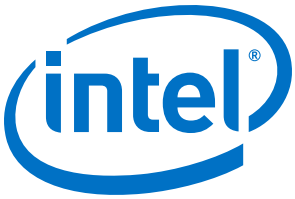


Intel Acceleration Stack Quick Start Guide for Intel® Programmable Acceleration Card with Intel® Arria® 10 GX FPGA

Updated for Intel® Acceleration Stack for Intel® Xeon® CPU with FPGAs: **1.0 Production**

Subscribe
Send Feedback

UG-20064 | 2018.04.11
Latest document on the web: **PDF** | **HTML**

Contents

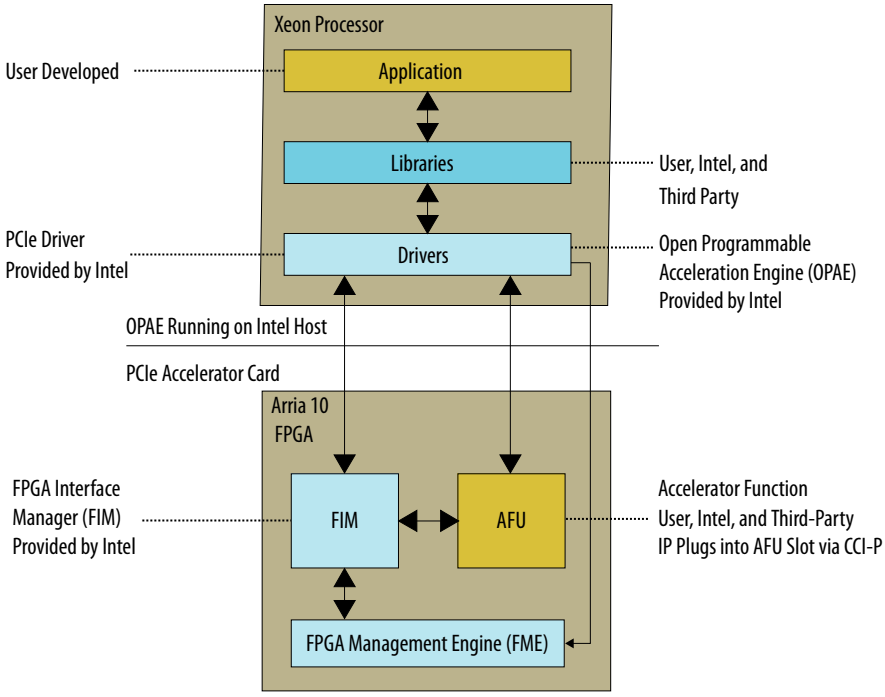
1. Introduction to the Intel® Acceleration Stack for Intel® Xeon® CPU with FPGAs

Intel® Acceleration Stack for Intel Xeon® CPU with FPGAs is a collection of software, firmware, and tools that allow software developers to leverage 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.

This guide specifically targets the Intel Programmable Acceleration Card with Intel Arria® 10 GX FPGA, referred to as Intel PAC with Intel Arria 10 GX FPGA throughout this document. The Intel PAC with Intel Arria 10 GX FPGA accelerator is connected to the Intel Xeon processor through the PCIe* interface on a motherboard.

Note: The Acceleration Stack targets the Intel PAC with Intel Arria 10 GX FPGA. This release does not support the A10PL4 cards.

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 special, 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. For

more information, refer to Figure 1 on page 3. The Open Programmable Acceleration Engine (OPAE) software running on the Intel Xeon processor handles all the details of the reconfiguration process.

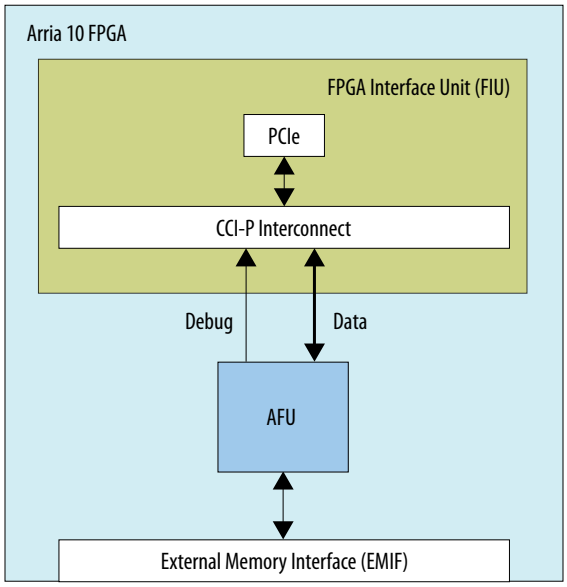
Reconfiguration is one of many utilities that the OPAE provides. The OPAE also provides libraries, drivers, and sample programs useful for AFUs development.

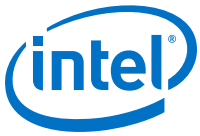
To facilitate dynamically loading AFUs, the Acceleration Stack includes the following two components:

- The FPGA Interface Manager (FIM), which provides a framework for loading AFUs within the Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA, referred to as Intel PAC with Intel Arria 10 GX FPGA throughout this document, 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). The FIM also includes the PR regions for loading AFUs. At power up, Intel PAC with Intel Arria 10 GX FPGA is configured from on-board FPGA configuration flash, containing the FIM bitstream image. The PR regions are initially empty until OPAE is used to load AFUs. The framework provided by the FIM cannot be altered. The current release of the Acceleration Stack for the Intel PAC with Intel Arria 10 GX FPGA supports a single PR region in the FIM.
- Loadable AFU images, which are dynamically loaded by OPAE into the PR regions in the FIM. The Acceleration Stack supports creating AFU images with RTL and OpenCL* design flows. An AFU image is the combination of the AFU's PR region bitstream generated from the supported design flows and metadata used to provide OPAE information on the AFU's characteristics and operational parameters used at load time. The current release of the Intel PAC with Intel Arria 10 GX FPGA supports dynamically swapping AFU images in a single PR region per installed Intel PAC with Intel Arria 10 GX FPGA.

Figure 2. Arria 10 with a Single AFU PR Region

This figure provides additional information about the AFU.





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 memory. Each DDR4 memory bank interface is 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 features a single QSFP+ network port, and two banks of DDR4. The accelerator card includes two banks of 4GB DDR4-SDRAM totaling 8GB. Each bank has a 72-bit memory which includes error correction codes (ECC).

The current release of the Acceleration Stack does not support the Quad Small-Form-Factor Pluggable (QSFP+) interface.

Related Information

- Avalon Memory-Mapped Interfaces
- Acceleration Stack for Intel Xeon CPU with FPGAs Core Cache Interface (CCI-P) Reference Manual

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 • Power monitoring • Performance monitoring • Partial reconfiguration • Global errors
Intel Xeon Scalable Platform with Integrated FPGA	Integrated FPGA Platform	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).
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 Intel Xeon Processor with Integrated FPGA and Intel PAC with Intel Arria 10 GX FPGA platforms include PR regions in the FPGA. You can reconfigure these regions at run time to implement different AFUs as system requirements dictate.
RAS	Reliability, Assessability 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 Programmable Acceleration Card with Intel Arria 10 GX FPGA	Intel PAC with Intel Arria 10 GX FPGA	PCIe accelerator card with an Intel Arria 10 FPGA. Programmable Acceleration Card is abbreviated PAC. Contains a FPGA Interface Manager (FIM) that connects to an Intel Xeon processor over PCIe bus.
Intel Xeon Scalable Platform with Integrated FPGA	Integrated FPGA Platform	A platform with the Intel Xeon and FPGA in a single package and sharing a coherent view of memory using the Intel Ultra Path Interconnect (UPI).

1.3. Differences Between the Hardware Platforms Supported by the Acceleration Stack

The following table lists the key differences between the hardware platforms supported by the Acceleration Stack.

Table 3. Hardware Platforms Supported by the Acceleration Stack Feature Comparison

Feature	Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA	Intel Xeon Scalable Platform with Integrated FPGA	Notes
Cached Memory Link	No	Yes	The Intel Xeon Processor with Integrated FPGA platform includes a low latency coherent link to host memory. The Intel PAC with Intel Arria 10 GX FPGA platform does not include this link; consequently, UMsg is not supported.
PCIe Interfaces	1, Gen3x8	2, Gen3x8	The Intel Xeon Processor with Integrated FPGA platform has two PCIe links.
Physical attachment	PCIe accelerator card	Internal on-die package	The Intel PAC with Intel Arria 10 GX FPGA shares the CCI-P interface for AFUs.This interface abstracts the physical link differences between the Intel Xeon Processor with Integrated FPGA and Intel PAC with Intel Arria 10 GX FPGA platforms.
Quick Path Interconnect (QPI)	Not supported	Supported	The Intel PAC with Intel Arria 10 GX FPGA shares the CCI-P interface for AFUs.This interface abstracts the physical link differences

continued...

Feature	Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA	Intel Xeon Scalable Platform with Integrated FPGA	Notes
			between the Intel Xeon Processor with Integrated FPGA and Intel PAC with Intel Arria 10 GX FPGA platforms.
Thermal shutdown	Supported	Supported	None
Remote JTAG	Supported	Supported	None
Reliability, Assessability and Serviceability (RAS)	Supported	Supported	None
Performance counters	Supported	Supported	None
Vtune integration	Not Supported	Not Supported	None
Telemetry – temperature and power reporting to Control and Status Registers (CSRs)	Temperature readings supported; but power readings require accessing BMC through an USB cable or sideband channel.	Supported	None
Private Memory	2 banks of 4GB DDR4 SDRAM	Not Supported	None
Physical Links between Host and FPGA	PCIe0	UPI, PCIe0, PCIe1	None

2. Getting Started

Related Information

- Intel Acceleration Stack for Intel Xeon CPU with FPGAs 1.0 Errata
  For more information on the latest errata
- Intel Acceleration Stack for Intel Xeon CPU with FPGAs 1.0 Production Release Notes
  For information on enhancements and known issues

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 image

The following servers have been tested for this release:

- Dell* R640
- Dell R740

The best known server configuration with the Intel PAC with Intel Arria 10 GX FPGA card must include:

- Intel Xeon processor
- A PCI Express* x16 Slot
- RAM: 48 GB
- Red Hat* Enterprise Linux* (RHEL) 7.4 or CentOS 7.4

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

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:

```
yum groupinstall
```

Table 4. Useful Linux Commands

The following Linux commands provide information about your system.

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

2.3. 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.

Both Acceleration Stack options are available on the Intel FPGA Acceleration Hub Download page.

The following table describes each Acceleration Stack package.

Table 5. Intel Acceleration Stack Download Options

Details	Acceleration Stack for Runtime	Acceleration Stack for Development
Purpose	Software development of runtime host application	Accelerator function development using Intel Quartus Prime Pro Edition and Acceleration Stack
Intel Acceleration Stack	Acceleration Stack version 1.0 Production	Acceleration Stack version 1.0 Production

continued...

Details	Acceleration Stack for Runtime	Acceleration Stack for Development
Intel Quartus Prime Software	Intel Quartus Prime Programmer Only	Intel Quartus Prime Pro Edition 17.0.0 including SR-IOV license
OpenCL Software	Intel FPGA Runtime Environment for OpenCL 17.0	Intel FPGA Software Development Kit (SDK) for OpenCL 17.0
Download Size	650 MB	12 GB

2.3.1. Installing the Intel Acceleration Stack Runtime package on the Host Machine

1. Extract the runtime archive file:
```
$ tar xvf a10_gx_pac_ias_1_0_prq_rte_installer.tar.gz
```
2. Go to the installation directory.
```
$ cd a10_gx_pac_ias_1_0_prq_rte_installer
```
3. Run setup.sh.
```
$ ./setup.sh
```
4. Accept the license.
5. When you receive an installation directory prompt, you can select an install directory. Otherwise, the installer uses the default directory at `/home/<username>/intelrtestack` to install Intel Quartus Prime Programmer and OpenCL RTE.
6. Run the initialization script to set the required environment variables.
```
$ source /home/<username>/intelrtestack/init_env.sh
```

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

1. Extract the runtime archive file:
```
$ tar xvf a10_gx_pac_ias_1_0_prq_dev_installer.tar.gz
```
2. Go to the installation directory.
```
$ cd a10_gx_pac_ias_1_0_prq_dev_installer
```
3. Run setup.sh.
```
$ ./setup.sh
```
4. Accept the license.
5. When you receive an installation directory prompt, you can select an install directory. Otherwise, the installer uses the default directory at `/home/<username>/inteldevstack` to install Intel Quartus Prime Pro Edition and OpenCL SDK.
6. Run the initialization script to set the required environment variables.
```
$ source /home/<username>/inteldevstack/init_env.sh
```

2.4. Extracting the Intel PAC with Intel Arria 10 GX FPGA Package

The archive file, `a10_gx_pac_ias_1_0_prq.tar.gz`, includes the files for the Intel PAC with Intel Arria 10 GX FPGA 1.0 Production Release.

Depending on your prior install selections, the archive file, `a10_gx_pac_ias_1_0_prq.tar.gz`, is placed in one of the following install directories:

- Acceleration Stack for Runtime: `/home/<username>/intelrtestack`
- Acceleration Stack for Development: `/home/<username>/inteldevstack`
- Custom install directory: `/<custom_install_directory>`

Complete the following commands to extract the archive file and define an environment variable named DCP_LOC pointing to the extracted release location.

1. `$ mkdir a10_gx_pac_ias_1_0_prq`

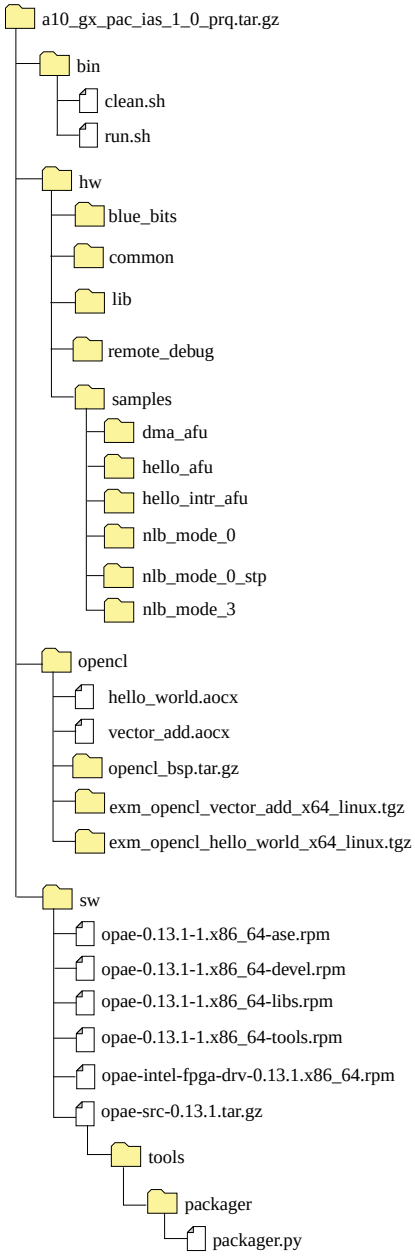
   Note: You may create this directory anywhere in your work area.

2. `$ cd a10_gx_pac_ias_1_0_prq`
3. `$ export DCP_LOC=`pwd``
4. `$ tar xf <path_to_install_directory>/a10_gx_pac_ias_1_0_prq.tar.gz`

   Note: The *<path_to_install_directory>* is a variable name that denotes your selected install directory from the bulleted list of install directories.

Figure 3. Intel PAC with Intel Arria 10 GX FPGA 1.0 Production Directory Structure

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



Each time you reboot your computer after your initial package extraction, you must source the initialization script and set the DCP_LOC environment variable with the following commands:

1. Run the initialization script to set the required environment variables.

- If you are using the Acceleration Stack runtime package, type:

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

- If you are using the Acceleration Stack development package, type:

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

2. Set the environment variable `DCP_LOC`.
   a. Change to directory where `a10_gx_pac_ias_1_0_prq.tar.gz` was extracted.
   b.
```
$ export DCP_LOC=`pwd`
```

2.5. 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.

Note: The Production release of the Acceleration Stack targets the Intel PAC with Intel Arria 10 GX FPGA . This release does not work with the older PL4 cards.

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.
3. A Micro USB cable connected to the Intel PAC with Intel Arria 10 GX FPGA and the host's USB port is required for the following reasons:
   - To update firmware on the card
   - If programming of flash over PCIe (fpgaflash tool) fails, flash has to be programmed over JTAG using the Intel Quartus Prime Programmer
   - If flash is empty or has the pre-alpha 1.0 version of the FPGA Interface Manager (FIM) image

3. Installing the OPAE Software Package

This section discusses steps required for the OPAE software includes the Intel FPGA driver, the OPAE software package.

After completing the OPAE software installation,

- Intel FPGA Driver is installed and loaded
- OPAE source is available at: `$DCP_LOC/sw/opae-0.13.1-1`.
- RPM flow will install OPAE binaries at `/usr/bin`
- RPM flow will install OPAE libraries at `/usr/lib`
- RPM flow will install OPAE headers at `/usr/include`

3.1. Installing the OPAE Intel FPGA Driver

Build and install the Intel FPGA Driver using the Dynamic Kernel Module Support (DKMS) framework.

Complete the following steps to install the Intel FPGA driver:

1. Remove any previous version of the OPAE FPGA driver by running the command:

```
$ sudo yum remove opae-intel-fpga-drv.x86_64
```

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

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

3. Go to directory with OPAE installation packages.

```
$ cd $DCP_LOC/sw
```

4. Install the Intel FPGA kernel drivers

```
$ sudo yum install $DCP_LOC/sw/opae-intel-fpga-drv-0.13.1.x86_64.rpm
```

5. Check the Linux kernel installation:

```
lsmod | grep fpga
```

Sample output:

```
lsmod | grep
fpgaintel_fpga_fme         51563  0
intel_fpga_afu         31735  0
fpga_mgr_mod           14693  1 intel_fpga_fme
intel_fpga_pci         25804  2 intel_fpga_afu,intel_fpga_fme
```

3.2. Installing and Building the OPAE Library

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

```
$ 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 rsync
```

Note: Some packages may already be installed. This command only installs the packages that are missing.

3.3. Installing the OPAE Library from Prebuilt Binaries

Installing the rpm allows for prebuilt OPAE SW binaries, libraries and required header to be installed in their respective default paths.

Complete the following steps to install the OPAE software:

1. Remove any previous version of OPAE library installed.

   ```
   $ sudo yum remove opae-ase.x86_64
   $ sudo yum remove opae-tools.x86_64
   $ sudo yum remove opae-devel.x86_64
   $ sudo yum remove opae-libs.x86_64
   ```

2. Install shared libraries at location `/usr/lib`, required for user applications to link against:

   ```
   $ sudo yum install opae-0.13.1-1.x86_64-libs.rpm
   ```

3. Install the OPAE header at location `/usr/include`:

   ```
   $ sudo yum install opae-0.13.1-1.x86_64-devel.rpm
   ```

4. Install the OPAE provided tools at location `/usr/bin` (For example: `fpgaconf` and `fpgainfo`):

   ```
   $ sudo yum install opae-0.13.1-1.x86_64-tools.rpm
   ```

   For more information about tools, refer to the OPAE tools document.

5. Install the ASE related shared libraries at location `/usr/lib`:

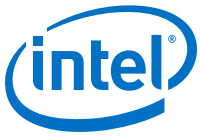
   ```
   $ sudo yum install opae-0.13.1-1.x86_64-ase.rpm
   ```

6. `Update dynamic linker run-time bindings`

   ```
   $ sudo ldconfig
   ```

7. List all of the `libopae` libraries in the `lib` directory:

   ```
   $ ls /usr/lib/libopae*
   /usr/lib/libopae-c-ase.so
   /usr/lib/libopae-c.so.0
   /usr/lib/libopae-c-ase.so.0
   /usr/lib/libopae-c++.so.0
   /usr/lib/libopae-c-ase.so.0.13.1
   /usr/lib/libopae-c.so.0.13.1
   /usr/lib/libopae-c++-nlb.so
   ```



```
/usr/lib/libopae-c++.so.0.13.1
/usr/lib/libopae-c++-nlb.so.0
/usr/lib/libopae-c++-utils.so
/usr/lib/libopae-c++-nlb.so.0.13.1
/usr/lib/libopae-c++-utils.so.0
/usr/lib/libopae-c.so
/usr/lib/libopae-c++-utils.so.0.13.1
/usr/lib/libopae-c++.so
```

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

If you prefer building the OPAE Software from source (NOT REQUIRED) and do not have access to the default install directory: `/usr/`, you must configure and build the Open Programmable Acceleration Engine (OPAE) software package using the `cmake` and `make` commands.

Complete the following steps to build the OPAE software:

1. `$ cd $DCP_LOC/sw`
2. `$ tar xf opae-src-0.13.1.tar.gz`
3. `$ cd opae-0.13.1-1`
4. `$ mkdir build && cd build`
5. `$ cmake .. -DBUILD_ASE=ON -DCMAKE_INSTALL_PREFIX=<path to install directory>`

   (For example: `$ cmake .. -DBUILD_ASE=ON -DCMAKE_INSTALL_PREFIX=/home/john/opaeinstall`)

   By default, if you followed the rpm install flow, the binaries, libraries and include files get installed under `/usr/`.

   Note: You may get an error because the `cmake` command cannot find the git repository. You can safely ignore this error message. The git repository is not required to successfully build the OPAE software.
6. `$ make install`
   This `make` command builds the following packages:
   - Executables in <path to install directory>
   - Libraries in <path to install directory>
7. `$ make doc`

   The headers are in `<path to install directory>/include`

   This `make` command makes the Doxygen documentation in `<path to install directory>`.

4. Identify the FPGA Interface Manager (FIM) Version Loaded

In order to identify the FIM version loaded, you must run the `fpgainfo` tool:

```
$ sudo fpgainfo fme
```

Sample Output:

```
//****** FME ******//
Class Path              : /sys/class/fpga/intel-fpga-dev.0/intel-fpga-fme.0
Device Path             : /sys/devices/pci0000:00/0000:00:03.0/0000:04:00.0/
fpga/intel-fpga-dev.0/intel-fpga-fme.0
Bus                     : 0x04
Device                  : 0x00
Function                : 0x00
Version                 : 0
Ports Num               : 1
Socket Id               : 0
Pr Interface Id         : ce489693-98f0-5f33-946d-560708be108a
Object Id               : 254803968
```

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

Table 6. Intel Acceleration Stack 1.0 Reference Table

Intel Acceleration Stack Version	FPGA Interface Manager (FIM) Version (Partial Reconfiguration (PR) Interface ID)	Open Programmable Acceleration Engine (OPAE) Version
1.0 Production(1)	ce489693-98f0-5f33-946d-560708be108a	0.13.1
1.0 Beta	3d949b98-7b30-5a9ab296-4530a780a3f9	0.13.0
1.0 Alpha	d4a76277-07da-528db623-8b9301feaffe	0.11.0

(1) The factory partition of the configuration flash contains the Acceleration Stack 1.0 Alpha version. When the image in the user partition cannot be loaded, a flash failover occurs and the factory image is loaded instead. After a flash failover occurs, the PR ID reads as d4a76277-07da-528d-b623-8b9301feaffe.

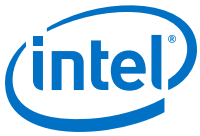


Table 7. Selecting the Correct Update Method

Acceleration Stack Release Version	FIM Update Instructions
Unknown or 1.0 Alpha and Earlier	1. Go to "Update the Board Management Controller (BMC) Configuration and Firmware". 2. Go to "Update Flash with FPGA Interface Manager (FIM) Image using Intel Quartus Prime Programmer".
1.0 Beta	1. Go to "Update the Board Management Controller (BMC) Configuration and Firmware". 2. Go to "Update Flash with FPGA Interface Manager (FIM) Image using `fpgaflash` Tool".
1.0 PRQ or Newer	1. Go to "Update Flash with FPGA Interface Manager (FIM) Image using `fpgaflash` Tool".

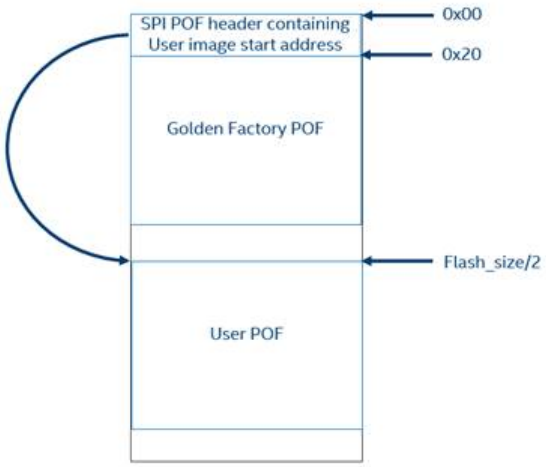
Related Information

- Open Programmable Acceleration Engine (OPAE) Tools Guide (fpgabist)
- Updating the Board Management Controller (BMC) Configuration and Firmware on page 33
- Update Flash with FPGA Interface Manager (FIM) Image using fpgaflash Tool on page 20
- Update Flash with FPGA Interface Manager (FIM) Image using Intel Quartus Prime Programmer on page 35

5. Update Flash with FPGA Interface Manager (FIM) Image using `fpgaflash` Tool

Flash layout has two partitions, one with the factory image (golden image) and the other with the user image. If the user image is corrupted, Intel Arria 10 FPGA automatically uses the factory image.

Figure 4. Flash Layout



Follow these steps to load the FIM into the user partition of the Intel PAC with Intel Arria 10 GX FPGA card onboard flash memory:

1. To update the user POF, use command:

```
# sudo fpgaflash user $DCP_LOC/hw/blue_bits/dcp_1_0.rpd
```

The flash erase, write and verify process takes several minutes to complete.

Expected Output:

```
flash size is 134217728
reversing bits
erasing flash
writing flash
reading back flash
verifying flash
flash successfully verified
```

Note: If you have multiple Intel PAC with Intel Arria 10 GX FPGA cards installed, you can flash a specific card by specifying the PCIe `bus:device.function`, for example:

```
sudo fpgaflash user $DCP_LOC/hw/blue_bits/dcp_1_0.rpd 04:00.0
```

Note: If a catastrophic system event such as power loss occurs during the flash programming process, first retry the `fpgaflash` command once the system recovers. If `fpgaflash` fails to find the Intel PAC with Intel Arria 10 GX FPGA due to it not enumerating, then warm reboot the system and retry the `fpgaflash` command. If the issue persists, then follow the procedure in the "Update Flash with FPGA Interface Manager (FIM) Image using Intel Quartus Prime Programmer" section to restore the FPGA configuration flash.

2. Power cycle host machine. A simple reboot is insufficient.
3. Run the following command to verify that PCIe enumeration has assigned a bus:

```
$ lspci -nn | grep 8086:09c4
```

Sample output:

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

If this command does not return one or more devices in its output as shown above, then

a. Reboot to see if PCIe enumeration has assigned a bus.
b. If reboot does not help, follow the steps in the "Update Flash with FPGA Interface Manager (FIM) Image using Intel Quartus Prime Programmer" section to update the flash using Intel Quartus Prime Programmer.

Related Information

- Installing the OPAE Library from Prebuilt Binaries on page 16
- Update Flash with FPGA Interface Manager (FIM) Image using Intel Quartus Prime Programmer on page 35
- Updating the Board Management Controller (BMC) Configuration and Firmware on page 33
- Identify the FPGA Interface Manager (FIM) Version Loaded on page 18

6. 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 required by the FPGA fpgadiag utility:

```
$ 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 $DCP_LOC/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 2887494796  2935937472
0            0             0             0          0      12000074611
6.160 GB/s     6.263 GB/s

                         VH0_Rd_Count VH0_Wr_Count VH1_Rd_Count
VH1_Wr_Count VL0_Rd_Count VL0_Wr_Count
                         2887494796   2935937473            0
0            0             0
Finished Executing NLB (FPGA DIAG)Tests

Built-in Self-Test Completed.
```

3. Configure and run diagnostics with DMA AFU image.

```
$ sudo fpgabist $DCP_LOC/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

Built-in Self-Test Completed.
```

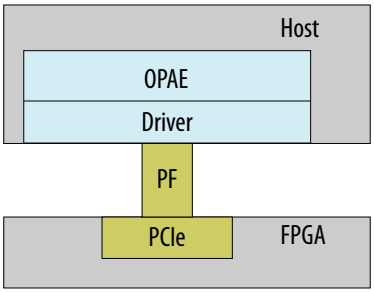
Related Information

OPAE FPGA Tools - fpgabist
For more information about the fpgabist utility

7. Using the OPAE in a Non-Virtualized Environment

This section shows OPAE examples running directly on the BareMetal operating system without a virtual machine nor SR-IOV.

Figure 5. OPAE Driver in Non-Virtualized Mode



7.1. 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 Intel Acceleration Stack for Intel Xeon CPU with FPGAs 1.0 Production installation includes the following AFU images in the `$DCP_LOC/hw/samples` directory:

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

Note: You must have an IP-PCIe/SR-IOV license to generate a custom AFU PR bitstream that is part of the loadable AFU image. For more information, refer to the "Intel FPGA Software Licensing Support" web page on the Intel FPGA website to get a license.

Related Information

Intel FPGA Software Licensing Support

7.2. OPAE Sample Application Programs

7.2.1. 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 Intel PAC with Intel Arria 10 GX 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 $DCP_LOC/hw/samples/nlb_mode_0/bin/nlb_mode_0.gbs
```

   Note: If you see an "Error enumerating FPGAs: not found" message, ensure that your FPGA Interface Manager version is compatible with your AFU image.

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"
```

To compile the source code for `hello_fpga` located at `$OPAE_LOC/samples/hello_fpga.c`:

3. `$ cd $DCP_LOC/sw`
4. `$ tar xf $DCP_LOC/sw/opae-src-0.13.1.tar.gz`

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

5. `$ cd opae-0.13.1-1`
6. `$ export OPAE_LOC=`pwd``
7. Type the following:

```
$ sudo gcc -o hello_fpga -std=gnu99 -rdynamic \
-ljson-c -luuid -lpthread \
-lopae-c -lm -Wl,-rpath -lopae-c -z noexecstack -z relro -z now \
-fstack-protector -fPIE -fPIC -pie -O2 -D_FORTIFY_SOURCE=2 \
-Wformat -Wformat-security $OPAE_LOC/samples/hello_fpga.c
```

   Note: If you built the OPAE from the source instead of the rpm install, you must provide the OPAE install path in the following command:

```
$ sudo gcc -o hello_fpga -std=gnu99 -rdynamic \
-ljson-c -luuid -lpthread\
-lopae-c -lm -Wl,-rpath -lopae-c -z noexecstack -z relro -z now \
-fstack-protector -fPIE -fPIC -pie -O2 -D_FORTIFY_SOURCE=2 \
-Wformat -Wformat-security\
-I <path to opae install>/include -L<path to opae install>/lib \
$OPAE_LOC/samples/hello_fpga.c
```

   As a sudo user:

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:<path to opae install>/lib
```

8. `$ sudo ./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_LOC/samples/hello_fpga.c`
- *Native Loopback Accelerator Functional Unit (AFU) User Guide* for AFU register descriptions.

Related Information

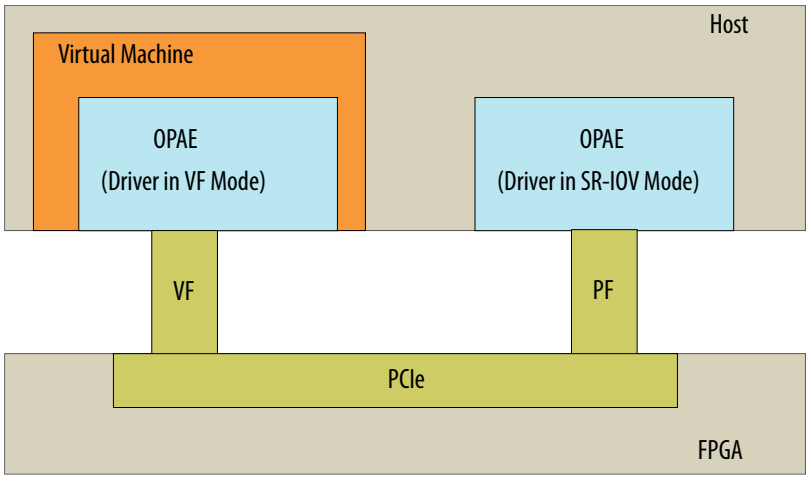
- Installing the OPAE Library from Prebuilt Binaries on page 16
- Documentation Available for the Intel Acceleration Stack for Intel Xeon CPU with FPGAs 1.0 Production Release on page 38
- Native Loopback Accelerator Functional Unit (AFU) User Guide

8. 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 6. 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 has been connected to a VF through OPAE is not able to initiate partial reconfiguration. This is because a permission table in the FME only allows partial reconfiguration through a PF. However, there is a mailbox capability that a VM connected to a VF can signal to the management application connected to the PF that it would like to initiate partial reconfiguration. That management process then as the ability and can arbitrate that request. Consequently, you must load the AFU image on the host before continuing with the steps to create a virtualized environment.

Note: The mailbox capability does not allow an AFU bitstream to be transferred to the PF directly as this exposes a potential security threat.

Run the following command on the host to load the AFU image. This is the AFU image required to run the example, *Running the Hello FPGA Example in a Virtualized Environment*, that you can run after setting up the virtualized environment.

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

Related Information

- Getting Started on page 9
- Installing the OPAE Software Package on page 15

8.1. 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. Reboot 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_jarrod--z620--lab-root ro intel_iommu=on
   crashkernel=auto rd.lvm.lv=cl_jarrod-z620-lab/root
   rd.lvm.lv=cl_jarrod-z620-lab/swap rhgb quiet
   ```

8.2. Configuring the VF Port on the Host

By default, the PF controls the AFU port. The following procedure transfers AFU control to the VF. The AFU under VF control is then accessible from the applications running on the VM.

1. `$ export port_path=$(find /sys/class/fpga/intel-fpga-dev.* \ -maxdepth 1 -follow -iname intel-fpga-port.0)`
2. `$ export link_path=$(readlink -m /$port_path/../)`
3. `$ export pci_path=$link_path/../../`
4. Release the port controlled by the PF using the `fpgaport` tool:

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

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

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

6. `$ 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]
   ```

   The VF is enabled. 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 is: 000:04:00.1

7. Load the vfio-pci driver:

   ```
   $ sudo modprobe vfio-pci
   ```

8. 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"
```

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

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

   Sample output:

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

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

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

8.3. Running the Hello FPGA Example on Virtual Machine

This section assumes that the Virtual Machine (VM) is setup 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.

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. `$ tar xf $DCP_LOC/sw/opae-src-0.13.1.tar.gz`
3. `$ cd opae-0.13.1-1`
4. `$ export OPAE_LOC=`pwd``
5. Type the following command:

```
$ sudo gcc -o hello_fpga -std=gnu99 -rdynamic \
-ljson-c -luuid -lpthread \
-lopae-c -lm -Wl,-rpath -lopae-c -z noexecstack -z relro -z now \
 -fstack-protector -fPIE -fPIC -pie -O2 -D_FORTIFY_SOURCE=2 \
-Wformat -Wformat-security $OPAE_LOC/samples/hello_fpga.c
```

6. `$ 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_LOC/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 15
- Running the Hello FPGA Example on page 24
- Documentation Available for the Intel Acceleration Stack for Intel Xeon CPU with FPGAs 1.0 Production Release on page 38
- Native Loopback Accelerator Functional Unit (AFU) User Guide

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

1. Uninstall the driver on the VM:

```
$ yum remove opae-intel-fpga-drv.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. Set to 0 VFs and disable SR-IOV:

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

5. Assign the port to PF using fpgaport tool:

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

A. Additional Flag Settings to AFU Makefiles

Note: Makefiles are located under `$DCP_LOC/hw/samples/<AFU_NAME>/sw`.

The AFU samples are provided as example code but the software has not been hardened for deployment in a production environment. The following compiler flags may help mitigate security risks and are recommended for the AFU software if used in a production environment:

```
# stack execution protection
LDFLAGS +=-z noexecstack

# data relocation and projection
LDFLAGS +=-z relro -z now

# stack buffer overrun detection
CFLAGS +=-fstack-protector

# Position independent execution
CFLAGS +=-fPIE -fPIC
LDFLAGS +=-pie

# fortify source
CFLAGS+=-O2 -D_FORTIFY_SOURCE=2

# format string vulnerabilities
CFLAGS+=-Wformat -Wformat-security
```

B. Updating the Board Management Controller (BMC) Configuration and Firmware

1. Loading FIM as a pre-requisite requires using the BittWorks II Toolkit-Lite along with an USB-Blaster cable to update the BMC configuration and firmware. For more information about the BittWorks II Toolkit-Lite, refer to the "BittWorks II Toolkit-Lite" web page.
2. To obtain the firmware and tools for updating the BMC firmware, refer to the BMC firmware information web page.
3. Run the following commands:
   - To determine if the Bittware tools are already installed:

     ```
     $ sudo yum list | grep bw2tk
     bw2tk-2017.4.x86_64                    1-full.el7.centos
     installed
     ```

   - To determine their versions:

     ```
     $ bwconfig –-version
     ```

4. If the BittWorks II Toolkit-Lite is installed, and the reported version is not 2017.4, remove the currently installed tools. The command varies with the OS release, but it typically is:

   ```
   $ sudo yum remove bw2tk-2017.4.x86_64
   ```

5. To install the tool, run the following command:

   ```
   $ sudo yum install bw2tk-lite-2017.4.el7.x86_64.rpm
   ```

6. To run Bittware tools without having to provide a complete path, type:

   ```
   $ export PATH=/opt/bwtk/2017.4L/bin/:$PATH
   ```

7. Run the following command to check if the the BMC configuration flag is set to `0x81`:

   ```
   $ bwmonitor --dev=0 --type=bmc --flags --read
   ```

   Expected Output:

   ```
   BMC flags set to 81
   ```

   Note: If you receive this error message:

   `"ERROR Item not found: could not open device 0"`, perform the following steps:

   a. `bwconfig --remove=0`

      Note: If you receive this error message: `"ERROR - 'remove' failed: Item not found".`, then you can ignore this message.

b. `bwconfig --scan=usb`

Sample output when the Intel PAC with Intel Arria 10 GX FPGA is detected:
`Scanning for devices`

```
[result]: Board Type (Name), Serial, VendorID, DeviceID, USB-Address
[0]: 0x5f (A10SA4) 201384 0x2528 0x0004 0x4
```

c. Run the `bwconfig` command to add the Intel PAC with Intel Arria 10 GX FPGA to the BittWorks II Toolkit-Lite managed device list:

```
$ bwconfig --add=usb --result=0
```

Sample output:

```
Scanning for devices

Board Type (Name), Serial, VendorID, DeviceID, USB-Address [0]: 0x50
(A10PL4) 832880 0x2528 0x0004 0x5 Device added as device "0"
```

8. If flag is not already set to 0x81, set the flag by running command:

```
$ bwmonitor --dev=0 --type=bmc --flags=0x81 --write
```

9. Verify firmware by typing the following command:

```
$ bwmonitor --dev=0 --version
```

Resultant output:

```
BwMonitor (cli) version 2017.4.233.31840
BMCLIB version : 2017.4.233.31840
MCU Firmware version : 0x68bf
```

If the firmware version is 0x68bf (26815), you may skip the rest of the steps in this section. If the firmware version is not 0x68bf (26815), proceed through the remaining steps in this section.

10. To update to the latest firmware version, 0x68bf (26815), run:

Note: You might see the following error message: `ERROR: Upgrade failed`. If so, you can ignore the message.

```
$ bwmonitor --dev=0 --type=bmc --write=1 --file=<path to location \
firmware hex file>
```

11. Power cycle host machine. A simple reboot is insufficient.
12. Verify firmware updated

Sample output:

```
bwmonitor --dev=0 --version
BwMonitor (cli) version 2017.4.233.31840
BMCLIB version : 2017.4.233.31840
MCU Firmware version : 0x68bf
```

Related Information

- Intel® Programmable Acceleration Card with Intel Arria® 10 GX FPGA
- BittWorks II Toolkit-Lite

C. Update Flash with FPGA Interface Manager (FIM) Image using Intel Quartus Prime Programmer

Note: This section is used to update flash with FIM if the current flash is empty or has pre-alpha FIM.

Note: A Micro USB cable is required.

1. `cd $DCP_LOC/hw/blue_bits/`
2. `export QUARTUS_HOME=<path to quartus installation>`
3. Find the Root port and End point of the Intel PAC with Intel Arria 10 GX FPGA card.

   ```
   $ lspci -tv | grep 09c4
   ```

   Example output 1:

   ```
   -+-[0000:d7]-+-00.0-[d8]----00.0 Intel Corporation Device 09c4
   ```

   Output shows that Endpoint is `d8:0.0` and Root port is `d7:0.0`.

   Example output 2:

   ```
    +-01.0-[03]----00.0 Intel Corporation Device 09c4
   ```

   Output shows that Endpoint is `3:0.0` and Root port is `0:1.0`.

   Example output 3:

   ```
   +-[0000:85]-+-02.0-[86]----00.0  Intel Corporation Device 09c4
   ```

   Output shows that Endpoint is `86:0.0` and Root port is `85:2.0`.

   Note: No output indicates that the flash is not programmed and the PCIe device enumeration did not occur. If this is the case, skip to step 5 on page 36 to program the FIM image and then go directly to step 7 on page 36 to power cycle the host machine.

4. Mask uncorrectable errors.
   a. Mask uncorrectable errors and correctable errors of FPGA:

      ```
      $ sudo setpci -s d8:0.0 ECAP_AER+0x08.L=0xFFFFFFFF
      $ sudo setpci -s d8:0.0 ECAP_AER+0x14.L=0xFFFFFFFF
      ```

   b. Mask uncorrectable errors and Mask correctable errors of RP:

      ```
      $ sudo setpci -s d7:0.0 ECAP_AER+0x08.L=0xFFFFFFFF
      $ sudo setpci -s d7:0.0 ECAP_AER+0x14.L=0xFFFFFFFF
      ```

5. Run the following Intel Quartus Prime Programmer command:

```
sudo $QUARTUS_HOME/bin/quartus_pgm -m JTAG -o 'pvbi;dcp_1_0.jic'
```

```
Info:
************************************************************
***** Info: Running Quartus Prime Programmer Info: Version
17.0.0 Build 290 04/26/2017 SJ Pro Edition Info: Copyright (C)
2017 Intel Corporation. All rights reserved. Info: Your use of
Intel Corporation's design tools, logic functions Info: and
other software and tools, and its AMPP partner logic Info:
functions, and any output files from any of the foregoing
Info: (including device programming or simulation files), and
any Info: associated documentation or information are
expressly subject Info: to the terms and conditions of the
Intel Program License Info: Subscription Agreement, the Intel
Quartus Prime License Agreement, Info: the Intel MegaCore
Function License Agreement, or other Info: applicable license
agreement, including, without limitation, Info: that your use
is for the sole purpose of programming logic Info: devices
manufactured by Intel and sold by Intel or its Info:
authorized distributors. Please refer to the applicable Info:
agreement for further details. Info: Processing started: Sun
Jan 21 06:39:24 2018 Info: Command: quartus_pgm -m JTAG -o
pvbi;dcp_1_0.jic Info (213045): Using programming cable
"A10SA4 [1-1.4.3.1]" Info (213011): Using programming file
dcp_1_0.jic with checksum 0x237FE3AA for device 10AX115N3@1
Info (209060): Started Programmer operation at Sun Jan 21
06:39:37 2018 Info (209016): Configuring device index 1 Info
(209017): Device 1 contains JTAG ID code 0x02E660DD Info
(209007): Configuration succeeded -- 1 device(s) configured
Info (209018): Device 1 silicon ID is 0x21 Info (209044):
Erasing ASP configuration device(s) Info (209019): Blankchecking
device(s) Info (209023): Programming device(s) Info
(209021): Performing CRC verification on device(s) Info
(209011): Successfully performed operation(s) Info (209061):
Ended Programmer operation at Sun Jan 21 06:45:38 2018 Info:
Quartus Prime Programmer was successful. 0 errors, 0 warnings
Info: Peak virtual memory: 1871 megabytes Info: Processing
ended: Sun Jan 21 06:45:38 2018 Info: Elapsed time: 00:06:14
Info: Total CPU time (on all processors): 00:00:45
```

6. To unMask uncorrectable errors and Mask correctable errors, run the following commands:
   a. UnMask uncorrectable errors and Mask correctable errors of FPGA:

```
$ sudo setpci -s d8:0.0 ECAP_AER+0x08.L=0x00000000
$ sudo setpci -s d8:0.0 ECAP_AER+0x14.L=0x00000000
```

   b. UnMask uncorrectable errors and Mask correctable errors of RP:

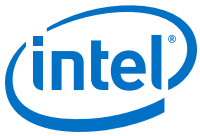
```
$ sudo setpci -s d7:0.0 ECAP_AER+0x08.L=0x00000000
$ sudo setpci -s d7:0.0 ECAP_AER+0x14.L=0x00000000
```

7. Power cycle the host machine.

Note: Now that Intel PAC with Intel Arria 10 GX FPGA has been updated, you can make future flash updates over PCI without needing a Micro USB cable using the `fpgaflash` tool. For more information, refer to the "Update Flash with FPGA Interface Manager (FIM) Image using `fpgaflash` Tool " section.

Related Information

- Identify the FPGA Interface Manager (FIM) Version Loaded on page 18



- Updating the Board Management Controller (BMC) Configuration and Firmware on page 33
- Update Flash with FPGA Interface Manager (FIM) Image using Intel Quartus Prime Programmer on page 35
- Installing the OPAE Software Package on page 15
- Update Flash with FPGA Interface Manager (FIM) Image using fpgaflash Tool on page 20

D. Documentation Available for the Intel Acceleration Stack for Intel Xeon CPU with FPGAs 1.0 Production Release

The following documents can be found on the Intel FPGA web page and can be accessed by clicking on the link.

Table 8. Documentation Available for the Intel Acceleration Stack for Intel Xeon CPU with FPGAs 1.0 Production Release

Document	Link to Access Document
Intel Acceleration Stack Quick Start Guide for Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA	https://www.altera.com/documentation/dnv1485190478614.html
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	https://www.altera.com/documentation/uux1498689964626.html
Acceleration Stack for Intel Xeon CPU with FPGAs Core Cache Interface (CCI-P) Reference Manual	https://www.altera.com/documentation/buf1506187769663.html
Accelerator Functional Unit AFU Developers Guide	https://www.altera.com/documentation/nms1512265268234.html
DMA Accelerator Functional Unit AFU User Guide	https://www.altera.com/documentation/tmv1511227122034.html
Native Loopback Accelerator Functional Unit (AFU) User Guide	https://www.altera.com/documentation/rbt1498764220522.html
OpenCL on Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA Quick Start User Guide	https://www.altera.com/documentation/qac1504285387466.html
Intel Acceleration Stack for Intel Xeon CPU with FPGAs 1.0 Errata	https://www.altera.com/documentation/ffv1519794536166.html
Intel Acceleration Stack for Intel Xeon CPU with FPGAs Release Notes	https://www.altera.com/documentation/mdk1519705276936.html
Intel Programmable Acceleration Card (PAC) with Intel Arria 10 GX FPGA Datasheet	https://www.altera.com/documentation/hhf1507759304946.html

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

Table 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 Acceleration Stack Version	Changes
2018.05.29	1.0 Production (supported with Intel Quartus Prime Pro Edition 17.0.0)	• Added implementation to install "rsync" on the host that is compiling the AFU. • Converted "Updating the Board Management Controller (BMC) Configuration and Firmware" into an Appendix. • Moved "Identifying the FIM Version Loaded" section to before "Updating Flash with the FIM Image using the `fpgaflash` Tool" section. • Updated the "Identifying the FIM Version Loaded" section: — Added steps to follow to update the Flash after identifying the FIM version. — Added the 1.0 Beta and 1.0 Alpha FIM and OPAE versions.
2018.04.11	1.0 Production (supported with Intel Quartus Prime Pro Edition 17.0.0)	• Modified the "Getting Started" section by removing the "Installing the Intel Quartus Prime Pro Edition Software" subsection and replacing it with "Installing the Intel Acceleration Stack section and subsections. • Modified instructions in the "Extracting the Package" section to account for new installation method. • Added a procedure to update the FIM. • Modified the "Updating the Board Management Controller (BMC) Configuration and Firmware" section to replace "Bittware II Toolkit" reference with "Bittware II Toolkit-Lite" and to add a step for verifying firmware.

continued...

Document Version	Intel Acceleration Stack Version	Changes
		• Updated steps, in "Appendix B: Update Flash with FPGA Interface Manager (FIM) Image using Intel Quartus Prime Programmer", to mask PCIe error to avoid system crash while programming the FIM image. • Updated Intel PAC with Intel Arria 10 GX FPGA features in Table 3: Hardware Platforms Supported by the Acceleration Stack Feature Comparison in the "Differences Between the Hardware Platforms Supported by the Acceleration Stack" section. • Replaced Quick Path Interconnect (QPI) with Ultra Path Interconnect (UPI) for the Intel Xeon Scalable Platform with Integrated FPGA term in the "Acceleration Glossary".
2018.01.19	1.0 Beta (supported with Intel Quartus Prime Pro Edition 17.0)	This version of the document was created to address deficiencies in the Beta version. The following updates were made in this version: • In the "Building the OPAE Software" section: — Added the "make install" step — Added the missing dependency Doxygen. • Corrected the hugepages setting to run `fpgabist` example in the "Running FPGA Diagnostics" section. • In the "Update the Board Management Controller (BMC) Configuration and Firmware" section: — Added steps for when an error is received. — Fixed a typo in the command in step 7. • In the "Update Flash with Beta FPGA Interface Manager (FIM) Image using Intel Quartus Prime Programmer" section: — Corrected typo in step 3. — Replaced Intel FPGA Download Cable with "Micro USB cable. — Added steps for when the OPAE is built from source instead of rpm install in the "Running the Hello FPGA Example".
2017.12.22	1.0 Beta (supported with Intel Quartus Prime Pro Edition 17.0)	• Updated populated memory on accelerator card in "Introduction to the Intel Acceleration Stack for Intel Xeon CPU with FPGAs" section. • Removed Alpha information from the "Update Flash with Beta FPGA Interface Manager (FIM) Image using Intel Quartus Prime Pro Edition Programmer" section. • Removed the "Installing the BitWare Linux Toolkit" section. • "Flashing the Card" and "Running FPGA Diagnostics" are no longer a sub-sections of the "Getting Started" section. They have been promoted to main sections listed under the "Installing the OPAE Software" section.

continued...

Document Version	Intel Acceleration Stack Version	Changes
		• Modified the FIM ID and Sample Output code in the "Building the OPAE Software" section. • Updated the $ sudo gcc command in the "Running the Hello FPGA Example" section. • Updated the $ sudo gcc command in the "Running the Hello FPGA Example on Virtual Machine" section. • Added "Additional Flag Settings to AFU Makefiles" appendix. • Updated "Documentation Available for the 1.0 Beta Release" section.
2017.11.10	1.0 Alpha (supported with Intel Quartus Prime Pro Edition 17.0)	Preliminary version