OpenCL* on Intel® Programmable Acceleration Card with Intel® Arria® 10 GX FPGA Quick Start User Guide
# Contents

1 About this Document ........................................................................................................ 3  
  1.1 Conventions ........................................................................................................... 3  
  1.2 Acceleration Glossary ............................................................................................... 3  
2 Introduction .................................................................................................................... 4  
  2.1 Release Content ..................................................................................................... 4  
3 Setting Up the Host Machine ............................................................................................ 5  
  3.1 Installing the OpenCL Runtime ................................................................................. 5  
  3.2 Installing the Release .............................................................................................. 5  
  3.3 Installing the OpenCL BSP ....................................................................................... 6  
  3.4 Setting Up Permissions ............................................................................................ 6  
  3.5 Initializing the Run Time Environment (RTE) ............................................................... 7  
  3.6 Setup Summary ..................................................................................................... 7  
4 Running Diagnostics ........................................................................................................ 8  
5 OpenCL Support for Multi-Card Systems ........................................................................ 10  
6 Running Samples ........................................................................................................... 12  
  6.1 Running Hello World .............................................................................................. 12  
  6.2 Running Vector Add ............................................................................................... 13  
7 Compiling OpenCL Kernels ............................................................................................. 14  
8 Document Revision History ............................................................................................ 15
1 About this Document

1.1 Conventions

Table 1. Document Conventions

<table>
<thead>
<tr>
<th>Convention</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>Precedes a command that indicates the command is to be entered as root.</td>
</tr>
<tr>
<td>$</td>
<td>Indicates a command is to be entered as a user.</td>
</tr>
<tr>
<td>This font</td>
<td>Filenames, commands, and keywords are printed in this font. Long command lines are printed in this font. Although long command lines may wrap to the next line, the return is not part of the command; do not press enter.</td>
</tr>
<tr>
<td>&lt;variable_name&gt;</td>
<td>Indicates the placeholder text that appears between the angle brackets must be replaced with an appropriate value. Do not enter the angle brackets.</td>
</tr>
</tbody>
</table>

1.2 Acceleration Glossary

Table 2. Acceleration Stack for Intel® Xeon® CPU with FPGAs Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acceleration Stack for Intel® Xeon® CPU with FPGAs</td>
<td>Acceleration Stack</td>
<td>A collection of software, firmware and tools that provides performance-optimized connectivity between an Intel FPGA and an Intel Xeon processor.</td>
</tr>
<tr>
<td>Intel Programmable Acceleration Card with Intel Arria® 10 GX FPGA</td>
<td>Intel PAC with Arria 10</td>
<td>PCIe* accelerator card with an Intel Arria 10 FPGA. Programmable Acceleration Card is abbreviated PAC. Contains a FPGA Interface Manager (FIM) that pairs with an Intel Xeon processor over PCIe bus.</td>
</tr>
<tr>
<td>Intel Xeon Processor with Integrated FPGA</td>
<td>Integrated FPGA Platform</td>
<td>Intel Xeon plus FPGA platform with the Intel Xeon and an FPGA in a single package and sharing a coherent view of memory via Quick Path Interconnect (QPI).</td>
</tr>
</tbody>
</table>

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

*Other names and brands may be claimed as the property of others.
2 Introduction

This user guide describes how to get started with the OpenCL* on the Intel PAC with Arria 10 1.0 Beta Release. The instructions use the precompiled OpenCL kernels included in this 1.0 Beta Release. This user guide also includes a brief introduction to compiling OpenCL kernels.

OpenCL designs comprise two components, the kernel and the host. The kernel includes the accelerator code. The host runs on the host machine. The accelerator card plugs into the host machine.

Note: You must have root permission on the host machine to setup OpenCL.

Related Links

- Intel FPGA SDK for Open Computing Language (OpenCL) web-page
  For more details about OpenCL.
- Intel FPGA SDK for OpenCL - Getting Started Guide
  For more information about installation and instructions on how to compile an example OpenCL application.

2.1 Release Content

The release includes the following files located in the dcp_1.0_rush_creek_b11/opencl folder:

- 1.0 OpenCL Board Support Package (BSP):
  - opencl_bsp_* .tar.gz
- OpenCL example designs tested with:
  - exm_opencl_hello_world_x64_linux.tgz
  - exm_opencl_vector_add_x64_linux.tgz
- Pre-compiled kernels <aocx>:
  - hello_world.aocx
  - vector_add.aocx
3 Setting Up the Host Machine

Note: You must have a working setup to proceed. Also, ensure that the Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA is up and running.

3.1 Installing the OpenCL Runtime

You must install the Linux OpenCL Runtime devices before running the example designs. Installation requires root permissions on the host machine.

1. Navigate to the Intel FPGA SDK for OpenCL download page.
2. Click on the RTE (runtime environment) tab.
3. Select Intel FPGA Runtime Environment for OpenCL Linux x86-64 RPM and click Download.
4. Accept the License Agreement.
5. Log in when requested. (Create an account if you do not already have one.)
6. Save the file, acl-rte-17.0.0-1.x86_64.rpm.
7. Install the runtime by running the command:
   
   ```bash
   # sudo yum install acl-rte-17.0.0-1.x86_64.rpm
   ```

3.2 Installing the Release

You build the OpenCL BSP provided with this release on top of the Acceleration Stack for Intel Xeon CPU with FPGAs. Follow instructions from the Intel Acceleration Stack Quick Start Guide for Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA to set up the Intel PAC with Arria 10 FPGA.

1. Install the Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA board.
2. Flash the Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA.
3. Install the FPGA driver.
4. Build the OPAE Open Programmable Acceleration Engine (OPAE) software.
5. Run the following command:
   
   ```bash
   # sudo ldconfig
   ```

6. You can now run the OPAE software in a non-virtualized environment or in a virtualized environment with Single Root IO Virtualization (SR-IOV) disabled. Complete the following additional steps for a virtualized environment that includes virtual functions (VFs) and SRIOV:
   a. For OpenCL functionality, load the OpenCL configuration before enabling SRIOV mode.
b. Set the `CL_CONTEXT_COMPILER_MODE_ALTERA` environment variable to disable FPGA configuration or reconfiguration during OpenCL host runtime:

```
$ CL_CONTEXT_COMPILER_MODE_ALTERA=3
```

After completing these steps, the following environment variable is available:

- `DCP_LOC` – points to the location of the extracted release archive.

*Note:* To avoid having to reset `DCP_LOC` after a reboot, save this variable to your shell initialization script.

### 3.3 Installing the OpenCL BSP

The OpenCL BSP is an archive file. To extract this file, create a working directory for the BSP and run the following commands:

```
$ tar xf $DCP_LOC/opencl/opencl_bsp_*.tar.gz
$ cd opencl_bsp
$ export AOCL_BOARD_PACKAGE_ROOT=`pwd`
```

To avoid having to reset the `AOCL_BOARD_PACKAGE_ROOT` environment variable after a reboot, save it to your shell initialization script.

### 3.4 Setting Up Permissions

Running OpenCL requires you to set various permissions and system parameters. Running the `setup_permissions.sh` script completes this task. The script uses `sudo` internally; consequently, it requires root privileges.

**Procedure:**

1. Run the script once, when you enable OpenCL for the first time on this host:

   ```
   $ $AOCL_BOARD_PACKAGE_ROOT/ linux64/libexec/setup_permissions.sh
   ```

2. Reboot the computer because some permanent settings only take effect after a reboot.

3. Some of the settings are not permanent. Consequently, you must rerun the `setup_permissions.sh` command after rebooting.

   ```
   $ $AOCL_BOARD_PACKAGE_ROOT/linux64/libexec/setup_permissions.sh
   ```
3.5 Initializing the Run Time Environment (RTE)

Before running OpenCL examples, you must initialize the RTE.

In addition to the previously mentioned variables, set the following additional environment variables for the RTE:

```
$ export ALTERAOCLSDKROOT=/opt/altera/aocl-rte
```

Run the OpenCL initialization script from the RTE:

```
$ source /opt/altera/aocl-rte/init_opencl.sh
```

3.6 Setup Summary

Each time you reboot the computer, you must complete the following steps to run the OpenCL examples:

- Set the following environment variables:
  - DCP_LOC
  - AOCL_BOARD_PACKAGE_ROOT
- Run the permissions script:
  ```
  $ $AOCL_BOARD_PACKAGE_ROOT/linux64/libexec/setup_permissions.sh
  ```
- Initialize RTE:
  ```
  $ source /opt/altera/aocl-rte/init_opencl.sh
  ```
4 Running Diagnostics

Before running diagnostics, load an OpenCL kernel to the board. The following instructions use the hello_world kernel, you may also use your own.

1. Load hello_world OpenCL kernel:

   ```bash
   $ aocl program acl0 $DCP_LOC/opencl/hello_world.aocx
   ```

2. Run the simple diagnostic utility:

   ```bash
   $ aocl diagnose
   ```

   Sample diagnostic output:

   ```
   aocl diagnose: Running diagnose from /mnt/Tools/<user_name>_tools/
dcp_1.0_rush_creek_b11/opencl/opencl_bsp/linux64/libexec
   ------------------------- acl0 -------------------------
   Vendor: Intel Corp
   Phys Dev Name  Status   Information
   pac_a10_f400000  Passed   PAC Arria 10 Platform (pac_a10_f400000)
   PCIe 04:00.0
   FPGA temperature = 47 degrees C.
   DIAGNOSTIC_PASSED
   ```

3. Run the advanced diagnostic:

   ```bash
   $ aocl diagnose acl0
   ```

   Sample advanced diagnostic output:

   ```
   aocl diagnose: Running diagnose from /mnt/Tools/<user_name>_tools/
dcp_1.0_rush_creek_b11/opencl/opencl_bsp/linux64/libexec
   Using platform: Intel(R) FPGA SDK for OpenCL(TM)
   Using Device with name: pac_a10 : PAC Arria 10 Platform (pac_a10_f400000)
   Using Device from vendor: Intel Corp clGetDeviceInfo
   CL_DEVICE_GLOBAL_MEM_SIZE = 8589934592
   clGetDeviceInfo CL_DEVICE_MAX_MEM_ALLOC_SIZE = 8588886016
   Memory consumed for internal use = 1048576
   Actual maximum buffer size 8588886016 bytes
   Writing 8191 MB to global memory...
   Allocated 1073741824 Bytes host buffer for large transfers
   Write speed: 6576.87 MB/s [6567.03 -> 6580.40]
   Reading and verifying 8191 MB from global memory ... 
   Read speed: 6915.93 MB/s [6885.51 -> 6926.66]
   Successfully wrote and readback 8191 MB buffer
   Transferring 262144 KBs in 512 512 KB blocks ... 3748.69 MB/s
   Transferring 262144 KBs in 256 1024 KB blocks ... 3870.31 MB/s
   Transferring 262144 KBs in 128 2048 KB blocks ... 4528.58 MB/s
   Transferring 262144 KBs in 64 4096 KB blocks ... 5405.89 MB/s
   Transferring 262144 KBs in 32 8192 KB blocks ... 5923.38 MB/s
   Transferring 262144 KBs in 16 16384 KB blocks ... 6246.18 MB/s
   ```
Transferring 262144 KBs in 8 32768 KB blocks ... 6427.20 MB/s
Transferring 262144 KBs in 4 65536 KB blocks ... 6624.94 MB/s
Transferring 262144 KBs in 2 131072 KB blocks ... 6774.19 MB/s
Transferring 262144 KBs in 1 262144 KB blocks ... 6855.82 MB/s

As a reference:
PCIe Gen1 peak speed: 250MB/s/lane
PCIe Gen2 peak speed: 500MB/s/lane
PCIe Gen3 peak speed: 985MB/s/lane

Writing 262144 KBs with block size (in bytes) below:

<table>
<thead>
<tr>
<th>Block Size</th>
<th>Avg</th>
<th>Max</th>
<th>Min</th>
<th>End-End (MB/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>524288</td>
<td>3489.94</td>
<td>3670.94</td>
<td>914.56</td>
<td>2766.34</td>
</tr>
<tr>
<td>1048576</td>
<td>3396.38</td>
<td>3473.50</td>
<td>3004.03</td>
<td>3040.98</td>
</tr>
<tr>
<td>2097152</td>
<td>4391.87</td>
<td>4528.58</td>
<td>3931.17</td>
<td>4104.98</td>
</tr>
<tr>
<td>4194304</td>
<td>5365.16</td>
<td>5405.89</td>
<td>4999.12</td>
<td>5157.90</td>
</tr>
<tr>
<td>8388608</td>
<td>5896.15</td>
<td>5923.38</td>
<td>5699.69</td>
<td>5816.82</td>
</tr>
<tr>
<td>16777216</td>
<td>6215.94</td>
<td>6246.18</td>
<td>6135.78</td>
<td>6150.34</td>
</tr>
<tr>
<td>33554432</td>
<td>6398.59</td>
<td>6427.20</td>
<td>6376.68</td>
<td>6358.53</td>
</tr>
<tr>
<td>67108864</td>
<td>6532.26</td>
<td>6542.09</td>
<td>6518.01</td>
<td>6516.37</td>
</tr>
<tr>
<td>134217728</td>
<td>6582.69</td>
<td>6590.64</td>
<td>6574.77</td>
<td>6580.38</td>
</tr>
<tr>
<td>268435456</td>
<td>6603.61</td>
<td>6603.61</td>
<td>6603.61</td>
<td>6603.61</td>
</tr>
</tbody>
</table>

As a reference:
PCIe Gen1 peak speed: 250MB/s/lane
PCIe Gen2 peak speed: 500MB/s/lane
PCIe Gen3 peak speed: 985MB/s/lane

Reading 262144 KBs with block size (in bytes) below:

<table>
<thead>
<tr>
<th>Block Size</th>
<th>Avg</th>
<th>Max</th>
<th>Min</th>
<th>End-End (MB/s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>524288</td>
<td>3396.10</td>
<td>3748.69</td>
<td>3144.31</td>
<td>2673.82</td>
</tr>
<tr>
<td>1048576</td>
<td>3396.38</td>
<td>3473.50</td>
<td>3004.03</td>
<td>3040.98</td>
</tr>
<tr>
<td>2097152</td>
<td>3682.68</td>
<td>4044.24</td>
<td>3623.88</td>
<td>3641.46</td>
</tr>
<tr>
<td>4194304</td>
<td>3981.96</td>
<td>4082.71</td>
<td>3878.14</td>
<td>3870.63</td>
</tr>
<tr>
<td>8388608</td>
<td>5006.64</td>
<td>5050.74</td>
<td>4849.07</td>
<td>4904.61</td>
</tr>
<tr>
<td>16777216</td>
<td>5835.33</td>
<td>5873.50</td>
<td>5783.57</td>
<td>5761.21</td>
</tr>
<tr>
<td>33554432</td>
<td>6319.69</td>
<td>6331.87</td>
<td>6305.91</td>
<td>6283.10</td>
</tr>
<tr>
<td>67108864</td>
<td>6621.20</td>
<td>6624.94</td>
<td>6618.82</td>
<td>6604.06</td>
</tr>
<tr>
<td>134217728</td>
<td>6772.23</td>
<td>6774.19</td>
<td>6770.28</td>
<td>6768.46</td>
</tr>
<tr>
<td>268435456</td>
<td>6855.82</td>
<td>6855.82</td>
<td>6855.82</td>
<td>6855.82</td>
</tr>
</tbody>
</table>

Write top speed = 6603.61 MB/s
Read top speed = 6855.82 MB/s
Throughput = 6729.71 MB/s

DIAGNOSTIC_PASSED
5 OpenCL Support for Multi-Card Systems

Before running an OpenCL application, program the PAC card with an Accelerator Function (AF) that includes the BSP logic. Use the aocl program command to load an aocx file to the PAC card. It is only necessary to program the AF one time per PAC card. After the initial programming, you can use the OpenCL API to load different applications to the PAC card using aocx program command.

Run the aocl diagnose -probe command to determine how many FPGAs the system includes. For example, running the aocl diagnose -probe command on a system with three PAC cards might show output similar to the following:

1. $ aocl diagnose -probe

   aocl diagnose: Running diagnose from /storage/shared/home_directories/
gsouther/regtest/2017-12-15/1101.53/adapt_remote_tests/dcp_1_0_skx/opencl/
boardtest/opencl_bsp_build/linux64/libexec
pac_a10_f200001
pac_a10_f200000
pac_a10_f200002

2. The following command programs the first card listed in 1 on page 10 Step 1:

   $ aocl program pac_a10_f200001 hello_world.aocx

   aocl program: Running program from /storage/shared/home_directories/
gsouther/regtest/2017-12-15/1101.53/adapt_remote_tests/dcp_1_0_skx/opencl/
boardtest/opencl_bsp_build/linux64/libexec
Program succeed.

3. The following command programs the second card listed in 1 on page 10 Step 1:

   $ aocl program pac_a10_f200000 hello_world.aocx

   aocl program: Running program from /storage/shared/home_directories/
gsouther/regtest/2017-12-15/1101.53/adapt_remote_tests/dcp_1_0_skx/opencl/
boardtest/opencl_bsp_build/linux64/libexec
Program succeed.

4. After programming the FPGAs, the aocl diagnose command provides information about them:

   $ aocl diagnose

   aocl diagnose: Running diagnose from /storage/shared/home_directories/
gsouther/regtest/2017-12-15/1101.53/adapt_remote_tests/dcp_1_0_skx/opencl/
boardtest/opencl_bsp_build/linux64/libexec

   ------------------------- acl0 -------------------------
   Vendor: Intel Corp
   Phys Dev Name  Status   Information
   pac_a10_f200001  Passed  PAC Arria 10 Platform (pac_a10_f200001)
PCIe 05:00.0  
FPGA temperature = 79 degrees C.

DIAGNOSTIC_PASSED

---------------------------------------------------------------------

------------------------- acl1 -------------------------
Vendor: Intel Corp

<table>
<thead>
<tr>
<th>Phys Dev Name</th>
<th>Status</th>
<th>Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>pac_a10_f200000</td>
<td>Passed</td>
<td>PAC Arria 10 Platform (pac_a10_f200000)</td>
</tr>
</tbody>
</table>

PCIE 03:00.0  
FPGA temperature = 79 degrees C.

DIAGNOSTIC_PASSED

---------------------------------------------------------------------
6 Running Samples

This section describes how to compile and run the host code for the provided samples using the precompiled OpenCL kernels.

6.1 Running Hello World

1. **Extract hello_world example:**

   ```
   $ cd $DCP_LOC/opencl
   $ mkdir exm_opencl_hello_world_x64_linux
   $ cd exm_opencl_hello_world_x64_linux
   $ tar xf ../exm_opencl_hello_world_x64_linux.tgz
   ```

2. **Build example:**

   ```
   $ cd hello_world
   $ make
   ```

3. **Copy aocx to example bin folder:**

   ```
   $ cp $DCP_LOC/opencl/hello_world.aocx ./bin/
   ```

4. **Run example:**

   ```
   $ ./bin/host
   ```

**Example sample output:**

```
Querying platform for info:
-------------------------------
CL_PLATFORM_NAME                         = Intel(R) FPGA SDK for OpenCL(TM)
CL_PLATFORM_VENDOR                       = Intel(R) Corporation
CL_PLATFORM_VERSION                      = OpenCL 1.0 Intel(R) FPGA SDK
CL_PLATFORM_NAME                         = Intel(R) FPGA SDK for OpenCL(TM), Version 17.0

Querying device for info:
-------------------------
CL_DEVICE_NAME                           = pac_a10 : PAC Arria 10 Platform (pac_a10_f400000)
CL_DEVICE_VENDOR                         = Intel Corp
CL_DEVICE_VENDOR_ID                      = 4466
CL_DEVICE_VERSION                        = OpenCL 1.0 Intel(R) FPGA SDK
CL_DEVICE_NAME                           = pac_a10 : PAC Arria 10 Platform
CL_DEVICE_VENDOR                         = Intel Corp
CL_DEVICE_VENDOR_ID                      = 4466
CL_DEVICE_VERSION                        = OpenCL 1.0 Intel(R) FPGA SDK
CL_DEVICE_NAME                           = pac_a10 : PAC Arria 10 Platform
CL_DEVICE_VENDOR                         = Intel Corp
CL_DEVICE_VENDOR_ID                      = 4466
CL_DEVICE_VERSION                        = OpenCL 1.0 Intel(R) FPGA SDK
CL_DRIVER_VERSION                        = 17.0
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
```
6.2 Running Vector Add

1. Extract example:

   $ cd $DCP_LOC/opencl
   $ mkdir exm_opencl_vector_add_x64_linux
   $ cd exm_opencl_vector_add_x64_linux
   $ tar xf ../exm_opencl_vector_add_x64_linux.tgz

2. Build example:

   $ cd vector_add
   $ make

3. Copy precompiled OpenCL kernel to bin folder:

   $ cp $DCP_LOC/opencl/vector_add.aocx ./bin

4. Run example:

   $ ./bin/host

Example sample output:

   Initializing OpenCL
   Platform: Intel(R) FPGA SDK for OpenCL(TM)
   Using 1 device(s)
     pac_a10 : PAC Arria 10 Platform (pac_a10_f400000)
   Using AOCX: vector_add.aocx
   Reprogramming device [0] with handle 1
   Launching for device 0 (1000000 elements)
   Time: 7.282 ms
   Kernel time (device 0): 3.451 ms
   Verification: PASS
7 Compiling OpenCL Kernels

Refer to the Intel FPGA SDK for OpenCL Getting Started Guide for more details, including how to compile OpenCL kernels.

After completing instructions from the current document and the Intel FPGA SDK for OpenCL Getting Started Guide, you can compile an OpenCL kernel to an aocx file using a command similar to the following:

```
$ aoc $DCP_LOC/opencl/exm_opencl_vector_add_x64_linux/vector_add/device/vector_add.cl
```

Before compilation, ensure that the environment is setup with correct BSP using the command:

```
aoc --list-boards
```

**Output**

Board list:
pac_a10

Related Links

- Intel FPGA SDK for Open Computing Language (OpenCL) web-page
  For more details about OpenCL.
- Intel FPGA SDK for OpenCL - Getting Started Guide
  For more information about installation and instructions on how to compile an example OpenCL application.
8 Document Revision History

<table>
<thead>
<tr>
<th>Date</th>
<th>Version</th>
<th>Changes</th>
</tr>
</thead>
</table>