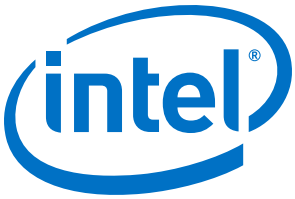


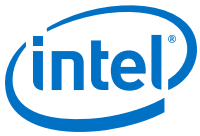
Intel Acceleration Stack Quick Start Guide for Intel® Programmable Acceleration Card with Intel® Arria® 10 GX FPGA

Updated for Intel® Acceleration Stack for Intel® Xeon® CPU with FPGAs: **1.2**

Subscribe
Send Feedback

UG-20166 | 2019.08.05
Latest document on the web: **PDF** | **HTML**

Contents



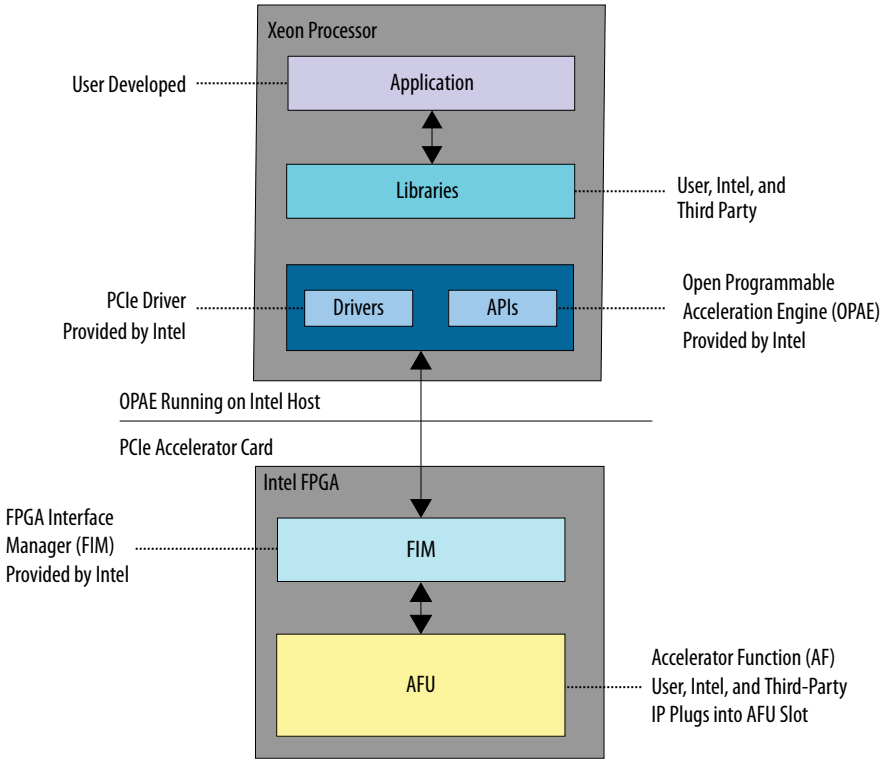
1. Introduction to the Intel® Acceleration Stack for Intel® Xeon® CPU with FPGAs

This guide provides a brief introduction to the Intel® Programmable Acceleration Card with Intel Arria® 10 GX FPGA, abbreviated as Intel PAC with Intel Arria 10 GX FPGA in this document. This guide provides the instructions to load and run a loopback test, *Hello FPGA*, in both non-virtualized and virtualized environments.

The Acceleration Stack is a collection of software, firmware, and tools that allows both software and RTL developers to take advantage of the power of Intel FPGAs. By offloading computationally intensive tasks to the FPGA, the acceleration platform frees the Intel Xeon® processor for other critical processing tasks.

The Intel PAC with Intel Arria 10 GX FPGA, an accelerator card, connects to the Intel Xeon processor through the PCIe* interface on the motherboard.

Figure 1. Overview of the Intel PAC with Intel Arria 10 GX FPGA Platform Hardware and Software



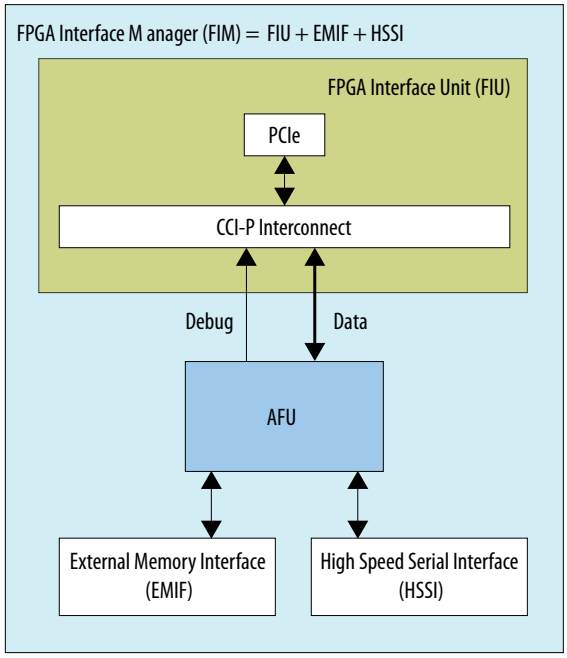
To take advantage of the flexibility of the FPGA, you can reconfigure a predefined, partial reconfiguration (PR) region of the Intel Arria 10 GX FPGA at run time. You can design multiple Accelerator Functional Units (AFUs) to swap in and out of this PR region.The Open Programmable Acceleration Engine (OPAE) software running on the Intel Xeon processor handles all the details of the reconfiguration process.

Reconfiguration is one of many utilities that the OPAE provides. The OPAE also provides libraries, drivers, and sample programs useful for AFU development.

To facilitate dynamically loading AFUs, the Acceleration Stack includes the following two components:

- The FIM. This component provides a framework to load AFUs on the Intel PAC. The FIM also includes the PR regions for the AFUs. The Intel PAC contains the FPGA logic to support the accelerators, including the PCIe IP core, the CCI-P fabric, the on-board DDR memory interfaces, and the FPGA Management Engine (FME). At power up, an on-board FPGA configuration flash containing the FIM bitstream image configures the FIM. The PR regions are empty until the OPAE software programs the AFU images. The FIM framework is fixed. The current release of the FIM for the Intel PAC supports a single PR region.
- The Acceleration Stack supports creation of AFU images with either RTL or OpenCL* design flows. An AFU image includes the AFU PR region bitstream and metadata that provides OPAE information on AFU characteristics and operational parameters. The current release supports dynamically swapping a single AFU image in a single PR region per installed Intel PAC.

Figure 2. Intel Arria 10 with a Single AFU PR Region



The AFU connects to the Intel Xeon processor through the CCI-P interface and then the PCIe link. The Intel PAC with Intel Arria 10 GX FPGA platform uses a simplified version of the CCI-P interface. For more information about the CCI-P interface, refer to the *Intel Acceleration Stack for Intel Xeon CPU with FPGAs Core Cache Interface (CCI-P) Reference Manual*.

The AFU also connects to two banks of private DDR4-SDRAM memory, totaling 8 GB. Each DDR4 memory bank interface has a standard Avalon® Memory-Mapped (Avalon-MM) interface. For more information about this interface, refer to the *Avalon-MM Interface Specifications*.

The Intel PAC with Intel Arria 10 GX FPGA supports a single QSFP+ network port.

Related Information

- Documentation Available for the Intel Acceleration Stack for Intel Xeon CPU with FPGAs 1.2 Release on page 46
- 10 Gbps Ethernet Accelerator Functional Unit (AFU) Design Example User Guide
- 40 Gbps Ethernet Accelerator Functional Unit (AFU) Design Example User Guide
- Intel Ethernet QSFP+ Cables Product Brief
- Avalon-MM Interfaces
  For more information about the Avalon-MM protocol, including extensive timing diagrams.
- Acceleration Stack for Intel Xeon CPU with FPGAs Core Cache Interface (CCI-P) Reference Manual
  CCI-P is a host interface bus for an AFU.
- Intel Programmable Acceleration Card (PAC) with Intel Arria 10 GX FPGA Datasheet
- Intel Acceleration Hub Knowledge Center
  For a comprehensive list of documentation available for the Intel Acceleration Task.

1.1. Acronym List for the Intel Acceleration Stack Quick Start Guide for Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA

Table 1. Intel Acceleration Stack for Intel Xeon CPU with FPGAs Glossary

AF	Accelerator Function	Compiled Hardware Accelerator image implemented in FPGA logic that accelerates an application.
AFU	Accelerator Functional Unit	Hardware accelerator implemented in FPGA logic which offloads a computational operation for an application from the CPU to improve performance.
ASE	AFU Simulation Environment	Co-simulation environment that allows you to use the same host application and AF in a simulation environment. ASE is part of the Intel Acceleration Stack for FPGAs.
CCI-P	Core Cache Interface	CCI-P is the standard interface AFUs use to communicate with the host.
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.
FME	FPGA Management Engine	Provides the following functions: • Thermal monitoring • Performance monitoring • Partial reconfiguration • Global errors

continued...

IOMMU	Input–Output Memory Management Unit	An IOMMU is a memory management unit that connects a Direct Memory Access (DMA) I/O bus to main memory. The IOMMU maps device-visible virtual addresses to physical addresses.
OPAE	Open Programmable Acceleration Engine	The OPAE is a software framework for managing and accessing AFs.
PR	Partial Reconfiguration	The ability to dynamically reconfigure a portion of an FPGA while the remaining FPGA design continues to function. The FPGA includes PR region. You can reprogram these regions at run time to implement different AFUs as system requirements dictate.
RAS	Reliability, Accessibility and Serviceability	RAS features ensure that Intel processor-based platforms perform reliably in complex, real-world environments; provide seamless support for enterprise-class security solutions; and heal themselves in response to a wide variety of errors that can bring down less protected platforms.
RBF	Raw Binary File	A binary file that is produced by the Intel Quartus® Prime Pro Edition software. It is the file format used for PR programming files.
Xeon + FPGA	Xeon + FPGA	A family of products pairing a Xeon with one or more FPGAs for acceleration, such as the Intel Xeon Processor with Integrated FPGA or the Intel PAC with Intel Arria 10 GX FPGA.

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. Contains an FPGA Interface Manager (FIM) that pairs with an Intel Xeon processor over the PCIe bus.

1.3. Intel Acceleration Stack Hardware Features

The Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA supports the following features:

- Two banks of 4 gigabyte (GB) private memory for a total memory of 8 GB
- One, Gen3 x8 PCIe link
- 4 x 10 Gbps Ethernet (10GbE) or 1 x 40 Gbps Ethernet (40GbE)
- Remote In-System Debug
- Reliability, Availability and Serviceability (RAS)
- Performance counters
- Temperature monitoring using a sideband channel

2. Getting Started

2.1. System Requirements

You can use the same server for all development, including the following activities:

- Developing software
- Running sample programs and diagnostics
- Creating and simulating AFUs
- Generating the loadable AFU images

The following servers and Linux releases have been tested for this release:

- Validated servers:
  - Dell* R640
  - Dell R740xd
  - Dell R740
  - Dell R840
  - Dell R940xa

  Note: For the most current list of validated servers, refer to the Intel FPGA Acceleration Hub Platforms Intel web page.
- Validated Linux* releases:
  - Red Hat* Enterprise Linux* (RHEL) 7.4, kernel version 3.10
  - CentOS 7.4, kernel version 3.10
  - Ubuntu 16.04, kernel version 4.4

Note: The system you use to compile the hardware design must have at least 48 GB of free memory.

Known good software and hardware combination configurations can be found at the Intel FPGA Acceleration Hub Software Download Intel web page.

2.2. Installing Required OS Packages and Components While Installing CentOS 7.4

You must install the software and select the following options and packages during initial installation:

- Development and Creative Workstation
- Additional Development
- Compatibility Libraries

- Development Tools
- Platform Development
- Python
- Virtualization Hypervisor

Or, you can use the following command:

```
sudo yum groupinstall <package from list above>
```

Table 3. Useful Linux Commands

The following Linux commands provide information about your system.

Command	Description
`sudo dmidecode -t bios`	BIOS information, including revision
`cat /proc/cpuinfo`	CPU information
`cat /etc/redhat-release`	CentOS version information
`cat /proc/version`	Linux kernel version

2.3. Installing the Intel PAC with Intel Arria 10 GX FPGA Card In the Host Machine

Follow these instructions to install the Intel PAC with Intel Arria 10 GX FPGA card.

1. Enable the following options in the BIOS:
   - Intel VT-x (Intel Virtualization Technology for IA-32 and Intel 64 Processors)
   - Intel VT-d (Intel Virtualization Technology for Directed I/O)
2. Plug the Intel PAC with Intel Arria 10 GX FPGA card into the x16 slot on the motherboard. Ensure that the slot is capable of operating the card in x8 mode.

2.4. Installing the Intel Acceleration Stack

You have the option of downloading the Acceleration Stack for Runtime or the Acceleration Stack for Development. If you are a software developer who develops and integrates your host application with accelerator functions, download the Acceleration Stack for Runtime. If you are an accelerator function developer who creates, debugs and simulates accelerator functions, download the Acceleration Stack for Development.

The following table describes each Acceleration Stack package.

Table 4. Intel Acceleration Stack Download Options

	Acceleration Stack for Runtime	Acceleration Stack for Development
Purpose	Software development of runtime host application	Develop the hardware Accelerator function using RTL or OpenCL BSP with Intel Quartus Prime Pro Edition and Acceleration Stack.
OPAE Software Development Kit (SDK) version for 1.2pv	OPAE SDK version 1.1.2 Production	OPAE SDK version 1.1.2 Production

continued...

	Acceleration Stack for Runtime	Acceleration Stack for Development
	You can access the download by clicking here: Acceleration Stack for Runtime.	You can access the download by clicking here: Acceleration Stack for Development.
Intel Quartus Prime Software	N/A	Intel Quartus Prime Pro Edition 17.1.1 including SR-IOV license
OpenCL Software	Intel FPGA Runtime Environment for OpenCL 17.1.1	Intel FPGA SDK for OpenCL 17.1.1
Download Size	~200 MB	~18 GB

For more information about where to find the Acceleration Stack package, refer to the Intel FPGA Acceleration Hub web page

Related Information

Intel FPGA Acceleration Hub Software Download web page

2.4.1. Installing the Intel Acceleration Stack Runtime Package on the Host Machine

1. Extract the runtime archive file:

```
tar xvf *rte_installer.tar.gz
```

2. Change to the installation directory.

```
cd *rte_installer
```

3. This step only applies to RHEL 7.4 (skip this step if you are using CentOS 7.4 or Ubuntu 16.04): Install Extra packages for Enterprise Linux(EPEL)

```
sudo yum install https://dl.fedoraproject.org/pub/epel/\
epel-release-latest-7.noarch.rpm
```

```
sudo subscription-manager repos --enable "rhel-*-optional-rpms"\
--enable "rhel-*-extras-rpms"
```

4. Run setup.sh.

```
./setup.sh
```

5. You are prompted with the following question: *Do you want to continue to install the software?* Answer **Yes**.
6. You are prompted with the following question: *Do you wish to install the OPAE?*

Option	Description
Answer Yes	If you have admin and network access.
Answer No	If you do not have admin and network access. After the installation, follow the manual steps listed in section *Installing the OPAE Software Package*.

7. Accept the license.
8. When you receive an installation directory prompt, you can specify an install directory. Otherwise, the installer uses the default directory at `/home/<username>/intelrtestack` to install Intel Quartus Prime Programmer and OpenCL RTE.
9. Run the initialization script to set the required environment variables.

```
source /home/<username>/intelrtestack/init_env.sh
```

Note: To avoid having to setup the environment variables after every reboot, save the export environment variable to your shell initialization script.

Note: The `init_env.sh` script calls another internal script, `setup_permissions.sh`. This internal script requires the Intel PAC to be plugged into the motherboard. Thus, if `init_env.sh` runs and the Intel PAC is not present, it may exit the current working terminal; or if `init_env.sh` runs as part of shell initialization script, it may prevent the user from successfully logging in.

To workaround this issue, replace:

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

with

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

Related Information

Installing the OPAE Software Package on page 14

2.4.2. Installing the Intel Acceleration Stack Development Package on the Host Machine

1. Extract the runtime archive file:

```
tar xvf *dev_installer.tar.gz
```

2. Change to the installation directory.

```
cd *dev_installer
```

3. This step only applies to RHEL 7.4 (skip this step if you are using CentOS 7.4 or Ubuntu 16.04): Install Extra packages for Enterprise Linux(EPEL)

```
sudo yum install https://dl.fedoraproject.org/pub/epel/epel-release\
-latest-7.noarch.rpm
```

```
sudo subscription-manager repos --enable "rhel-*-optional-rpms"\
--enable "rhel-*-extras-rpms"
```

4. Run setup.sh.

```
./setup.sh
```

5. You are prompted with the following question: `Do you wish to install the OPAE?`

Option	Description
Select Yes	If you have admin and network access.
Select No	If you do not have admin and network access. After the installation, follow the manual steps listed in section *Installing the OPAE Software Package*.

6. Accept the license.
7. When you receive an installation directory prompt, you can specify an install directory. Otherwise, the installer uses the default directory at `/home/<username>/inteldevstack` to install Intel Quartus Prime Pro Edition and OpenCL SDK.
8. Run the initialization script to set the required environment variables, `QUARTUS_HOME`, `OPAE_PLATFORM_ROOT` and other OpenCL variables.

```
source /home/<username>/inteldevstack/init_env.sh
```

Note: To avoid having to setup the environment variables after every reboot, source the script from your shell initialization script.

Note: The `init_env.sh` script calls another internal script, `setup_permissions.sh`. This internal script requires the Intel PAC to be plugged into the motherboard. Thus, if `init_env.sh` runs and the Intel PAC is not present, it may exit the current working terminal; or if `init_env.sh` runs as part of shell initialization script, it may prevent the user from successfully logging in.

To workaround this issue, replace:

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

with

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

2.4.3. Understanding the Extracted Intel PAC with Intel Arria 10 GX FPGA Release Package

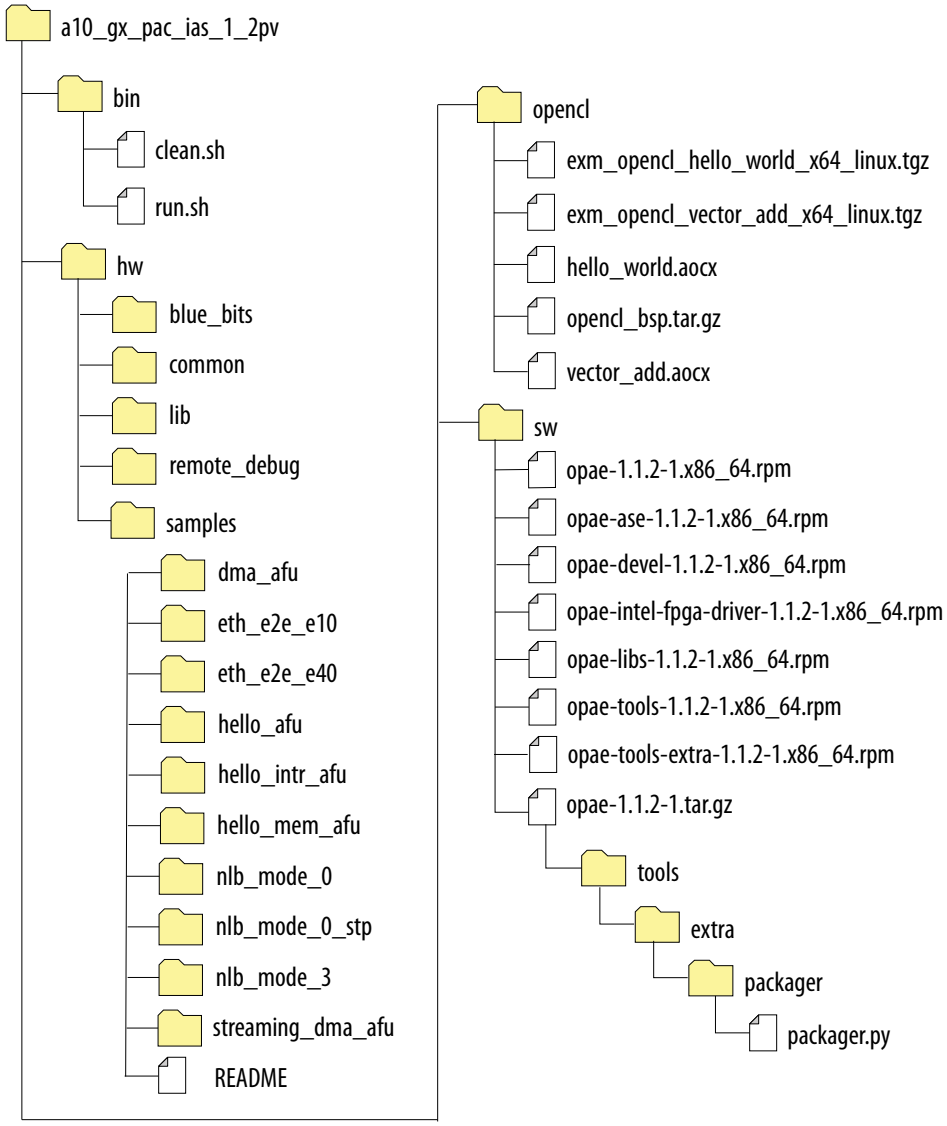
The `init_env.sh` defines `OPAE_PLATFORM_ROOT` environment variable. `OPAE_PLATFORM_ROOT` points to the extracted `a10_gx_pac_ias*` release directory. Depending on your previous choice, `a10_gx_pac_ias*` is available in one of the following directories:

- If you installed the Acceleration Stack for Runtime: `/home/<username>/intelrtestack/*`
- If you installed the Acceleration Stack for Development: `/home/<username>/inteldevstack/*`
- If you chose a custom installation directory, `a10_gx_pac_ias*`: `/<custom_install_directory>/*`

Note: If installation fails, please rerun the installer and select **No** when prompted with: *Do you wish to install the OPAE?* After installation completes, follow the manual steps to install OPAE as detailed in the Installing the OPAE Software Package on page 14 and Troubleshooting OPAE Installation on CentOS on page 44 sections.

Figure 3. Intel PAC with Intel Arria 10 GX FPGA 1.2 Directory Structure

This figure shows extracted directory structure and some of the most important files:



Related Information

- Troubleshooting OPAE Installation on CentOS on page 44
- Installing the OPAE Software Package on page 14

3. Installing the OPAE Software Package

The Intel OPAE is a software framework for managing and accessing programmable accelerators (FPGAs).

This section can be **skipped** if you have already installed OPAE by answering **Yes** when prompted by the script `setup.sh` (Acceleration Stack installer package).

After completing the OPAE framework installation, the following software and libraries are available:

- The Intel FPGA Driver
- The OPAE source at: `$OPAE_PLATFORM_ROOT/sw/opae*`
- The OPAE software development kit (SDK)

3.1. CentOS/RHEL 7.4: Installing the OPAE Framework from Prebuilt Binaries (RPM)

Important: Before you can install and build the OPAE software, you must install the required packages by running the following commands:

- If you are using RHEL 7.4:

```
sudo yum install https://dl.fedoraproject.org/pub/epel/\
epel-release-latest-7.noarch.rpm
```

```
sudo yum install gcc gcc-c++ cmake make autoconf automake libxml2\
libxml2-devel json-c-devel boost ncurses ncurses-devel\
ncurses-libs boost-devel libuuid libuuid-devel python2-jsonschema\
doxygen hwloc-devel libpng12 rsync python2-pip
```

```
sudo pip install intelhex
```

- If you are using CentOS 7.4:

```
sudo yum install gcc gcc-c++ cmake make autoconf automake libxml2\
libxml2-devel json-c-devel boost ncurses ncurses-devel\
ncurses-libs boost-devel libuuid libuuid-devel python2-jsonschema\
doxygen hwloc-devel libpng12 rsync python2-pip
```

```
sudo pip install intelhex
```

Note: These commands only install the missing packages.

Complete the following steps to install the OPAE framework:

1. Install the FPGA driver:
   a. Remove any previous version of the OPAE framework

```
sudo yum remove opae*.x86_64
```

b. Install the Extra Packages for Enterprise Linux (EPEL):

```
sudo yum install epel-release
```

c. Change to the OPAE installation software directory:

```
cd $OPAE_PLATFORM_ROOT/sw
```

d. Install the driver:

```
sudo yum install opae-intel-fpga*.rpm
```

2. Install the latest OPAE framework:

```
sudo yum install opae*.rpm
```

3. Copy updated `fpgaflash` to `/usr/bin`

```
sudo cp fpgaflash /usr/bin
sudo chmod 755 /usr/bin/fpgaflash
```

4. Update dynamic linker run-time bindings:

```
sudo ldconfig
```

5. Check the Linux kernel installation:

```
lsmod | grep fpga
```

Sample output:

```
intel_fpga_pac_iopll   13392  0
intel_fpga_pac_hssi    18347  0
intel_fpga_fme         54120  0
intel_fpga_afu         32062  0
intel_fpga_pci         26439  2 intel_fpga_afu,intel_fpga_fme
fpga_mgr_mod           14693  1 intel_fpga_fme
```

After completing the OPAE installation, the binaries and libraries are available in the following directories:

Directory	Binary Files or Libraries
/usr/bin	opae-tools* opae-tools-extra*
/usr/include	opae-devel*
/usr/lib64	opae-libs* opae-ase*

6. Verify rpm installation:

```
rpm -qa | grep opae
```

Sample output:

```
opae-tools-1.1.2-1.x86_64
opae-devel-1.1.2-1.x86_64
opae-libs-1.1.2-1.x86_64
opae-1.1.2-1.x86_64
opae-tools-extra-1.1.2-1.x86_64
opae-intel-fpga-driver-1.1.2-1.x86_64
opae-ase-1.1.2-1.x86_64
```

For more information, refer to the "Troubleshooting OPAE Installation on CentOS" section.

Related Information

Troubleshooting OPAE Installation on CentOS on page 44

3.2. Ubuntu: Installing the OPAE framework from prebuilt binaries (deb)

Before you can install and build the OPAE software, you must install the required packages by running the following two commands:

```
sudo apt-get -f install dkms libjson0 uuid-dev php7.0-dev php7.0-cli \
python-pip libjson-c-dev libhwloc-dev
```

```
sudo pip install intelhex
```

Complete the following steps to install the OPAE framework:

1. Remove any previous version of the OPAE framework.

```
sudo dpkg -r opae-intel-fpga-driver
sudo dpkg -r opae-ase
sudo dpkg -r opae-tools-extra
sudo dpkg -r opae-tools
sudo dpkg -r opae-devel
sudo dpkg -r opae-libs
```

2. Change to the OPAE installation software directory.

```
cd $OPAE_PLATFORM_ROOT/sw
```

3. Install the OPAE FPGA Intel driver.

```
sudo dpkg -i opae-intel-fpga-driver_*.deb
```

4. Check the Linux kernel installation.

```
lsmod | grep fpga
```

Sample Output:

```
intel_fpga_pac_hssi    20480  0
intel_fpga_fme         61440  0
intel_fpga_afu         36864  0
fpga_mgr_mod           16384  1 intel_fpga_fme
intel_fpga_pci         32768  2 intel_fpga_fme,intel_fpga_afu
```

5. Install the OPAE framework.

```
sudo dpkg -i opae-libs-*.x86_64.deb
sudo dpkg -i opae-devel-*.x86_64.deb
sudo dpkg -i opae-tools-*.x86_64.deb
sudo dpkg -i opae-tools-extra-*.x86_64.deb
sudo dpkg -i opae-ase-*.x86_64.deb
```

6. Copy updated `fpgaflash` to `/usr/bin`

```
sudo cp fpgaflash /usr/bin
sudo chmod 755 /usr/bin/fpgaflash
```

7. After completing the OPAE installation, the binaries and libraries are available in the following directories:

Directory	Binary Files or Libraries
/usr/bin	• opae-tools* • opae-tools-extra*
/usr/include	opae-devel*
/usr/lib	• opae-libs* • opae-ase*

8. Verify deb packages installation:

```
dpkg -l | grep opae
```

Sample Output:

```
ii  opae-ase
1.1.2                                   amd64        OPAE AFU
Simulation Environment
ii  opae-devel
1.1.2                                   amd64        OPAE headers,
sample source, and documentation
ii  opae-intel-fpga-driver
1.1.2-1                                 amd64        DKMS-enabled
Intel FPGA driver source code.
ii  opae-libs
1.1.2                                   amd64        OPAE runtime
ii  opae-samplesrasescriptsc
1.1.2                                   amd64        Open Programmable
Acceleration Engine
ii  opae-tools
1.1.2                                   amd64        OPAE base tool
binaries
ii  opae-tools-extra
1.1.2                                   amd64        OPAE extra tool
binaries
```

3.3. (Optional) Building and Installing the OPAE Software from Source Code

1. Complete the following steps to install Intel FPGA Driver:
   a. Remove any previous version:

      CentOS:

      ```
      sudo yum remove opae*-intel-fpga*.x86_64
      ```

      Ubuntu:

      ```
      sudo dpkg -r opae-intel-fpga-driver
      ```

   b. Install the Extra Packages for Enterprise Linux (EPEL):

      CentOS:

      ```
      sudo yum install epel-release
      ```

   c. Change to the OPAE installation software directory:

      ```
      cd $OPAE_PLATFORM_ROOT/sw
      ```

d. Install the driver:

CentOS:

```
sudo yum install opae-intel-fpga*.rpm
```

Ubuntu:

```
sudo dpkg -i opae-intel-fpga-driver_*.deb
```

2. Build and install the OPAE SDK from source:

a. Change to the OPAE software directory and extract the .tar file:

```
cd $OPAE_PLATFORM_ROOT/sw
tar xf opae*.tar.gz
```

b. Complete the following steps to build the OPAE software:

```
cd opae*
mkdir build && cd build
cmake .. -DBUILD_ASE=OFF -DCMAKE_INSTALL_PREFIX=<path to install
directory> DCMAKE_BUILD_TYPE=Release
```

For example:

```
cmake .. -DBUILD_ASE=ON -DCMAKE_INSTALL_PREFIX=/home/john/ \
opaeinstall -DCMAKE_BUILD_TYPE=Release
```

Note: You may get an error because the `cmake` command cannot find the git repository. You can safely ignore this error message. You do not need the git repository to successfully build the OPAE software.

c. Run the following command to build the executables and libraries:

```
make install
```

d. Copy updated `fpgaflash` to `<Path to install directory>`

```
sudo cp fpgaflash <path to install directory>/bin/
sudo chmod 755 <path to install directory>/bin/fpgaflash
```

e. Run the following command to generate documentation:

```
make doc
```

Documentation is in the current directory.

Note: By default, if you choose the RPM installation flow, the binaries, libraries and include files are under `/usr/`. If you build and install the OPAE from the source flow, the binaries, libraries and include files are under `<path to install directory>`.

f. Set the appropriate environment variable to ensure tools, libraries, and include files are in your search path. To avoid rerunning this command whenever you restart or open a new terminal, add these directory environment variables to your shell configuration file, `/etc/bashrc`.

```
export PATH=<path to OPAE install directory>/bin:$PATH
```

```
export C_INCLUDE_PATH=<path to OPAE install directory>/include:\
$C_INCLUDE_PATH
```

To check for static libraries use the following paths:

- CentOS:

```
export LIBRARY_PATH=<path to OPAE install directory>\
/lib64:$LIBRARY_PATH
```

- Ubuntu:

```
export LIBRARY_PATH=<path to OPAE install directory>\
/lib:$LIBRARY_PATH
```

To check for shared libraries use the following paths:

- CentOS:

```
export LD_LIBRARY_PATH=<path to OPAE install directory>/\
lib64:$LD_LIBRARY_PATH
```

- Ubuntu:

```
export LD_LIBRARY_PATH=<path to OPAE install directory>/\
lib:$LD_LIBRARY_PATH
```

4. Identifying the Flash Image and BMC Firmware

Each Acceleration Stack Release requires a different version of the FIM. Run the `fpgainfo` tool to **identify the FIM (PR Interface ID) and BMC firmware** currently loaded.

```
sudo fpgainfo fme
```

Sample Output (after updating to 1.2 FIM):

```
[sudo] Your password:
Board Management Controller, microcontroller FW version 26889
Last Power Down Cause: POK_CORE
Last Reset Cause: None
//****** FME ******//
Object Id                        : 0xEE00000
PCIe s:b:d:f                     : 0000:D8:00:0
Device Id                        : 0x09C4
Socket Id                        : 0x00
Ports Num                        : 01
Bitstream Id                     : 0x121000200000154
Bitstream Version                : 0x55B200010201
Pr Interface Id                  : 69528db6-eb31-577a-8c36-68f9faa081f6
```

Table 5. Correspondence Between Acceleration Stack, FIM, and OPAE Versions

Note: Intel recommends porting AFUs and workloads to the Intel Acceleration Stack v1.2. You must recompile and validate when upgrading to the new release. The Intel Acceleration Stack v1.2 cannot be downgraded to previous versions.

Acceleration Stack Version	FIM Version (PR Interface ID)	OPAE Version	BMC Firmware Version
1.2 Production	69528db6-eb31-577a-8c36-68f9faa081f6	1.1.2-1	26889 (bootloader version 26879)
1.2 Alpha	93abeb6a-30c8-5f77-8172-d828c3a699ca	1.1.1-1	
1.1 Production	9926ab6d-6c92-5a68-aabc-a7d84c545738	1.0.2	26822
1.1 Beta	0f17997f-199b-5f75-9713-2653d3ce0176	1.0.1	
1.1 Alpha	8fd6574f-8f82-5164-9336-69c4bdaba437	0.14.0	
1.0 Production	ce489693-98f0-5f33-946d-560708be108a	0.13.1	26815
1.0 Beta	3d949b98-7b30-5a9ab296-4530a780a3f9	0.13.0	
1.0 Alpha	d4a76277-07da-528db623-8b9301feaffe	0.11.0	

If your FIM and BMC firmware version correspond to the most recent version for 1.2 Production, then proceed to the next section: *Running FPGA Diagnostics*. If your FIM version is out of date, go to *Appendix A: Updating the FIM and BMC Firmware* for further instructions.

5. Running FPGA Diagnostics

This section presents instructions on how to run the FPGA diagnostics by using the `fpgabist` utility. The current AFUs accepted are `nlb_mode_3` and `dma_afu`, running `fpgadiag` and `fpga_dma_test` tests, respectively.

1. Configure the number of system hugepages the FPGA fpgadiag utility requires:

```
sudo sh -c "echo 20 > /sys/kernel/mm/hugepages/hugepages-\
2048kB/nr_hugepages"
```

2. Configure and run diagnostics with NLB_3 AFU image.

```
sudo fpgabist $OPAE_PLATFORM_ROOT/hw/samples/nlb_mode_3/bin/\
nlb_mode_3.gbs
```

Sample output:

```
Cachelines Read_Count Write_Count Cache_Rd_Hit Cache_Wr_Hit Cache_Rd_Miss
Cache_Wr_Miss   Eviction 'Clocks(@400 MHz)'   Rd_Bandwidth   Wr_Bandwidth
1024  480797340   488815296             0            0
0             0          0       1000021563      6.234 GB/s    6.256 GB/s

VH0_Rd_Count VH0_Wr_Count VH1_Rd_Count VH1_Wr_Count VL0_Rd_Count
VL0_Wr_Count480797340        488815297                        0
0            0            0

Built-in Self-Test Completed.
```

Note: If you get Error Message: "Exception caught: `stoi` - could not convert `af` to a number", then follow these steps to fix the issue:

Fix:

a. Edit file `bist_nlb3.py`:

```
sudo vim /usr/bin/bist_nlb3.py
```

b. Change Line48:

```
cmd = "fpgadiag -B {} {}".format(bus_num, param)
---
cmd = "fpgadiag -B 0x{} {}".format(bus_num, param)
```

3. Configure and run diagnostics with DMA AFU image.

```
sudo fpgabist $OPAE_PLATFORM_ROOT/hw/samples/dma_afu/bin/dma_afu.gbs
```

Sample output:

```
Running test in HW mode
Buffer Verification Success!
Buffer Verification Success!
Running DDR sweep test
Allocated test buffer
Fill test buffer
DDR Sweep Host to FPGA
Measured bandwidth = 6616.881910 Megabytes/sec
Clear buffer
```

```
DDR Sweep FPGA to Host
Measured bandwidth = 6932.201347 Megabytes/sec
Verifying buffer.
Buffer Verification Success!
Finished Executing DMA Tests
```

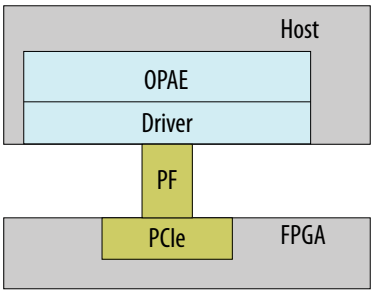
Related Information

OPAE FPGA Tools - fpgabist

6. Running the OPAE in a Non-Virtualized Environment

This section shows OPAE examples running directly on the Bare Metal operating system without a virtual machine nor SR-IOV. The host links to the FPGA with a single PCIe physical function (PF).

Figure 4. OPAE Driver in Non-Virtualized Mode



6.1. Loading the AFU Image into the FPGA

Use the `fpgaconf` utility to load the AFU image. The AFU image's filename is the only parameter:

```
sudo fpgaconf <AFU image>
```

The Acceleration Stack 1.2 release includes the following AFU images in the `$OPAE_PLATFORM_ROOT/hw/samples` directory:

- `dma_afu/bin/dma_afu.gbs`
- `eth_e2e_e10/bin/eth_e2e_e10.gbs`
- `eth_e2e_e40/bin/eth_e2e_e40.gbs`
- `hello_afu/bin/hello_afu.gbs`
- `hello_intr_afu/bin/hello_intr_afu.gbs`
- `nlb_mode_0/bin/nlb_0.gbs`
- `nlb_mode_0_stp/bin/nlb_0_stp.gbs`
- `nlb_mode_3/bin/nlb_3.gbs`
- `streaming_dma_afu/bin/streaming_dma_afu.gbs`

Related Information

Intel FPGA Software Licensing Support

6.2. OPAE Sample Application Programs

6.2.1. Running the Hello FPGA Example

The `hello_fpga` sample host application uses the OPAE library to test the hardware in native loopback mode (NLB). Load the FPGA with the `nlb_mode_0` AFU image to run this example.

Run the following commands to test the `hello_fpga` sample host application:

1. Run the following command to load the AFU image:

```
sudo fpgaconf $OPAE_PLATFORM_ROOT/hw/samples/nlb_mode_0/bin/\
nlb_mode_0.gbs
```

2. Configure the system hugepage to allocate 20, 2 MB hugepages that this utility requires. This command requires root privileges:

```
sudo sh -c "echo 20 > /sys/kernel/mm/hugepages/\
hugepages-2048kB/nr_hugepages"
```

3. To compile the source code for `hello_fpga` located at `$OPAE_PLATFORM_ROOT/sw/opae*/samples/hello_fpga.c`:

```
cd $OPAE_PLATFORM_ROOT/sw
```

4. Extract the `tar` file:

```
tar xf opae*.tar.gz
```

Note: This step is only necessary if you installed the OPAE software from binaries. For more information, refer to the *Installing the OPAE Software from Prebuilt Binaries* section.

5. Change to the OPAE directory:

```
cd opae*
```

6. Compile the example:

CentOS:

```
gcc -o hello_fpga -std=gnu99 -rdynamic \
-ljson-c -luuid -lpthread -lopae-c -lm -Wl,-rpath \
-lopae-c $OPAE_PLATFORM_ROOT/sw/opae*/samples/hello_fpga.c
```

Ubuntu:

```
gcc -o hello_fpga -std=gnu99 -rdynamic \
-ljson-c -luuid -lpthread -lopae-c -lm -Wl,--no-as-needed \
-lopae-c -luuid $OPAE_PLATFORM_ROOT/sw/opae*/samples/hello_fpga.c
```

7. To run the example, type the following command:

Option	Description
For the OPAE RPM installation:	`sudo ./hello_fpga`
For an OPAE installation from source:	`sudo LD_LIBRARY_PATH=\ $LD_LIBRARY_PATH:/<path to opae install>/\ lib64 ./hello_fpga`

Sample output:

```
Running Test
Done Running Test
```

For more information about the `hello_fpga` example, refer to the following files:

- Source code located at `$OPAE_PLATFORM_ROOT/sw/opae*/samples/hello_fpga.c`
- *Native Loopback Accelerator Functional Unit (AFU) User Guide* for AFU register descriptions.

Related Information

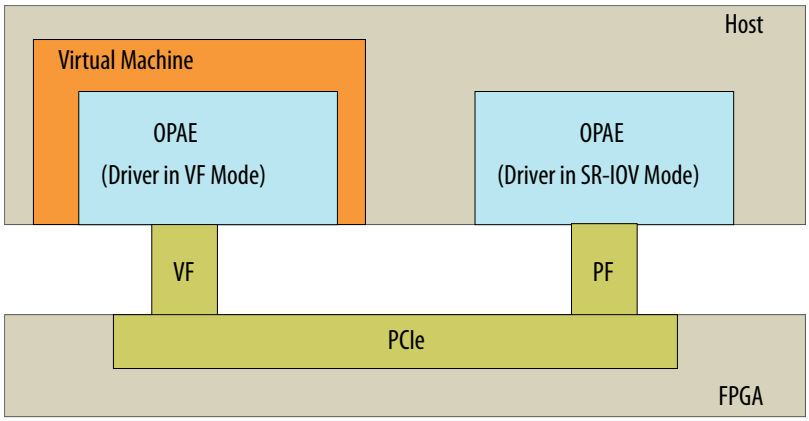
- Identifying the Flash Image and BMC Firmware on page 20
- CentOS/RHEL 7.4: Installing the OPAE Framework from Prebuilt Binaries (RPM) on page 14

7. Running the OPAE in a Virtualized Environment

In SR-IOV mode, a host processor uses a physical function (PF) to access management functions. A virtual machine (VM) uses a virtual function (VF) to access the AFU.

Note: Partial reconfiguration (PR) is not available in this mode.

Figure 5. OPAE Driver in SR-IOV Mode



You must complete all the steps in the *Getting Started* and *Installing the OPAE Software* chapters before you can set up a virtualized environment. An application running in a virtual machine that connects to a VF through OPAE cannot initiate partial reconfiguration. The permission table in the FME enforces this restriction. The permission table only allows partial reconfiguration through a PF. Consequently, you must load the AFU image on the host before continuing with the steps to create a virtualized environment.

Run the following command on the host to load the AFU image.

```
sudo fpgaconf \
$OPAE_PLATFORM_ROOT/hw/samples/nlb_mode_0/bin/nlb_mode_0.gbs
```

Related Information

- Getting Started on page 8
- Installing the OPAE Software Package on page 14

7.1. Updating Settings Required for VFs

To use SR-IOV and pass a VF to a virtual machine, you must enable the Intel IOMMU driver on the host. Complete the following steps to enable the Intel IOMMU driver:

1. Add `intel_iommu=on` to the kernel command line by updating the GRUB configuration.
2. Restart to apply the new GRUB configuration file.
3. To verify the GRUB update, run the following command:

   ```
   cat /proc/cmdline
   ```

   The sample output below shows *`intel_iommu=on`* on the kernel command line.

   Sample output:

   ```
   BOOT_IMAGE=/vmlinuz-3.10.0-514.21.1.el7.x86_64
   root=/dev/mapper/cl_<server-name>-root ro intel_iommu=on
   crashkernel=auto rd.lvm.lv=cl_<server-name>/root
   rd.lvm.lv=cl_<server-name>/swap rhgb quiet
   ```

7.2. Configuring the VF Port on the Host

By default, the PF controls the AFU port. The following procedure transfers AFU control to the VF. After the transfer to VF control, applications running on the VM can access the AFU.

In a multicard system, if you want to configure the VF on only a single PCIe device, run below command to find the device mapping for the specific PCIe:

```
ls -l /sys/class/fpga/intel-fpga-dev.*
```

Sample output:

```
/sys/class/fpga/intel-fpga-dev.0 -> ../../devices/
pci0000:36/0000:36:00.0/0000:37:00.0/fpga/intel-fpga-dev.0

/sys/class/fpga/intel-fpga-dev.1 -> ../../devices/pci0000:ae/
0000:ae:00.0/0000:af:00.0/fpga/intel-fpga-dev.1
```

To target PCIe B:D.F (AF:00.0) and B:D.F (37:00.0) in the following commands, use instance id **1** and **0** instead of ***** respectively.

1. Run the following three commands, individually, to export the required paths:

   ```
   export port_path=$(find /sys/class/fpga/intel-fpga-dev.* \
   -maxdepth 1 -follow -iname intel-fpga-port.*)
   ```

   ```
   export link_path=$(readlink -m /$port_path/../)
   ```

   ```
   export pci_path=$link_path/../../
   ```

2. Release the port controlled by the PF using the `fpgaport` tool:

   ```
   sudo fpgaport release /dev/intel-fpga-fme.* 0
   ```

3. Enable SR-IOV and VFs. Each VF has 1 AFU Port:

   ```
   sudo sh -c "echo 1 > $pci_path/sriov_numvfs"
   ```

4. Find the additional device number for the VF device:

```
lspci -nn | grep :09c[45]
```

Sample output:

```
04:00.0 Processing accelerators [1200]: Intel Corporation Device [8086:09c4]
04:00.1 Processing accelerators [1200]: Intel Corporation Device [8086:09c5]
```

`lspci` shows an additional device number, 09c5. This is the VF device you assign to a VM. The original bus and device numbers for the PF remains 09c4.

Note that the Domain:Bus:Device.Function (BDF) notation for the VF device in this example is: 000:04:00.1. Replace this BDF with the appropriate BDF for your system.

5. Load the vfio-pci driver:

```
sudo modprobe vfio-pci
```

6. Unbind the VF device from its driver:

```
sudo sh -c "echo 0000:04:00.1 > \
/sys/bus/pci/devices/0000:04:00.1/driver/unbind"
```

7. Find the vendor and device ID for the VF device:

```
lspci -n -s 04:00.1
```

Sample output:

```
04:00.1 1200: 8086:09c5
```

8. Bind the VF to the `vfio-pci` driver:

```
sudo sh -c "echo 8086 09c5 > \
/sys/bus/pci/drivers/vfio-pci/new_id"
```

7.3. Running the Hello FPGA Example on Virtual Machine

This section assumes that you have set up the Virtual Machine (VM) and connected to the virtual function (VF) device with ID 09c5. On the virtual machine, install the Intel FPGA Driver and OPAE Software. Refer to *Installing the OPAE Software Package* section for instructions.

Complete the following steps to test the operation of the NLB mode 0 AFU in a virtualized environment:

1. Configure the system hugepage to allocate 20, 2 MB hugepages that this utility requires. This command requires root privileges:

```
sudo sh -c "echo 20 > /sys/kernel/mm/hugepages/hugepages-\
2048kB/nr_hugepages"
```

2. Complete the following commands to extract the `.tar` file:

```
tar xf $OPAE_PLATFORM_ROOT/sw/opae*.tar.gz
cd opae*
```

3. To compile, type the following command:

```
gcc -o hello_fpga -std=gnu99 -rdynamic \
-ljson-c -luuid -lpthread -lopae-c -lm -Wl,-rpath -lopae-c \
$OPAE_PLATFORM_ROOT/sw/opae*/samples/hello_fpga.c
```

4. Run the example:

```
sudo ./hello_fpga
```

   Sample output:

```
Running Test
Done Running Test
```

   For more information about the `hello_fpga` sample host application, refer to the following files:

   - Source code located at `$OPAE_PLATFORM_ROOT/sw/opae*/samples/hello_fpga.c`
   - *Native Loopback Accelerator Functional Unit (AFU) User Guide* for AFU register descriptions.

Related Information

- Installing the OPAE Software Package on page 14
- Running the Hello FPGA Example on page 24
- Native Loopback Accelerator Functional Unit (AFU) User Guide
- Documentation Available for the Intel Acceleration Stack for Intel Xeon CPU with FPGAs 1.2 Release on page 46

7.3.1. Disconnecting the VF from the VM and Reconnecting to the PF

1. Uninstall the driver on the VM:

```
yum remove opae-intel-fpga-driver.x86_64
```

2. Detach the VF from the VM.

   On the host machine, unbind the VF PCI device from the vfio-pci driver:

```
sudo sh -c "echo -n 0000:04:00.1 > /sys/bus/pci/drivers/vfio-pci/unbind"
```

3. Bind the VF to the intel-fpga driver:

```
sudo sh -c "echo -n 0000:04:00.1 > \
/sys/bus/pci/drivers/intel-fpga-pci/bind"
```

4. To ensure you have the correct `$pci_path` for disconnection, type:

```
export port_path=$(find /sys/class/fpga/intel-fpga-dev.* \
-maxdepth 1 -follow -iname intel-fpga-port.*)
export link_path=$(readlink -m /$port_path/../)
export pci_path=$link_path/../../
```

5. Set to 0 VFs and disable SR-IOV:

```
sudo sh -c "echo 0 > $pci_path/sriov_numvfs"
```

6. Assign the port to PF using fpgaport tool:

```
sudo fpgaport assign /dev/intel-fpga-fme.* 0
```

8. Intel Acceleration Stack Quick Start Guide for Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA Archives

Intel Acceleration Stack Version	User Guide (PDF)
1.1 Production	Intel Acceleration Stack Quick Start Guide for Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA

Related Information

Intel Acceleration Stack Quick Start Guide for Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA Archives on page 31

Provides a list of user guides for previous versions of the Intel Acceleration Stack Quick Start Guide for Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA IP core.

9. Document Revision History for Intel Acceleration Stack Quick Start Guide for Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA

Document Version	Intel Quartus Prime Version	Changes
2019.08.05	1.2 (supported with Intel Quartus Prime Pro Edition 17.1.1)	Corrected a document title in appendix *Documentation Available for the Intel Acceleration Stack for Intel Xeon CPU with FPGAs 1.2 Release*: From *HSSI User Guide for Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA* to *Networking Interface for Open Programmable Acceleration Engine: Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA*.
2019.05.30	1.2 (supported with Intel Quartus Prime Pro Edition 17.1.1)	• Updated steps under section: — *Configuring the VF Port on the Host* — *Updating FPGA Flash and BMC Firmware* — *Troubleshooting OPAE Installation on CentOS*
2019.03.08	1.2 (supported with Intel Quartus Prime Pro Edition 17.1.1)	Added a note regarding use of pacd at the top of the *Handling Graceful Thermal Shutdown* section..
2019.02.23	1.2 (supported with Intel Quartus Prime Pro Edition 17.1.1)	Added a note regarding upgrading to the Intel Acceleration Stack v1.2 Production in the *Correspondence Between Acceleration Stack, FIM, and OPAE Versions* table in the *Identify the Flash Image and BMC Firmware* section.
2019.02.06	1.2 (supported with Intel Quartus Prime Pro Edition 17.1.1)	• Added a note about the `init_env.sh` script in the *Installing the Intel Acceleration Stack Runtime Package on the Host Machine* and *Intel Acceleration Stack Development Package on the Host Machine*. • Moved the "Updating the Flash Image and BMC Firmware" section to an appendix. • Added a important note about BMC firmware version permissions after upgrading to version 26889. • Added a *Troubleshooting Frequently Asked Questions (FAQ)* section.
2018.12.04	1.2 (supported with Intel Quartus Prime Pro Edition 17.1.1)	• Added the "Intel Acceleration Stack Quick Start Guide for Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA Archives" section that contains the archived versions of this document. • Removed the "Updating the Board Management Controller (BMC) Configuration and Firmware" section.
2018.10.15	1.2 Alpha supported with Intel Quartus Prime Pro Edition 17.1.1)	• In the "Handling Graceful Thermal Shutdown" section — Added a note about partial reconfiguration cannot be initiated from a VM. — Added a note about conditions for when the server can reboot or hang.

continued...

Document Version	Intel Quartus Prime Version	Changes
2018.10.01	1.2 Pre-alpha supported with Intel Quartus Prime Pro Edition 17.1.1)	Added the following chapters: • Ubuntu: Installing the OPAE framework from prebuilt binaries (deb) • Handling Graceful Thermal Shutdown
2018.08.27	1.1 Production supported with Intel Quartus Prime Pro Edition 17.1.1)	Added the following chapters: • FPGA Device Access Permission • Memlock Unit • Hugepage Settings
2018.08.14	1.1 Production supported with Intel Quartus Prime Pro Edition 17.1.1)	Made the following changes: • Changed `a10_gx_pac_ias_1_1_pv_eth.pv` to `a10_gx_pac_ias_1_1_pv_eth.patch` in *Installing the Intel Acceleration Stack Runtime package on the Host Machine* and *Installing the Intel Acceleration Stack Development Package on the Host Machine* • Corrected the OPAE Software Development Kit (SDK) version 1.1 Production `.tar` file names and Download sizes in *Installing the Intel Acceleration Stack*
2018.08.06	1.1 Production supported with Intel Quartus Prime Pro Edition 17.1.1)	Initial release.

A. Updating the FIM and BMC Firmware

A.1. Selecting the Correct Update Method

The following table provides instructions to update the FIM based on the release currently running on the Intel PAC with Intel Arria 10 GX FPGA:

Table 6. Selecting the Correct Update Method

Acceleration Stack Release Version	FIM Update Instructions
1.1PV or newer	Use the file `setup_fim_and_bmc.sh`, provided with the release, as described in the "Updating FPGA Flash and BMC Firmware" section.
Older than 1.1PV	1. Follow the "Identifying and Updating FIM" section of the 1.1PV version of the *Intel Acceleration Stack Quick Start Guide for Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA* to first update to 1.1PV FIM (`dcp_1_1.rpd`). *Note:* • It is acceptable to leverage the 1.2 OPAE packages when following the FIM update instructions from the 1.1PV version of the Intel Acceleration Stack Quick Start Guide for Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA. • 1.1PV runtime download package can be located here. For access to the previous version of the *Intel Acceleration Stack Quick Start Guide for Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA*, refer to the "Intel Acceleration Stack Quick Start Guide for Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA Archives" section. 2. Follow the instructions in the "Updating FPGA Flash and BMC Firmware" section in the current version of this document.

For more information about the `fpgaflash` tool, refer to the *Open Programmable Acceleration Engine (OPAE) Tools Guide* located on the Intel FPGA Acceleration Hub.

Related Information

- Open Programmable Acceleration Engine - Documentation web page on GIT
- Intel Acceleration Stack Quick Start Guide for Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA Archives on page 31
  Provides a list of user guides for previous versions of the Intel Acceleration Stack Quick Start Guide for Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA IP core.

A.1.1. Updating FPGA Flash and BMC Firmware

Important: Before you begin:

- OPAE Software package (Version 1.1.2) must be installed

  Note: Refer to Table 6 on page 34 for the correct upgrade process based on the state of the Intel PAC card.
- You must install a micro USB cable between the Intel PAC and the server.
- Only one Intel PAC card must be connected to the host server using the Intel FPGA Download Cable II.
- Do not interrupt the script.
- This script must be run on a Host Machine and not on a Virtual Machine.
- Stop any service or daemon accessing the FPGA before updating the FPGA flash.

  For example: pacd service using command:

```
systemctl stop pacd.service
```

Important: You must not downgrade your BMC firmware after upgrading to version 26889. BMC version 26889 only supports the Acceleration Stack 1.2 FIM (69528db6-eb31-577a-8c36-68f9faa081f6) and no prior Acceleration Stack release is supported. Please check compatibility of intended AFU workloads with Acceleration Stack 1.2 before upgrading the board. For questions, please contact your Intel sales representative or workload solutions partner.

The following steps update the Intel PAC card:

1. Obtain and install the appropriate BittWorks II Toolkit-Lite software, firmware, and bootloader.

Table 7. OS-Compatible BittWorks II ToolkitLite Version

Operating System	Release	BittWorks II Toolkit-Lite Version	Install Command
CentOS 7.4/RHEL 7.4	2018.6 Enterprise Linux 7 (64-bit)	bw2tk-lite-2018.6.el7.x86_64.rpm	`sudo yum install bw2tk-\ lite-2018.6.el7.x86_64.rpm`
Ubuntu 16.04	2018.6 Ubuntu 16.04 (64-bit)	bw2tk-lite-2018.6.u1604.amd64.deb	`sudo dpkg -i bw2tk-\ 2018.6.u1604.amd64.deb`

Follow the steps to download BMC firmware and tools:

- BMC Firmware version: 26889
- BMC Bootloader version: 26879

Save the files to a known location on the host machine. The following script prompts for this location.

2. Add Bittware tool to PATH:

```
export PATH=/opt/bwtk/2018.6.0L/bin/:$PATH
```

3. Change directory:

```
cd $OPAE_PLATFORM_ROOT
```

4. Identify the b, d, and f for Intel PAC card.

```
lspci | grep 09c4
```

Output:

```
d8:00.0 Processing accelerators: Intel Corporation Device 09c4
```

5. Execute script. The script prompts to run part1, followed by a mandatory power cycle and part 2. You must follow the prompts.

```
./setup_fim_and_bmc -b <bus id> -d <device id> -f <function id> -p
$OPAE_PLATFORM_ROOT
```

For example:

```
./setup_fim_and_bmc -b D8 -d 0 -f 0 -p $OPAE_PLATFORM_ROOT
```

Note: You can ignore the Bittworks II Tool version prerequisites that list when running this script. Instead, use the versions found in Table 7 on page 35.

Note: While running part 1 of this script, the error below appears. You can ignore this error:

```
>>> Running cmd:
      bwmonitor --dev=0 --upgrade-mode

Entering upgrade mode...Success!
ERROR Item not found: could not re-open device 0
Tue Dec 11 12:24:16 2018
```

6. Power cycle the server for the changes to take effect.
7. Ensure BMC version (26889) and PR Interface ID or FIM version (69528db6-eb31-577a-8c36-68f9faa081f6) is displayed while running:

```
sudo fpgainfo fme
```

- Use the following commands for **future upgrades** of the FIM and BMC:
  - FIM:

    ```
    sudo fpgaflash user $OPAE_PLATFORM_ROOT/hw/blue_bits/dcp*.rpd
    ```

  - BMC:

    ```
    sudo fpgaflash bmc_app <path to file>/*.hex
    ```

Note:
- `fpgaflash` accepts `bdf` as an argument to target a specific card in a multi-card system.
- Ensure `pacd` service is stopped while updating `bmc_app` using `fpgaflash`.

For more information about the `fpgaflash` tool, refer to the Open Programmable Acceleration Engine - Documentation web page on GIT.

B. Handling Graceful Thermal Shutdown

Note:

- Qualified OEM server systems provide adequate cooling for standard workloads and the use of `pacd` may be optional.
- Refer to the OPAE pacd documentation for more details on using `pacd`, including considerations that may lead to an unexpected system reboot.

The Intel PAC Daemon (`pacd`) is a program that can be used to help protect the user's server from crashing due to hardware reaching an upper non-recoverable or lower non-recoverable sensor threshold. `pacd` is capable of monitoring any of the 23 sensors reported by Board Management Controller. `pacd` can be run standalone, as a daemon, or as a *systemd* service. When the OPAE tools-extra package is installed, `pacd` gets placed in the OPAE binaries directory (default: */usr/bin*) along with a configuration and service file – `pacd.conf` and `pacd.service`, respectively.

On startup, the `pacd` sets its threshold to the BMC's default sensor threshold values. The BMC threshold values are readjusted such that the threshold range is expanded. This is to pass on the Graceful Thermal Shutdown responsibility to `pacd`.

`pacd` periodically reads the sensor values and if the values exceed the threshold, it resets the FPGA. This sends a `SIGHUP` signal to all running processes and makes the board inaccessible from the host. The daemon waits for a configurable time specified by `-c` in `pacd.conf`, as described below, to cool down the board. After this configurable wait time elapses, the `pacd` service programs the specified AFU. Ensure that the AFU host application that you develop monitors for a `SIGHUP` signal and exits.

`pacd` can be set up as a *systemd* service as follows (using a shell with elevated privileges (sudo)):

1. Edit the `pacd.conf` file to update the "`DefaultGBSOptions`" entry with a list of AFUs appropriate for your FIM. Use the full absolute path to each AFU file and precede each file name with '`-n`'.

   ```
   sudo vim /usr/bin/pacd.conf
   ```

   Edit entry:

   ```
   DefaultGBSOptions=-n /home/<username>/intelrtestack/\
   a10_gx_pac_ias_1_2_pv/hw/samples/nlb_mode_3/bin/nlb_mode_3.gbs
   ```

Note: Optional settings include:

- PCIe address (For example: -S, -B, -D, -F), `pacd` monitors all Intel PACs matching the PCIe address components specified. For example, if you specify -B 5 only, all Intel PACs on PCIe bus 5 becomes monitored.
- Sensor Threshold—The thresholds are global, so specifying -T 11:95.0:93.0 monitors sensor 11 on all selected Intel PACs. When the value exceeds 95.0, it causes the default bitstream specified with -n in pacd.conf to be programmed (PR). The sensor is considered triggered (and no PR is performed) until its value drops below 93.0
- Specify `-c <time period>` for `ThresholdOptions` in `pacd.conf` to vary the cool down period or change `CooldownInterval=<time period>`.
- The Sensor Number can be found by running this command:

```
sudo fpgainfo bmc
```

Examine the remaining option variables and adjust as appropriate for your system.

2. Copy `pacd.conf` to the default *systemd* service configuration directory (typically */etc/sysconfig*).

   CentOS:

   ```
   sudo cp /usr/bin/pacd.conf /etc/sysconfig/
   ```

   Ubuntu:

   ```
   sudo cp /usr/bin/pacd.conf /etc/default
   ```

3. Edit the `pacd.service` file to update "`EnvironmentFile`" entry to reflect where the `pacd.conf` file was copied. Prepend the path name with a single dash '-', and specify the path as absolute.

   ```
   sudo vim /usr/bin/pacd.service
   ```

   Edit entry:
   CentOS:

   ```
   EnvironmentFile=-/etc/sysconfig/pacd.conf
   ```

   Ubuntu:

   ```
   EnvironmentFile=-/etc/default/pacd.conf
   ```

4. Copy `pacd.service` to /etc/`systemd/system/pacd.service`. This will make `pacd` visible to *systemd*.

   CentOS:

   ```
   sudo cp /usr/bin/pacd.service /etc/systemd/system/
   ```

   Ubuntu:

   ```
   sudo cp /usr/bin/pacd.service /lib/systemd/system/
   ```

5. Start `pacd` as a *systemd* service.

Note: Please use `sudo` if command cannot be run in regular user mode.

```
systemctl daemon-reload
systemctl start pacd.service
```

6. Optional: To enable `pacd` to re-start on boot, execute

```
systemctl enable pacd.service
```

To check whether **`pacd`** has started and to check state or actions, please examine the log file (specified in **`pacd.conf`** on the "**`LogFile`**" line).

For a full list of `systemctl` commands, run the following command:

```
systemctl -h
```

7. To verify that the service is running, run the following command:

```
systemctl status pacd.service
```

Sample Output:

Note: Partial reconfiguration cannot be initiated from a Virtual Machine. Hence, `pacd` cannot run on a Virtual Machine.

8. To stop the service:

```
systemctl stop pacd.service
```

For more information about the `pacd` tool, refer to the Open Programmable Acceleration Engine - Documentation web page on GIT.

Related Information

Open Programmable Acceleration Engine - Documentation web page on GIT

C. FPGA Device Access Permission

To be able to run AFU samples and other OPAE tools as non-root user, you can run the below command to change file access permissions.

Use file access permissions on the Intel FPGA device file directories, `/dev/intel-fpga-fme.*` and `/dev/intel-fpga-port.*` to control access to FPGA accelerators and devices. Use the same file access permissions to control access to the files reachable through `/sys/class/fpga/`.

The * denotes the respective socket, for example 0 or 1.

Typically, you must change these permissions after every restart. To make the changes permanent, add these permissions to `/etc/bashrc` as well.

Here are the commands to run:

```
sudo chmod 666 /dev/intel-fpga-fme.*
sudo chmod 666 /dev/intel-fpga-port.*
sudo chmod 666 /sys/class/fpga/intel-fpga-dev.*/intel-fpga-port.*\
/userclk_freqcmd
sudo chmod 666 /sys/class/fpga/intel-fpga-dev.*/intel-fpga-port.*\
/userclk_freqcntrcmd
sudo chmod 666 /sys/class/fpga/intel-fpga-dev.*/intel-fpga-port.*/errors/clear
```

D. Memlock Limit

Depending on the requirements of your application, you may also want to increase the maximum amount of memory that a user process can lock. The exact method may vary with your Linux distribution.

Use `ulimit -l` to check the current memlock setting:

```
ulimit -l
```

To permanently remove the locked memory limit for a regular user, add the following lines to `/etc/security/limits.conf`:

```
user1    hard   memlock           unlimited
user1    soft   memlock           unlimited
```

The previous commands remove the limit on locked memory for `user1`. To remove memory locks for all users, replace `user1` with `*`:

```
*    hard   memlock           unlimited
*    soft   memlock           unlimited
```

Note: Settings in the `/etc/security/limits.conf` file do not apply to services. To increase the locked memory limit for a service, modify the application's `systemd` service file to add the following line:

```
[Service]
LimitMEMLOCK=infinity
```

E. Hugepage Settings

Use the hugepage command to to reserve `2 MB-hugepages` or `1 GB-hugepages`. For example, the `hello_fpga` sample requires several `2 MB-hugepages`.

The following command below reserves 20, `2 MB-hugepages`:

```
sudo sh -c 'echo 20 > /sys/kernel/mm/hugepages/hugepages-2048kB/nr_hugepages'
```

The following command below reserves 4, `1 GB-hugepages`:

```
sudo sh -c 'echo 4 > /sys/kernel/mm/hugepages/hugepages-1048576kB/nr_hugepages'
```

Note: To make these changes permanent, include them as part of `/etc/bashrc`.

F. Troubleshooting Frequently Asked Questions (FAQ)

F.1. Why do I see a "No Suitable slots found" message when running `fpgaconf` on my AFU image?

If you see a **No suitable slots found** message, ensure that your FIM version is compatible with your AFU image by completing the following steps:

1. Refer to Identifying the Flash Image and BMC Firmware on page 20 to determine the required *<FIM version>*.
2. To verify that the AFU is compatible with the FIM version, run the following command:

```
packager gbs-info --gbs=<gbs-file>
```

For example, for `nlb_mode_0.gbs` run the following command:

```
packager gbs-info --gbs=$OPAE_PLATFORM_ROOT/hw/samples/nlb_mode_0/bin/
\nlb_mode_0.gbs
```

Sample output:

```
{
    "version": 1,
    "afu-image": {
        "interface-uuid": "69528db6-eb31-577a-8c36-68f9faa081f6",
        "afu-top-interface": {
            "class": "ccip_std_afu"
        },
        "magic-no": 488605312,
        "power": 0,
        "accelerator-clusters": [
            {
                "total-contexts": 1,
                "name": "nlb_400",
                "accelerator-type-uuid": "d8424dc4-a4a3-c413-
f89e-433683f9040b"
            }
        ]
    }
}
```

The interface-uuid should match the FIM version (PR interface ID) you found in Identifying the Flash Image and BMC Firmware on page 20.

F.2. How do I flash the FIM or program the AFU in a multicard system?

The OPAE commands `fpgaflash`, `fpgabist`, and `fpgaconf` use the PCIe bus, device and function numbers as arguments to target the specific Intel PAC.

Use the `--help` option for details on how to use these commands. For example:

```
fpgaflash --help
```

F.3. Which environment variables are required?

To ensure all environment variables are set, you must source the initialization script that is provided as part of the installer.

```
source init_env.sh
```

F.4. What actions do I take if I see the error message "Error enumerating resources: no driver available"?

1. Validate that your card is detected by PCIe.

   ```
   lspci | grep 09c4
   ```

   If it is not detected, remove the card and then plug it back in again.
2. Reinstall OPAE by following steps listed the *Installing the OPAE Software* section.

Related Information

Installing the OPAE Software Package on page 14

F.5. Troubleshooting OPAE Installation on CentOS

Find the Intel FPGA drivers loaded.

```
sudo lsmod | grep fpga
```

If no output is returned, follow the below troubleshooting steps.

1. Find the kernel version used by the system

   ```
   uname -a
   ```

   Sample Output:

   ```
   Linux  3.10.0-957.el7.x86_64 #1 SMP Mon Sept 21 23:36:36 UTC 2018 x86_64
   x86_64 x86_64 GNU/Linux
   ```
2. Update the kernel source

   ```
   sudo yum install "kernel-devel-uname-r == $(uname -r)"
   ```
3. Remove the old kernel header and install the relevant kernel headers.

   ```
   sudo yum remove kernel-headers.x86_64
   sudo yum  install kernel-headers-`uname -r`
   ```
4. Remove the Intel FPGA driver.

   ```
   sudo yum remove opae-*.x86_64
   ```

5. Reinstall the driver by either running the install script (`setup.sh`) or the command below:

```
cd $OPAE_PLATFORM_ROOT/sw
sudo yum install opae-*.rpm
```

In some cases, if you don't have the right kernel, you may need to update kernel using:

```
sudo yum update
```

Reboot the system and select the new kernel in the grub menu. Then, you follow the above steps 2 to 5.

G. Documentation Available for the Intel Acceleration Stack for Intel Xeon CPU with FPGAs 1.2 Release

The following documents are on the Intel FPGA web page. To access a document, click the link.

Table 8. Documentation Available for the Intel Acceleration Stack for Intel Xeon CPU with FPGAs 1.2 Release

Document	Link to Access Document
Intel Acceleration Stack Quick Start Guide for Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA	Intel Acceleration Stack Quick Start Guide for Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA
10 Gbps Ethernet AFU Design Example User Guide	10 Gbps Ethernet Accelerator Functional Unit (AFU) Design Example User Guide
40 Gbps Ethernet AFU Design Example User Guide	40 Gbps Ethernet Accelerator Functional Unit (AFU) Design Example User Guide
Open Programmable Acceleration Engine (OPAE) C API Programming Guide	GitHub Link
Open Programmable Acceleration Engine (OPAE) Linux Device Driver Architecture Guide	GitHub Link
Open Programmable Acceleration Engine (OPAE) Tools Guide	GitHub Link
Intel Accelerator Functional Unit (AFU) Simulation Environment (ASE) User Guide	GitHub Link
Intel Accelerator Functional Unit (AFU) Simulation Environment (ASE) Quick Start User Guide	Intel Accelerator Functional Unit (AFU) Simulation Environment (ASE) Quick Start User Guide
Acceleration Stack for Intel Xeon CPU with FPGAs Core Cache Interface (CCI-P) Reference Manual	Acceleration Stack for Intel Xeon CPU with FPGAs Core Cache Interface (CCI-P) Reference Manual
Accelerator Functional Unit (AFU) Developer's User Guide	Accelerator Functional Unit (AFU) Developer's User Guide
Streaming DMA Accelerator Functional Unit (AFU) User Guide	Streaming DMA Accelerator Functional Unit AFU User Guide
Native Loopback Accelerator Functional Unit (AFU) User Guide	Native Loopback Accelerator Functional Unit (AFU) User Guide
Networking Interface for Open Programmable Acceleration Engine: Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA Previously known as *HSSI User Guide for Intel Programmable Acceleration Card (PAC) Intel Arria 10 GX FPGA*	Networking Interface for Open Programmable Acceleration Engine: Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA
OpenCL on Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA Quick Start User Guide	OpenCL on Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA Quick Start User Guide

continued...

Document	Link to Access Document
Intel Acceleration Stack Quick Start Guide for Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA with Intel Arria 10 GX FPGA	Intel Acceleration Stack Quick Start Guide for Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA with Intel Arria 10 GX FPGA
Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA with Intel Arria 10 GX FPGA Datasheet	Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA with Intel Arria 10 GX FPGA Datasheet
Intel Acceleration Stack for Intel Xeon CPU with FPGAs v1.2 Release Notes	Intel Acceleration Stack for Intel Xeon CPU with FPGAs v1.2 Release Notes
Intel Acceleration Stack for Intel Xeon CPU with FPGAs v1.2 Errata	Intel Acceleration Stack for Intel Xeon CPU with FPGAs v1.2 Errata