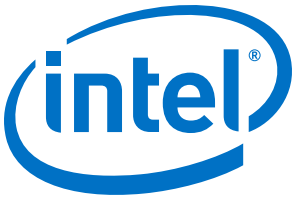


OpenCL* Quick Start User Guide

Intel FPGA Programmable Acceleration Card D5005

Updated for Intel® Acceleration Stack for Intel® Xeon® CPU with FPGAs: **2.0**

Subscribe
Send Feedback

UG-20221 | 2019.10.02
Latest document on the web: **PDF** | **HTML**

Contents

1. About this Document

1.1. Conventions

Table 1. Document Conventions

Convention	Description
#	Precedes a command that indicates the command is to be entered as root.
$	Indicates a command is to be entered as a user.
`This font`	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.
<variable_name>	Indicates the placeholder text that appears between the angle brackets must be replaced with an appropriate value. Do not enter the angle brackets.

1.2. Acceleration Glossary

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

Term	Abbreviation	Description
Intel® Acceleration Stack for Intel Xeon® CPU with FPGAs	Acceleration Stack	A collection of software, firmware, and tools that provides performance-optimized connectivity between an Intel FPGA and an Intel Xeon processor.
Intel FPGA Programmable Acceleration Card (Intel FPGA PAC)	Intel FPGA PAC	PCIe* FPGA accelerator card with Intel FPGA PAC. Contains a FPGA Interface Manager (FIM) that connects to an Intel Xeon processor over PCIe bus.

1.3. Acronyms

Table 3. Acronyms

Acronyms	Expansion	Description
AFU	Accelerator Functional Unit	Hardware Accelerator implemented in FPGA logic which offloads a computational operation for an application from the CPU to improve performance.
AF	Accelerator Function	Compiled Hardware Accelerator image implemented in FPGA logic that accelerates an application.
API	Application Programming Interface	A set of subroutine definitions, protocols, and tools for building software applications.
FIM	FPGA Interface Manager	The FPGA hardware containing the FPGA Interface Unit (FIU) and external interfaces for memory, networking, etc. The Accelerator Function (AF) interfaces with the FIM at run time.
OPAE	Open Programmable Acceleration Engine	The OPAE is a software framework for managing and accessing AFs.

2. Introduction

This user guide describes how to get started with the OpenCL* on the Intel FPGA Programmable Acceleration Card D5005. The instructions use the precompiled OpenCL kernels included in this version 2.0 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 Information

- Intel FPGA SDK for Open Computing Language (OpenCL) web-page
- Intel FPGA SDK for OpenCL Pro Edition Getting Started Guide

2.1. Release Content

When you install the Intel Acceleration Stack package and run the initialization script, the environment variables are set. The `$OPAE_PLATFORM_ROOT` points to the extracted installation package. The release includes the following files for OpenCL located in the `$OPAE_PLATFORM_ROOT/opencl` folder:

- 2.0 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

Prerequisites: Before running OpenCL, you must follow the instructions from the *Getting Started* section of the *Intel Acceleration Stack Quick Start Guide for Intel FPGA Programmable Acceleration Card D5005*, referred to as *Quick Start Guide* through out this document.

Attention:

- If you need the OpenCL compiler and tools to build and run OpenCL AFUs, download and install the Intel Acceleration Stack for Development. Installing the development software ensures that the OpenCL SDK is available under/home/<username>/inteldevstack/ or a Custom Directory, /<custom Directory>. This user guide refers to this path as /<dev Install Path>.
- If you only require the Intel FPGA SDK for the OpenCL deployment functionality, download and install the Intel Acceleration Stack for Runtime. Installing the runtime environment ensures that the OpenCL RTE is installed under /home/<username>/intelrtestack/ or a Custom Directory, /<custom Directory>. This user guide refers to this path as /<RTE Install Path>.
- Do not install the RTE and the DEV on the same host system. The DEV already contains the RTE.

Related Information

- Intel Acceleration Stack Quick Start Guide for Intel FPGA Programmable Acceleration Card D5005
- Installing the Runtime Package on the Host Machine
- Installing the Development Package on the Host Machine

3.1. Installing the Release

Follow the installation instructions from the *Quick Start Guide* to set up the Intel FPGA PAC D5005.

You can run the OPAE software in a non-virtualized environment with the Single Root I/O Virtualization (SR-IOV) disabled or in a virtualized environment with the SR-IOV enabled.

To run the OpenCL reference design in a virtualized environment that includes SR-IOV, complete the following additional steps:

1. Program the required OpenCL configuration from the host machine by typing the following command:

```
aocl program <device name> <filename>
```

Note: The 2.0 Release does not support partial reconfiguration in virtualized environment.

2. Enable virtualization using the instructions from section *Updating Settings Required for VFs* and section *Configuring the VF Port on the Host* of the *Quick Start Guide*.
3. Set the `CL_CONTEXT_COMPILER_MODE_INTELFPGA` environment variable in the virtual machine to disable FPGA configuration or reconfiguration during OpenCL host runtime:

```
export CL_CONTEXT_COMPILER_MODE_INTELFPGA=3
```

4. Run the required application from the virtual machine.
5. Disable virtualization using the instruction from section *Disconnecting the VF from the VM and Reconnecting to the PF* of the *Quick Start Guide*.

Related Information

- Updating Settings Required for VFs
- Configuring the VF Port on the Host
- Disconnecting the VF from the VM and Reconnecting to the PF

3.2. Initializing the Intel Acceleration Stack for OpenCL

The `init_env.sh` script performs all the initialization and setup for the Acceleration Stack for OpenCL. The script is available in either `/<RTE install path>/` or `/<DEV install path>/`.

Note: If this is your first time running `init_env.sh`, you must restart and rerun the script for permanent permissions and system parameter settings to take effect.

Note: Each time you restart the host or start a new shell, rerun the `init_env.sh` script. Most settings are temporary.

The script completes the following tasks:

- Exports the following environment variables:

Environment Variables	Description
OPAE_PLATFORM_ROOT	Points to the extracted Intel Acceleration Stack release.
AOCL_BOARD_PACKAGE_ROOT	Points to the unpacked OpenCL BSP.
INTELFPGAOCLSKDROOT	The Intel FPGA SDK for OpenCL installation directory.
ALTERAOCLSDKROOT	Builds and runs the OpenCL samples in the installation directory. *Note:* If you are using the Intel Acceleration Stack for Runtime Release 2.0, add the following line at the end of `init_env.sh` script: `export ALTERAOCLSDKROOT= $INTELFPGAOCLSDKROOT`
QUARTUS_HOME	Exported only if you are using the Intel Acceleration Stack for Development. Points to Intel Quartus® Prime installation used for compiles. *Note:* The Acceleration Stack for Development includes Intel Quartus Prime software. Use this version for all your OpenCL development and compiles as FIM on the board is developed using this particular Intel Quartus Prime version. The `init_env.sh` points to this version by default.
CL_CONTEXT_COMPILER_MODE_INTELFPGA	Set this to three to disable FPGA configuration or reconfiguration during OpenCL host runtime (only exported in the Development Stack Release 2.0 `init_env.sh` script). *Note:* If you are using the Intel Acceleration Stack for Runtime Release 2.0 in a non-virtualized environment, remove the following export statement from `init_env.sh`: `CL_CONTEXT_COMPILER_MODE_FPGA= 3`

- Runs the OpenCL initialization script to enable the runtime environment or the development environment (if installed) by running `init_opencl.sh`

  Note: If you are using the Intel Acceleration Stack for Development Release 2.0, move the following lines to the end of the `init_env.sh` script, after Quartus bin has been added to the PATH:

  ```
  echo source $INTELFPGAOCLSDKROOT/init_opencl.sh
  source $INTELFPGAOCLSDKROOT/init_opencl.sh >> /dev/null
  ```

- Sets various permissions and system parameters by running `setup_permissions.sh`
- Adds the Intel SDK for OpenCL (`aocl`) utility located at `$INTELFPGAOCLSDKROOT/bin` to your PATH

  Note: Ensure that you install the FPGA driver as per the instructions in the *Intel Acceleration Stack Quick Start Guide for Intel FPGA Programmable Acceleration Card D5005* and the `init_env.sh` script sources the `setup_permission.sh` script. Otherwise, add the following lines to the end of your `init_env.sh` script:

  ```
  if ls /dev/intel-fpga-* 1> /dev/null 2>&1; then
  source $AOCL_BOARD_PACKAGE_ROOT/linux64/libexec/setup_permissions.sh
  fi
  ```

  You must execute `setup_permisssion.sh` script after every reboot. Intel recommends you to include it as part of the init_env.sh script.

4. Running Diagnostics

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

1. Load `hello_world` OpenCL kernel:

```
aocl program acl0  $OPAE_PLATFORM_ROOT/opencl/hello_world.aocx
```

Sample program output:

```
aocl program: Running program from $OPAE_PLATFORM_ROOT/opencl/opencl_bsp \
/linux64/libexec
Program succeed.
```

2. Run the simple diagnostic utility:

```
aocl diagnose
```

Sample diagnostic output:

```

-------------------------------------------------
Device
Name:

acl0


BSP Install Location:
/home/inteldevstack/d5005_ias_2_0_pv/opencl/opencl_bsp

Vendor: Intel Corp

Physical Dev Name   Status            Information

pac_ee0000         Passed            Intel PAC Platform (pac_ee00001)
                                     PCIe 55:00.0
                                     FPGA temperature = 0 degrees C.

DIAGNOSTIC_PASSED

-------------------------------------------------------------
```

3. Run the advanced diagnostic:

```
aocl diagnose acl0
```

Sample advanced diagnostic output:

```
Using platform: Intel(R) FPGA SDK for OpenCL(TM)
Using Device with name: pac_s10_dc : Intel PAC Platform (pac_ee00001)
Using Device from vendor: Intel Corp
clGetDeviceInfo CL_DEVICE_GLOBAL_MEM_SIZE = 34359738368
clGetDeviceInfo CL_DEVICE_MAX_MEM_ALLOC_SIZE = 34359738368
Allocated 34359738368 bytes
Actual maximum buffer size = 34359738368 bytes
Writing 32768 MB to global memory ...
```

```
Allocated 1073741824 Bytes host buffer for large transfers
Write speed: 6887.51 MB/s [6662.47 -> 6981.26]
Reading and verifying 32768 MB from global memory ...
Read speed: 6569.58 MB/s [6349.56 -> 6637.72]
Successfully wrote and readback 32768 MB buffer

Transferring 262144 KBs in 512 512 KB blocks ... 4396.40 MB/s
Transferring 262144 KBs in 256 1024 KB blocks ... 4518.01 MB/s
Transferring 262144 KBs in 128 2048 KB blocks ... 4500.23 MB/s
Transferring 262144 KBs in 64 4096 KB blocks ... 5248.65 MB/s
Transferring 262144 KBs in 32 8192 KB blocks ... 5807.63 MB/s
Transferring 262144 KBs in 16 16384 KB blocks ... 6241.43 MB/s
Transferring 262144 KBs in 8 32768 KB blocks ... 6544.45 MB/s
Transferring 262144 KBs in 4 65536 KB blocks ... 6717.46 MB/s
Transferring 262144 KBs in 2 131072 KB blocks ... 6768.89 MB/s
Transferring 262144 KBs in 1 262144 KB blocks ... 6556.16 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:

Block_Size Avg    Max    Min    End-End (MB/s)
  524288 2776.84 3721.44 693.83 2356.68
 1048576 2832.61 3985.10 2151.36 2579.25
 2097152 3949.87 4084.18 3413.17 3854.91
 4194304 5121.64 5248.65 4355.35 5058.67
 8388608 5193.31 5807.63 4543.09 5136.54
16777216 5871.76 6241.43 5172.29 5832.21
33554432 6372.32 6544.45 6010.24 6350.70
67108864 6665.69 6717.46 6609.32 6655.30
134217728 6707.64 6768.89 6647.48 6702.68
268435456 6556.16 6556.16 6556.16 6556.16

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

Block_Size Avg    Max    Min    End-End (MB/s)
  524288 4067.56 4396.40 3540.33 3550.93
 1048576 4397.34 4518.01 3734.66 4145.41
 2097152 4402.93 4500.23 4077.02 4301.13
 4194304 4551.87 4648.28 4338.68 4513.96
 8388608 4461.48 4547.12 4341.88 4445.99
16777216 5251.08 5309.37 5164.30 5229.90
33554432 5852.69 5979.23 5761.73 5839.81
67108864 6205.49 6295.10 6134.20 6199.51
134217728 6259.76 6416.61 6110.39 6257.51
268435456 6236.09 6236.09 6236.09 6236.09

Write top speed = 6768.89 MB/s
Read top speed = 6416.61 MB/s
Throughput = 6592.75 MB/s

DIAGNOSTIC_PASSED
```

5. OpenCL Support for Multi-Card Systems

Before running an OpenCL application, program the Intel FPGA PAC with an Accelerator Function (AF) that includes the BSP logic. Use the `aocl` program command to load an `aocx` file to the Intel FPGA PAC.

Note: For a system with one Intel FPGA PAC, Intel recommends that you allocate the number of hugepages to 20. If your system has multiple Intel FPGA PACs, you must allocate 20 hugepages per card. For example, a system with four Intel FPGA PAC requires of total 80 hugepages.

To set the hugepages to 80, enter the following command:

```
sudo sh -c "echo 80 > /sys/kernel/mm/hugepages/hugepages-2048kB \
/nr_hugepages"
```

Run the `aocl diagnose` command to determine how many FPGAs the system includes. For example, running the `aocl diagnose` command on a system with two Intel FPGA PAC might show output similar to the following:

1. `aocl diagnose`

```
------------------------------------------------
Device Name:
acl0

BSP Install Location:
$OPAE_PLATFORM_ROOT/opencl/opencl_bsp

Vendor: Intel Corp

Phys Dev Name    Status           Information

pac_ee00000      Uninitialized   OpenCL BSP not loaded. Must load BSP
                                  using command:
                                  'aocl program <device_name> <aocx_file>'
                                  before running OpenCL programs using
                                  this device

DIAGNOSTIC_PASSED
------------------------------------------------------------

------------------------------------------------
Device Name:
acl1

BSP Install Location:
$OPAE_PLATFORM_ROOT/opencl/opencl_bsp

Vendor: Intel Corp

Phys Dev Name    Status           Information

pac_ee00001  Uninitialized    OpenCL BSP not loaded. Must load BSP
                                  using command:)
                                  'aocl program <device_name> <aocx_file>'
```

```
                                    before running OpenCL programs using
                                    this device

DIAGNOSTIC_PASSED
------------------------------------------------------------
```

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

   ```
   aocl program acl0 $OPAE_PLATFORM_ROOT/opencl/hello_world.aocx
   ```

   ```
   aocl program: Running program from $OPAE_PLATFORM_ROOT/opencl \
   /opencl_bsp

   Program succeed.
   ```

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

   ```
   aocl program acl1 $OPAE_PLATFORM_ROOT/opencl/hello_world.aocx
   ```

   ```
   aocl program: Running program from $OPAE_PLATFORM_ROOT/opencl \
   /opencl_bsp

   Program succeed.
   ```

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

   ```
   aocl diagnose
   ```

   ```
   ------------------------------------------------
   Device Name:
   acl0

   BSP Install Location:
   $OPAE_PLATFORM_ROOT/opencl/opencl_bsp

   Vendor: Intel Corp

   Phys Dev Name   Status   Information

   pac_ee00000 Passed   Intel PAC Platform (pac_ee00000)
                            PCIe 55:00.0
                            FPGA temperature = 0 degrees C.

   DIAGNOSTIC_PASSED
   ------------------------------------------------------------

   Device Name:
   acl1

   BSP Install Location:
   $OPAE_PLATFORM_ROOT/opencl/opencl_bsp

   Vendor: Intel Corp

   Phys Dev Name   Status   Information

   pac_ee00001 Passed  Intel PAC Platform (pac_ee00001)
                            PCIe 55:00.0
                            FPGA temperature = 0 degrees C.

   DIAGNOSTIC_PASSED
   ------------------------------------------------------------
   ```

   Note: You can run the advanced diagnostic on any specific device in your multi-card system using the following command:

   ```
   aocl diagnose <device name>
   ```

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 $OPAE_PLATFORM_ROOT/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 $OPAE_PLATFORM_ROOT/opencl/hello_world.aocx ./bin/
```

4. Program the aocx file:

```
aocl program acl0 ./bin/hello_world.aocx
```

5. 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 for
OpenCL(TM), Version 18.1.2

Querying device for info:
========================
sched_setaffinity: Invalid argument
CL_DEVICE_NAME                           = pac_s10_dc : Intel PAC Platform
(pac_ee00001)
CL_DEVICE_VENDOR                         = Intel Corp
CL_DEVICE_VENDOR_ID                      = 4466
CL_DEVICE_VERSION                        = OpenCL 1.0 Intel(R) FPGA SDK for
OpenCL(TM), Version 18.1.2
CL_DRIVER_VERSION                        = 18.1
CL_DEVICE_ADDRESS_BITS                   = 64
sched_setaffinity: Invalid argument
```

```
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                 = 34359738368
CL_DEVICE_IMAGE_SUPPORT                   = false
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        = 8589934592
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
sched_setaffinity: Invalid argument
Using AOCX: hello_world.aocx

Kernel initialization is complete.
Launching the kernel...

sched_setaffinity: Invalid argument
Thread #2: Hello from Altera's OpenCL Compiler!

Kernel execution is complete.
```

6.2. Running Vector Add

1. Extract example:

```
cd $OPAE_PLATFORM_ROOT/opencl

mkdir exm_opencl_vector_add_x64_linux

cd exm_opencl_vector_add_x64_linux

tar xzvf ../exm_opencl_vector_add_x64_linux.tgz
```

2. Build example:

```
cd vector_add

make
```

3. Copy precompiled OpenCL kernel to bin folder:

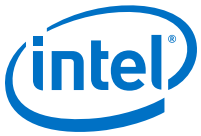
```
cp $OPAE_PLATFORM_ROOT/opencl/vector_add.aocx ./bin
```

4. Program the aocx file:

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

5. Run example:

```
./bin/host
```



Example sample output:

```
Initializing OpenCL
Platform: Intel(R) FPGA SDK for OpenCL(TM)
Using 2 device(s)
  pac_s10_dc : Intel PAC Platform (pac_ee00001)
  pac_s10_dc : Intel PAC Platform (pac_ee00000)
Using AOCX: vector_add.aocx
Launching for device 0 (500000 elements)
Launching for device 1 (500000 elements)

Time: 6.814 ms
Kernel time (device 0): 1.817 ms
Kernel time (device 1): 2.094 ms

Verification: PASS
```

7. Compiling OpenCL Kernels

Before compiling an OpenCL kernel, you must install the Intel Acceleration Stack for Development.

1. Set the user environment variable using one of the following commands:

```
source <DEV Install Path>init_env.sh
source <DEV Install Path>/intelFPGA_pro/hld/init_opencl.sh
export ALTERAOCLSDKROOT=$INTELFPGAOCLSDKROOT
```

2. Ensure that the environment is setup with correct BSP using the following command:

```
aoc -list-boards

Output
Board list:
pac_s10_dc
Board Package: /home/inteldevstack/d5005_ias_2_0_pv/opencl/opencl_bsp
```

3. Compile an OpenCL Kernel to an `aocx` using commands similar to the following:

```
cd $OPAE_PLATFROM_ROOT/opencl/vector_add

aoc device/vector_add.cl -o bin/vector_add.aocx -board pac_s10_dc
```

Related Information

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

7.1. Checking Timing Results

Intel recommends that you check for timing failures after compilation of the `aocx` file.

Check the compilation directory for the presence of the following report files:

```
afu_default.failing_clocks.rpt
```

```
afu_default.failing_paths.rpt
```

For example, after compiling `vector_add.cl`, locate the `$OPAE_PLATFORM_ROOT/opencl/vector_add/bin/vector_add` directory. If there is a timing violation, this directory contains the failing report files. The failing report files indicate that the timing is not clean and the functional correctness cannot be guaranteed.

If OpenCL kernel compilation results in timing violations, Intel recommends to retry compilation with a different seed (`aoc <kernel.cl> -seed= <value>`).

For example,

```
aoc vector_add.cl -seed 2
aoc vector_add.cl -seed 3
aoc vector_add.cl -seed 63
```

A. Disabling Non-Uniform Memory Access (NUMA) and DMA Worker Threads to Optimize PCIe Bandwidth

OpenCL transfers data from the host to the FPGA device using the DMA. By default, there is a DMA worker thread that performs the transaction and triggers a callback function when the transaction completes. The DMA worker thread tends to improve the overall performance by allowing the host program to continue working while the DMA transfer is in progress.

By default, the runtime uses `numactl` to keep the DMA worker thread on the same NUMA node as the main OpenCL thread. This reduces performance overhead associated with transferring data to a worker thread on a different node.

Although the defaults are intended to provide best performance, on some systems, users may want to disable the worker thread to improve PCIe bandwidth, or the NUMA affinity to allow the OS more freedom in scheduling threads.

To experiment with tuning the memory transfer performance, the OpenCL MMD provides two environment variables:

- `DISABLE_NUMA_AFFINITY_ENV`: Disables the settings of the CPU affinity. This allows the Operating System (OS) to schedule the DMA thread on any core. You can enable this environment variable by typing the following command:

  ```
  export DISABLE_NUMA_AFFINITY_ENV=yes
  ```

- `DISABLE_DMA_WORK_THREAD_ENV`: Disables the DMA worker thread entirely. This converts large data transfers into a blocking operation in the host code. You can enable this environment variable by typing the following command:

  ```
  export DISABLE_DMA_WORK_THREAD_ENV=yes
  ```

B. Document Revision History for OpenCL Quick Start User Guide

Document Version	Intel Acceleration Stack Version	Changes
2019.10.02	2.0 (supported with Intel Quartus Prime Pro Edition Edition 18.1.2)	• Added missing exports in section *Initializing the Intel Acceleration Stack for OpenCL*. • Modified command to set the hugepages in section *OpenCL Support for Multi-Card Systems*. • Added `aocl program` step in section *Running Hello World*. • Made the following changes in section *Checking Timing Results*: — Corrected the report file names. — Modified the command to do compilation with a different seed.
2019.08.05	2.0 (supported with Intel Quartus Prime Pro Edition Edition 18.1.2)	Initial release.