Intel® FPGA SDK for OpenCL™ Pro Edition

Getting Started Guide

Updated for Intel® Quartus® Prime Design Suite: **21.1**

Subscribe
Send Feedback

UG-OCL001 | 2021.03.29
Latest document on the web: PDF | HTML

Contents

1. Intel® FPGA SDK for OpenCL™ Pro Edition Getting Started Guide..... 4
1.1. Prerequisites for the Intel FPGA SDK for OpenCL Pro Edition..... 5
1.1.1. GCC Requirement..... 7
1.2. Intel FPGA SDK for OpenCL Pro Edition and OpenCL BSP Backwards Compatibility..... 10
1.3. Contents of the Intel FPGA SDK for OpenCL Pro Edition..... 11
1.4. Overview of the Intel FPGA SDK for OpenCL Pro Edition Setup Process..... 12
2. Getting Started with the Intel FPGA SDK for OpenCL Pro Edition for Windows.....15
2.1. Downloading the Intel FPGA SDK for OpenCL Pro Edition 15
2.2. Installing the Intel FPGA SDK for OpenCL Pro Edition 16
2.3. Setting the Intel FPGA SDK for OpenCL Pro Edition User Environment Variables.....17
2.4. Verifying Software Installation..... 18
2.5. Installing an FPGA Board..... 19
2.5.1. Enabling Backwards Compatibility for an FPGA Board..... 20
2.6. Verifying Host Runtime Functionality via Emulation..... 21
2.6.1. Accessing an OpenCL Design Example..... 22
2.6.2. Compiling a Kernel for Emulation..... 22
2.6.3. Building the Host Application..... 22
2.6.4. Emulating Your OpenCL Kernel..... 24
2.7. Creating the FPGA Hardware Configuration File of an OpenCL Kernel..... 24
2.8. Updating the Hardware Image on the FPGA..... 27
2.8.1. Querying the Device Name of Your FPGA Board..... 27
2.8.2. Programming the Flash Memory of an FPGA..... 28
2.9. Executing an OpenCL Kernel on an FPGA..... 29
2.9.1. Running the Host Application..... 29
2.9.2. Output from Successful Kernel Execution..... 29
2.10. Uninstalling an FPGA Board 30
2.11. Uninstalling the Software..... 30
3. Getting Started with the Intel FPGA SDK for OpenCL Pro Edition for Linux..... 32
3.1. Downloading the Intel FPGA SDK for OpenCL Pro Edition..... 32
3.2. Installing the Intel FPGA SDK for OpenCL Pro Edition..... 33
3.3. Setting the Intel FPGA SDK for OpenCL Pro Edition User Environment Variables.....34
3.4. Verifying Software Installation..... 35
3.5. Installing an FPGA Board..... 35
3.5.1. Enabling Backwards Compatibility for an FPGA Board..... 37
3.6. Verifying Host Runtime Functionality via Emulation..... 37
3.6.1. Accessing an OpenCL Design Example..... 38
3.6.2. Compiling a Kernel for Emulation..... 38
3.6.3. Building the Host Application..... 38
3.6.4. Emulating Your OpenCL Kernel..... 39
3.7. Creating the FPGA Hardware Configuration File of an OpenCL Kernel..... 40
3.8. Updating the Hardware Image on the FPGA..... 42
3.8.1. Querying the Device Name of Your FPGA Board..... 43
3.8.2. Programming the Flash Memory of an FPGA..... 43
3.9. Executing an OpenCL Kernel on an FPGA..... 44
3.9.1. Running the Host Application..... 44
3.9.2. Output from Successful Kernel Execution..... 45

3.10. Uninstalling an FPGA Board..... 46
3.11. Uninstalling the Software..... 46

A. Intel FPGA SDK for OpenCL Pro Edition Getting Started Guide Archives.....47

B. Document Revision History of the Intel FPGA SDK for OpenCL Pro Edition Getting Started Guide..... 48

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.

Tip: If you are looking for instructions to get started with oneAPI toolkits, use the following links:

- Intel oneAPI Base Toolkit: Includes the Intel oneAPI DPC++/C++ Compiler and provides tools and libraries for developing high-performance, data-centric applications across diverse architectures.
- Intel FPGA Add-on for oneAPI Base Toolkit: Specialized toolkit for programming FPGAs.
- Intel DevCloud: A cloud-based sandbox to actively prototype and experiment with workloads on Intel hardware.
- Intel oneAPI Toolkits Installation Guides: Provides instructions for installing the toolkits.
- Get Started with Intel oneAPI Base Toolkit on the DevCloud: Provides instructions to get started with the Intel DevCloud development sandbox.
- Get Started with the Intel oneAPI Base Toolkit for Linux*: Provides Linux-specific getting started instructions.
- Get Started with the Intel oneAPI Base Toolkit for Windows*: Provides Windows-specific getting started instructions.

(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™.

Intel Corporation. All rights reserved. Agilex, Altera, Arria, Cyclone, eASIC, Intel, the Intel logo, MAX, Nios, Quartus and Stratix words and logos are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.
*Other names and brands may be claimed as the property of others.

ISO 9001:2015 Registered

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.
- If you want to use the Intel FPGA SDK for OpenCL Pro Edition with the Intel Stratix® 10 GX Development Kit, refer to *Intel Stratix 10 Development Kit Initialization guide* (bringup guide included in the `<path_to_BSP>/s10_ref/bringup` directory) and *Intel Stratix 10 GX FPGA Development Kit User Guide* 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
- Intel Stratix 10 GX FPGA Development Kit User Guide

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.
- For RAM requirements on the deployment system, refer to the Download Center for FPGAs.

  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 and Microsoft Visual C++ versions 2015 or later.

  Note: Intel FPGA SDK for OpenCL plug-ins support Windows* 10 and Windows* Server 2016 systems, with installed Microsoft Visual Studio* 2017 or 2019 version. For correct working of Visual Studio plug-ins, you should have installed extensions for C/C++ Development, for example, Desktop development with C++ and Microsoft .Net Framework 4.5 or higher. For more information about the IDE workflow on Microsoft Visual Studio, refer to FPGA Workflows in Microsoft Visual Studio.

- For Linux systems, use Eclipse 2019-12 or later and GCC 7.2.0. For more information about GCC, refer to GCC Requirement on page 7.

  Note: For correct working of Eclipse extensions, you should have installed C/C++ Development Tools and Java SE 11 or higher. For more information about the IDE workflow on Eclipse, refer to FPGA Workflows in Eclipse.

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. GCC Requirement

For Linux host systems, the Intel FPGA Emulation Platform for OpenCL software requires that you have at least GCC 7.2.0 on your system. GCC 7.2.0 provides the `libstdc++.so.6` shared library. Newer versions may work as well. Specifically, `libstdc++.so.6.0.24` library or later is required that defines `GLIBCXX_3.4.24` and `CXXABI_1.3.11` symbol versions.

If you do not have a sufficiently new GCC and `libstdc++.so.6` library installed, you may encounter issues when trying to run OpenCL host programs that target the Intel FPGA Emulation Platform for OpenCL software. If the correct version of `libstdc++.so` library is not found at run time, the call to `clGetPlatformIDs` function fails to load the emulation platform and returns `CL_PLATFORM_NOT_FOUND_KHR` (error code -1001). Depending on the version of `libstdc++.so` library found, the call to `clGetPlatformIDs` function may succeed, but a later call to the `clCreateContext` function might fail with `CL_DEVICE_NOT_AVAILABLE` (error code -2).

To verify the GCC version you are using, run the following command:

```
$ gcc -dumpversion
```

Ensure that the command reports version 7.2.0 or greater. If you do not have a sufficiently recent GCC, use the following instructions to install a newer GCC on your system.

Installing GCC 7

The process for installing the required GCC differs depending on the host operating system as described in the following sections.

Ubuntu*

If your Ubuntu version does not include GCC 7.x or later by default, you can obtain GCC 7.x by using the following commands:

```
$ sudo add-apt-repository ppa:ubuntu-toolchain-r/test
$ sudo apt-get update
$ sudo apt-get install gcc-7 g++-7
$ sudo update-alternatives --install /usr/bin/gcc gcc /usr/bin/gcc-7 60 \
--slave /usr/bin/g++ g++ /usr/bin/g++-7
$ sudo update-alternatives --config gcc
```

Note: Ensure that you select `gcc-7` for the `sudo update-alternatives` command. The `update-alternatives` utility can be used to swap between GCC versions as required.

CentOS and Red Hat Enterprise Linux* Version 7

Note: Unfortunately, installing the Developer Toolset* (for example, devtoolset-7), does not provide the desired result because it does not include a newer `libstdc++.so.6` library. The Developer Toolset* provides a custom version of the GCC that uses the older `libstdc++.so.6` library originally installed on the system.

For CentOS and Red Hat Enterprise Linux version 7, Intel recommends that you compile GCC from source and install it using the following instructions:

1. Download the tarball of the GCC version you want to install.

```
wget http://www.netgull.com/gcc/releases/gcc-7.2.0/gcc-7.2.0.tar.gz
```

2. Unpack the tar archive and change the current working directory.

```
tar zxf gcc-7.2.0.tar.gz
cd gcc-7.2.0
```

3. Install the `bzip2` package.

```
yum -y install bzip2
```

4. Run the `download_prerequisites` script to download some prerequisites required by the GCC.

   Note: You must run the script from the top-level of the GCC source tree.

```
./contrib/download_prerequisites
```

5. Once the prerequisites are downloaded, execute the following command to start configuring the GCC build environment:

```
./configure --disable-multilib --enable-languages=c,c++
```

6. Run the following command to compile the source code. Compilation may take a few hours to complete.

```
make -j 4
make install
```

7. Refer to Verifying Your GCC Installation on page 8 section for instructions to verify if your GCC installation was successful.

SUSE Enterprise Linux* 12

For SUSE, Intel recommends that you compile GCC from source and install it by following the instructions provided for CentOS and Red Hat Enterprise Linux.

Verifying Your GCC Installation

To verify your GCC installation, ensure the following:

- Shell can find the correct GCC binaries.
- The newer GCC is using correct shared libraries (in particular, `libstdc++.so.6`).

This depends on how you installed the GCC and set the $PATH and $LD_LIBRARY_PATH environment variables. You might have installed the GCC binaries and libraries to the default locations for your operating system.

Verifying the Correct GCC Binaries are Used

Perform these steps:

1. Verify if your newly-installed GCC binaries are being used.

```
$ which gcc
```

2. Ensure the output points to the correct GCC.

```
$ gcc –dumpversion
```

3. Ensure GCC reports a version of 7.2.0 or higher.

If the above commands do not report the correct GCC version, modify your $PATH environment variable to point to the location of the newly installed GCC binaries.

For Ubuntu, the default install location for GCC binaries is `/usr/bin;` and libraries is `/usr/lib64`. However, binaries are versioned (for example, `gcc-7`). Use the `update-alternatives` utility to ensure `gcc` is a symbolic link to the desired GCC version. On Ubuntu systems, the $PATH and $LD_LIBRARY_PATH environment variables need not be updated because the shell and dynamic linker should automatically search in the correct places for the binaries and libraries, respectively.

Tip: You can place the following command in your shell startup script or an environment setup script so that the development environment is automatically set up when you enter it.

```
$ export PATH=</path/to/new/gcc>/bin:$PATH
```

To verify that your $PATH environment variable has been modified appropriately, use the following command:

```
$ echo $PATH
```

Check the output to ensure that the path to your newly installed GCC is part of your $PATH. Repeat these steps to ensure the correct GCC binaries are being used.

Verifying the Use of Correct Shared Libraries

To verify that the new GCC libraries are being used, check that a test compile links against the correct `libstdc++.so.6` library version by using the following steps:

```
$ echo "int main() { return 0; }" > test.cpp
$ g++ -Wl,--no-as-needed test.cpp -o test
$ ldd test
        linux-vdso.so.1 =>  (0x00007fff6df35000)
        libstdc++.so.6 => /usr/lib64/libstdc++.so.6 (0x0000003451600000)
        libm.so.6 => /lib64/libm.so.6 (0x0000003446a00000)
        libgcc_s.so.1 => /lib64/libgcc_s.so.1 (0x000000344d600000)
        libc.so.6 => /lib64/libc.so.6 (0x0000003446600000)
        /lib64/ld-linux-x86-64.so.2 (0x000055ff56398000)
$ strings /usr/lib64/libstdc++.so.6 | grep GLIBCXX_3.4.24
GLIBCXX_3.4.24
```

Use the output from the `ldd` command to determine the path to the `libstdc++.so.6` library that you need to use with the `strings` command (see highlighted strings; your output may be different). Ensure that the `strings` command returns `GLIBCXX_3.4.24`.

Alternatively, the `libstdc++.so.6` file is often a symbolic link to the actual library, which has the version number appended. Therefore, instead of calling the `strings` command on `libstdc++.so.6` library, you can use the following `readlink` command to verify `libstdc++.so.6` points to a sufficiently new version of `libstdc++`:

```
$ readlink /usr/lib64/libstdc++.so.6
libstdc++.so.6.0.24
```

Use the output from the `ldd` command (executed earlier) to determine the path to `libstdc++.so.6` that you need to use with the `readlink` command. Ensure that `readlink` command reports that your `libstdc++.so.6` is a symbolic link to `libstdc++.so.6.0.24`. Versions newer than 6.0.24 may also work.

If the above commands do not report the correct shared libraries are being used, you might need to modify your $LD_LIBRARY_PATH environment variable to point to the location of the newly installed GCC libraries.

Tip: You can place the following command in your shell startup script or an environment setup script so that the development environment is automatically set up when you enter it.

```
$ export LD_LIBRARY_PATH=</path/to/new/gcc>/lib64:$LD_LIBRARY_PATH
```

To verify that your $LD_LIBRARY_PATH environment variable has been modified appropriately, run the following command:

```
$ echo $LD_LIBRARY_PATH
```

Check the output to ensure the path to your newly installed GCC libraries are included in $LD_LIBRARY_PATH. Repeat the above steps to ensure the correct GCC libraries (in particular `libstdc++.so.6`) are being used.

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

The Intel FPGA SDK for OpenCL Pro Edition Version 21.1 compiler is compatible with OpenCL board support package (BSP) Version 20.2 and Intel Acceleration Stack Version 2.0.1 and Version 1.2.1. This backwards compatibility allows you to use older OpenCL BSPs with newer compilers. However, some newer OpenCL compiler features might not be available to use with older OpenCL BSPs.

Note: Starting from the 20.3 release, support for Windows and Linux OpenCL BSPs is removed. Use version 20.2 or older OpenCL BSPs available at Download Center for FPGAs as a reference. If you want to migrate your OpenCL BSP to a newer version, follow the recommended steps provided in the Reference Platform Porting Guides available under Intel FPGA SDK for OpenCL documentation.

To use an older OpenCL 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 OpenCL BSP. For example, to use a Version 20.2 BSP with Intel FPGA SDK for OpenCL Pro Edition Version 21.1, you need Intel Quartus Prime Pro Edition Version 20.2.

Related Information

- Enabling Backwards Compatibility for an FPGA Board on Windows on page 20
- Enabling Backwards Compatibility for an FPGA Board on Linux on page 37

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

The Intel FPGA SDK for OpenCL Pro Edition provides programs 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\custom_platform_toolkit\tests`	`board/custom_platform_toolkit/tests`	The path to the Intel FPGA SDK for OpenCL Pro Edition kernel code files to test platforms. *Note:* Starting from the 20.3 release, support for Windows and Linux OpenCL BSPs is removed. Use version 20.2 or older OpenCL BSPs available at Download Center for FPGAs as a reference. If you want to migrate your OpenCL BSP to a newer version, follow the recommended steps provided in the Reference Platform Porting Guides available under Intel FPGA SDK for OpenCL documentation.
`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.

continued...

Windows Folder	Linux Directory	Description
`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

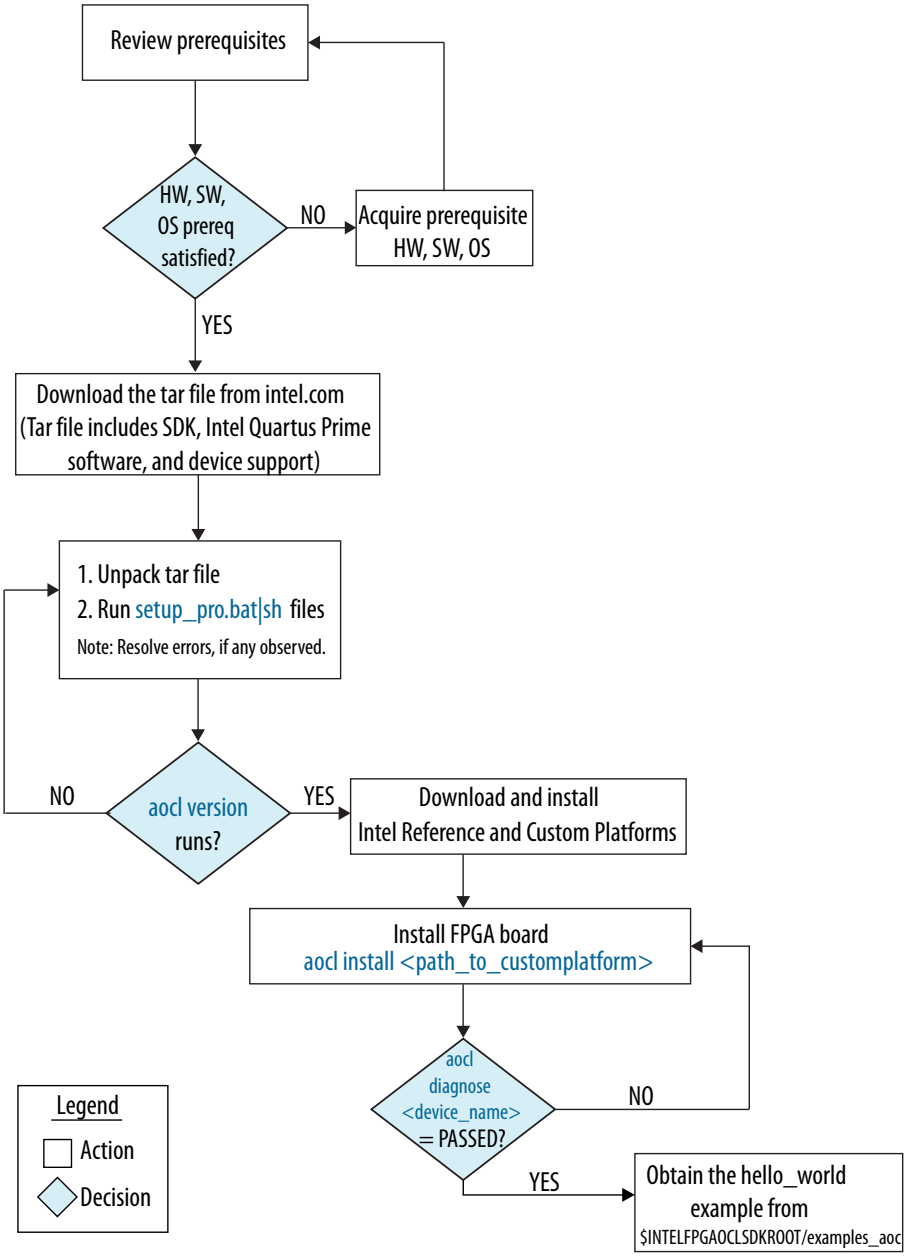
You can refer to example OpenCL applications from the `examples_aoc` OpenCL design examples directory under `INTELFPGAOCLSDKROOT`.

1.4. 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



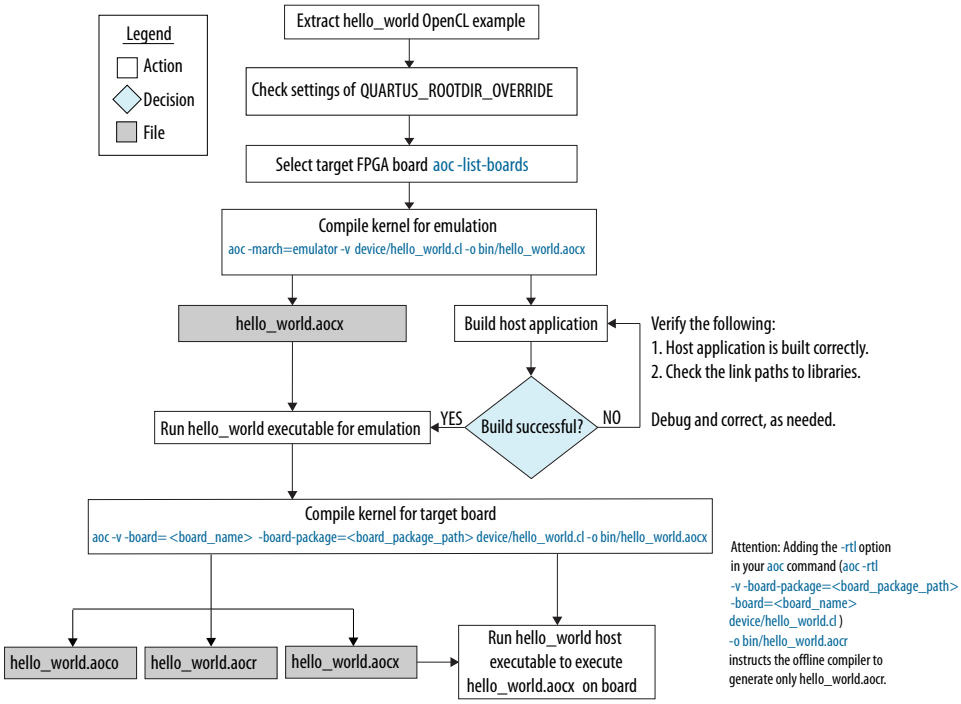
Attention: For possible errors after implementing `aocl diagnose` utility, refer to Possible Errors After Running the `diagnose` Utility section.

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 27

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.

Tip: If you are looking for instructions to get started with oneAPI toolkits on Windows, refer to the following links:

- Intel oneAPI Toolkits Installation Guide for Windows*
- Get Started with the Intel oneAPI Base Toolkit for Windows*

1. Downloading the Intel FPGA SDK for OpenCL Pro Edition on page 15
2. Installing the Intel FPGA SDK for OpenCL Pro Edition on page 16
3. Setting the Intel FPGA SDK for OpenCL Pro Edition User Environment Variables on page 17
4. Verifying Software Installation on page 18
5. Installing an FPGA Board on page 19
6. Verifying Host Runtime Functionality via Emulation on page 21
7. Creating the FPGA Hardware Configuration File of an OpenCL Kernel on page 24
8. Updating the Hardware Image on the FPGA on page 27
9. Executing an OpenCL Kernel on an FPGA on page 29
10. Uninstalling an FPGA Board on page 30
11. Uninstalling the Software on page 30

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:

Intel Corporation. All rights reserved. Agilex, Altera, Arria, Cyclone, eASIC, Intel, the Intel logo, MAX, Nios, Quartus and Stratix words and logos are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.
*Other names and brands may be claimed as the property of others.

ISO 9001:2015 Registered

https://fpgasoftware.intel.com/opencl/

2. Select the Pro edition.
3. Select the software version. The default selection is the current version.
4. On the **Windows SDK** tab, select the **Intel FPGA SDK for OpenCL (includes Intel Quartus Prime Pro Edition software and devices)** package.

   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.

   Click **More** beside **Download and install instructions** if you want to see the download and installation instructions.
5. 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 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*.

Installing the Microsoft Visual Studio* Plug-in

After installing the Intel FPGA SDK for OpenCL, perform these steps to install the Microsoft Visual Studio* plug-in:

1. Navigate to the *$INTELFPGAOCLSDKROOT*/ide_plugins directory.
2. Execute the plugins_install.bat script.
3. Follow the prompts to indicate which version of the Microsoft Visual Studio* plug-in you want to install.

Related Information

Setting the Intel FPGA SDK for OpenCL Pro Edition User Environment Variables on page 17

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**.
  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(R) 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 Intel Reference Platform or 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.

   Note: Starting from 20.3 release, support for Windows OpenCL BSPs is removed. Use 20.2 or older OpenCL BSPs available at Download Center for FPGAs as a reference. If you want to migrate your OpenCL BSP to a newer version, follow the recommended steps provided in the Reference Platform Porting Guides available under Intel FPGA SDK for OpenCL documentation.
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. 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.
5. Add the paths to the Custom Platform libraries (for example, path to the MMD library of the board support package resembles `<path_to_customplatform>/windows64/bin`) 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>* installs both the FCD and 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

- If the system already has the driver installed and you need to install FCD without the administrative rights, you can invoke the `aocl install` command with the flag `-fcd-only` as shown below and follow the prompt for FCD installation:

```
aocl install <path_to_customplatform> -fcd-only
```

7. Query a list of FPGA devices installed in your machine by invoking 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. Verify the successful installation of the FPGA board by invoking 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 27
- Intel FPGA SDK for OpenCL FPGA Platforms
- Setting the Intel FPGA SDK for OpenCL Pro Edition User Environment Variables on page 17

2.5.1. Enabling Backwards Compatibility for an FPGA Board

To enable a board with a downlevel OpenCL BSP, you need a version of Intel Quartus Prime Pro Edition with the same version number as the OpenCL BSP version.

For example, you can use Version 20.2 OpenCL BSP with Intel FPGA SDK for OpenCL Version 21.1 and Intel Acceleration Stack Version 2.0.1 and Version 1.2.1.

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 OpenCL 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 OpenCL BSP needs.

Also, install your FPGA board or boards following the instructions in Installing an FPGA Board on page 19.

To enable a board with a backwards compatible OpenCL BSP:

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

   For example, if you have Version 20.2 OpenCL BSP, QUARTUS_ROOTDIR_OVERRIDE must point at Intel Quartus Prime Pro Edition Version 20.2:

   ```
   set QUARTUS_ROOTDIR_OVERRIDE=<path_to_quartus_install>\quartus
   ```

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

   For example, to point to a Version 20.2 OpenCL BSP for the Intel Arria 10 GX FPGA Development Kit Reference Platform, install the board with the following command:

   ```
   aocl install <YOUR_PATH_TO_V20.2_OpenCL_BSP>\a10_ref
   ```

After you complete these steps, Intel SDK for OpenCL commands like `aocl diagnose` should work correctly with your backlevel OpenCL 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. Add the path to the `LINK.exe` file in Microsoft Visual Studio to the *PATH* user environment variable setting.
2. Add the path to the Microsoft compilation time libraries in Microsoft Visual Studio to the *LIB* user environment variable setting.

1. Accessing an OpenCL Design Example on page 22
2. Compiling a Kernel for Emulation on page 22
3. Building the Host Application on page 22
4. Emulating Your OpenCL Kernel on page 24

2.6.1. Accessing an OpenCL Design Example

The `INTELFPGAOCLSDKROOT\examples_aoc` directory contains sample applications of varying complexities that you can copy to your local and run on your FPGA.

The following instructions are for accessing the Hello World design:

1. Access the `hello_world` folder.
2. Copy it to a folder that you have write access for.

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

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 device/hello_world.cl -o bin/
  hello_world.aocx
  ```

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

2.6.3. Building the Host Application

The *<local_path_to exm_opencl_hello_world>*`\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. The ICD driver links the host against the `OpenCL.dll`, which requires the FPGA Client Driver (FCD) installed. For installing FCD, refer to Installing an FPGA Board on page 19.

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 using the `aocl diagnose -icd-only` command. If both FCD and ICD are installed correctly, you should see the following output:

```
ICD System Diagnostics
-------------------------------------------------------------------------
Using the following location for ICD installation:
    HKEY_LOCAL_MACHINE\Software\Khronos\OpenCL\Vendors

Found 1 icd entry at that location:
    <INTELFPGAOCLSDKROOT>\windows64\bin\alteracl_icd.dll   REG_DWORD   0x0

Checking validity of found icd entries:
    <INTELFPGAOCLSDKROOT>\windows64\bin\alteracl_icd.dll   REG_DWORD   0x0
is
correctly registered on the system
```

```
Using the following location for fcd installations:
    HKEY_LOCAL_MACHINE\Software\Intel\OpenCL\Boards

Found 1 fcd entry at that location:
    <path_to_OpenCL_BSP>\a10_ref\windows64\bin\altera_a10_ref_mmd.dll
REG_DWORD 0x0

Checking validity of found fcd entries:
    <path_to_OpenCL_bsp>\a10_ref\windows64\bin\altera_a10_ref_mmd.dll
REG_DWORD 0x0 is correctly registered on the system

Number of Platforms = 1
    1. Intel(R) FPGA SDK for OpenCL(TM)             | Intel(R) Corporation
|
OpenCL 1.0 Intel(R) FPGA SDK for OpenCL(TM), Version  <version_number>
---------------------------------------------------------------------
ICD diagnostics PASSED
---------------------------------------------------------------------
```

2. If neither the FCD nor ICD is setup correctly, refer to the *Installing an FPGA Board* for FCD setup, and refer to the *Accessing Custom Platform-Specific Functions* and *Linking Your Host Application to the Khronos ICD Loader Library* sections of the *Intel FPGA SDK for OpenCL Pro Edition Programming Guide* for more information.
3. Link the host application to the `OpenCL.lib` library.
   a. Under the Solution, right-click the host application name and select **Properties**, and then 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 <*local_path_to_exm_opencl_hello_world*>`\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 is in the <*local_path_to_exm_opencl_hello_world*>`\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 Your Host Application to the Khronos ICD Loader Library

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. To run the host application for emulation, first define the number of emulated devices by invoking the `set CL_CONFIG_CPU_EMULATE_DEVICES=<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. By default, the emulator platform provides a single emulator device. Therefore, if your host code requires only a single device, you do not need to set the `CL_CONFIG_CPU_EMULATE_DEVICES` environment variable. The host application for the `hello_world` design example only requires a single device, so you may skip setting this environment variable.

   Remember: When the host is compiled and running for the emulator platform, only the emulated devices are available, that is, access to all physical boards is disabled.
2. After you run the host application, unset the `CL_CONFIG_CPU_EMULATE_DEVICES` variable by invoking the `set CL_CONFIG_CPU_EMULATE_DEVICES=` command.

Upon successful emulation of your kernel, you should see the following output:

```
...
...
...

Using AOCX: hello_world.aocx

Kernel initialization is complete.

Launching the kernel...

Thread #2: Hello from Intel FPGA OpenCL Compiler!

Kernel execution is complete.
```

Related Information

Running the Host Application on page 29

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

Access the hello_world OpenCL design example from the `INTELFPGAOCLSDKROOT\examples_aoc` folder. . 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.

Note: The Intel FPGA SDK for OpenCL Offline Compiler compiles an OpenCL kernel for a target FPGA board. The Intel FPGA SDK for OpenCL compiles for the installed board package. If no board package is installed, use the `-board-package=<board_package_path>` option to point to the OpenCL BSP, otherwise the Intel FPGA SDK for OpenCL throws an error. . Refer to the following steps for more details. To install an 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. If no OpenCL BSP is installed, `No board support packages installed` message is displayed. Use the `-board-package=<board_package_path>` option to point to the OpenCL BSP.

   If you have multiple OpenCL BSPs installed, you can compile your kernel with the board variant from a specific Custom Platform by including `-board-package=<custom_platform_path>` option with `-board=<board_name>`. At the command prompt, invoke the following command:

   ```
   aoc -board-package=<board_package_path> -board=<board_name> <kernel.cl>
   ```

   The Intel FPGA SDK for OpenCL Offline Compiler compiles the kernel with the board specified in the <board_package_path>. To list Custom Platforms available in the system, include the `-list-board-packages` option in the aoc command.

   For more information about `-list-boards`, `-board-package`, and `-list-board-packages` options of the aoc command, refer to the *Listing the Available FPGA Boards in Your Custom Platform (`-list-boards` and `-list-board-packages`)* 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-package=<board_package_path> -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 `.aocx` file.

Tip:
- If you have only one FPGA board installed, `-board=<board_name>` is optional.
- If you have only one OpenCL BSP installed, `-board-package=<board_package_path>` 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 and Custom Platform (-board=<board_name>) and (-board-package=<board_package_path>)* 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 and Custom Platforms (-list-boards and -list-board-packages)
- Compiling a Kernel for a Specific FPGA Board and Custom Platform (-board=<board_name>) and (-board-package=<board_package_path>)

- Generating Compilation Progress Report (-v)
- Specifying the Name of an Intel FPGA SDK for OpenCL Offline Compiler Output File (-o <filename>)
- Setting the Intel FPGA SDK for OpenCL Pro Edition User Environment Variables on page 17
- Installing an FPGA Board on page 19

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 <board_package_path>/<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 fails to enumerate the PCIe endpoint, or the driver does 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. 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.

   For more information about compiling an `aocx` file, refer to Creating the FPGA Hardware Configuration File of an OpenCL kernel in the *Intel FPGA SDK for OpenCL Getting Started Guide*.

3. 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 re-enumerate 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
- Setting the Intel FPGA SDK for OpenCL Pro Edition User Environment Variables on page 17

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 22

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 `<local_path_to_exm_opencl_hello_world>\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 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. Follow 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.

   Remember: You need administrative rights to uninstall the Custom Platform. If you want to keep the driver while removing the installed FCD, you can invoke the `aocl uninstall` command with the flag `-fcd-only` as shown below and follow the prompt for FCD uninstall:

   ```
   aocl uninstall <path_to_customplatform> -fcd-only
   ```
3. Uninstall the Custom Platform.
4. Modify *PATH* environment variable to remove paths that have the *INTELFPGAOCLSDKROOT* environment variable.

2.11. 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.

Note: Before uninstalling the Intel FPGA SDK for OpenCL Pro Edition for Windows, you must first uninstall the FPGA board. Refer to Uninstalling an FPGA Board on page 30 for more information.

1. From the Windows Start Menu shortcut, navigate to the `intelFPGA_pro \<version> <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.

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.

Tip: If you are looking for instructions to get started with oneAPI toolkits on Linux, refer to the following guides:

- Intel oneAPI Toolkits Installation Guide for Linux*
- Get Started with the Intel oneAPI Base Toolkit for Linux*

1. Downloading the Intel FPGA SDK for OpenCL Pro Edition on page 32
2. Installing the Intel FPGA SDK for OpenCL Pro Edition on page 33
3. Setting the Intel FPGA SDK for OpenCL Pro Edition User Environment Variables on page 34
4. Verifying Software Installation on page 35
5. Installing an FPGA Board on page 35
6. Verifying Host Runtime Functionality via Emulation on page 37
7. Creating the FPGA Hardware Configuration File of an OpenCL Kernel on page 40
8. Updating the Hardware Image on the FPGA on page 42
9. Executing an OpenCL Kernel on an FPGA on page 44
10. Uninstalling an FPGA Board on page 46
11. Uninstalling the Software on page 46

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:

Intel Corporation. All rights reserved. Agilex, Altera, Arria, Cyclone, eASIC, Intel, the Intel logo, MAX, Nios, Quartus and Stratix words and logos are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.
*Other names and brands may be claimed as the property of others.

ISO 9001:2015 Registered

https://fpgasoftware.intel.com/opencl/

2. Select the Pro edition.
3. Select the software version. The default selection is the current version.
4. On the **Linux SDK** tab, select the **Intel FPGA SDK for OpenCL (includes Intel Quartus Prime Pro Edition software and devices)** package.

   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.

   Click **More** beside **Download and install instructions** if you want to see the download and installation instructions.
5. 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 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.

Important: Run this script with `sudo` privileges to ensure the Installable Client Driver (ICD) is setup successfully.

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

   Note:
   - The installer sets the environment variable *INTELFPGAOCLSDKROOT* to point to the path of the software installation.
   - If there are dependency issues, use the `–nodeps` option.

   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*.

Installing the Eclipse Plug-in

After installing the Intel FPGA SDK for OpenCL, perform these steps to install the Eclipse plug-in:

1. Navigate to the *$INTELFPGAOCLSDKROOT*`/ide_plugins` directory.
2. Execute the `plugins_install.sh` script.
3. Follow the prompts to indicate where your installation of Eclipse is.

Note: If the installation indicates that it finished successfully but you do not see the plug-ins in the IDE, ensure the Java Runtime version you have installed is supported by the Eclipse IDE version, and that you have all of the prerequisites described in Prerequisites for the Intel FPGA SDK for OpenCL Pro Edition on page 5.

Related Information

Setting the Intel FPGA SDK for OpenCL Pro Edition User Environment Variables on page 34

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 variable in your `.bashrc` or `.cshrc` file, the settings are applied to all subsequent sessions. 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 automatically initialize the environment at the beginning of each session, add the following export statements to your `.bashrc` or `.cshrc` file:

  `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> Intel(R) 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 Intel Reference Platform or the Custom Platform from your board vendor. Most Custom Platform installers require administrator privileges. Visit the Download Center for FPGAs site for downloading the Intel Reference Platform. 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.

Note: Starting from 20.3 release, support for Linux OpenCL BSPs is removed. Use 20.2 or older OpenCL BSPs available at Download Center for FPGAs as a reference. If you want to migrate your OpenCL BSP to a newer version, follow the recommended steps provided in the Reference Platform Porting Guides available under Intel FPGA SDK for OpenCL documentation.

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. 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, the path to the MMD library resembles `<path_to_customplatform>/linux64/lib`) 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. Invoke the command `aocl install <path_to_customplatform>` at a command prompt.

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

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

   If the system already has the drivers installed, and you need to install FCD without the `root` privilege, you can perform the following:

   a. Export the *ACL_BOARD_VENDOR_PATH=<path_to_install_fcd>* environment variable to specify the installation directory of FCD other than the default location.

      Note: If you set the *ACL_BOARD_VENDOR_PATH* environmental variable, set it every time after you enter the OpenCL development environment via the `source init_opencl.sh` file.

   b. Invoke the `aocl install` command with the flag `-fcd-only` as shown below and follow the prompt for FCD installation:

      ```
      aocl install <path_to_customplatform> -fcd-only
      ```

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. 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 OpenCL BSP, you need a version of Intel Quartus Prime Pro Edition with the same version number as the OpenCL BSP version. For example, you can use Version 20.2 OpenCL BSP with Intel FPGA SDK for OpenCL Version 21.1 and Intel Acceleration Stack Version 2.0.1 and Version 1.2.1.

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 OpenCL 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 OpenCL BSP needs.

Also, install your FPGA board or boards following the instructions in Installing an FPGA Board on page 35.

To enable a board with a backwards compatible OpenCL BSP:

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

   For example, if you have Version 20.2 OpenCL BSP, `QUARTUS_ROOTDIR_OVERRIDE` must point at Intel Quartus Prime Pro Edition Version 20.2:

   ```
   export QUARTUS_ROOTDIR_OVERRIDE=<path_to_quartus_install>/quartus
   ```

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

   For example, to point to a Version 20.2 OpenCL BSP for the Intel Arria 10 GX FPGA Development Kit Reference Platform, install the board as follows:

   ```
   aocl install <YOUR_PATH_TO_V20.2_OPENCL_BSPs>/a10_ref
   ```

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

Important: Before you proceed, review Emulating and Debugging Your OpenCL Kernel topic of the *Intel FPGA SDK OpenCL Pro Edition: Programming Guide*.

1. 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.

1. Accessing an OpenCL Design Example on page 38
2. Compiling a Kernel for Emulation on page 38
3. Building the Host Application on page 38
4. Emulating Your OpenCL Kernel on page 39

3.6.1. Accessing an OpenCL Design Example

The `$INTELFPGAOCLSDKROOT/examples_aoc` contains sample applications of varying complexities that you can copy to your local and run on your FPGA.

The following instructions are for accessing the Hello World design:

1. Access the `hello_world` directory.
2. Copy it to a directory to which you have write access.

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

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 device/hello_world.cl -o bin/hello_world.aocx
  ```

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

3.6.3. Building the Host Application

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

To build the host application, perform the following tasks:

1. If you did not run the `aocl install` command when performing the steps from Installing an FPGA Board on page 35, you must execute the following command:

   ```
   aocl install -fcd-only <board_package_path>/a10_ref
   ```

The ICD driver links the host against the `libOpenCL.so`, which requires the FPGA Client Driver (FCD) installed. For installing FCD, refer to Installing an FPGA Board on page 35.

Note: If you do not have the `root` privilege, refer to Installing an FPGA Board on page 35.

2. Navigate to the `hello_world` directory.
3. Invoke the `make` command.
   The `hello_world` executable is generated in the `bin` sub-directory.
4. If you performed the step 1, then you must execute the following command:

```
aocl uninstall -fcd-only <board_package_path>/a10_ref
```

3.6.4. Emulating Your OpenCL Kernel

To emulate your OpenCL kernel, run the emulation `.aocx` file on the platform on which you built your kernel. The OpenCL Emulator uses a different OpenCL platform than when targeting FPGA hardware.

To emulate your kernel, perform the following steps:

1. To run the host application for emulation, invoke the `env CL_CONFIG_CPU_EMULATE_DEVICES=<number_of_devices> <host_application_filename>` command.
   This command specifies the number of identical emulation devices that the Emulator needs to provide. By default, the emulator platform provides a single emulator device. Therefore, if your host code requires only a single device, you do not need to set the `CL_CONFIG_CPU_EMULATE_DEVICES` environment variable.

   For the `hello_world` design example, run `bin/host -emulator`. The `-emulator` argument informs the host program to select the Intel FPGA Emulation Platform for OpenCL.

Upon successful emulation of your kernel, you see the following output:

```
...
...
...

Using AOCX: hello_world.aocx

Kernel initialization is complete.

Launching the kernel...

Thread #2: Hello from Intel FPGA OpenCL Compiler!

Kernel execution is complete.
```

Note:

- The emulator in Pro Edition is built with GCC 7.2.0 as part of the offline compiler. When executing the host program for an emulated OpenCL device, the version of `libstdc++.so` must be at least that of GCC 7.2.0. In other words, the *LD_LIBRARY_PATH* environment variable must ensure that the correct version of `libstdc++.so` is found.

  If the correct version of `libstdc++.so` is not found, the call to `clGetPlatformIDs` function fails to load the FPGA emulator platform and returns `CL_PLATFORM_NOT_FOUND_KHR (error code -1001)`. Depending on which version of `libstdc++.so` is found, the call to `clGetPlatformIDs` may succeed, but a later call to the `clCreateContext` function may fail with `CL_DEVICE_NOT_AVAILABLE (error code -2)`.

  If `LD_LIBRARY_PATH` does not point to a sufficiently new `libstdc++.so`, use the following syntax to invoke the host program:

  ```
  env LD_LIBRARY_PATH=<path to sufficiently new libstdc+
  +.so>:$LD_LIBRARY_PATH <host> [host arguments]
  ```

- To enable debugging of kernel code, optimizations are disabled by default for the FPGA emulator. This can lead to sub-optimal execution speed when emulating kernel code.

  You can pass the `-g0` flag to the `aoc` compile command to disable debugging and enable optimizations. This enables faster emulator execution.

Related Information

Running the Host Application on page 44

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

Copy the hello_world OpenCL design example from the `$INTELFPGAOCLSDKROOT/examples_aoc` directory. 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.

Note: The Intel FPGA SDK for OpenCL Offline Compiler compiles an OpenCL kernel for a target FPGA board. The Intel FPGA SDK for OpenCL compiles for the installed board package. If you have not installed the FPGA board, refer to *Installing an FPGA Board* for more instructions. If you do not install the board package, use the `-board-package=`*<board_package_path>* option to point to the OpenCL BSP, otherwise the Intel FPGA SDK for OpenCL throws an error. Refer to the following steps for more details.

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. If no OpenCL BSP is installed, `No board support packages installed` message is displayed. Use the `-board-package=<board_package_path>` option to point to the OpenCL BSP.

   If you have multiple OpenCL BSPs installed, you can compile your kernel with the board variant from a specific Custom Platform by including `-board-package=<custom_platform_path>` option with `-board=<board_name>`. At the command prompt, invoke the following command:

   ```
   aoc -board-package=<board_package_path> -board=<board_name> kernel.cl
   ```

   The Intel FPGA SDK for OpenCL Offline Compiler compiles the kernel with the board specified in the <board_package_path>. To list Custom Platforms available in the system, include the `-list-board-packages` option in the `aoc` command.

   For more information about `-list-boards`, `-board-package`, and `-list-board-packages` options of the `aoc` command, refer to the *Listing the Available FPGA Boards in Your Custom Platform (-list-boards and -list-board-packages)* 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-package=<board_package_path> -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 `.aocx` file.

   Tip:
   - If you have only one FPGA board installed, `-board=<board_name>` is optional.
   - If you have only one OpenCL BSP installed, `-board-package=<board_package_path>` 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 and Custom Platform (-board=<board_name>) and (-board-package=<board_package_path>)* 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 35
- Listing the Available FPGA Boards and Custom Platforms (-list-boards and -list-board-packages)
- Compiling a Kernel for a Specific FPGA Board and Custom Platform (-board=<board_name>) and (-board-package=<board_package_path>)
- 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 <board_package_path>/<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 fails to enumerate the PCIe endpoint, or the driver does 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. 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.

   For more information about compiling an `aocx` file, refer to Creating the FPGA Hardware Configuration File of an OpenCL Kernel in the *Intel FPGA SDK for OpenCL Pro Edition: Getting Started Guide*.
3. 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 re-enumerate the PCIe endpoint. Intel recommends that you power cycle the complete host system after programming the flash memory.

Related Information

Setting the Intel FPGA SDK for OpenCL Pro Edition User Environment Variables on page 34

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 38

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 <*local_path_to_exm_opencl_hello_world*>/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 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.

   Remember:
   - You need `root` privileges to uninstall the Custom Platform. If you want to keep the driver while removing the installed FCD, you can invoke the `aocl uninstall` command with the flag `-fcd-only` as shown below and follow the prompt for FCD uninstall:

     ```
     aocl uninstall <path_to_customplatform> -fcd-only
     ```

   - For Linux systems, if you had installed the FCD to a specific directory, then prior to uninstalling, you need to ensure that you have set an environment variable ACL_BOARD_VENDOR_PATH that points to that specific FCD installation directory.
3. Uninstall the Custom Platform.
4. Unset the *LD_LIBRARY_PATH* environment variable.

3.11. 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.

Note: Before uninstalling the Intel FPGA SDK for OpenCL Pro Edition for Linux, you must first uninstall the FPGA board. Refer to Uninstalling an FPGA Board on page 46 for more information.

1. Remove the software package by running 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. This uninstalls the SDK.
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.

A. Intel FPGA SDK for OpenCL Pro Edition Getting Started Guide Archives

If the table does not list a software version, the user guide for the previous software version applies.

Intel Quartus Prime Version	User Guide
20.4	Intel FPGA SDK for OpenCL Pro Edition Getting Started Guide
20.3	Intel FPGA SDK for OpenCL Pro Edition Getting Started Guide
20.2	Intel FPGA SDK for OpenCL Pro Edition Getting Started Guide
19.4	Intel FPGA SDK for OpenCL Pro Edition Getting Started Guide
19.3	Intel FPGA SDK for OpenCL Pro Edition Getting Started Guide
19.1	Intel FPGA SDK for OpenCL Pro Edition Getting Started Guide
18.1	Intel FPGA SDK for OpenCL Pro Edition Getting Started Guide
18.0	Intel FPGA SDK for OpenCL Pro Edition Getting Started Guide
17.1	Intel FPGA SDK for OpenCL Getting Started Guide
17.0	Intel FPGA SDK for OpenCL Getting Started Guide
16.1	Altera SDK for OpenCL Getting Started Guide
16.0	Altera SDK for OpenCL Getting Started Guide
15.1	Altera SDK for OpenCL Getting Started Guide
15.0	Altera SDK for OpenCL Getting Started Guide
14.1	Altera SDK for OpenCL Getting Started Guide

Intel Corporation. All rights reserved. Agilex, Altera, Arria, Cyclone, eASIC, Intel, the Intel logo, MAX, Nios, Quartus and Stratix words and logos are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.
*Other names and brands may be claimed as the property of others.

ISO 9001:2015 Registered

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

Document Version	Intel Quartus Prime Version	Changes
2021.03.29	21.1	• Added a note about running the script with `sudo` privileges in step 2 of *Installing the Intel FPGA SDK for OpenCL Pro Edition*. • Updated the BSP and SDK versions throughout the guide. • Changed all occurrences of BSP to OpenCL BSP for clarity.
2020.12.14	20.4	• Maintenance release.
2020.09.28	20.3	• Added instructions for installing GCC from the source for CentOS, Red Hat Enterprise Linux, and SUSE Enterprise Linux. • Updated the SDK versions throughout the guide. • Added a note about the Microsoft Visual Studio and Eclipse plug-ins in *Prerequisites for the Intel FPGA SDK for OpenCL Pro Edition*. • Added Microsoft Visual Studio and Eclipse plug-ins install instructions in *Installing the Intel FPGA SDK for OpenCL Pro Edition* topics for Windows and Linux. • In multiple topics, added or updated a note about the lack of support for Windows and Linux BSPs. • Updated the backward compatibility options in *Enabling Backwards Compatibility for an FPGA Board* and *Intel FPGA SDK for OpenCL Pro Edition and BSP Backwards Compatibility* topics.
2020.06.22	20.2	• Updated the SDK versions throughout the guide. • Updated the Intel Acceleration Stack version in *Intel FPGA SDK for OpenCL Pro Edition and BSP Backwards Compatibility.* • Added reference to the Intel Stratix 10 bring-up guide in the introduction. • RAM details and GCC version updated in *Prerequisites for the Intel FPGA SDK for OpenCL Pro Edition*. • Updated the `board` directory details in *Contents of the Intel FPGA SDK for OpenCL Pro Edition*. • Modified the Intel FPGA SDK for OpenCL Pro Edition Installation Process Overview figure to include downloading both Intel and Custom Platforms in *Overview of the Intel FPGA SDK for OpenCL Pro Edition Setup Process.*. • Modified the FPGA Programming Overview figure to include `-board-package=<board_package_path>` in *Overview of the Intel FPGA SDK for OpenCL Pro Edition Setup Process.*. • Since the BSPs are no longer part of the SDK, updated all occurrences of the path to BSP. • Modified the title and instructions in the *Downloading an OpenCL Design Example* topic. • Removed references to downloading OpenCL design examples from Intel website and replaced them with instructions to access examples from the SDK. • Added a new topic *GCC Requirement*.

continued...

Intel Corporation. All rights reserved. Agilex, Altera, Arria, Cyclone, eASIC, Intel, the Intel logo, MAX, Nios, Quartus and Stratix words and logos are trademarks of Intel Corporation or its subsidiaries in the U.S. and/or other countries. Intel warrants performance of its FPGA and semiconductor products to current specifications in accordance with Intel's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Intel assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Intel. Intel customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.
*Other names and brands may be claimed as the property of others.

ISO 9001:2015 Registered

Document Version	Intel Quartus Prime Version	Changes
2019.12.16	19.4	• Made minor updates in steps 1 and 4 in *Building the Host Application*. • Added more information in step 6 of *Installing an FPGA Board*. • Updated the backward compatibility details in *Intel FPGA SDK for OpenCL Pro Edition and BSP Backwards Compatibility*. • Changed the BSP path to *INTELFPGAOCLSDKROOT*`/board` in *Installing an FPGA Board*. • Updated the backwards compatibility version details in *Enabling Backwards Compatibility for an FPGA Board*. • Made the following changes in *Creating the FPGA Hardware Configuration File of an OpenCL Kernel*: — Added more information to the description. — Added more information to Step 2. — In Step 4, removed a task about `.aoco` file. • Added more information about ICD and FCD in *Building the Host Application*. • Updated the details about environment variables in *Setting the Intel FPGA SDK for OpenCL Pro Edition User Environment Variables* • Included GCC 6.3.0 requirement in *Prerequisites for the Intel FPGA SDK for OpenCL Pro Edition*
2019.09.30	19.3	• In *Creating the FPGA Hardware Configuration File of an OpenCL Kernel*, updated two commands. • In *Overview of the Intel FPGA SDK for OpenCL Pro Edition Setup Process*, updated few commands in the "FPGA Programming Overview" image. • In *Verifying Host Runtime Functionality via Emulation*, removed steps about installing a custom or reference platform and verifying the *QUARTUS_ROOTDIR_OVERRIDE* environment variable. • In *Compiling a Kernel for Emulation*, removed `-fast-emulator` option. • In *Creating the FPGA Hardware Configuration File of an OpenCL Kernel*, updated the commands. • In *Emulating Your OpenCL Kernel*, made the following updates: — Removed a steps about running `aocl linkflags`, linking host application, and moving the `hello_world.aocx file`. — Changed `CL_CONTEXT_EMULATOR_DEVICE_INTELFPGA` to `CL_CONFIG_CPU_EMULATE_DEVICES`. — Added more information in the step about running the host application for emulation. — Removed information about shared library copy. — Only in *Emulating Your OpenCL Kernel* topic for Linux, added a note about the version of `libstdc++.so` that must be at least that of GCC 6.3.0. • In *Getting Started with the SDK on Windows* chapter, swapped topics *Uninstalling an FPGA Board* and *Uninstalling the Software*. • In *Uninstalling the Software*, added a note about uninstalling the FPGA board. • In *Building the Host Application*, completely updated this topic by adding new steps and modifying existing steps.

continued...

Document Version	Intel Quartus Prime Version	Changes
2019.04.01	19.1	• Following topics were updated with information about `-fcd-only`: — Installing an FPGA Board on page 19 — Uninstalling an FPGA Board on page 30 — Installing an FPGA Board on page 35 — Uninstalling an FPGA Board on page 46 • Updated the compatibility details in Intel FPGA SDK for OpenCL Pro Edition and OpenCL BSP Backwards Compatibility on page 10. • In Building the Host Application on page 22, steps 1 - 3 were updated for setting up the MS Visual Studios, showing output when FCD and ICD are correctly setup. • In Prerequisites for the Intel FPGA SDK for OpenCL Pro Edition on page 5, added support for MS Visual C++ 2015 or later. • In Compiling a Kernel for Emulation on page 22, Emulating Your OpenCL Kernel on page 24, and Emulating Your OpenCL Kernel, changed `hello_world_emulation.aocx` to `hello_world.aocx`. • In Downloading the Intel FPGA SDK for OpenCL Pro Edition on page 15 and Downloading the Intel FPGA SDK for OpenCL Pro Edition on page 32, removed a step about downloading Intel FPGA SDK for OpenCL with Intel Code Builder. • In Figure 2 on page 14, changed `hello_world_emulation.aocx` to `hello_world.aocx` • In Installing an FPGA Board on page 19 and Installing an FPGA Board on page 35, added an example for the step about adding the paths to the Custom Platform libraries. • In Uninstalling the Software on page 30, updated the path to the installation folder. • In Uninstalling an FPGA Board on page 30, updated the last step about the *PATH* variable for better clarity. • In Creating the FPGA Hardware Configuration File of an OpenCL Kernel on page 24 and Creating the FPGA Hardware Configuration File of an OpenCL Kernel on page 40, updated step 2 with more information about what you can when you have multiple BSPs. • In Emulating Your OpenCL Kernel on page 24, and Emulating Your OpenCL Kernel, added the output observed upon successful emulation of the kernel.
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.

continued...

Document Version	Intel Quartus Prime Version	Changes
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 OpenCL BSP Backwards Compatibility on page 10 — 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 12, 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 15 — Downloading the Intel FPGA SDK for OpenCL Pro Edition on page 32 • 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 33
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".

continued...

Date	Version	Changes
		• In Building the Host Application on page 22, updated references to Microsoft Visual Studio 2015 as Microsoft Visual Studio. • In Executing an OpenCL Kernel on an FPGA on page 29, 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 15 — Installing an FPGA Board on page 19 — Verifying Host Runtime Functionality via Emulation on page 21 — Creating the FPGA Hardware Configuration File of an OpenCL Kernel on page 24 — Downloading the Intel FPGA SDK for OpenCL Pro Edition on page 32 — Installing an FPGA Board on page 35 — Verifying Host Runtime Functionality via Emulation on page 37 — Creating the FPGA Hardware Configuration File of an OpenCL Kernel on page 40 • 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 12 — Creating the FPGA Hardware Configuration File of an OpenCL Kernel on page 24 — Compiling a Kernel for Emulation on page 22 • Removed references to *AOCL_BOARD_PACKAGE_ROOT* throughout the guide since it is deprecated. • 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 12, 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 27, 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.

continued...

Date	Version	Changes
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:December 2014 — Getting Started with the AOCL on Windows — Getting Started with the AOCL on Linux
	14.1.0	• Reorganized information flow. • Updated Red Hat Enterprise Linux (RHEL) version support. • Included the *Contents of the AOCL* section. • 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*.

continued...

Date	Version	Changes
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*.
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*.

continued...

Date	Version	Changes
		• 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.