version 18.0, and Version 17.1.1. This backwards compatibility lets you to use older BSPs with newer compilers. However, some newer OpenCL compiler features might not be available to use with older BSPs.

To use an older BSP with the Intel FPGA SDK for OpenCL, you must have a version of Intel Quartus Prime Pro Edition with the same version number as your BSP. For example, to use a Version 18.0 BSP with Intel FPGA SDK for OpenCL Pro Edition Version 18.1, you need Intel Quartus Prime Pro Edition Version 18.0.

Related Information

- Enabling Backwards Compatibility for an FPGA Board on Windows on page 15
- Enabling Backwards Compatibility for an FPGA Board on Linux on page 30

1.2. Contents of the Intel FPGA SDK for OpenCL Pro Edition

The Intel FPGA SDK for OpenCL Pro Edition provides programs, drivers, and SDK-specific libraries and files.

Logic Components

- The *Intel FPGA SDK for OpenCL Offline Compiler* translates your OpenCL device code into a hardware configuration file that the system loads onto an Intel FPGA product.
- The *Intel FPGA SDK for OpenCL Pro Edition* utility includes a set of commands you can invoke to perform high-level tasks such as running diagnostic tests.
- The *host runtime* provides the OpenCL host and runtime API for your OpenCL host application.

  The host runtime consists of libraries that provide OpenCL APIs, hardware abstractions, and helper libraries.

Drivers, Libraries and Files

The software installation process installs the Intel FPGA SDK for OpenCL Pro Edition into a directory that you own. The *INTELFPGAOCLSDKROOT* environment variable references the path to the SDK's installation directory.

Table 1. Select Contents of the Intel FPGA SDK for OpenCL Pro Edition Installation Directory

Windows Folder	Linux Directory	Description
`bin`	`bin`	User commands in the SDK. Include this directory in your *PATH* environment variable setting.
`board`	`board`	The Intel FPGA SDK for OpenCL Pro Edition Custom Platform Toolkit and Reference Platforms available with the software.

continued...

Windows Folder	Linux Directory	Description
		• The path to the Custom Platform Toolkit is *INTELFPGAOCLSDKROOT*/board/custom_platform_toolkit • The path to the a10_ref Reference Platform is *INTELFPGAOCLSDKROOT*/board/a10_ref
ip	ip	Intellectual property (IP) cores used to compile device kernels.
host	host	Files necessary for compiling and running your host application.
host\include	host/include	OpenCL Specification version 1.0 header files and software interface files necessary for compiling and linking your host application. The host/include/CL subdirectory also includes the C++ header file cl.hpp. The file contains an OpenCL version 1.1 C++ wrapper API. These C++ bindings enable a C++ host program to access the OpenCL runtime APIs using native C++ classes and methods. *Important:* The OpenCL version 1.1 C++ bindings are compatible with OpenCL Specification versions 1.0 and 1.1. Add this path to the include file search path in your development environment.
host\windows64\lib	host/linux64/lib	OpenCL host runtime libraries that provide the OpenCL platform and runtime APIs. These libraries are necessary for linking your host application. To run an OpenCL application on Linux, include this directory in the *LD_LIBRARY_PATH* environment variable setting.
host\windows64\bin	host/linux64/bin	Runtime commands and libraries necessary for running your host application, wherever applicable. For 64-bit Windows system, include this directory in your *PATH* environment variable setting. For Windows system, this folder contains runtime libraries. For Linux system, this directory contains platform-specific binary for the aocl utility command.
share	share	Architecture-independent support files.

Example OpenCL Applications

You can download example OpenCL applications from the OpenCL Design Examples page.

Related Information

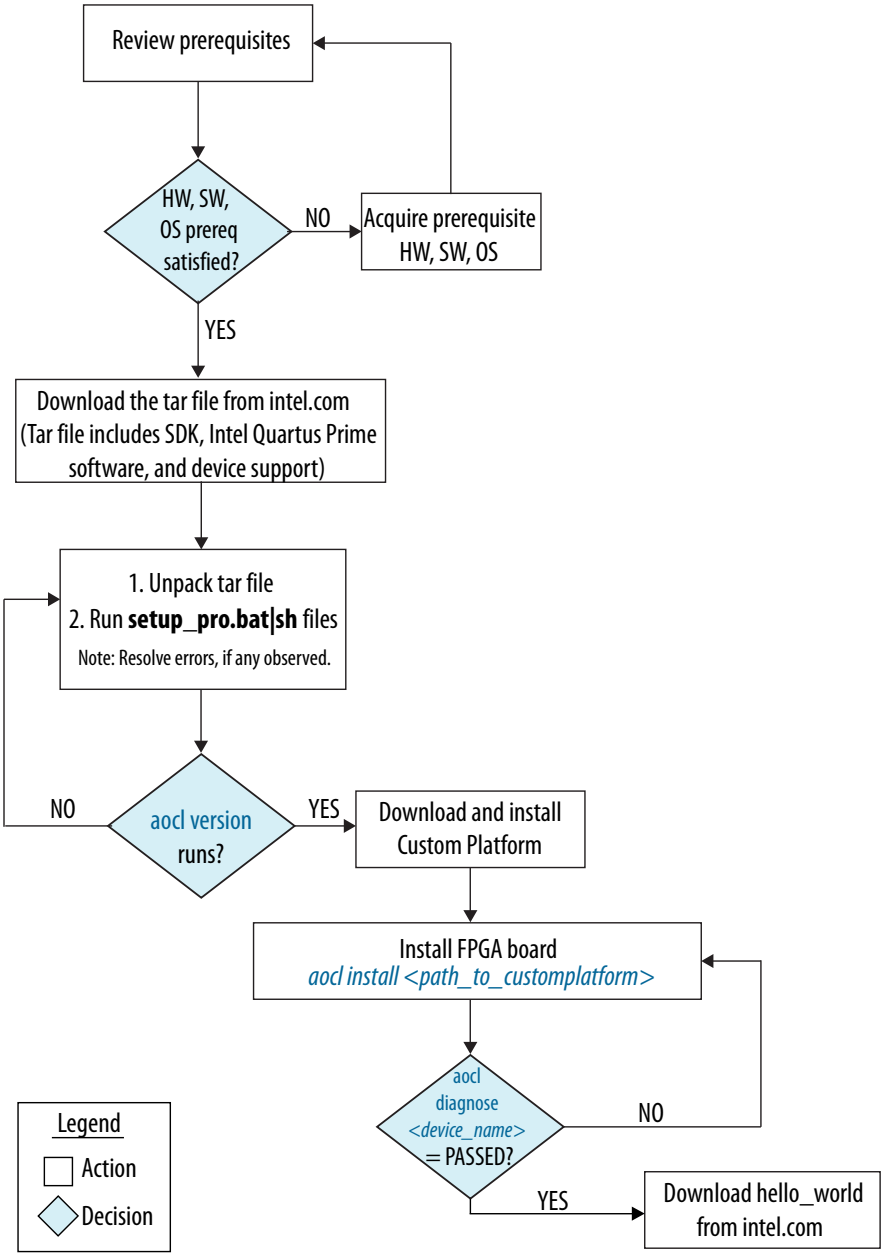
OpenCL Design Examples

1.3. Overview of the Intel FPGA SDK for OpenCL Pro Edition Setup Process

The *Intel FPGA SDK for OpenCL Pro Edition Getting Started Guide* outlines the procedures for installing the Intel FPGA SDK for OpenCL Pro Edition and programming your FPGA.

The following figure summarizes the steps for setting up the necessary software and installing the FPGA board.

Figure 1. Intel FPGA SDK for OpenCL Pro Edition Installation Process Overview



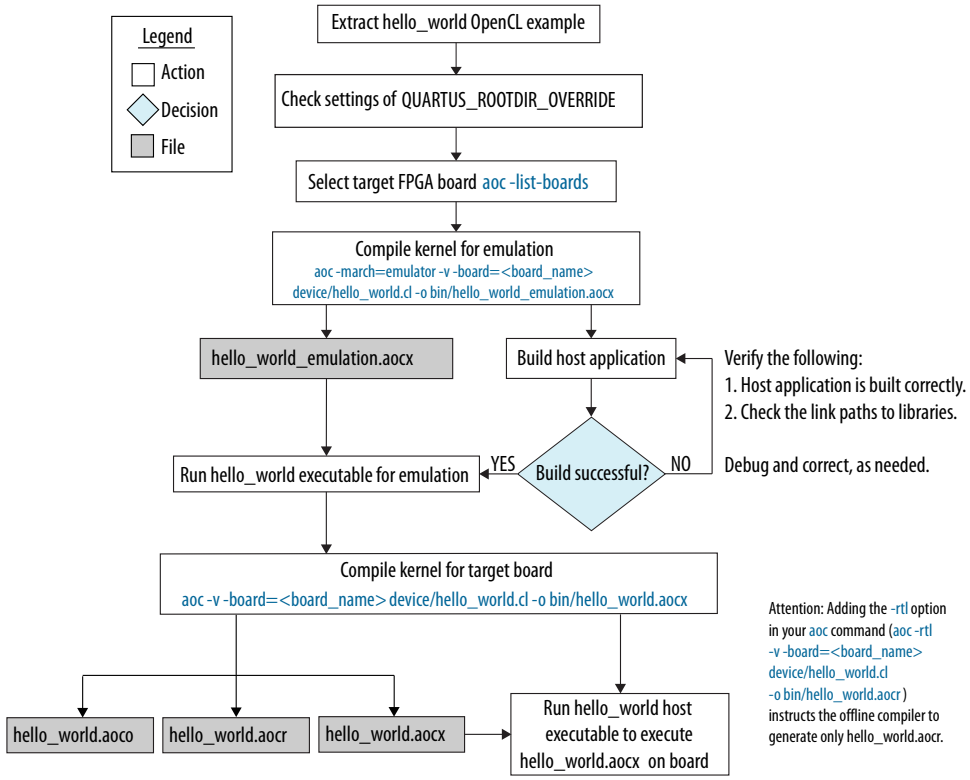
Attention: For possible errors after implementing `aocl diagnose` utility, refer to Possible Errors After Running the `diagnose` Utility section.

After you complete the initial software and hardware setup successfully, you can create a hardware image from the *hello_world* OpenCL design example.

Attention: Before you program your FPGA with the hardware image, ensure that your FPGA contains an image created using a current version of the Intel FPGA SDK for OpenCL. Refer to *Updating the Hardware Image on the FPGA* for more information

The following figure summarizes the steps you perform to program your FPGA.

Figure 2. FPGA Programming Overview



Related Information

Updating the Hardware Image on the FPGA on page 21

2. Getting Started with the Intel FPGA SDK for OpenCL Pro Edition for Windows

The Intel FPGA SDK for OpenCL Pro Edition setup process includes downloading and installing the software, installing the FPGA board, and then executing an OpenCL kernel on the FPGA.

2.1. Downloading the Intel FPGA SDK for OpenCL Pro Edition

Download the installation package that contains the Intel FPGA SDK for OpenCL Pro Edition and all related software for Windows from the Intel FPGA SDK for OpenCL Download Center.

The Intel FPGA SDK for OpenCL Download Center provides a tar file that includes all of the following software and files:

- Intel FPGA SDK for OpenCL Pro Edition
- Intel Quartus Prime Pro Edition software
- Device support

1. Go to the Intel FPGA SDK for OpenCL Download Center at the following URL:
   https://fpgasoftware.intel.com/opencl/
2. Select the Pro edition.
3. Select the software version. The default selection is the current version.
4. Select one of the following download methods:
   — **Akamai DLM3 Download Manager**

— **Direct Download**

5. On the **Windows SDK** tab, select one of the following packages:
   - **Intel FPGA SDK for OpenCL (includes Quartus Prime Pro Edition software and devices)**

     This package provides core functions of the Intel FPGA SDK for OpenCL including the Intel FPGA SDK for OpenCL offline compiler and the Intel FPGA SDK for OpenCL utility.
   - **Intel FPGA SDK for OpenCL (includes Quartus Prime Pro Edition software and devices with Intel Code Builder)**

     This package provides all of the functions of the Intel FPGA SDK for OpenCL as well as integration with Intel Code Builder to provide an IDE for developing your OpenCL kernels.

   Click **More** beside **Download and install instructions** if you want to see the download and installation instructions.
6. Perform the steps outlined in the download and installation instructions on the download page.

Related Information

Intel FPGA website

2.2. Installing the Intel FPGA SDK for OpenCL Pro Edition

In this section, you learn how to unpack the downloaded tar archive and run the installation files to install all the software and files.
Install the Windows version of the Intel FPGA SDK for OpenCL Pro Edition in a folder that you own.

You must have administrator privileges to execute these instructions.

To install the SDK, Intel Quartus Prime software, and device support files, perform the following tasks:

1. Unpack the downloaded `AOCL-<version>-<build>-windows.tar` archive into a temporary folder.
2. Run the `setup_pro.bat` file to install the SDK with the Intel Quartus Prime Pro Edition software.
3. *Note:* The installer sets the user environment variable *INTELFPGAOCLSDKROOT* to point to the path of the software installation.

   Verify that *INTELFPGAOCLSDKROOT* points to the current version of the software. Open a Windows command window and then type `echo %INTELFPGAOCLSDKROOT%` at the command prompt.

   If the returned path does not point to the location of the SDK installation, edit the *INTELFPGAOCLSDKROOT* setting.

   For instructions on modifying environment variable settings, refer to *Setting the Intel FPGA SDK for OpenCL Pro Edition User Environment Variables*.

Related Information

Setting the Intel FPGA SDK for OpenCL Pro Edition User Environment Variables on page 12

2.3. Setting the Intel FPGA SDK for OpenCL Pro Edition User Environment Variables

You have the option to set the Intel FPGA SDK for OpenCL Pro Edition Windows user environment variables permanently or transiently. The environment variable settings describe the FPGA board and the host runtime to the software.

Attention: If you set the environment variables permanently, you apply the settings once during installation. If you set the environment variables transiently, you must apply the settings during installation and during every subsequent session you run.

Table 2. Intel FPGA SDK for OpenCL Windows User Environment Variable Settings

Environment Variable	Path to Include
PATH	1. `%INTELFPGAOCLSDKROOT%\bin` 2. `%INTELFPGAOCLSDKROOT%\windows64\bin` 3. `%INTELFPGAOCLSDKROOT%\host\windows64\bin` where *INTELFPGAOCLSDKROOT* points to the path of the software installation

- To apply permanent environment variable settings, perform the following tasks:
  a. Click **Windows Start menu ➤ Control Panel** (or search for and then open the Control Panel application in Windows 8.1 and Windows 10).
  b. Click **System and Security ➤ System**.
  c. In the **System** window, click **Advanced system settings**.
  d. Click the **Advanced** tab in the **System Properties** dialog box.
  e. Click **Environment Variables**.
  The **Environment Variables** dialog box appears.
  f. To modify an existing environment variable setting, select the variable under **User variables for** <*user_name*> and then click **Edit**. In the **Edit User Variable** dialog box, type the environment variable setting in the **Variable value** field.
  g. If you add a new environment variable, click **New** under **User variables for** <*user_name*>. In the **New User Variable** dialog box, type the environment variable name and setting in the **Variable name** and **Variable value** fields, respectively.

  For an environment variable with multiple settings, add semicolons to separate the settings.
- To apply transient environment variable settings, open a command window and run the `%INTELFPGAOCLSDKROOT%\init_opencl.bat` script.

  Example script output:

```
Adding %INTELFPGAOCLSDKROOT%\bin to PATH
Adding %INTELFPGAOCLSDKROOT%\host\windows64\bin to PATH
```

  Running the `init_opencl.bat` script only affects the current command window. The script performs the following tasks:
  — Finds the Intel Quartus Prime Pro Edition software installation
  — Finds the Microsoft Visual Studio installation
  — Imports the Microsoft Visual Studio environment to properly set the *LIB* environment variable
  — Ensures that the *PATH* environment variable includes the path to the Microsoft `LINK.EXE` file and the `aoc.exe` file

2.4. Verifying Software Installation

Invoke the `version` utility command and verify that the correct version of the OpenCL software is installed.

- At a command prompt, invoke the `aocl version` utility command.
  An output similar to the one below notifies you of a successful installation:

```
aocl <version>.<build> (Intel(R) FPGA SDK for OpenCL(TM), Version <version>
Build <build>, Copyright (C) <year> Intel Corporation)
```

- If installation was unsuccessful, reinstall the software. You can also refer to the *Intel FPGA Software Installation and Licensing* manual and the Intel FPGA Knowledge Base for more information.

Related Information

- Intel FPGA Software Installation and Licensing
- Intel FPGA Knowledge Base

2.5. Installing an FPGA Board

Before creating an OpenCL application for an FPGA accelerator board or SoC device, you must first download and install the Custom Platform from your board vendor. Most Custom Platform installers require administrator privileges. To install your board into a Windows host system, invoke the `aocl install` <*path_to_customplatform*> utility command.

The steps below outline the board installation procedure. Some Custom Platforms require additional installation tasks. Consult your board vendor's documentation for further information on board installation.

1. Follow your board vendor's instructions to connect the FPGA board to your system.
2. Download the Custom Platform for your FPGA board from your board vendor's website. To download an Intel FPGA SDK for OpenCL Reference Platform, refer to the Intel FPGA SDK for OpenCL FPGA Platforms page.

   Tip: If you are installing a BSP that is provided with the Intel FPGA SDK for OpenCL (such as a10_ref, s10_ref, or a10soc, you do not need to download and install a Custom Platform. The BSP files are in *INTELFPGAOCLSDKROOT*/hld/boards. You can skip to 4 on page 14.
3. Install the Custom Platform in a folder that you own (that is, not a system folder).

   You can install multiple Custom Platforms simultaneously on the same system using the SDK utilities, such as `aocl diagnose` with multiple Custom Platforms. The Custom Platform subdirectory contains the `board_env.xml` file.

   In a system with multiple Custom Platforms, ensure that the host program uses the FPGA Client Driver (FCD) to discover the boards rather than linking to the Custom Platforms' memory-mapped device (MMD) libraries directly. As long as FCD is correctly set up for Custom Platform, FCD finds all the installed boards at runtime.
4. Set the *QUARTUS_ROOTDIR_OVERRIDE* user environment variable to point to the Intel Quartus Prime Pro Edition software installation directory.
5. Add the paths to the Custom Platform libraries (for example, the memory-mapped (MMD) library) to the *PATH* environment variable setting.

   For information on setting user environment variables and running the `init_opencl` script, refer to the *Setting the Intel FPGA SDK for OpenCL Pro Edition User Environment Variables* section.

6. Invoke the command `aocl install <path_to_customplatform>` at a command prompt.

   Invoking `aocl install <path_to_customplatform>` also installs a board driver that allows communication between host applications and hardware kernel programs.

   Remember: You need administrative rights to install a board. To run a Windows command prompt as an administrator, click **Start ➤ All Programs ➤ Accessories**. Under **Accessories**, right click **Command Prompt**, In the right-click menu, click **Run as Administrator**.

   On Windows 8.1 or Windows 10 systems, you might also need to disable signed driver verification. For details, see the following articles:

   - Windows 8: https://www.intel.com/content/altera-www/global/en_us/index/support/support-resources/knowledge-base/solutions/fb321729.html
   - Windows 10: https://www.intel.com/content/altera-www/global/en_us/index/support/support-resources/knowledge-base/embedded/2017/Why-does-aocl-diagnose-fail-while-using-Windows-10.html

7. To query a list of FPGA devices installed in your machine, invoke the `aocl diagnose` command.
   The software generates an output that includes the *<device_name>*, which is an acl number that ranges from acl0 to acl127.

   Attention: For possible errors after implementing the `aocl diagnose` utility, refer to Possible Errors After Running the `diagnose` Utility section in the *Intel Arria® 10 GX FPGA Development Kit Reference Platform Porting Guide*. For more information on querying the *<device_name>* of your accelerator board, refer to the *Querying the Device Name of Your FPGA Board* section.

8. To verify the successful installation of the FPGA board, invoke the command `aocl diagnose <device_name>` to run any board vendor-recommended diagnostic test.

Related Information

- Querying the Device Name of Your FPGA Board on page 21
- Intel FPGA SDK for OpenCL FPGA Platforms
- Setting the Intel FPGA SDK for OpenCL Pro Edition User Environment Variables on page 12

2.5.1. Enabling Backwards Compatibility for an FPGA Board

To enable a board with a downlevel BSP, you need a version of Intel Quartus Prime Pro Edition with the same version number as the BSP version. You can use BSPs with versions up to two versions lower than your Intel SDK for OpenCL version.For example, you can use Version 18.0, and Version 17.1.1 BSPs with Intel FPGA SDK for OpenCL Version 18.1, as well as Version 18.1 BSPs.

After you have installed and configured the required version or versions of Intel Quartus Prime Pro Edition, you enable your backwards compatible boards by setting the QUARTUS_ROOTDIR_OVERRIDE environment variable and running the `aocl install` command so that your Intel Quartus Prime Pro Edition version matches your BSP version number.

Before you set the QUARTUS_ROOTDIR_OVERRIDE environment variable and run the `aocl install` command, install and configure the version of Intel Quartus Prime Pro Edition that your BSP needs.

Also, install your FPGA board or boards following the instructions in Installing an FPGA Board on page 14.

To enable a board with a backwards compatible BSP:

1. Set the QUARTUS_ROOTDIR_OVERRIDE to point to the version of Intel Quartus Prime Pro Edition that your BSP needs.

   For example, if you have Version 18.0 BSP, QUARTUS_ROOTDIR_OVERRIDE must point at Intel Quartus Prime Pro Edition Version 18.0:

   ```
   set QUARTUS_ROOTDIR_OVERRIDE=C:\intalFPGA_pro\18.0\quartus
   ```

2. Run the `aocl install <path_to_customplatform>` command with the path to your backward compatible BSP as the value for *<path_to_customplatform>*.

   For example, to point to a Version 18.0 BSP for the Intel Arria 10 GX FPGA Development Kit Reference Platform, install the board with the following command:

   ```
   aocl install <YOUR_PATH_TO_V18.0_BSPs>\a10_ref
   ```

After you complete these steps, Intel SDK for OpenCL commands like `aocl diagnose` should work correctly with your backlevel BSP.

2.6. Verifying Host Runtime Functionality via Emulation

Test the functionality of the host runtime by emulating an OpenCL design example using the Intel FPGA SDK for OpenCL Pro Edition Emulator.

1. Install a Custom or Reference Platform because emulation targets a specific FPGA board.
2. Verify that the *QUARTUS_ROOTDIR_OVERRIDE* environment variable points to the correct edition of the Intel Quartus Prime Pro Edition software. Open a Windows command window and then type `echo %QUARTUS_ROOTDIR_OVERRIDE%` at the command prompt.

   If the path to the installation folder of the Intel Quartus Prime Pro Edition software is not returned, add it to the *QUARTUS_ROOTDIR_OVERRIDE* setting.
3. Add the path to the `LINK.exe` file in Microsoft Visual Studio to the *PATH* user environment variable setting.
4. Add the path to the Microsoft compilation time libraries in Microsoft Visual Studio to the *LIB* user environment variable setting.

1. Downloading an OpenCL Design Example on page 17
2. Compiling a Kernel for Emulation on page 17

3. Building the Host Application on page 17
4. Emulating Your OpenCL Kernel on page 18

2.6.1. Downloading an OpenCL Design Example

The OpenCL Design Examples page contains sample applications of varying complexities that you can download and run on your FPGA.

The following instructions are for downloading the Hello World design.

1. In the OpenCL Design Examples page, under Basic Examples, click **Hello World**.
2. In the Hello World Design Example page, under Downloads, click ***<version>* x64 Windows package (.zip)** to download the compressed file for your platform.
3. Unzip the `exm_opencl_hello_world_x64_windows_<version>.zip` file and save it in a folder that you have write access for.

   Important: Ensure that the folder name does not contain spaces.

Related Information

OpenCL Design Examples

2.6.2. Compiling a Kernel for Emulation

To compile an OpenCL kernel for emulation, include the `-march=emulator` option in your `aoc` command.

- To compile the kernel for your target FPGA board, at a command prompt, navigate to the hello_world design and then invoke the following command:

  ```
  aoc -march=emulator -v -board=<board_name> device/
  hello_world.cl -o bin/hello_world_emulation.aocx
  ```

  The Intel FPGA SDK for OpenCL Offline Compiler compiles the `hello_world.cl` kernel source file and creates the `hello_world_emulation.aocx` emulation-specific executable file in the `bin` subdirectory.

2.6.3. Building the Host Application

The `<path_to_exm_opencl_hello_world_x64_windows_<version>>\hello_world\hello_world.sln` file contains the host solution. After you open this `.sln` file in Microsoft Visual Studio, you can build the OpenCL host application in the `main.cpp` file.

If you are using Microsoft Visual Studio, you need the FCD, and the Installable Client Driver (ICD) from Khronos. To set up Microsoft Visual Studio with FCD and ICD, perform the following tasks prior to building the host application:

1. Verify that FCD and ICD are set up correctly. You must set up FCD and ICD manually if invoking the `aocl install <path_to_customplatform>` utility command fails to set them up. For instructions, refer to the *Accessing Custom Platform-Specific Functions* and *Linking to the ICD Loader Library on Windows* sections of the *Intel FPGA SDK for OpenCL Pro Edition Programming Guide* for more information.
2. Link the host application to the `OpenCL.lib` library.

a. Under the solution properties, select **Configuration Properties ➤ Linker ➤ Input**.

b. In the **Additional Dependencies** field, enter `OpenCL.lib`.

Attention: Because you are using FCD and ICD, do not link the host program to `alteracl.lib` or to your Custom Platform's MMD libraries directly.

To build the hello_world host application, perform the following tasks:

1. Open the *<path_to_exm_opencl_hello_world_x64_windows_<version>>*`\hello_world\hello_world.sln` file in Microsoft Visual Studio.
2. Verify that the build configuration is correct. The default build configuration is **Debug**, but you can use **Release**. You must select the appropriate option as the solution platform (for example, for x64 architecture, select **x64**).
3. Build the solution by selecting the **Build ➤ Build Solution** menu option, or by pressing the F7 key.
   The `hello_world.exe` executable will be in the *<path_to_exm_opencl_hello_world_x64_windows_<version>>*`\hello_world\bin` folder.
4. Verify that the build is correct. An output ending with a message similar to the one shown below notifies you of a successful build:

```
1> Build succeeded.
1>
1> Time Elapsed 00:00:03:29
========== Build: 1 succeeded, 0 failed, 0 up-to-date, 0 skipped ==========
```

Attention: You can ignore the `LNK4009: PDB 'vc90.pdb' was not found with...` warnings because they have no effect on the build. The compiler might issue this type of warning messages if you have built your Windows libraries using a previous version of Microsoft Visual Studio.

Related Information

- Accessing Custom Platform-Specific Functions
- Linking to the ICD Loader Library on Windows

2.6.4. Emulating Your OpenCL Kernel

To emulate your OpenCL kernel, run the emulation `.aocx` file on the platform on which you build your kernel.

To emulate your kernel, perform the following steps:

1. Run the utility command `aocl linkflags` to find out which libraries are necessary for building a host application. The software lists the libraries for both emulation and regular kernel compilation flows.
2. Link your host application to the libraries returned by the `aocl linkflags` utility command.
3. Move the `hello_world_emulation.aocx` file to the current working directory so that the host can locate it easily.
4. To run the host application for emulation, first define the number of emulated devices by invoking the `set CL_CONTEXT_EMULATOR_DEVICE_INTELFPGA=<number_of_devices>` command and then run the host application.
   This command specifies the number of identical emulation devices that the Emulator needs to provide.

   Remember: When the environment variable *CL_CONTEXT_EMULATOR_DEVICE_INTELFPGA* is set, only the emulated devices are available, that is, access to all physical boards is disabled.
5. After you run the host application, unset the *CL_CONTEXT_EMULATOR_DEVICE_INTELFPGA* variable by invoking the `set CL_CONTEXT_EMULATOR_DEVICE_INTELFPGA=` command.

Each invocation of the emulated kernel creates a shared library copy called `<process_ID>-libkernel.so` in a default temporary directory, where *<process_ID>* is a unique numerical value assigned to each emulation run. You may override the default directory by setting the *TMP* or *TEMP* environment variable.

Related Information

Running the Host Application on page 23

2.7. Creating the FPGA Hardware Configuration File of an OpenCL Kernel

Download the Windows version of the hello_world OpenCL design example from the OpenCL Design Examples page. Compile the hello_world kernel program using the Intel FPGA SDK for OpenCL Offline Compiler to create the `hello_world.aocx` executable file. The `.aocx` file is the FPGA hardware configuration file.

After you successfully install your FPGA board and emulate your kernel, you can create a `.aocx` file that executes on the device. The steps below describe the process of creating a `.aocx` file from the hello_world example design. For more information on the OpenCL design examples, refer to the OpenCL Design Examples page.

Note: The Intel FPGA SDK for OpenCL Offline Compiler compiles an OpenCL kernel for a target FPGA board. You must install the FPGA board before creating a hardware configuration file. If you have not installed the FPGA board, refer to *Installing an FPGA Board* for more instructions.

1. Verify that the *QUARTUS_ROOTDIR_OVERRIDE* environment variable points to the Intel Quartus Prime Pro Edition software. Open a Windows command window and then type `echo %QUARTUS_ROOTDIR_OVERRIDE%` at the command prompt.

   If the path to the installation folder of the Intel Quartus Prime Pro Edition software is not returned, add it to the *QUARTUS_ROOTDIR_OVERRIDE* setting.
2. Select your target FPGA board. To list the FPGA boards available in your Custom Platform, invoke the command `aoc -list-boards` at a command prompt.

   For more information on the `-list-boards` option of the `aoc` command, refer to the *Listing the Available FPGA Boards in Your Custom Platform (-list-boards)* section of the *Intel FPGA SDK for OpenCL Pro Edition Programming Guide*.
3. At a command prompt, navigate to the hello_world design folder containing the `hello_world.cl` file that you used for emulation.
4. To compile the kernel for your target FPGA board, invoke the following command:

   `aoc -v -board=<board_name> device/hello_world.cl -o bin/hello_world.aocx`

   This command performs the following tasks:

   - Generates the Intel Quartus Prime design project files from the OpenCL source code.
   - Checks for initial syntax errors.
   - Performs basic optimizations.
   - Creates a `hello_world` folder containing necessary intermediate files.
   - Creates the *Intel FPGA SDK for OpenCL Offline Compiler Object File* (`.aoco`).
   - Creates the `.aocx` file.

   Tip: If you have only one FPGA board installed, `-board=<board_name>` is optional.

   Attention: The `.aocx` file might take hours to build, depending on the complexity of the kernel. To view the progress of the compilation on-screen, include the `-v` flag in your `aoc` command. An example output is shown below.

   ```
   aoc: Environment checks are completed successfully.
   You are now compiling the full flow!!
   aoc: Selected target board <board_name>
   aoc: Running OpenCL parser....
   aoc: OpenCL parser completed successfully.
   aoc: Compiling....
   aoc: Linking with IP library ...
   aoc: First stage compilation completed successfully.
   aoc: Setting up project for CvP revision flow....
   aoc: Hardware generation completed successfully.
   ```

   The offline compiler displays the line `aoc: Hardware generation completed successfully.` to signify the completion of the compilation process.

For more information about the `-board=<board_name>` option of the `aoc` command, refer to the *Compiling a Kernel for a Specific FPGA Board (-board=<board_name>)* section of the *Intel FPGA SDK for OpenCL Pro Edition Programming Guide*.

For more information about the `-v` option of the `aoc` command, refer to the *Generating Compilation Progress Report (-v)* section of the *Intel FPGA SDK for OpenCL Pro Edition Programming Guide*.

For more information about the `-o <filename>` option of the `aoc` command, refer to the *Specifying the Name of an Intel FPGA SDK for OpenCL Offline Compiler Output File (-o <filename>)* section of the *Intel FPGA SDK for OpenCL Pro Edition Programming Guide*.

Related Information

- Listing the Available FPGA Boards in Your Custom Platform (-list-boards)
- Compiling a Kernel for a Specific FPGA Board (-board=<board_name>)
- Generating Compilation Progress Report (-v)
- Specifying the Name of an Intel FPGA SDK for OpenCL Offline Compiler Output File (-o <filename>)
- OpenCL Design Examples
- Setting the Intel FPGA SDK for OpenCL Pro Edition User Environment Variables on page 12
- Installing an FPGA Board on page 14

2.8. Updating the Hardware Image on the FPGA

If applicable, before you execute an OpenCL kernel program on the FPGA, ensure that the flash memory of the FPGA contains a hardware image created using a current version of the OpenCL software.

Remember:

- If your Custom Platform requires that you preload a valid OpenCL image into the flash memory, for every major release of the Intel Quartus Prime Design Suite, program the flash memory of the FPGA with a hardware image compatible with the current version of the software.

Related Information

Configuring the Intel Arria 10 GX FPGA Development Kit for the Intel FPGA SDK for OpenCL

2.8.1. Querying the Device Name of Your FPGA Board

Some OpenCL software utility commands require you to specify the device name (*<device_name>*). The *<device_name>* refers to the acl number (e.g. acl0 to acl127) that corresponds to the FPGA device. When you query a list of accelerator boards, the OpenCL software produces a list of installed devices on your machine in the order of their device names.

- To query a list of installed devices on your machine, type `aocl diagnose` at a command prompt.
  The software generates an output that resembles the example shown below:

```
aocl diagnose: Running diagnostic from INTELFPGAOCLSDKROOT/board/
<board_name>/<platform>/libexec

Verified that the kernel mode driver is installed on the host machine.

Using board package from vendor: <board_vendor_name>
Querying information for all supported devices that are installed on the
host machine ...

device_name  Status  Information

acl0         Passed  <descriptive_board_name>
                     PCIe dev_id = <device_ID>, bus:slot.func = 02:00.00,
                       at Gen 2 with 8 lanes.
                     FPGA temperature = 43.0 degrees C.

acl1         Passed  <descriptive_board_name>
                     PCIe dev_id = <device_ID>, bus:slot.func = 03:00.00,
                       at Gen 2 with 8 lanes.
                     FPGA temperature = 35.0 degrees C.

Found 2 active device(s) installed on the host machine, to perform a full
diagnostic on a specific device, please run aocl diagnose <device_name>

DIAGNOSTIC_PASSED
```

2.8.2. Programming the Flash Memory of an FPGA

Configure the FPGA by loading the hardware image of an Intel FPGA SDK for OpenCL design example into the flash memory of the device. When there is no power, the FPGA retains the hardware configuration file in the flash memory. When you power up the system, it configures the FPGA circuitry based on this hardware image in the flash memory. Therefore, it is imperative that an OpenCL-compatible hardware configuration file is loaded into the flash memory of your FPGA.

Preloading an OpenCL image into the flash memory is necessary for the proper functioning of many Custom Platforms. For example, most PCIe-based boards require a valid OpenCL image in flash memory so that hardware on the board can use the image to configure the FPGA device when the host system powers up for the first time. If the FPGA is not configured with a valid OpenCL image, the system will fail to enumerate the PCIe endpoint, or the driver will not function.

Before running any designs, ensure that the flash memory of your board has a valid OpenCL image that is compatible with the current OpenCL software version. Consult your board vendor's documentation for board-specific requirements.

Caution: When you load the hardware configuration file into the flash memory of the FPGA, maintain system power for the entire loading process, which might take a few minutes. Also, do not launch any host code that calls OpenCL kernels or might otherwise communicate with the FPGA board.

To load your hardware configuration file into the flash memory of your FPGA board, perform the following tasks:

1. Install any drivers or utilities that your Custom Platform requires.

For example, some Custom Platforms require you to install the Intel FPGA Download Cable driver to load your hardware configuration file into the flash memory. For installation instructions, refer to the *Intel FPGA Download Cable II User Guide*.

2. Download a design example for your Custom Platform.

   Remember: Download design examples from the OpenCL Design Examples page, and extract the example to a location that you have write access for. Ensure that the location name does not contain spaces.

3. To load the hardware configuration file into the flash memory, invoke the `aocl flash` *<device_name>* *<design_example_filename>*`.aocx` command, where *<device_name>* refers to the acl number (e.g. acl0 to acl127) that corresponds to your FPGA device, and *<design_example_filename>*`.aocx` is the hardware configuration file you create from the *<design_example_filename>*`.cl` file in the design example package.

4. Power down your device or computer and then power it up again.

   Power cycling ensures that the FPGA configuration device retrieves the hardware configuration file from the flash memory and configures it into the FPGA.

   Warning: Some Custom Platforms require you to power cycle the entire host system after programming the flash memory. For example, PCIe-based Custom Platforms might require a host system restart to reenumerate the PCIe endpoint. Intel recommends that you power cycle the complete host system after programming the flash memory.

Related Information

- Intel FPGA Download Cable II User Guide
- OpenCL Design Examples
- Setting the Intel FPGA SDK for OpenCL Pro Edition User Environment Variables on page 12

2.9. Executing an OpenCL Kernel on an FPGA

Build your OpenCL host application in Microsoft Visual Studio, and run the application by invoking the `hello_world.exe` executable. The Intel FPGA SDK for OpenCL is compatible with 64-bit host binaries only.

Related Information

Building the Host Application on page 17

2.9.1. Running the Host Application

To execute the OpenCL kernel on the FPGA, run the Windows host application that you built from the `.sln` file.

1. Add the path *%INTELFPGAOCLSDKROOT%*`\host\windows64\bin` to the *PATH* environment variable.
2. At a command prompt, navigate to the host executable within the *<path_to_exm_opencl_hello_world_x64_windows_<version>>*`\hello_world\bin` folder.
3. Invoke the `hello_world.exe` executable.

The `hello_world` executable executes the kernel code on the FPGA.

2.9.2. Output from Successful Kernel Execution

When you run the host application to execute your OpenCL kernel on the target FPGA, the OpenCL software notifies you of a successful kernel execution.

Example output:

```
Reprogramming device [0] with handle 1
Querying platform for info:
==========================
CL_PLATFORM_NAME                         = Intel(R) FPGA SDK for OpenCL(TM)
CL_PLATFORM_VENDOR                       = Intel Corporation
CL_PLATFORM_VERSION = OpenCL 1.0 Intel(R) FPGA SDK for OpenCL(TM), <version>

Querying device for info:
========================
CL_DEVICE_NAME                           = <board name> : <descriptive board
name>
CL_DEVICE_VENDOR                         = <board vendor name>
CL_DEVICE_VENDOR_ID                      = <board vendor ID>
CL_DEVICE_VERSION   = OpenCL 1.0 Intel(R) FPGA SDK for OpenCL(TM), <version>
CL_DRIVER_VERSION                        = <version>
CL_DEVICE_ADDRESS_BITS                   = 64
CL_DEVICE_AVAILABLE                      = true
CL_DEVICE_ENDIAN_LITTLE                  = true
CL_DEVICE_GLOBAL_MEM_CACHE_SIZE          = 32768
CL_DEVICE_GLOBAL_MEM_CACHELINE_SIZE      = 0
CL_DEVICE_GLOBAL_MEM_SIZE                = 8589934592
CL_DEVICE_IMAGE_SUPPORT                  = true
CL_DEVICE_LOCAL_MEM_SIZE                 = 16384
CL_DEVICE_MAX_CLOCK_FREQUENCY            = 1000
CL_DEVICE_MAX_COMPUTE_UNITS              = 1
CL_DEVICE_MAX_CONSTANT_ARGS              = 8
CL_DEVICE_MAX_CONSTANT_BUFFER_SIZE       = 2147483648
CL_DEVICE_MAX_WORK_ITEM_DIMENSIONS       = 3
CL_DEVICE_MEM_BASE_ADDR_ALIGN            = 8192
CL_DEVICE_MIN_DATA_TYPE_ALIGN_SIZE       = 1024
CL_DEVICE_PREFERRED_VECTOR_WIDTH_CHAR    = 4
CL_DEVICE_PREFERRED_VECTOR_WIDTH_SHORT   = 2
CL_DEVICE_PREFERRED_VECTOR_WIDTH_INT     = 1
CL_DEVICE_PREFERRED_VECTOR_WIDTH_LONG    = 1
CL_DEVICE_PREFERRED_VECTOR_WIDTH_FLOAT   = 1
CL_DEVICE_PREFERRED_VECTOR_WIDTH_DOUBLE  = 0
Command queue out of order?              = false
Command queue profiling enabled?         = true
Using AOCX: hello_world.aocx

Kernel initialization is complete.
Launching the kernel...

Thread #2: Hello from the Intel(R) FPGA OpenCL(TM) compiler!

Kernel execution is complete.
```

2.10. Uninstalling the Software

To uninstall the Intel FPGA SDK for OpenCL Pro Edition for Windows, run the uninstaller, and restore all modified environment variables to their previous settings.

1. From the Windows Start Menu shortcut, navigate to the `Altera <version> Pro Edition` installation folder.
2. Select **Uninstall Intel Quartus Prime Pro Edition**.

The uninstallation wizard appears.

3. In the uninstallation wizard, perform the following tasks:
   a. Select **Individual components** and then click **Next**.
   b. Select the SDK and then click **Next**.
      The uninstaller uninstalls the Intel FPGA SDK for OpenCL.
4. Remove the following paths from the *PATH* environment variable:
   a. `%INTELFPGAOCLSDKROOT%\bin`
   b. `%INTELFPGAOCLSDKROOT%\host\windows64\bin`
5. Remove the *INTELFPGAOCLSDKROOT* environment variable.
6. Remove the *QUARTUS_ROOTDIR_OVERRIDE* environment variable.

2.11. Uninstalling an FPGA Board

To uninstall an FPGA board for Windows, invoke the `uninstall` utility command, uninstall the Custom Platform, and unset the relevant environment variables. You must uninstall the existing FPGA board if you migrate your OpenCL application to another FPGA board that belongs to a different Custom Platform.

To uninstall your FPGA board, perform the following tasks:

1. Following your board vendor's instructions to disconnect the board from your machine.
2. Invoke the `aocl uninstall <path_to_customplatform>` utility command to remove the current host computer drivers (for example, PCIe drivers). The Intel FPGA SDK for OpenCL uses these drivers to communicate with the FPGA board.
3. Uninstall the Custom Platform.
4. Unset the *PATH* environment variable.

3. Getting Started with the Intel FPGA SDK for OpenCL Pro Edition for Linux

The Intel FPGA SDK for OpenCL Pro Edition setup process includes downloading and installing the software, installing the FPGA board, and then executing an OpenCL kernel on the FPGA.

3.1. Downloading the Intel FPGA SDK for OpenCL Pro Edition

Download the installation package that contains the Intel FPGA SDK for OpenCL Pro Edition and all related software for Linux from the Download Center.

The Download Center provides a tar file that includes all of the following software and files:

- Intel FPGA SDK for OpenCL
- Intel Quartus Prime Pro Edition software
- Device support

1. Go to the Intel FPGA SDK for OpenCL Download Center at the following URL:
   https://fpgasoftware.intel.com/opencl/
2. Select the Pro edition.
3. Select the software version. The default selection is the current version.
4. Select **Direct Download**.
5. On the **Linux SDK** tab, select one of the following packages:

- **Intel FPGA SDK for OpenCL (includes Intel Quartus Prime Pro Edition software and devices)**

  This package provides core functions of the Intel FPGA SDK for OpenCL including the Intel FPGA SDK for OpenCL Offline Compiler and the Intel FPGA SDK for OpenCL utility.
- **Intel FPGA SDK for OpenCL (includes Intel Quartus Prime Pro Edition software and devices with Intel Code Builder)**

  This package provides all of the functions of the Intel FPGA SDK for OpenCL as well as integration with Intel Code Builder to provide an integrated development environment (IDE) for developing your OpenCL kernels.

Click **More** beside **Download and install instructions** if you want to see the download and installation instructions.

6. Perform the steps outlined in the download and installation instructions on the download page.

Related Information

Intel FPGA website

3.2. Installing the Intel FPGA SDK for OpenCL Pro Edition

Unpack the downloaded tar file and and run the installation files to install all the software and files.
Install the Linux version of the Intel FPGA SDK for OpenCL Pro Edition in a directory that you own.

- You must have `sudo` or `root` privileges.
- You must install the Linux OS kernel source and headers (for example, `kernel-devel.x86_64` and `kernel-headers.x86_64`), and the GNU Compiler Collection (GCC) (`gcc.x86_64`).
- If you are installing a package that includes Intel Code Builder, you must have Java SE 1.8.71 or later installed to run Intel Code Builder. If you have an earlier version of Java SE installed, you can still complete the installation of Intel Code Builder. However, you must meet the Java version prerequisite to run Intel Code Builder.

Attention: If you install the software on a system that does not contain any C Shell Run Commands file (`.cshrc`) or Bash Run Commands file (`.bashrc`) in your directory, you must set the environment variables *INTELFPGAOCLSDKROOT* and *PATH* manually. Alternatively, you may create the `.cshrc` and `.bashrc` files, and then append the environment variables to them. To ensure that the updates take effect, restart your terminal after you set the environment variables.

To install the Intel FPGA SDK for OpenCL, Intel Quartus Prime software, and device support files simultaneously, perform the following tasks:

1. Unpack the downloaded tar file into a temporary directory.
2. Run the `setup_pro.sh` file to install the SDK with the Intel Quartus Prime Pro Edition software.
3. *Note:* The installer sets the environment variable *INTELFPGAOCLSDKROOT* to point to the path of the software installation.

Verify that *INTELFPGAOCLSDKROOT* points to the current version of the software. Open a shell and then type `echo $INTELFPGAOCLSDKROOT` at the command prompt.

If the returned path does not point to the location of the Intel FPGA SDK for OpenCL installation , edit the *INTELFPGAOCLSDKROOT* setting.

For instructions on modifying environment variable settings, refer to *Setting the Intel FPGA SDK for OpenCL Pro Edition User Environment Variables*.

Related Information

Setting the Intel FPGA SDK for OpenCL Pro Edition User Environment Variables on page 28

3.3. Setting the Intel FPGA SDK for OpenCL Pro Edition User Environment Variables

You have the option to set the Intel FPGA SDK for OpenCL Pro Edition Linux user environment variables permanently or transiently. The environment variable settings describe the FPGA board and the host runtime to the software.

Attention: If you set the environment variables permanently, you apply the settings once during installation. If you set the environment variables transiently, you must apply the settings during installation and during every subsequent session you run.

Table 3. Intel FPGA SDK for OpenCL Linux User Environment Variable Settings

Environment Variable	Path to Include
PATH	`$INTELFPGAOCLSDKROOT/bin` where *INTELFPGAOCLSDKROOT* points to the path of the software installation
LD_LIBRARY_PATH	`$INTELFPGAOCLSDKROOT/host/linux64/lib`

- To apply permanent environment variable settings, open a shell and then type the `export <variable_name>="<variable_setting>":$<variable_name>` command.

  For example, the command `export PATH="$INTELFPGAOCLSDKROOT/bin":$PATH` adds `$INTELFPGAOCLSDKROOT/bin` to the list of *PATH* settings.
- To apply transient environment variable settings, open a bash-shell command-line terminal and run the `source $INTELFPGAOCLSDKROOT/init_opencl.sh` command. This command does not work in other shells.

  Example script output:

```
Adding $INTELFPGAOCLSDKROOT/bin to PATH
Adding $INTELFPGAOCLSDKROOT/host/linux64/lib to LD_LIBRARY_PATH
```

3.4. Verifying Software Installation

Invoke the `version` utility command and verify that the correct version of the OpenCL software is installed.

- At a command prompt, invoke the `aocl version` utility command.
  An output similar to the one below notifies you of a successful installation:

```
aocl <version>.<build> (Intel(R) FPGA SDK for OpenCL(TM), Version <version>
Build <build>, Copyright (C) <year>  Corporation)
```

- If installation was unsuccessful, reinstall the software. You can also refer to the *Intel FPGA Software Installation and Licensing* manual and the Intel FPGA Knowledge Base for more information.

Related Information

- Intel FPGA Software Installation and Licensing
- Intel FPGA Knowledge Base

3.5. Installing an FPGA Board

Before creating an OpenCL application for an FPGA board on Linux, you must first download and install the Custom Platform from your board vendor. Most Custom Platform installers require administrator privileges. To install your board into a Linux host system, invoke the `install` utility command.

The steps below outline the board installation procedure. Some Custom Platforms require additional installation tasks. Consult your board vendor's documentation for further information on board installation.

1. Follow your board vendor's instructions to connect the FPGA board to your system.
2. Download the Custom Platform for your FPGA board from your board vendor's website. To download an Intel FPGA SDK for OpenCL Reference Platform, refer to the Intel FPGA SDK for OpenCL FPGA Platforms page.

   Tip: If you are installing a BSP that is provided with the Intel FPGA SDK for OpenCL (such as a10_ref, s10_ref, or a10soc, you do not need to download and install a Custom Platform. The BSP files are in *INTELFPGAOCLSDKROOT*/hld/boards. You can skip to 4 on page 29.
3. Install the Custom Platform in a directory that you own (that is, not a system directory).

   You can install multiple Custom Platforms simultaneously on the same system. To use the RTE utilities, such as `aocl diagnose` with multiple Custom Platforms. The Custom Platform subdirectory contains the `board_env.xml` file.

   In a system with multiple Custom Platforms, ensure that the host program uses the FPGA Client Drivers (FCD) to discover the boards rather than linking to the Custom Platforms' memory-mapped device (MMD) libraries directly. If FCD is correctly set up for Custom Platform, FCD finds all the installed boards at runtime.
4. Set the *QUARTUS_ROOTDIR_OVERRIDE* user environment variable to point to the Intel Quartus Prime Pro Edition software installation directory. Open a shell and then type `echo $QUARTUS_ROOTDIR_OVERRIDE` at the command prompt.
5. Add the paths to the Custom Platform libraries (for example, memory-mapped (MMD) library) to the *LD_LIBRARY_PATH* environment variable setting.

   For information on setting Linux user environment variables and running the `init_opencl` script, refer to the *Setting the Intel FPGA SDK for OpenCL Pro Edition User Environment Variables* section.

6. *Remember:* You need `sudo` or `root` privileges to install a board.

   Invoke the command `aocl install <path_to_customplatform>` at a command prompt.

   Invoking `aocl install <path_to_customplatform>` also installs a board driver that allows communication between host applications and hardware kernel programs.

7. To query a list of FPGA devices installed in your machine, invoke the `aocl diagnose` command.

   The software generates an output that includes the <*device_name*>, which is an acl number that ranges from acl0 to acl31.

   Attention: For possible errors after implementing the `aocl diagnose` utility, refer to Possible Errors After Running the `diagnose` Utility section. For more information on querying the <*device_name*> of your accelerator board, refer to the *Querying the Device Name of Your FPGA Board* section.

8. To verify the successful installation of the FPGA board, invoke the command `aocl diagnose <device_name>` to run any board vendor-recommended diagnostic test.

3.5.1. Enabling Backwards Compatibility for an FPGA Board

To enable a board with a downlevel BSP, you need a version of Intel Quartus Prime Pro Edition with the same version number as the BSP version. You can use BSPs with versions up to two versions lower than your Intel SDK for OpenCL version.For example, you can use Version 18.0, and Version 17.1.1 BSPs with Intel FPGA SDK for OpenCL Version 18.1, as well as Version 18.1 BSPs.

After you have installed and configured the required version or versions of Intel Quartus Prime Pro Edition, you enable your backwards compatible boards by setting the `QUARTUS_ROOTDIR_OVERRIDE` environment variable and running the `aocl install` command so that your Intel Quartus Prime Pro Edition version matches your BSP version number.

Before you set the `QUARTUS_ROOTDIR_OVERRIDE` environment variable and run the `aocl install` command, install and configure the version of Intel Quartus Prime Pro Edition that your BSP needs.

Also, install your FPGA board or boards following the instructions in Installing an FPGA Board on page 29.

To enable a board with a backwards compatible BSP:

1. Set the `QUARTUS_ROOTDIR_OVERRIDE` to point to the version of Intel Quartus Prime Pro Edition that your BSP needs.

   For example, if you have Version 18.0 BSP, `QUARTUS_ROOTDIR_OVERRIDE` must point at Intel Quartus Prime Pro Edition Version 18.0:

   ```
   export QUARTUS_ROOTDIR_OVERRIDE=/opt/quartus/intelfpga/18.0/quartus
   ```

2. Run the `aocl install <path_to_customplatform>` command with the path to your backward compatible BSP as the value for <*path_to_customplatform*>.

For example, to point to a Version 18.0 BSP for the Intel Arria 10 GX FPGA Development Kit Reference Platform, install the board as follows:

```
aocl install <YOUR_PATH_TO_V18.0_BSPs>/a10_ref
```

After you complete these steps, Intel SDK for OpenCL commands like `aocl diagnose` should work correctly with your backlevel BSP.

3.6. Verifying Host Runtime Functionality via Emulation

Test the functionality of the host runtime by emulating an OpenCL design example using the Intel FPGA SDK for OpenCL Emulator.

1. Install a Custom or Reference Platform because emulation targets a specific FPGA board.
2. Verify that the *QUARTUS_ROOTDIR_OVERRIDE* environment variable points to your Intel Quartus Prime software. Open a shell and then type `echo $QUARTUS_ROOTDIR_OVERRIDE` at the command prompt.

   If the path to the installation directory of the Intel Quartus Prime software is not returned, add it to the *QUARTUS_ROOTDIR_OVERRIDE* setting.
3. Verify that the *LD_LIBRARY_PATH* environment variable setting includes the paths identified in *Setting the Intel FPGA SDK for OpenCL User Environment Variables*. Open a shell and then type `echo $LD_LIBRARY_PATH` at the command prompt.

   If the returned path do not include `$INTELFPGAOCLSDKROOT/host/linux64/lib`, add it to the *LD_LIBRARY_PATH* setting.

Each invocation of the emulated kernel creates a shared library copy called `<process_ID>-libkernel.so` in a default temporary directory, where `<process_ID>` is a unique numerical value assigned to each emulation run. You may override the default directory by setting the *TMPDIR* environment variable.

1. Downloading an OpenCL Design Example on page 31
2. Compiling a Kernel for Emulation on page 32
3. Building the Host Application on page 32
4. Emulating Your OpenCL Kernel on page 32

3.6.1. Downloading an OpenCL Design Example

The OpenCL Design Examples page contains sample applications of varying complexities that you can download and run on your FPGA.

The following instructions are for downloading the Hello World design.

1. In the OpenCL Design Examples page, under Basic Examples, click **Hello World**.
2. In the Hello World Design Example page, under Downloads, click ***<version>* x64 Linux package (.tgz)** to download the compressed file for your platform.
3. Unpack the `.tgz` file and save it in a directory to which you have write access.

   Important: Ensure that the directory name does not contain spaces.

Related Information

OpenCL Design Examples

3.6.2. Compiling a Kernel for Emulation

To compile an OpenCL kernel for emulation, include the `-march=emulator` option in your `aoc` command.

- To compile the kernel for your target FPGA board, at a command prompt, navigate to the hello_world design and then invoke the following command:

  ```
  aoc -march=emulator -v -board=<board_name> device/
  hello_world.cl -o bin/hello_world_emulation.aocx
  ```

  The compiles the `hello_world.cl` kernel source file and creates the `hello_world_emulation.aocx` emulation-specific executable file in the `bin` subdirectory.

3.6.3. Building the Host Application

Build the host executable with the *<path_to_exm_opencl_hello_world_x64_linux_<version>>*`/hello_world/Makefile` file.

To build the host application, perform the following tasks:

1. Navigate to the `hello_world` directory.
2. Invoke the `$ make -f Makefile` command. Alternatively, you can simply invoke the `make` command.
   The `hello_world` executable will be in the *<path_to_exm_opencl_hello_world_x64_linux_<version>>*`/hello_world/bin` directory.

3.6.4. Emulating Your OpenCL Kernel

To emulate your OpenCL kernel, run the emulation `.aocx` file on the platform on which you build your kernel.

To emulate your kernel, perform the following steps:

1. Run the utility command `aocl linkflags` to find out which libraries are necessary for building a host application. The software lists the libraries for both emulation and regular kernel compilation flows.
2. Link your host application to the libraries returned by the `aocl linkflags` utility command.
3. Move the `hello_world_emulation.aocx` file to the current working directory so that the host can locate it easily.
4. To run the host application for emulation, invoke the `env CL_CONTEXT_EMULATOR_DEVICE_INTELFPGA=`*<number_of_devices>* *<host_application_filename>* command.
   This command specifies the number of identical emulation devices that the Emulator needs to provide.

Each invocation of the emulated kernel creates a shared library copy called `<process_ID>-libkernel.so` in a default temporary directory, where *<process_ID>* is a unique numerical value assigned to each emulation run. You may override the default directory by setting the *TMP* or *TEMP* environment variable on Windows, or setting *TMPDIR* on Linux.

Related Information

Running the Host Application on page 37

3.7. Creating the FPGA Hardware Configuration File of an OpenCL Kernel

Download the Linux version of the hello_world OpenCL design example from the OpenCL Design Examples page. Compile the hello_world kernel program using the Intel FPGA SDK for OpenCL Offline Compiler to create the `hello_world.aocx` executable file. The `.aocx` file is the FPGA hardware configuration file.

After you successfully install your FPGA board and emulate your kernel, you can create a `.aocx` file that executes on the device. The steps below describe the process of creating a `.aocx` file from the hello_world example design. For more information on the OpenCL design examples, refer to the OpenCL Design Examples page.

Note: The Intel FPGA SDK for OpenCL Offline Compiler compiles an OpenCL kernel for a target FPGA board. You must install the FPGA board before creating a hardware configuration file. If you have not installed the FPGA board, refer to *Installing an FPGA Board* for more instructions.

1. Verify that the *QUARTUS_ROOTDIR_OVERRIDE* environment variable points to your Intel Quartus Prime software. Open a shell and then type `echo $QUARTUS_ROOTDIR_OVERRIDE` at the command prompt.

   If the path to the installation directory of the Intel Quartus Prime software is not returned, add it to the *QUARTUS_ROOTDIR_OVERRIDE* setting.
2. Select your target FPGA board. To list the FPGA boards available in your Custom Platform, invoke the command `aoc -list-boards` at a command prompt.

   For more information on the `-list-boards` option of the `aoc` command, refer to the *Listing the Available FPGA Boards in Your Custom Platform (-list-boards)* section of the *Intel FPGA SDK for OpenCL Pro Edition Programming Guide*.
3. At a command prompt, navigate to the hello_world design directory containing the `hello_world.cl` file that you used for emulation.
4. To compile the kernel for your target FPGA board, invoke the following command:

   `aoc -v -board=<board_name> device/hello_world.cl -o bin/hello_world.aocx`

   This command performs the following tasks:

   - Generates the Intel Quartus Prime design project files from the OpenCL source code.
   - Checks for initial syntax errors.
   - Performs basic optimizations.

- Creates a `hello_world` directory containing necessary intermediate files.
- Creates the *Intel FPGA SDK for OpenCL Offline Compiler Object File* (`.aoco`).
- Creates the `.aocx` file.

Tip: If you have only one FPGA board installed, `-board=<board_name>` is optional.

Attention: The `.aocx` file might take hours to build, depending on the complexity of the kernel. To view the progress of the compilation on-screen, include the `-v` flag in your `aoc` command. An example output is shown below.

```
aoc: Environment checks are completed successfully.
You are now compiling the full flow!!
aoc: Selected target board <board_name>
aoc: Running OpenCL parser....
aoc: OpenCL parser completed successfully.
aoc: Compiling....
aoc: Linking with IP library ...
aoc: First stage compilation completed successfully.
aoc: Setting up project for CvP revision flow....
aoc: Hardware generation completed successfully.
```

The offline compiler displays the line `aoc: Hardware generation completed successfully.` to signify the completion of the compilation process.

For more information about the `-board=<board_name>` option of the `aoc` command, refer to the *Compiling a Kernel for a Specific FPGA Board (-board=<board_name>)* section of the *Intel FPGA SDK for OpenCL Pro Edition Programming Guide*.

For more information about the `-v` option of the `aoc` command, refer to the *Generating Compilation Progress Report (-v)* section of the *Intel FPGA SDK for OpenCL Pro Edition Programming Guide*.

For more information about the `-o <filename>` option of the `aoc` command, refer to the *Specifying the Name of an Intel FPGA SDK for OpenCL Offline Compiler Output File (-o <filename>)* section of the *Intel FPGA SDK for OpenCL Pro Edition Programming Guide*.

Related Information

- Installing an FPGA Board on page 29
- OpenCL Design Examples page
- Listing the Available FPGA Boards in Your Custom Platform (-list-boards)
- Compiling a Kernel for a Specific FPGA Board (-board=<board_name>)
- Generating Compilation Progress Report (-v)
- Specifying the Name of an Intel FPGA SDK for OpenCL Offline Compiler Output File (-o <filename>)

3.8. Updating the Hardware Image on the FPGA

If applicable, before you execute an OpenCL kernel program on the FPGA, ensure that the flash memory of the FPGA contains a hardware image created using a current version of the OpenCL software.

Remember:

- If your Custom Platform requires that you preload a valid OpenCL image into the flash memory, for every major release of the Intel Quartus Prime Design Suite, program the flash memory of the FPGA with a hardware image compatible with the current version of the software.

3.8.1. Querying the Device Name of Your FPGA Board

Some OpenCL software utility commands require you to specify the device name (<*device_name*>). The <*device_name*> refers to the acl number (e.g. acl0 to acl127) that corresponds to the FPGA device. When you query a list of accelerator boards, the OpenCL software produces a list of installed devices on your machine in the order of their device names.

- To query a list of installed devices on your machine, type `aocl diagnose` at a command prompt.
  The software generates an output that resembles the example shown below:

```
aocl diagnose: Running diagnostic from INTELFPGAOCLSDKROOT/board/
<board_name>/<platform>/libexec

Verified that the kernel mode driver is installed on the host machine.

Using board package from vendor: <board_vendor_name>
Querying information for all supported devices that are installed on the
host machine ...

device_name  Status  Information

acl0         Passed  <descriptive_board_name>
                     PCIe dev_id = <device_ID>, bus:slot.func = 02:00.00,
                       at Gen 2 with 8 lanes.
                     FPGA temperature = 43.0 degrees C.

acl1         Passed  <descriptive_board_name>
                     PCIe dev_id = <device_ID>, bus:slot.func = 03:00.00,
                       at Gen 2 with 8 lanes.
                     FPGA temperature = 35.0 degrees C.

Found 2 active device(s) installed on the host machine, to perform a full
diagnostic on a specific device, please run aocl diagnose <device_name>

DIAGNOSTIC_PASSED
```

3.8.2. Programming the Flash Memory of an FPGA

Configure the FPGA by loading the hardware image of an Intel FPGA SDK for OpenCL design example into the flash memory of the device. When there is no power, the FPGA retains the hardware configuration file in the flash memory. When you power up the system, it configures the FPGA circuitry based on this hardware image in the flash memory. Therefore, it is imperative that an OpenCL-compatible hardware configuration file is loaded into the flash memory of your FPGA.

Preloading an OpenCL image into the flash memory is necessary for the proper functioning of many Custom Platforms. For example, most PCIe-based boards require a valid OpenCL image in flash memory so that hardware on the board can use the image to configure the FPGA device when the host system powers up for the first time. If the FPGA is not configured with a valid OpenCL image, the system will fail to enumerate the PCIe endpoint, or the driver will not function.

Before running any designs, ensure that the flash memory of your board has a valid OpenCL image that is compatible with the current OpenCL software version. Consult your board vendor's documentation for board-specific requirements.

Caution: When you load the hardware configuration file into the flash memory of the FPGA, maintain system power for the entire loading process, which might take a few minutes. Also, do not launch any host code that calls OpenCL kernels or might otherwise communicate with the FPGA board.

To load your hardware configuration file into the flash memory of your FPGA board, perform the following tasks:

1. Install any drivers or utilities that your Custom Platform requires.
2. Download a design example for your Custom Platform.

   Remember: Download design examples from the OpenCL Design Examples page, and extract the example to a location to which you have write access. Ensure that the location name does not contain spaces.
3. To load the hardware configuration file into the flash memory, invoke the `aocl flash` *`<device_name>`* *`<design_example_filename>`*`.aocx` command, where *`<device_name>`* refers to the acl number (e.g. acl0 to acl127) that corresponds to your FPGA device, and *`<design_example_filename>`*`.aocx` is the hardware configuration file you create from the *`<design_example_filename>`*`.cl` file in the example design package.
4. Power down your device or computer and then power it up again.

   Power cycling ensures that the FPGA configuration device retrieves the hardware configuration file from the flash memory and configures it into the FPGA.

   Warning: Some Custom Platforms require you to power cycle the entire host system after programming the flash memory. For example, PCIe-based Custom Platforms might require a host system restart to reenumerate the PCIe endpoint. Intel recommends that you power cycle the complete host system after programming the flash memory.

Related Information

- OpenCL Design Examples
- Setting the Intel FPGA SDK for OpenCL Pro Edition User Environment Variables on page 28

3.9. Executing an OpenCL Kernel on an FPGA

You must build your OpenCL host application with the `Makefile` file, and run the application by invoking the `hello_world` executable. You need GNU development tools such as `gcc` and `make` to build the OpenCL application.

Related Information

Building the Host Application on page 32

3.9.1. Running the Host Application

To execute the OpenCL kernel on the FPGA , run the Linux host application that you built from the `Makefile`.

1. Add the path `$INTELFPGAOCLSDKROOT/host/linux64/lib` to the *LD_LIBRARY_PATH* environment variable.
2. At a command prompt, navigate to the host executable within the *<path_to_exm_opencl_hello_world_x64_linux_<version>>/*`hello_world/bin` directory.
3. Invoke the `hello_world` executable.
   The `hello_world` executable executes the kernel code on the FPGA.

3.9.2. Output from Successful Kernel Execution

When you run the host application to execute your OpenCL kernel on the target FPGA, the OpenCL software notifies you of a successful kernel execution.

Example output:

```
Reprogramming device [0] with handle 1
Querying platform for info:
==========================
CL_PLATFORM_NAME                         = Intel(R) FPGA SDK for OpenCL(TM)
CL_PLATFORM_VENDOR                       = Intel Corporation
CL_PLATFORM_VERSION = OpenCL 1.0 Intel(R) FPGA SDK for OpenCL(TM), <version>

Querying device for info:
========================
CL_DEVICE_NAME                           = <board name> : <descriptive board
name>
CL_DEVICE_VENDOR                         = <board vendor name>
CL_DEVICE_VENDOR_ID                      = <board vendor ID>
CL_DEVICE_VERSION   = OpenCL 1.0 Intel(R) FPGA SDK for OpenCL(TM), <version>
CL_DRIVER_VERSION                        = <version>
CL_DEVICE_ADDRESS_BITS                   = 64
CL_DEVICE_AVAILABLE                      = true
CL_DEVICE_ENDIAN_LITTLE                  = true
CL_DEVICE_GLOBAL_MEM_CACHE_SIZE          = 32768
CL_DEVICE_GLOBAL_MEM_CACHELINE_SIZE      = 0
CL_DEVICE_GLOBAL_MEM_SIZE                = 8589934592
CL_DEVICE_IMAGE_SUPPORT                  = true
CL_DEVICE_LOCAL_MEM_SIZE                 = 16384
CL_DEVICE_MAX_CLOCK_FREQUENCY            = 1000
CL_DEVICE_MAX_COMPUTE_UNITS              = 1
CL_DEVICE_MAX_CONSTANT_ARGS              = 8
CL_DEVICE_MAX_CONSTANT_BUFFER_SIZE       = 2147483648
CL_DEVICE_MAX_WORK_ITEM_DIMENSIONS       = 3
CL_DEVICE_MEM_BASE_ADDR_ALIGN            = 8192
CL_DEVICE_MIN_DATA_TYPE_ALIGN_SIZE       = 1024
CL_DEVICE_PREFERRED_VECTOR_WIDTH_CHAR    = 4
CL_DEVICE_PREFERRED_VECTOR_WIDTH_SHORT   = 2
CL_DEVICE_PREFERRED_VECTOR_WIDTH_INT     = 1
CL_DEVICE_PREFERRED_VECTOR_WIDTH_LONG    = 1
CL_DEVICE_PREFERRED_VECTOR_WIDTH_FLOAT   = 1
CL_DEVICE_PREFERRED_VECTOR_WIDTH_DOUBLE  = 0
Command queue out of order?              = false
Command queue profiling enabled?         = true
Using AOCX: hello_world.aocx
```

```
Kernel initialization is complete.
Launching the kernel...

Thread #2: Hello from the Intel(R) FPGA OpenCL(TM) compiler!

Kernel execution is complete.
```

3.10. Uninstalling the Software

To uninstall the Intel FPGA SDK for OpenCL for Linux, remove the software package via the GUI uninstaller, then delete the software directory and restore all modified environment variables to their previous settings.

1. Remove the software package by performing one of the following tasks:
   a. To uninstall the SDK, run the `aocl-<version>-uninstall.run` program located in the `<install directory>/uninstall` directories for the Intel Quartus Prime Standard Edition software and the Intel Quartus Prime Pro Edition software.
2. Remove $*INTELFPGAOCLSDKROOT*`/bin` from the *PATH* environment variable.
3. Remove $*INTELFPGAOCLSDKROOT*`/host/linux64/lib` from the *LD_LIBRARY_PATH* environment variable.
4. Remove the *INTELFPGAOCLSDKROOT* environment variable.
5. Remove the *QUARTUS_ROOTDIR_OVERRIDE* environment variable.

3.11. Uninstalling an FPGA Board

To uninstall an FPGA board for Linux, invoke the `uninstall` utility command, uninstall the Custom Platform, and unset the relevant environment variables.

To uninstall your FPGA board, perform the following tasks:

1. Disconnect the board from your machine by following the instructions provided by your board vendor.
2. Invoke the `aocl uninstall` *<path_to_customplatform>* utility command to remove the current host computer drivers (for example, PCIe drivers). The Intel FPGA SDK for OpenCL uses these drivers to communicate with the FPGA board.
3. Uninstall the Custom Platform.
4. Unset the *LD_LIBRARY_PATH* environment variable.

A. Document Revision History of the Intel FPGA SDK for OpenCL Pro Edition Getting Started Guide

Document Version	Intel Quartus Prime Version	Changes
2018.01.14	18.1	• In Prerequisites for the Intel FPGA SDK for OpenCL Pro Edition on page 5, updated Microsoft* Visual Studio* version required to Visual Studio 2015 or later.
2018.12.24	18.1	• Removed references to passing channels by reference since it is no longer supported.
2018.12.10	18.1	• Added information about backwards compatibility for OpenCL board support packages (BSPs). The following topic new topics were added: — Intel FPGA SDK for OpenCL Pro Edition and BSP Backwards Compatibility on page 6 — Enabling Backwards Compatibility for an FPGA Board on Windows — Enabling Backwards Compatibility for an FPGA Board on Linux
2018.09.24	18.1	• In Intel FPGA SDK for OpenCL Pro Edition, the Intel FPGA SDK for OpenCL Offline Compiler has a new front end. For a summary of changes introduced by this new front end, see *Improved Intel FPGA SDK for OpenCL Compiler Front End* in the Intel FPGA SDK for OpenCL Pro Edition Release Notes. • Replaced altera.com links with intel.com links. For example, http://dl.altera.com/opencl/ is now http://fpgasoftware.intel.com/opencl/.
2018.05.04	18.0	• Removed Intel Quartus Prime Standard Edition software-related information. • In Overview of the Intel FPGA SDK for OpenCL Pro Edition Setup Process on page 7, updated the figure *FPGA Programming Overview* to include the `.aocr` file as an output and changed the `-c` flag to `-rtl` flag in the Attention note. • For Windows and Linux, updated the maximum number of devices (that is, acl number) from 32 to 128 in the following topics: — Querying the Device Name of Your FPGA Board — Programming the Flash Memory of an FPGA

Date	Version	Changes
December 2017	2017.12.08	• Updated the following sections to include the availability of Intel FPGA SDK for OpenCL installation packages that include Intel Code Builder: — Downloading the Intel FPGA SDK for OpenCL Pro Edition on page 10 — Downloading the Intel FPGA SDK for OpenCL Pro Edition on page 26 • Updated the following sections to include the Java SE version prerequisite required to run Intel Code Builder: — Prerequisites for the Intel FPGA SDK for OpenCL Pro Edition on page 5 — Installing the Intel FPGA SDK for OpenCL Pro Edition on page 27
November 2017	2017.11.06	• Rebranded the following: — ALTERAOCLSDKROOT to INTELFPGAOCLSDKROOT — *CL_CONTEXT_EMULATOR_DEVICE_ALTERA* to *CL_CONTEXT_EMULATOR_DEVICE_INTELFPGA* — Quartus Prime to Intel Quartus Prime — Arria 10 to Intel Arria 10 — USB Blaster to Intel FPGA Download Cable • In Intel® FPGA SDK for OpenCL™ Pro Edition Getting Started Guide on page 4, changed OpenCL Reference Pages as OpenCL 1.0 reference pages to improve clarity and added reference to Intel Arria 10 GX Application Note. • In Prerequisites for the Intel FPGA SDK for OpenCL: — Changed Intel preferred accelerator board to Intel FPGA Preferred Board for OpenCL. — Changed Microsoft Visual Studio version 2010 Professional as Microsoft Visual Studio Professional version 2010 or later. • In Contents of the Intel FPGA SDK for OpenCL: — In the sentence "The Intel FPGA SDK for OpenCL provides logic components, drivers, and SDK-specific libraries and files.", changed logical components to programs. — Under the Logic Components section, changed "host platform API and runtime API" as "host and runtime API". • In Building the Host Application on page 17, updated references to Microsoft Visual Studio 2015 as Microsoft Visual Studio. • In Executing an OpenCL Kernel on an FPGA on page 23, updated reference to Microsoft Visual Studio version 2010 Professional as Microsoft Visual Studio. • Removed topics on Licensing the Intel FPGA SDK for OpenCL in both OpenCL for Linux and Windows sections. • Added support for Intel Stratix® 10 devices in the following topics: — Downloading the Intel FPGA SDK for OpenCL Pro Edition on page 10 — Installing an FPGA Board on page 14 — Verifying Host Runtime Functionality via Emulation on page 16 — Creating the FPGA Hardware Configuration File of an OpenCL Kernel on page 19 — Downloading the Intel FPGA SDK for OpenCL Pro Edition on page 26 — Installing an FPGA Board on page 29 — Verifying Host Runtime Functionality via Emulation on page 31 — Creating the FPGA Hardware Configuration File of an OpenCL Kernel on page 33 • Implemented single dash and -option=<value> conventions in the following topics: — Overview of the Intel FPGA SDK for OpenCL Pro Edition Setup Process on page 7 — Creating the FPGA Hardware Configuration File of an OpenCL Kernel on page 19 — Compiling a Kernel for Emulation on page 17 • Removed references to *AOCL_BOARD_PACKAGE_ROOT* throughout the guide since it is deprecated.

continued...

Date	Version	Changes
		• Updated instances of `aocl install` to `aocl install <path_to_customplatform>`. • Updated instances of `aocl uninstall` to `aocl uninstall <path_to_customplatform>`. • In Overview of the Intel FPGA SDK for OpenCL Pro Edition Setup Process on page 7, added a note after the Installation Process Overview diagram about possible errors after implementing `aocl diagnose`. • In Updating the Hardware Image on the FPGA on page 21, added a note and related links to *Configuring the Intel Arria 10 GX FPGA Development Kit for the Intel FPGA SDK for OpenCL* application note.
May 2017	2017.05.05	• Rebranded the Altera Client Driver (ACD) to the FPGA Client Driver (FCD). • Updated the download instructions in *Downloading the Intel FPGA SDK for OpenCL* for Windows and Linux. • Added reminders that folder names where you uncompress downloaded OpenCL design examples must not contain spaces.
October 2016	2016.10.31	• Rebranded the Altera SDK for OpenCL to Intel FPGA SDK for OpenCL. • Rebranded the Altera Offline Compiler to Intel FPGA SDK for OpenCL Offline Compiler. • In *Installing an FPGA Board* for Windows and Linux, provided the following updates: — Noted that the SDK supports multi-Custom Platform installation. To use the SDK utilities on each board in a multi-Custom Platform installation, the *AOCL_BOARD_PACKAGE_ROOT* environment variable setting must correspond to the Custom Platform subdirectory of the associated board. — Noted that in a multi-Custom Platform system, the host program should use ACD to discover the boards instead of directly linking to the MMD libraries. • In *Building the Host Application* for Windows, outlined the prerequisite tasks for setting up ACD and ICD for use with Microsoft Visual Studio 2015 prior to building the host application.
May 2016	2016.05.02	• Replaced the lists of supported Windows and Linux versions to a link to the Operating System Support page on the Altera website. • Added the *%ALTERAOCLSDKROOT%*`\windows64\bin` setting to the list of Windows environment variables. • Corrected the Windows instructions for setting the *CL_CONTEXT_EMULATOR_DEVICE_ALTERA* variable for emulating multiple devices.
November 2015	2015.11.02	• Changed instances of *Quartus II* to *Quartus Prime*. • Added Windows 8.1 to supported Windows versions. • Modified download and installation instructions for the tar file that includes the AOCL, Quartus Prime software, and device support. • Deprecated and removed AOCL-only installation instructions because they are invalid for the current version. • Added instructions to verify host runtime functionality by emulating the hello_world example design. • Modified the figure *FPGA Programming Overview* to include emulation in the programming flow. • Updated uninstallation instructions.
May 2015	15.0.0	• Reorganized instructions into the following sections: — Getting Started with the AOCL on Windows — Getting Started with the AOCL on Linux
December 2014	14.1.0	• Reorganized information flow. • Updated Red Hat Enterprise Linux (RHEL) version support. • Included the *Contents of the AOCL* section.

continued...

Date	Version	Changes
		• Updated licensing instructions for the new Altera Software Development Kit (SDK) for OpenCL (AOCL) single license. • Updated board uninstallation instructions to include the `aocl uninstall` utility command. • Included information on the `init_opencl` script for setting environment variables. • Grouped software and board uninstallation instructions under *Uninstalling the Software and the FPGA Board*.
June 2014	14.0.0	• Updated the *Prerequisites* section. • Updated the figure *AOCL Installation Process Overview*. • Updated software download instructions. • Updated AOCL installation and uninstallation instructions for Windows. • For Linux systems: — Added the sections *Installing the AOCL on Linux Using RPM* and *Uninstalling the AOCL on Linux Using RPM*. — Updated the section *Installing the AOCL on Linux Using the GUI Installer*. • Added the section *Licensing the Software*. • Updated the section *Installing an FPGA Board* with updated instructions on querying your devices and running diagnostic tests. • Updated the section *Creating the FPGA Hardware Configuration File of an OpenCL Kernel*: — Updated path that you have to set for *AOCL_BOARD_PACKAGE_ROOT*. — Updated example AOC output for compiling `hello_world.cl` with the `-v` option. — Updated AOC output. • Added the section *Identifying the Device Name of Your FPGA Board*. • Modified instructions for building and running the host application with updated hello_world directory and file names. • Added the section *Uninstalling an FPGA Board*.
December 2013	13.1.1	• Updated the *Prerequisites* section to include a reminder to install Linux kernel source, headers, and GCC.
November 2013	13.1.0	• Reorganized information flow. • Updated the *Prerequisites* section. • Updated board installation instructions. • Updated software download instructions. • Inserted the section *Installing the AOCL and the Quartus II Software Together*. • Updated software installation and uninstallation instructions. • Inserted the following figures: — *AOCL Installation Process Overview* — *FPGA Programming Overview* • Removed the *Licensing* section. • Removed all board-specific installation and configuration instructions. • Changed example OpenCL application used to demonstrate kernel configuration and FPGA programming from moving_average to hello_world. • Inserted the section *Updating the Hardware Image on the FPGA*, which contained the updated flash programming instructions. • Removed the section *Installing the USB-Blaster Driver on Windows.* • Updated output from successful execution of hello_world kernel on FPGA for Windows and Linux systems. • Removed the figure *Contents of the Moving Average Example*. • Removed the figure *Opening host.sln in Visual Studio*.

continued...

Date	Version	Changes
June 2013	13.0 SP1.0	• Updated requisite Quartus II and AOCL software versions from 13.0 to 13.0 SP1. • Inserted the figure *A Correct Windows Device Manager After Complete Board Driver Installation for a BittWare Board*. • Updated the vendor and device IDs in the *Verifying the Functionality of the BittWare Board* section for Windows. • Updated the AOCL installation instructions for Linux systems that do not contain a `.cshrc` or a `.bashrc` file in the directory. • Updated path to the AOCL design examples. • Updated the figure *Contents of the Moving Average Example*. • Updated flash programming instructions.
May 2013	13.0.1	• Renamed the `OpenCL_SDK` folder or directory to `AOCL`. • Inserted warning about the AOCL installation dependency on `.cshrc` and `.bashrc` files for Linux systems. • Included reminder to BittWare board users about installing the BittWare development software. • Inserted warning about potential Jungo WinDriver installation failure for systems running on Windows 7. Included reinstallation instructions. • Inserted warnings about error messages displayed for `aocl` commands that have not been implemented for the BittWare FPGA board. • Inserted caution message about setting the environment variable *AOCL_BOARD_PACKAGE_ROOT*. • Updated board driver installation procedures for Windows and Linux systems. • Modified the path to the default location of the AOCL for Windows and Linux systems. • Modified the path name added to the *PATH* environment variable when installing the AOCL on Linux systems. The path name should be `$QUARTUS_ROOTDIR/bin` instead of `$QUARTUS_ROOTDIR/bin64`.
May 2013	13.0.0	• Updated installation and compilation procedures. • Incorporated licensing procedure. • Updated flash programming procedure and moved it to Appendix A. • Updated links to software and documentation download pages.
November 2012	12.1.0	Initial release.