Intel® FPGA RTE for OpenCL™ Pro Edition

Getting Started Guide

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

Subscribe
Send Feedback

UG-OCL005 | 2021.03.29
Latest document on the web: PDF | HTML

Contents

UG-OCL005 | 2021.03.29

Send Feedback

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

The *Intel® FPGA RTE for OpenCL™ Pro Edition Getting Started Guide* describes the procedures you follow to install the Intel FPGA Runtime Environment (RTE) for OpenCL Pro Edition. This document also contains instructions on how to deploy an OpenCL [(1)] application with the RTE.

The RTE is a subset of the Intel FPGA Software Development Kit (SDK) for OpenCL Pro Edition[(2)] . Unlike the SDK, which provides an environment that enables the development and deployment of OpenCL kernel programs, the RTE provides tools and runtime components that enable you to build and execute a host program, and execute precompiled OpenCL kernel programs on target accelerator boards.

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

Attention: If you require OpenCL kernel development and deployment functionalities, download the Intel FPGA SDK for OpenCL Pro Edition. Refer to the *Intel FPGA SDK for OpenCL Pro Edition Getting Started Guide* for more information.

Do not install the RTE and the SDK on the same host system.

Related Information

- OpenCL Reference Pages
- OpenCL Specification version 1.0
- Intel FPGA SDK for OpenCL Getting Started Guide
- Intel FPGA SDK for OpenCL Cyclone V SoC Getting Started Guide

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

To install the Intel FPGA RTE for OpenCL Pro Edition and deploy an application on an Intel preferred accelerator board, your system must meet certain hardware, target platform, and software requirements.

---

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

(2) 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.

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.

Deployment system requirements:

- You must have administrator privileges on the development system to install the necessary packages and drivers.
- The deployment system has at least 70 megabytes (MB) 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.

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.
- Linux versions as supported on Intel SoC FPGA products on the Arm* ARMv7-A architecture.

  Important: For x86_64 Linux systems, 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`).

  To install the Linux kernel source or header package, invoke the `yum install <kernel_package_name>` command.

  You must have administrator privileges on the host system to install the necessary packages and drivers.

Software Prerequisites

Develop your host application using one of the following RTE-compatible C compiler or software development environment:

- For Windows systems, use Microsoft Visual Studio Professional and Microsoft Visual C++ versions 2015 or later.
- 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 6.
- For SoC applications, use the GCC cross-compiler available with the Intel SoC FPGA Embedded Development Suite (EDS).

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

Related Information

- OpenCL Platforms
- Intel FPGA SDK for OpenCL Pro Edition Getting Started Guide

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 7 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. Contents of the Intel FPGA RTE for OpenCL Pro Edition

The Intel FPGA RTE for OpenCL Pro Edition provides utilities, host runtime libraries, drivers, and RTE-specific libraries and files.

Utilities and Host Runtime Libraries

- The RTE Utility includes commands you can invoke to perform high-level tasks. The RTE utilities are a subset of of the Intel FPGA SDK for OpenCL Pro Edition utilities.
- The host runtime provides the OpenCL host platform API and runtime API for your OpenCL host application.

  The host runtime consists of the following libraries:

  — *Statically-linked libraries* provide OpenCL host APIs, hardware abstractions and helper libraries.

  — *Dynamic link libraries* (DLLs) provide hardware abstractions and helper libraries.

Drivers, Libraries and Files

The RTE installation process installs the RTE into a directory that you own. The path to the software installation directory is referenced by the *INTELFPGAOCLSDKROOT* environment variable.

Table 1. Contents of the RTE Installation Directory

Windows Folder	Linux Directory	ARM Directory	Description
`bin`	`bin`	`bin`	High-level utilities. Include this directory in your *PATH* environment variable setting.
`host`	`host`	`host`	Files necessary for compiling your host program.
`host\include`	`host/include`	`host/include`	OpenCL Specification version 1.0 header files and software interface files necessary for compiling and linking your host application. The `host/include/CL` subdirectory also includes the C++ header file `cl.hpp`. The file contains an OpenCL version 1.1 C++ wrapper API. These C++ bindings enable a C++ host program to access the OpenCL runtime APIs using native C++ classes and methods. *Important:* The OpenCL version 1.1 C++ bindings are compatible with OpenCL Specification versions 1.0 and 1.1. Add this path to the `include` file search path in your development environment.
`host\windows64\lib`	`host/linux64/lib`	`host/arm32/lib`	OpenCL host runtime libraries for the given target platform that provide the OpenCL platform and runtime APIs. These libraries are necessary for linking and running your host application. Prior to running your host application, include this directory in the library search path. • For Linux and ARM®, add the path to the *LD_LIBRARY_PATH* environment variable setting. • For Windows, add the path to the *PATH* environment variable setting.
`host\windows64\bin`	`host/linux64/bin`	`host/arm32/bin`	Platform-specific binary for the RTE Utility, runtime commands, and DLLs (for Windows) necessary for running your host application, wherever applicable. Include this directory in your *PATH* environment variable setting.
`share\lib\perl`	`share/lib/perl`	`share/lib/perl`	Perl scripts and support libraries for the RTE Utility.

Example OpenCL Applications

You can refer to example OpenCL applications from the `examples_aoc` directory in the Intel FPGA SDK for OpenCL installation. For more information, refer to Contents of the Intel FPGA SDK for OpenCL Pro Edition in the *Intel FPGA SDK for OpenCL Getting Started Guide*.

1.3. RTE Utility

The Intel FPGA RTE for OpenCL utility is a subset of the Intel FPGA SDK for OpenCL utility. It provides you with tools and information to perform high-level tasks such as configuring the host application development flow.

Displaying the Software Version on page 11

Listing the Intel FPGA RTE for OpenCL Utility Command Options on page 11

Managing an FPGA Board on page 11

Managing Host Application on page 11

1.3.1. Displaying the Software Version

To display the version of the Intel FPGA RTE for OpenCL, invoke the `version` utility command.

Note: The ARM processor on an Intel SoC FPGA board does not support this utility.

- At the command prompt, invoke the `aocl version` command.
  Example output:

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

1.3.2. Listing the Intel FPGA RTE for OpenCL Utility Command Options

To display information on the Intel FPGA RTE for OpenCL utility command options, invoke the `help` utility command.

Attention: The ARM processor on an Intel SoC board does not support this utility.

- At a command prompt, invoke the `aocl help` command.
  The RTE categorizes the utility command options based on their functions. It also provides a description for each option.

1.3.3. Managing an FPGA Board

The Intel FPGA RTE for OpenCL includes utility commands you can invoke to install, uninstall, diagnose, and program your FPGA board.

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.

For more information about the `install`, `uninstall`, `diagnose`, `program` and `flash` utility commands, refer to the *Managing an FPGA Board* section of the *Intel FPGA SDK for OpenCL Pro Edition Programming Guide*.

Related Information

Managing an FPGA Board

1.3.4. Managing Host Application

The Intel FPGA RTE for OpenCL includes utility commands you can invoke to obtain information on flags and libraries necessary for compiling and linking your host application.

Attention: To cross-compile your host application to an SoC FPGA board, include the `--arm` option in your utility command.

Caution: For Linux systems, if you debug your host application using the GNU Project Debugger (GDB), invoke the following command prior to running the host application:

```
handle SIG44 nostop
```

Without this command, the GDB debugging process terminates with the following error message:

```
Program received signal SIG44, Real-time event 44.
```

For information on the following utility command options, refer to the *Managing Host Application* section of the *Intel FPGA SDK for OpenCL Programming Guide*:

- `example-makefile` or `makefile`
- `compile-config`
- `ldflags`
- `ldlibs`
- `link-config` or `linkflags`

Related Information

Managing Host Application

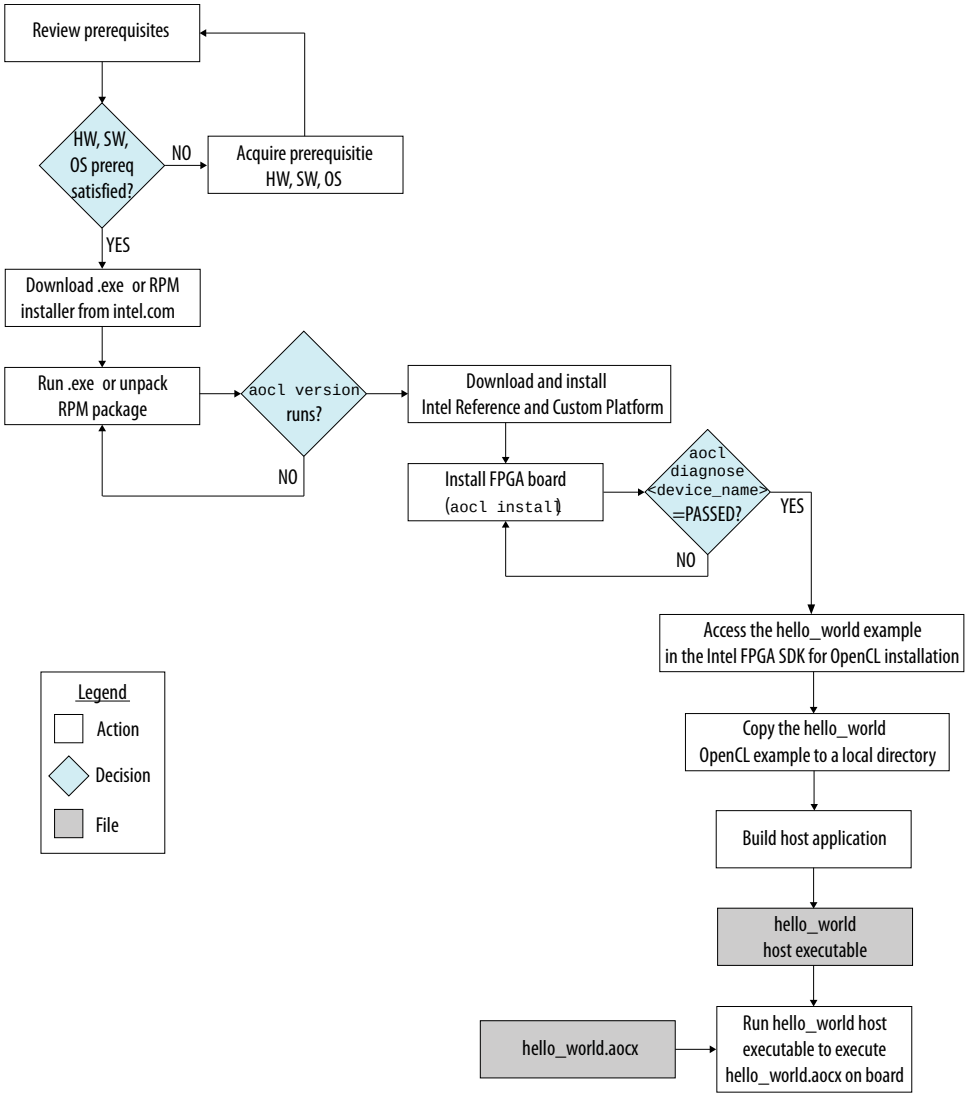
1.4. Overview of the Intel FPGA RTE for OpenCL Pro Edition Setup Process

The *Intel FPGA RTE for OpenCL Pro Edition Getting Started Guide* outlines the procedures for installing the Intel FPGA RTE for OpenCL Pro Edition and deploying an OpenCL example design onto your device.

Important: The RTE does not include the Intel FPGA SDK for OpenCL Offline Compiler; therefore, you cannot use the RTE to compile an OpenCL kernel. You must use the Intel FPGA SDK for OpenCL on a separate development machine to create an executable file (`.aocx`) from the `.cl` kernel source file. Refer to the *Intel FPGA SDK for OpenCL Pro Edition Getting Started Guide* for instructions on setting up the SDK and compiling an OpenCL kernel.

Figure 1. RTE Setup Process for x86-64 Systems

The figure below summarizes the steps for installing the RTE and the FPGA board, and in executing an OpenCL kernel on the board.



Note: The FPGA boards are no longer shipped with the SDK and you need to download them from https://fpgasoftware.intel.com/opencl/ page and select **Windows BSP** or **Linux BSP** tab.

For an overview of the RTE setup process for SoC, refer to *Getting Started with the Intel FPGA RTE for OpenCL for Intel ARMv7-A SoC*.

Related Information

- Getting Started with the Intel FPGA RTE for OpenCL Pro Edition for Intel ARMv7-A SoC FPGA on page 35
- Intel FPGA SDK for OpenCL Pro Edition Getting Started Guide

2. Getting Started with the Intel FPGA RTE for OpenCL Pro Edition for 64-Bit Windows

Figure 1 on page 13 outlines the RTE setup process for 64-bit Windows systems.

1. Downloading the Intel FPGA RTE for OpenCL Pro Edition on page 14
2. Installing the Intel FPGA RTE for OpenCL Pro Edition on page 14
3. Setting the Intel FPGA RTE for OpenCL Pro Edition User Environment Variables on page 15
4. Verifying Software Installation on page 16
5. Installing an FPGA Board on page 17
6. Updating the Hardware Image on the FPGA on page 19
7. Executing an OpenCL Kernel on an FPGA on page 20
8. Uninstalling the Software on page 23
9. Uninstalling an FPGA Board on page 24

2.1. Downloading the Intel FPGA RTE for OpenCL Pro Edition

Download the Intel FPGA RTE for OpenCL Pro Edition for Windows from the Intel FPGA SDK for OpenCL Download Center.

1. Go to the Intel FPGA SDK for OpenCL Download Center at the following URL:

   https://fpgasoftware.intel.com/opencl/
2. Select the Pro edition.
3. Select the software version. The default selection is the current version.
4. Click the **RTE** tab and select **Intel FPGA Runtime Environment for OpenCL Windows x86-64**. Click **More** beside **Download and install instructions** to view the download and installation procedure.
5. Click the download button to start the download process.
6. Perform the steps outlined in the download and installation instructions on the download page.

Related Information

Intel FPGA website

2.2. Installing the Intel FPGA RTE for OpenCL Pro Edition

Install the Windows version of the Intel FPGA RTE for OpenCL Pro Edition in a folder that you own.

You must have administrator privileges to execute these instructions.

To install the Intel FPGA RTE for OpenCL, perform the following tasks:

1. Run the `.exe` installer. Direct the installer to extract the software to an empty folder that you own (that is, not a system folder).

   Note: The installation path must not contain any spaces (for example, `<home_directory>\intelFPGA_pro\<version>\aclrte-windows64`).

2. *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 RTE installation, edit the *INTELFPGAOCLSDKROOT* setting.

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

Related Information

Setting the Intel FPGA RTE for OpenCL Pro Edition User Environment Variables on page 15

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

You have the option to set the Intel FPGA RTE 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 RTE 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 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 `aocl.exe` file

2.4. Verifying Software Installation

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

Attention: Intel FPGA RTE for OpenCL-supported Intel SoC boards do not support the `version` utility.

- 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 Runtime Environment 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
- 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 `aocl install` utility. 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. 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 RTE for OpenCL Pro Edition User Environment Variables* section.

5. 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
     ```

6. 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.

7. 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

- Setting the Intel FPGA RTE for OpenCL Pro Edition User Environment Variables on page 15
- Querying the Device Name of Your FPGA Board on page 19
- Intel FPGA SDK for OpenCL FPGA Platforms

2.6. 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.

2.6.1. 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
```

2.6.2. Programming the Flash Memory of an FPGA

Configure the FPGA by loading the hardware image of an Intel FPGA RTE 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

2.7. 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 RTE for OpenCL is compatible with 64-bit host binaries only.

2.7.1. 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 17.

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 to the ICD Loader Library on Windows

2.7.2. 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.

2.7.3. 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.
```

2.8. Uninstalling the Software

To uninstall the Intel FPGA RTE for OpenCL Pro Edition for Windows, delete the RTE folder and restore all modified environment variables to their previous settings.

Note: Before uninstalling the Intel FPGA RTE for OpenCL Pro Edition for Windows, you must first uninstall the FPGA board. Refer to Uninstalling an FPGA Board on page 24 for more information.

1. In Windows Explorer, navigate to the `intelFPGA_pro\<version> <edition>` folder.
2. Delete the `aclrte-windows64` folder.
3. Remove the following paths from the *PATH* environment variable:
   a. `%INTELFPGAOCLSDKROOT%\bin`

b. `%INTELFPGAOCLSDKROOT%\host\windows64\bin`

4. Remove the *INTELFPGAOCLSDKROOT* environment variable.

2.9. 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 RTE 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.

3. Getting Started with the Intel FPGA RTE for OpenCL Pro Edition for x86_64 Linux Systems

Figure 1 on page 13 outlines the RTE setup process for x86_64 Linux systems.

3.1. Downloading the Intel FPGA RTE for OpenCL Pro Edition

Download the Intel FPGA RTE for OpenCL Pro Edition for Linux from the Download Center.

1. Go to the Intel FPGA RTE for OpenCL Download Center at the following URL:
   https://fpgasoftware.intel.com/opencl/
2. Select the Pro edition.
3. Select the software version. The default selection is the current version.
4. Click the **RTE** tab and select the installation package you want to download. Click **More** beside **Download and install instructions** to view the download and installation procedure.
5. Click the download button to start the download process.
6. Perform the steps outlined in the download and installation instructions on the download page.

Related Information

Intel FPGA website

3.2. Installing the Intel FPGA RTE for OpenCL Pro Edition

Install the Linux version of the Intel FPGA RTE 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.

To install the Intel FPGA RTE for OpenCL, perform the following tasks:

1. 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.

2. At the command prompt, type the RPM command to install the downloaded RPM package.

   Note: The installation path must not contain any spaces (for example, `/usr/intel/<version>/intel-fpga-opencl-pro-rte`).

   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 software using the Red Hat Package Manager (RPM), at the command prompt, type the `rpm -i intel-fpga-opencl-pro-rte-64bit-linux.rpm` command.

   The RPM installs the software in the default location (for example, `/opt/intel/intel-fpga-opencl-pro-rte`).

   — To install the software in the default location with verbose progress reporting, type `rpm -ivh intel-fpga-opencl-pro-rte-64bit-linux.rpm`

   — To install the software in an alternate directory that you own (that is, not a system directory), type `rpm -i --prefix <rte_destination_directory> intel-fpga-opencl-pro-rte-64bit-linux.rpm`

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 RTE for OpenCL installation, edit the *INTELFPGAOCLSDKROOT* setting.

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

Related Information

Setting the Intel FPGA RTE for OpenCL Pro Edition User Environment Variables on page 27

3.2.1. Setting the Intel FPGA RTE for OpenCL Pro Edition User Environment Variables

You have the option to set the Intel FPGA RTE 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 RTE 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.

3.3. Verifying Software Installation

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

Attention: FPGA RTE for OpenCL-supported SoC boards do not support the `version` utility.

- 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 Runtime Environment 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
- Knowledge Base

3.4. 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 install` 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. 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 RTE for OpenCL Pro Edition User Environment Variables* section.

5. 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
      ```

6. 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.

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

Related Information

- Querying the Device Name of Your FPGA Board on page 30
- Setting the Intel FPGA RTE for OpenCL Pro Edition User Environment Variables on page 27
- Intel FPGA SDK for OpenCL FPGA Platforms

3.5. 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.

3.5.1. 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
```

3.5.2. Programming the Flash Memory of an FPGA

Configure the FPGA by loading the hardware image of an Intel FPGA RTE 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.

3.6. 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.

3.6.1. 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 28, 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 28.

   Note: If you do not have the `root` privilege, refer to Installing an FPGA Board on page 28.

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
```

3.6.2. 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.

3.6.3. 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.
```

3.7. Uninstalling the Software

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

Note: Before uninstalling the Intel FPGA RTE for OpenCL Pro Edition for Linux, you must first uninstall the FPGA board. Refer to Uninstalling an FPGA Board on page 33 for more information.

1. Remove the software package by executing the `rpm -e intel-fpga-opencl-pro-rte` command. This uninstalls the RTE.
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.

3.8. 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 RTE 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.

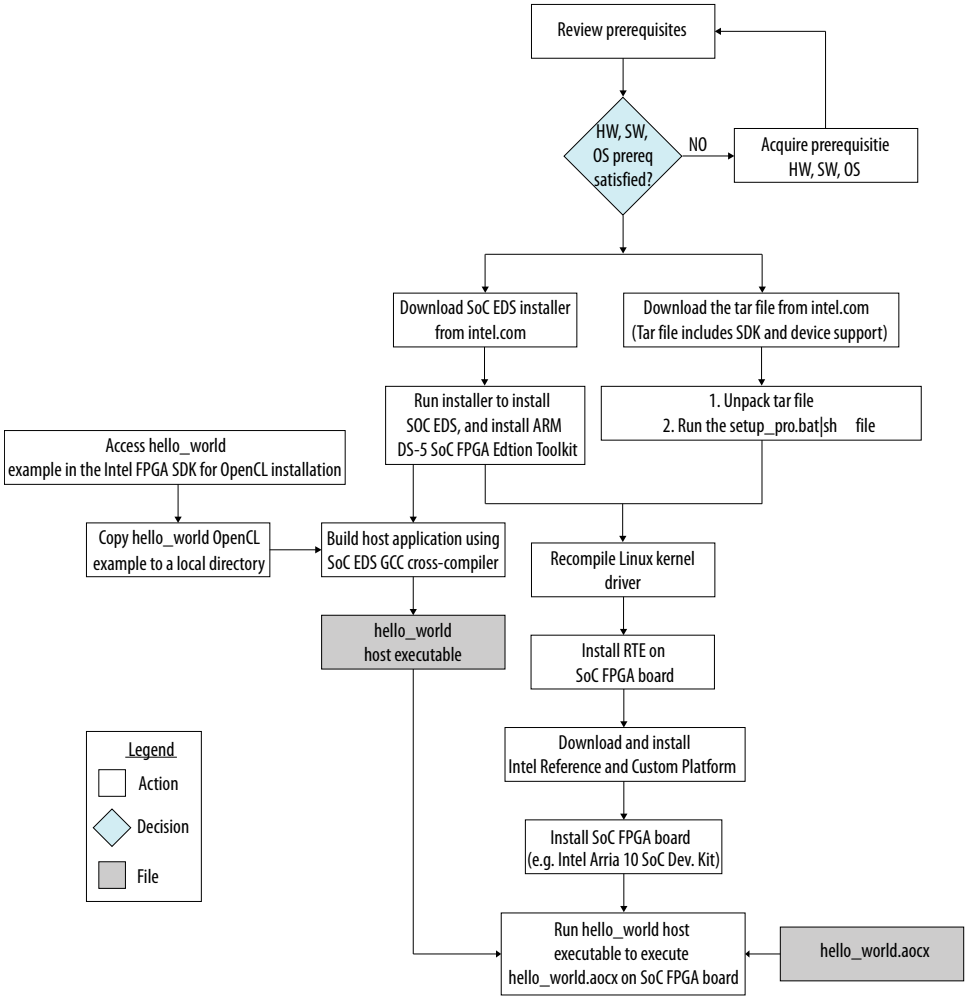
4. Getting Started with the Intel FPGA RTE for OpenCL Pro Edition for Intel ARMv7-A SoC FPGA

The following sections provide instructions for setting up Windows and Linux versions of the RTE for use with the Intel Arria 10 SoC Development Kit.

For information on the key components of the Intel Arria 10 SoC Development Kit, refer to the Intel Arria 10 SoC Development Kit product page on the Intel FPGA website.

Figure 2. RTE Setup Process for SoC FPGA

The figure below outlines the steps for installing the software and the SoC FPGA board, and in executing an OpenCL kernel on the SoC FPGA board.



Related Information

Intel Arria 10 SoC Development Kit product page

4.1. Getting Started with the Intel FPGA RTE for OpenCL Pro Edition for SoC FPGA on Windows

To execute an OpenCL kernel onto an SoC FPGA, install Windows versions of the Intel FPGA SDK for OpenCL Pro Edition and the Intel SoC FPGA Embedded Development Suite (EDS) Pro Edition on your host system, and install the RTE on your SoC FPGA board. You must also build your host application using an ARM-specific Makefile.

4.1.1. Downloading the Intel FPGA SDK for OpenCL and the SoC EDS

To get started with the Intel FPGA RTE for OpenCL Pro Edition on the Intel Arria 10 SoC Development Kit, download the Intel FPGA SDK for OpenCL Pro Edition and the SoC EDS Pro Edition for Windows from the Intel FPGA Download Center.

The SDK includes the SD card image you need to recompile the OpenCL Linux kernel driver. If you want to recompile the Linux kernel driver and write the SD card image on your own, download the RTE for SoC instead.

- To download the SDK, follow the instructions outlined in the *Downloading the Intel FPGA SDK for OpenCL* Pro Edition section of the *Intel FPGA SDK for OpenCL Pro Edition Getting Started Guide*.
- To download the RTE, perform the following tasks:
  a. Go to the Intel FPGA SDK for OpenCLDownload Center at the following URL:

     http://fpgasoftware.intel.com/opencl/
  b. Select the Pro edition.
  c. Select the software version. The default selection is the current version.
  d. Click the **RTE** tab. Click **More** beside **Download and install instructions** to view the download and installation procedure.
  e. Click the download button beside **Intel FPGA Runtime Environment for OpenCL Linux Cyclone V SoC TGZ** to start the download process.
  f. Perform the steps outlined in the download and installation instructions on the download page.
- Download the SoC EDS by performing the following steps:
  a. From the Download Center , click **Embedded Software ➤ SoC EDS** to enter the download page for the subscription edition of the SoC EDS.
  b. Select the Pro Edition.
  c. Select the software version.
  d. Select **Windows** as the operating system.
  e. Click **Intel SoC FPGA Embedded Development Suite Pro Edition** to download.
  f. Perform the steps outlined in the download and installation instructions.

Related Information

- Intel FPGA website

- Downloading the Intel FPGA SDK for OpenCL

4.1.2. Installing the Intel FPGA SDK for OpenCL Pro Edition for SoC FPGA

The Intel FPGA SDK for OpenCL Intel Arria 10 SoC Development Kit Reference Platform (a10soc) includes an SD flash card image necessary for running OpenCL applications on the board. The SD flash card image includes the recompiled Linux kernel driver, preinstalled version of the Intel FPGA RTE for OpenCL, and a script for setting environment variables.
To get started with the RTE on the Intel Arria 10 SoC Development Kit using the SD flash card image that comes with the SDK, install the SDK for Windows. If you want to create your own SD card image, install the RTE.

You must have administrator privileges.

- To install the SDK, perform the following tasks:
  a. Unpack the downloaded `AOCL-<version>-<build>-windows.tar` file into a folder that you own.

     The installation path must not contain any spaces (for example, `<home_directory>\intelFPGA_pro\<version>\hld`).

  b. Run the `setup_pro.bat` file to install the SDK and device support.
- To install the RTE, unpack the `.tgz` file install the RTE in a folder that you own.
- *Note:* The installer sets the environment variable *INTELFPGAOCLSDKROOT* to point to the path of the software installation.

  Verify that the installer sets the user environment variable *INTELFPGAOCLSDKROOT* to point 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 current SDK installation, or if the path is not set, modify the *INTELFPGAOCLSDKROOT* setting.

Related Information

Setting the Intel FPGA RTE for OpenCL Pro Edition User Environment Variables on page 15

4.1.3. Installing the Intel SoC FPGA Embedded Development Suite Pro Edition

Install the SoC EDS for Windows to build your host application for OpenCL kernel deployment on an SoC board.

1. Run the installer. Follow the installation instructions in the `SoCEDSSetup-<version>-windows.exe` executable. For more information, refer to the *Installing the SoC EDS* section of the *Intel SoC FPGA Embedded Design Suite User Guide*.
2. Perform the tasks outlined in the *Installing the Arm DS-5* Intel SoC FPGA Edition Toolkit* section of the *Intel SoC FPGA Embedded Design Suite User Guide* to install the Arm Development Studio 5* (DS-5) Intel SoC FPGA Edition Toolkit for your operating system.

For more information about the Arm DS-5 Intel SoC FPGA Edition Toolkit, refer to the Arm DS-5 Intel SoC FPGA Edition page of the ARM website.

3. Consult the *Licensing* section of the *Intel SoC FPGA Embedded Design Suite User Guide* for licensing instructions for the SoC EDS and the Arm DS-5 Intel SoC FPGA Edition Toolkit.

Related Information

- Installing the SoC FPGA EDS on Windows
- Installing the ARM DS-5 Intel SoC FPGA Edition Toolkit
- SoC FPGA EDS Licensing

4.1.4. Recompiling the Linux Kernel Driver

Compile the OpenCL Linux kernel driver against the compiled kernel source. If you need to rebuild the Linux kernel driver, recompile the `aclsoc` Linux kernel driver to the exact version of the Linux kernel running on the SoC FPGA.

The driver source is available in the Intel FPGA RTE for OpenCL installation directory. Compile the driver yourself on a host machine that has `sudo` and the most recent version of the SoC EDS.

1. Copy the driver source from `$INTELFPGAOCLSDKROOT/board/a10soc/arm32/driver/` to a new directory.
2. Set the `KDIR` value in the driver `Makefile` to the directory containing the Linux kernel source files that you downloaded in Compiling the Linux Kernel for the Intel Arria 10 SoC Development Kit.
3. In the new directory that contains the driver source files, run the `make clean` command.
4. Run the `make` command to create the `aclsoc_drv.ko` file.

   This file is used later in Building the SD Card Image.

   The driver might need to be updated to work with newer version of the Linux kernel if you see the following message while building the kernel driver:

   ```
   aclsoc_cmd.c:165:14: error: too many arguments to function
   'get_user_pages_unlocked'
   In file included from aclsoc_cmd.c:50:0
   ```

   To update the driver, make the following changes to `$INTELFPGAOCLSDKROOT/board/a10soc/arm32/driver/aclsoc_cmd.c`:

   a. Find the following code in `aclsoc_cmd.c`:

   ```
   ret = get_user_pages_unlocked(target_task, target_task->mm,
               start_page + got * PAGE_SIZE,
               num_pages - got, write, 1,
               p + got);
   ```

   b. Replace that code with the following code:

   ```
   ret = get_user_pages_remote(target_task, target_task->mm,
               start_page + got * PAGE_SIZE,
               num_pages - got, FOLL_WRITE|FOLL_FORCE,
               p + got, vma);
   ```

Related Information

- Recompiling the Linux Kernel for the Intel Arria 10 SoC Development Kit
- Compiling and Installing the OpenCL Linux Kernel Driver

4.1.5. Installing the Intel FPGA RTE for OpenCL Pro Edition onto the SoC FPGA Board

The RTE installation package for Intel SoC FPGAs with 32-bit ARM processor is available in tar format. To install the software, you must install it in a directory that you own, and set all the necessary environment variables.

1. Create an RTE directory on the board's file system by typing the `mkdir <rte_destination_directory>` command.
2. Move the downloaded installation package `intel-fpga-opencl-pro-rte-32bit-arm.tgz` to the RTE directory by typing the `mvintel-fpga-opencl-pro-rte-32bit-arm.tgz <rte_destination_directory>` command.
3. Type `cd <rte_destination_directory>` to navigate to the RTE directory.
4. To unpack the tarball, type `tar -xzvf intel-fpga-opencl-pro-rte-32bit-arm.tgz` at the command prompt.
5. Transfer the `aclsoc_drv.ko` file you built on your development machine into the `<rte_destination_directory>/board/a10soc/driver` directory on the SoC FPGA board.
6. Set the environment variables, as shown below.

   Intel recommends that you consolidate the settings of the environment variables into a file called `init_opencl.sh`. Then, run the command `source ./init_opencl.sh` to load all the environment variables and the OpenCL Linux kernel driver simultaneously.

```
export INTELFPGAOCLSDKROOT=<rte_destination_directory>
export PATH=$INTELFPGAOCLSDKROOT/bin:$PATH
export LD_LIBRARY_PATH=$INTELFPGAOCLSDKROOT/host/arm32/lib:$LD_LIBRARY_PATH
insmod $INTELFPGAOCLSDKROOT/board/c5soc/driver/aclsoc_drv.ko
```

4.1.6. Installing the Intel Arria 10 SoC Development Kit

For information on the setup of the Intel Arria 10 SoC Development Kit, refer to the *Arria 10 SoC Development Kit User Guide*.

Related Information

Arria 10 SoC Development Kit User Guide

4.1.7. Executing an OpenCL Kernel on an SoC FPGA

The procedures outlined in this document are for building and running the host application for the hello_world example design. To execute the hello_world OpenCL kernel on your SoC FPGA, you must first create an `hello_world.aocx` file. For instructions on obtaining the hello_world example design and creating the

hello_world.aocx file, refer to the *Creating the FPGA Hardware Configuration File of an OpenCL Kernel* section of the *Intel FPGA SDK for OpenCL Standard Edition Cyclone V SoC Getting Started Guide*.
Build your host application using the GCC cross-compiler available with the SoC EDS.

Related Information

Creating the Hardware Configuration File of an OpenCL Kernel for SoC FPGA

4.1.7.1. Building the Host Application

Build your SoC FPGA-specific OpenCL host application using the GCC cross-compiler available with the Windows version of the SoC EDS.

1. Perform the following tasks to download the hello_world design example.
   a. Download the SoC FPGA-specific hello_world design example (***<version>* Arm32 Linux package (.tgz)**) from the Hello World Design Example page.
   b. Extract exm_opencl_hello_world_arm32_linux_*<version>*.tar to a location to which you have write access.

      Important: Ensure that the location name does not contain spaces.
2. At a command prompt, invoke the following command to set the *PATH* environment variable:

   SET PATH=%PATH%;*<path_to_SoCEDS_installation_dir>*\ds-5\sw\gcc\bin
3. Navigate to the *<path_to_exm_opencl_hello_world_arm32_linux_<version>>*\hello_world folder.
4. Invoke the make -f Makefile command. Alternatively, you can simply invoke the make command.
   The hello_world executable is in the *<path_to_exm_opencl_hello_world_arm32_linux_<version>>*\hello_world\bin folder.

Related Information

Hello World Design Example

4.1.7.2. Running the Host Application

For Windows systems, execute the hello_world.aocx executable file on the SoC FPGA by running the host application you built from the ARM-specific Makefile.

1. Log into your SoC FPGA board.
2. Copy the hello_world.aocx hardware configuration file and the hello_world host executable from their current folders to the board.
3. Verify that the *LD_LIBRARY_PATH* environment variable setting includes *%INTELFPGAOCLSDKROOT%*\host\arm32\lib. Run the command echo $LD_LIBRARY_PATH.
   If you ran the init_opencl.sh script, the *LD_LIBRARY_PATH* setting should point to *%INTELFPGAOCLSDKROOT%*\host\arm32\lib.
4. To execute the kernel on the SoC FPGA, at a command prompt, navigate to the host executable folder and run the hello_world host executable.

4.1.8. Uninstalling the Intel FPGA RTE for OpenCL

To uninstall the RTE from the SoC FPGA board, delete the RTE directory and restore all modified environment variables to their previous settings.

1. Navigate to the root directory in the SoC FPGA board's file system that contains the <*rte_destination_directory*> directory.
2. Type `rm -rf <`*rte_destination_directory*`>` to remove the RTE directory.
3. Remove the environment variable settings by typing the following commands:

   `unset` *INTELFPGAOCLSDKROOT*

   `unset` *PATH*

   `unset` *LD_LIBRARY_PATH*
4. Uninstall the Intel FPGA SDK for OpenCL on your host system and unset the corresponding environment variables.

Related Information

Uninstalling the Intel FPGA SDK for OpenCL

4.2. Getting Started with the Intel FPGA RTE for OpenCL Pro Edition for SoC FPGA on Linux

To execute an OpenCL kernel onto an SoC FPGA, install Linux versions of the Intel FPGA SDK for OpenCL Pro Edition and the Intel SoC FPGA Embedded Development Suite (EDS) Pro Edition, and install the RTE on your SoC FPGA board. You must also build your host application using an ARM-specific Makefile.

4.2.1. Downloading the Intel FPGA SDK for OpenCL and the SoC EDS

To get started with the Intel FPGA RTE for OpenCL Pro Edition on the Intel Arria 10 SoC Development Kit, download the Intel FPGA SDK for OpenCL Pro Edition and the SoC EDS Pro Edition for Linux from the Intel FPGA Download Center.

The SDK includes the SD card image you need to recompile the OpenCL Linux kernel driver. If you want to recompile the Linux kernel driver and write the SD card image on your own, download the RTE for SoC instead.

- To download the SDK, follow the instructions outlined in the *Downloading the Intel FPGA SDK for OpenCL* Pro Edition section of the *Intel FPGA SDK for OpenCL Pro Edition Getting Started Guide*.
- To download the RTE, perform the following tasks:
  a. Go to the Intel FPGA SDK for OpenCLDownload Center at the following URL:
     http://fpgasoftware.intel.com/opencl/
  b. Select the Pro edition.
  c. Select the software version. The default selection is the current version.
  d. Click the **RTE** tab. Click **More** beside **Download and install instructions** to view the download and installation procedure.
  e. Click the download button beside **Intel FPGA Runtime Environment for OpenCL Linux Cyclone V SoC TGZ** to start the download process.
  f. Perform the steps outlined in the download and installation instructions on the download page.
- Download the SoC EDS by performing the following steps:
  a. From the Download Center , click **Embedded Software ➤ SoC EDS** to enter the download page for the subscription edition of the SoC EDS.
  b. Select the Pro edition.
  c. Select the software version.
  d. Select **Linux** as the operating system.
  e. Click **Intel SoC FPGA Embedded Development Suite Pro Edition**. Download begins immediately.
  f. Perform the steps outlined in the download and installation instructions.

Related Information

- Intel FPGA website
- Downloading the Intel FPGA SDK for OpenCL

4.2.2. Installing the Intel FPGA SDK for OpenCL Pro Edition for SoC FPGA

The Intel FPGA SDK for OpenCL Intel Arria 10 SoC Development Kit Reference Platform (a10soc) includes an SD flash card image necessary for running OpenCL applications on the board. The SD flash card image includes the recompiled Linux kernel driver, a preinstalled version of the Intel FPGA RTE for OpenCL, and a script for setting environment variables.
To get started with the RTE on the Intel Arria 10 SoC Development Kit using the SD flash card image that comes with the SDK, install the SDK for Linux. If you want to create your own SD card image, install the RTE.

You must have sudo or root privileges.

- To install the SDK, perform the following tasks:
  a. Unpack the downloaded `AOCL-<version>-<build>-linux.tar` file.
  b. Run the `setup_pro.sh` file to install the SDK and device support.
- To install the RTE, unpack the `.tgz` file install the RTE in a directory that you own.
- *Note:* The installer sets the environment variable *INTELFPGAOCLSDKROOT* to point to the path of the software installation.

  Verify that the installer sets the user environment variable *INTELFPGAOCLSDKROOT* to point 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 current SDK installation, or if the path is not set, modify the *INTELFPGAOCLSDKROOT* setting.

Related Information

Setting the Intel FPGA RTE for OpenCL Pro Edition User Environment Variables on page 27

4.2.3. Installing the Intel SoC FPGA Embedded Development Suite Pro Edition

Install the Intel SoC EDS for Linux to build your host application for OpenCL kernel deployment on an SoC FPGA board.

The GCC tool chain is part of the SoC EDS installation package.

1. Run the `SoCEDSSetup-<version>-linux.run` installer. For more information, refer to the *Installing the SoC EDS* section of the *Intel SoC FPGA Embedded Development Suite User Guide*.
2. Perform the tasks outlined in the *Installing the Arm DS-5 Intel SoC FPGA Edition Toolkit* section of the *Intel SoC FPGA Embedded Development Suite User Guide* to install the Arm Development Studio 5 (DS-5) Intel SoC FPGA Edition Toolkit for your operating system.

   For more information about the Arm DS-5 Intel SoC FPGA Edition Toolkit, refer to the Arm DS-5 Intel SoC FPGA Edition page of the ARM website.
3. Consult the *Licensing* section of the *Intel SoC FPGA Embedded Development Suite User Guide* for licensing instructions for the SoC EDS and the Arm DS-5 Intel SoC FPGA Edition Toolkit.

Related Information

- Installing the SoC FPGA EDS on Windows
- Installing the ARM DS-5 Intel SoC FPGA Edition Toolkit
- SoC FPGA EDS Licensing

4.2.4. Recompiling the Linux Kernel Driver

Compile the OpenCL Linux kernel driver against the compiled kernel source. If you need to rebuild the Linux kernel driver, recompile the `aclsoc` Linux kernel driver to the exact version of the Linux kernel running on the SoC FPGA.

The driver source is available in the installation directory. Compile the driver yourself on a host machine that has `sudo` and the most recent version of the SoC EDS.

1. Copy the driver source from `$INTELFPGAOCLSDKROOT/board/a10soc/arm32/driver/` to a new directory.
2. Set the KDIR value in the driver `Makefile` to the directory containing the Linux kernel source files that you downloaded in Compiling the Linux Kernel for the Intel Arria 10 SoC Development Kit.
3. In the new directory that contains the driver source files, run the `make clean` command.
4. Run the `make` command to create the `aclsoc_drv.ko` file.

   This file is used later in Building the SD Card Image.

   The driver might need to be updated to work with newer version of the Linux kernel if you see the following message while building the kernel driver:

   ```
   aclsoc_cmd.c:165:14: error: too many arguments to function
   'get_user_pages_unlocked'
   In file included from aclsoc_cmd.c:50:0
   ```

   To update the driver, make the following changes to `$INTELFPGAOCLSDKROOT/board/a10soc/arm32/driver/aclsoc_cmd.c`:

   a. Find the following code in `aclsoc_cmd.c`:

   ```
   ret = get_user_pages_unlocked(target_task, target_task->mm,
               start_page + got * PAGE_SIZE,
               num_pages - got, write, 1,
               p + got);
   ```

   b. Replace that code with the following code:

   ```
   ret = get_user_pages_remote(target_task, target_task->mm,
               start_page + got * PAGE_SIZE,
               num_pages - got, FOLL_WRITE|FOLL_FORCE,
               p + got, vma);
   ```

Related Information

- Recompiling the Linux Kernel for the Intel Arria 10 SoC Development Kit
- Compiling and Installing the OpenCL Linux Kernel Driver

4.2.5. Installing the Intel FPGA RTE for OpenCL Pro Edition onto the SoC FPGA Board

The RTE installation package for SoC FPGAs with 32-bit ARM processor is available in tar format. To install the software, you must install it in a directory that you own, and set all the necessary environment variables.

1. Create an RTE directory on the board's file system by typing the `mkdir <rte_destination_directory>` command.
2. Move the downloaded installation package `intel-fpga-opencl-pro-rte-32bit-arm.tgz` to the RTE directory by typing the `mvintel-fpga-opencl-pro-rte-32bit-arm.tgz <rte_destination_directory>` command.
3. Type `cd <rte_destination_directory>` to navigate to the RTE directory.

4. To unpack the tarball, type `tar -xzvf intel-fpga-opencl-pro-rte-32bit-arm.tgz` at the command prompt.
5. Transfer the `aclsoc_drv.ko` file you built on your development machine into the *<rte_destination_directory>*`/board/a10soc/driver` directory on the SoC FPGA board.
6. Set the environment variables, as shown below.

   Intel recommends that you consolidate the settings of the environment variables into a file called `init_opencl.sh`. Then, run the command `source ./init_opencl.sh` to load all the environment variables and the OpenCL Linux kernel driver simultaneously.

```
export INTELFPGAOCLSDKROOT=<rte_destination_directory>
export PATH=$INTELFPGAOCLSDKROOT/bin:$PATH
export LD_LIBRARY_PATH=$INTELFPGAOCLSDKROOT/host/arm32/lib:$LD_LIBRARY_PATH
insmod $INTELFPGAOCLSDKROOT/board/c5soc/driver/aclsoc_drv.ko
```

4.2.6. Installing the SoC Development Kit

For information on the setup of the SoC Development Kit, refer to the *Arria 10 SoC Development Kit User Guide*.

Related Information

Arria 10 SoC Development Kit User Guide

4.2.7. Executing an OpenCL Kernel on an SoC FPGA

The procedures outlined in this document are for building and running the host application for the hello_world example design. To execute the hello_world OpenCL kernel on your SoC FPGA, you must first create an `hello_world.aocx` file. For instructions on obtaining the hello_world example design and creating the `hello_world.aocx` file, refer to the *Creating the FPGA Hardware Configuration File of an OpenCL Kernel* section of the *Intel FPGA SDK for OpenCL Standard Edition Cyclone V SoC Getting Started Guide*.
Build your host application using the GCC cross-compiler available with the SoC EDS.

Related Information

Creating the Hardware Configuration File of an OpenCL Kernel for SoC FPGA

4.2.7.1. Building the Host Application

Build your SoC FPGA-specific OpenCL host application using the GCC cross-compiler available with the Linux version of the SoC EDS.

1. At a command prompt, invoke the following command to set the *PATH* environment variable:

   `export PATH=`*<path_to_SoCEDS_installation_dir>*`/ds-5/sw/gcc/bin:$PATH`
2. Navigate to the *<local_path_to_exm_opencl_hello_world>*`/hello_world` directory.
3. Invoke the `make -f Makefile` command. Alternatively, you can simply invoke the `make` command.

The hello_world executable is in the <*local_path_to_exm_opencl_hello_world*>/hello_world/bin directory.

4.2.7.2. Running the Host Application

For Linux systems, execute the hello_world.aocx file on the SoC FPGA by running the host application you built from the ARM-specific Makefile.

1. Log into your SoC FPGA board.
2. Copy the hello_world.aocx hardware configuration file and the hello_world host executable from their current directories to the board.
3. Verify that the *LD_LIBRARY_PATH* environment variable setting includes $*INTELFPGAOCLSDKROOT*/host/arm32/lib. Run the command echo $LD_LIBRARY_PATH.
   If you ran the init_opencl.sh script, the *LD_LIBRARY_PATH* setting should point to $*INTELFPGAOCLSDKROOT*/host/arm32/lib.
4. To execute the kernel on the SoC FPGA, at a command prompt, navigate to the host executable directory and run the hello_world host executable.

4.2.8. Uninstalling the Intel FPGA RTE for OpenCL

To uninstall the RTE from the SoC FPGA board, delete the RTE directory and restore all modified environment variables to their previous settings.

1. Navigate to the root directory in the SoC FPGA board's file system that contains the <*rte_destination_directory*> directory.
2. Type rm -rf <*rte_destination_directory*> to remove the RTE directory.
3. Remove the environment variable settings by typing the following commands:

   unset *INTELFPGAOCLSDKROOT*

   unset *PATH*

   unset *LD_LIBRARY_PATH*
4. Uninstall the Intel FPGA SDK for OpenCL on your host system and unset the corresponding environment variables.

Related Information

Uninstalling the Intel FPGA SDK for OpenCL

A. Intel FPGA RTE 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 RTE for OpenCL Pro Edition Getting Started Guide
20.3	Intel FPGA RTE for OpenCL Pro Edition Getting Started Guide
20.2	Intel FPGA RTE for OpenCL Pro Edition Getting Started Guide
19.1	Intel FPGA RTE for OpenCL Pro Edition Getting Started Guide
18.1	Intel FPGA RTE for OpenCL Pro Edition Getting Started Guide
18.0	Intel FPGA RTE for OpenCL Pro Edition Getting Started Guide
17.1	Intel FPGA RTE for OpenCL Getting Started Guide
17.0	Intel FPGA RTE for OpenCL Getting Started Guide
16.1	Intel FPGA RTE for OpenCL Getting Started Guide
16.0	Altera RTE for OpenCL Getting Started Guide
15.1	Altera RTE for OpenCL Getting Started Guide
15.0	Altera RTE for OpenCL Getting Started Guide
14.1	Altera RTE for OpenCL Getting Started Guide

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

Document Version	Intel Quartus Prime Version	Changes
2021.03.29	21.1	• Maintenance release.
2020.12.14	20.4	• Updated the RTE versions.
2020.09.28	20.3	• Added more information to the *GCC Requirement* page. • Updated the RTE versions. • Updated a note about the lack of support for Windows and Linux BSPs in *Managing an FPGA Board* and *Installing an FPGA Board*.
2020.06.22	20.2	• Removed all references to Akamai and direct download methods. • Added a note about `-nodeps` option and made minor updates to the output of FCD and ICD setup output in *Building the Host Application*. • Added a note about downloading the BSPs online since they are no longer shipped with SDK in *Overview of the Intel FPGA RTE for OpenCL Pro Edition Setup Process* and *Managing an FPGA board* topics. • Updated some of the prerequisites in *Prerequisites for the Intel FPGA RTE for OpenCL Pro Edition*. • Updated the `init_opencl.bat` script output in *Setting the Intel FPGA RTE for OpenCL Pro Edition User Environment Variables* • Removed the `board` directory from *Contents of the Intel FPGA RTE for OpenCL Pro Edition*. • Updated all references to downloading OpenCL design examples from Intel website and replaced them with instructions to access examples from the *Intel FPGA SDK for OpenCL* installation. • Updated *RTE Setup Process for x86-64 Systems* and *RTE Setup Process for SoC FPGA* diagrams. • Added a new topic *GCC Requirement*.
2020.04.15	19.1	• Changed `rpm -e aocl-rte` to `rpm -e intel-fpga-opencl-pro-rte` in *Uninstalling the Software*. • Changed `aclrte-arm32.tgz` to `intel-fpga-opencl-pro-rte-32bit-arm.tgz` in *Installing the Intel FPGA RTE for OpenCL Pro Edition onto the SoC FPGA Board*. • Changed `mv aclrte-arm32.tgz <rte_destination_directory>` to `mvintel-fpga-opencl-pro-rte-32bit-arm.tgz <rte_destination_directory>` in *Installing the Intel FPGA RTE for OpenCL Pro Edition onto the SoC FPGA Board*. • Changed `tar -xvfz aclrte-arm32.tgz` to `tar -xzvf intel-fpga-opencl-pro-rte-32bit-arm.tgz` in *Installing the Intel FPGA RTE for OpenCL Pro Edition onto the SoC FPGA Board*. • Updated the list of variables to be set, in *Installing the Intel FPGA RTE for OpenCL Pro Edition onto the SoC FPGA Board*.
2019.09.30	19.1	In *Installing the Intel FPGA RTE for OpenCL Pro Edition* topic, made the following updates:

continued...

Document Version	Intel Quartus Prime Version	Changes
		• Changed `/usr/intelFPGA_pro/<version>/aclrte_linux64` to `/usr/intel/<version>/intel-fpga-opencl-pro-rte`. • Changed `aocl-rte-<version>.x86_64.rpm` to `intel-fpga-opencl-pro-rte-64bit-linux.rpm`. • Changed `opt/intelFPGA_pro/aclrte-linux64` to `/opt/intel/intel-fpga-opencl-pro-rte`.
2019.04.01	19.1	• In Installing an FPGA Board on page 17 and Installing an FPGA Board on page 28, removed information/steps about the environment variable AOCL_BOARD_PACKAGE_ROOT and added more information to step 5 about what you can do when you do not have administrative privilege. • In Building the Host Application on page 21, mentioned about `aocl diagnose` command. • In Uninstalling an FPGA Board on page 24 and Uninstalling an FPGA Board on page 33 mentioned about administrative rights and root privileges, respectively, mentioned about `-fcd-only` in a note, added a step and removed a step about the environment variable AOCL_BOARD_PACKAGE_ROOT. • In Installing an FPGA Board on page 17, changed `<home_directory>\altera\<version>\aclrt-windows64` to `<home_directory>\intelFPGA_pro\<version>\aclrte-windows64` • In Setting the Intel FPGA RTE for OpenCL Pro Edition User Environment Variables on page 15, removed *AOCL_BOARD_PACKAGE_ROOT*. • In Programming the Flash Memory of an FPGA on page 19 and Programming the Flash Memory of an FPGA on page 30, removed a step about verify the *AOCL_BOARD_PACKAGE_ROOT* setting. • In Uninstalling the Software on page 23, changed `altera` to `intelFPGA_pro` • In Installing the Intel FPGA RTE for OpenCL Pro Edition on page 25, changed `/usr/altera/<version>/aclrte_linux64` to `/usr/intelFPGA_pro/<version>/aclrte_linux64` and `opt/altera/aclrte-linux64` to `opt/intelFPGA_pro/aclrte-linux64`. • In Setting the Intel FPGA RTE for OpenCL Pro Edition User Environment Variables on page 27, removed *AOCL_BOARD_PACKAGE_ROOT* variable and its example script output. • In Installing the Intel FPGA SDK for OpenCL Pro Edition for SoC FPGA on page 37, changed `<home_directory>\alteraa\<version>\hld` to `<home_directory>\intelFPGA_pro\<version>\hld` • In Uninstalling the Intel FPGA RTE for OpenCL on page 41 and Uninstalling the Intel FPGA RTE for OpenCL on page 46, removed *AOCL_BOARD_PACKAGE_ROOT* variable. • In Building the Host Application on page 40 and Building the Host Application on page 45, removed step about modifying and verifying *AOCL_BOARD_PACKAGE_ROOT* variable. • Outdated instructions in Recompiling the Linux Kernel Driver on page 38 were corrected.
2019.01.14	18.1	• In Prerequisites for the Intel FPGA RTE for OpenCL Pro Edition on page 4, updated Microsoft* Visual Studio* version required to Visual Studio 2015 or later.

continued...

Document Version	Intel Quartus Prime Version	Changes
2018.12.24	18.1	• Rectified the instructions for Windows in 4 on page 17 to reflect the correct syntax. • Updated the download center path in Downloading the Intel FPGA SDK for OpenCL and the SoC EDS on page 41 and Downloading the Intel FPGA SDK for OpenCL and the SoC EDS on page 36.
2018.09.24	18.1	• Maintenance release
2018.05.04	18.0	• Removed information related to the Intel FPGA RTE for OpenCL Standard Edition, the Intel FPGA SDK for OpenCL Standard Edition, and the Intel SoC FPGA Embedded Development Suite Standard Edition. • Updated the *Getting Started with the Intel FPGA RTE for OpenCL Pro Edition for ARMv7-A SoC FPGA* chapter to include Intel Arria 10 SoC information.

Date	Version	Changes
November 2017	2017.11.04	• Rebranded the following: — `ALTERAOCLSDKROOT` to `INTELFPGAOCLSDKROOT` — *CL_CONTEXT_EMULATOR_DEVICE_ALTERA* to *CL_CONTEXT_EMULATOR_DEVICE_INTELFPGA* • In Prerequisites for the Intel FPGA RTE for OpenCL Pro Edition on page 4, changed Microsoft Visual Studio version 2010 Professional as Microsoft Visual Studio Professional version 2010 or later. • In Building the Host Application on page 21, updated references to Microsoft Visual Studio 2015 as Microsoft Visual Studio.
May 2017	2017.05.05	• Rebranded the Altera Client Driver (ACD) to FPGA Client Driver (FCD). • Updated the download instructions in *Downloading the Intel FPGA RTE 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 Altera RTE for OpenCL to Intel FPGA RTE for OpenCL. • Rebranded Altera SoC Embedded Design Suite to Intel SoC FPGA Embedded Design Suite. • Rebranded ARM DS-5 Altera Edition Toolkit to Arm DS-5 Intel SoC FPGA Edition Toolkit. • Deprecated and removed support for big-endian system, resulting in the following documentation changes: — In *Prerequisites for the Altera RTE for OpenCL*, removed "Red Hat Enterprise Linux 6 on big-endian system" from the list of supported operating systems. — In *Contents of the Altera RTE for OpenCL*, removed the Big-Endian System Directory column from the table of contents in the RTE installation directory. — In *Downloading the Altera RTE for OpenCL* for Linux, removed from Step 6 the choice to select the OpenCL PowerPC RPM installation package. — Removed the topics *Installing the Altera RTE for OpenCL on Big-Endian Systems* and *Setting the Environment Variables on Big-Endian Systems*. — In *Installing and FPGA Board* for Linux, removed information on running the `init_opencl` script on big-endian systems, and removed related link to *Setting the Environment Variables on Big-Endian Systems*.

continued...

Date	Version	Changes
		• In *Installing an FPGA Board* for 64-bit Windows and Linux, provided the following updates: — Noted that the SDK supports installation of multiple Custom Platforms. To use the SDK utilities on each board in a system with multiple Custom Platforms, the *AOCL_BOARD_PACKAGE_ROOT* environment variable setting must correspond to the Custom Platform subdirectory of the associated board. — Noted that in a system with multiple Custom Platforms, the host program should use ACD to discover the boards instead of directly linking to the MMD libraries. • In *Building the Host Application* for 64-bit Windows, outlined the prerequisite tasks for setting up ACD and ICD for use with Microsoft Visual Studio 2015 prior to building the host application. • Updated all RTE output for a successful kernel execution.
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.
November 2015	2015.11.02	• Added Windows 8.1 to supported Windows versions. • Added the following figures to illustrate the RTE setup processes for x86-64, big-endian, and SoC systems: — *RTE Setup Process for x86-64 and Big-Endian Systems* — *RTE Setup Process for SoC* • Modified software download and installation instructions for SoC to include the new tar file installation package. • Modified instructions for executing the hello_world OpenCL example design onto a device. You must create your own `.aocx` file from the `hello_world.cl` file on a separate development machine, and then use the RTE to deploy the `.aocx` file onto the device. • Removed licensing sections because an AOCL license is not necessary to run the RTE.
May 2015	15.0.0	• Reorganized instructions into the following sections: — Introduction to the RTE — Getting Started with the RTE for Windows — Getting Started with the RTE for Linux and Big-Endian Systems — Getting Started with the RTE for Altera ARMv7-A SoC, which is further divided into Windows and Linux instructions
December 2014	14.1.0	• Reorganized information flow. • Updated Red Hat Enterprise Linux (RHEL) version support. • Added licensing information in the *Licensing the Software* section. • Included information on the `init_opencl` script for setting environment variables. • Updated the board uninstallation instructions to include the invocation of the `aocl uninstall` utility command.
June 2014	14.0.0	Initial release.