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: **1.1 Production**

UG-20166 | 2018.08.27
Latest document on the web: PDF | HTML

Contents

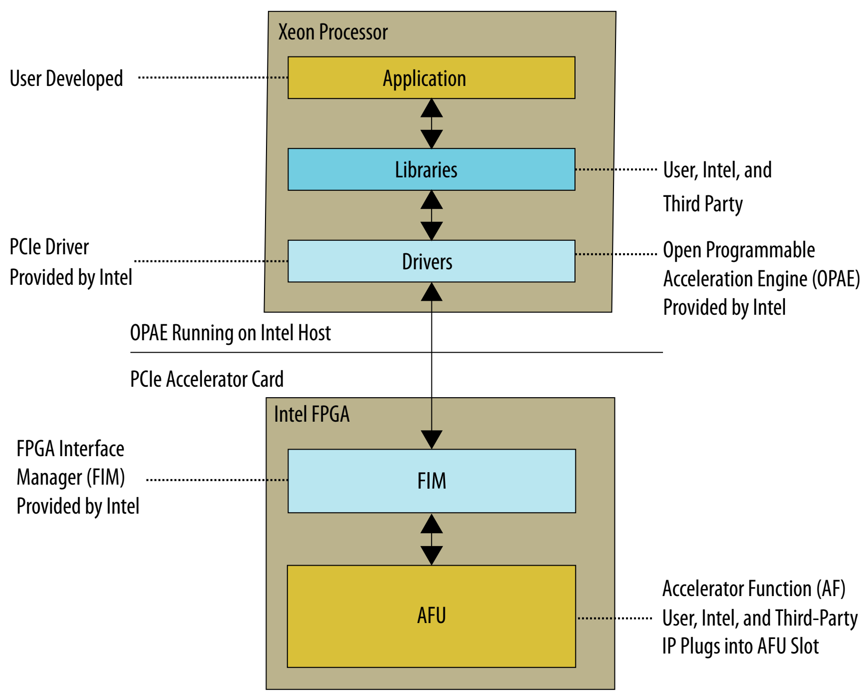
1. Introduction to the Intel® Acceleration Stack for Intel® Xeon® CPU with FPGAs

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

The Intel PAC with Intel Arria 10 GX FPGA 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.

This guide targets 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 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



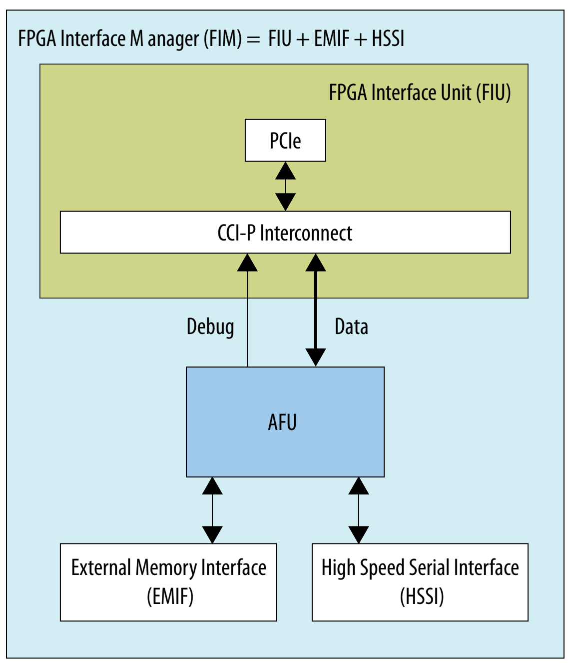
To take advantage of the flexibility of the FPGA, you can reconfigure a special, partial reconfiguration (PR) region of the Intel Arria 10 GX FPGA at run time. You can design multiple Accelerator Functional Units (AFUs) to swap in and out of this PR region.The Open Programmable Acceleration Engine (OPAE) 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 AFU images 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-Maped (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 1.1 Production Release on page 37
- 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, 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. 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, Assessability and Serviceability (RAS)
- Performance counters
- Temperature monitoring using a USB cable or 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:

- Dell* R640
- Dell R740
- Red Hat* Enterprise Linux* (RHEL) 7.4
- Cent OS 7.4

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

Refer to the Intel FPGA Acceleration Hub Platforms Intel web page for best-known configurations.

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`	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 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.

Related Information

- Identifying and Updating the FIM on page 17
- Updating Flash using the fpgaflash Tool on page 18

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.

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

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	Accelerator function development using Intel Quartus Prime Pro Edition and Acceleration Stack
OPAE Software Development Kit (SDK) version 1.1 Production	OPAE SDK version 1.0.2 Production (`a10_gx_pac_ias_1_1_pv_rte.tar.gz`)	OPAE SDK version 1.0.2 Production `a10_gx_pac_ias_1_1_pv_dev_installer.tar.gz`

continued...

	Acceleration Stack for Runtime	Acceleration Stack for Development
Intel Quartus Prime Software	Intel Quartus Prime Programmer Only	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	619 MB	16.9 GB

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. Run setup.sh.

```
./setup.sh
```

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

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>/intelrtestack` to install Intel Quartus Prime Programmer and OpenCL RTE.
8. Run the initialization script to set the required environment variables.

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

9. Download `a10_gx_pac_ias_1_1_pv_eth.patch` from the Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA webpage.
10. Copy the `a10_gx_pac_ias_1_1_pv_eth.patch` to the $OPAE_PLATFORM_ROOT/hw directory:

```
cp a10_gx_pac_ias_1_1_pv_eth.patch $OPAE_PLATFORM_ROOT/hw/
```

11. Change to the $OPAE_PLATFORM_ROOT/hw directory:

```
cd $OPAE_PLATFORM_ROOT/hw/
```

12. Install the patch:

```
patch -s -p0 < a10_gx_pac_ias_1_1_pv_eth.patch
```

Related Information

Installing the OPAE Software Package on page 13

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. Run setup.sh.

```
./setup.sh
```

4. 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*.

5. Accept the license.
6. 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.
7. Run the initialization script to set the required environment variables, `Quartus_Home` and `OPAE_PLATFORM_ROOT`.

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

8. Download `a10_gx_pac_ias_1_1_pv_eth.patch` from the Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA webpage.
9. Copy the `a10_gx_pac_ias_1_1_pv_eth.patch` to the `$OPAE_PLATFORM_ROOT/hw` directory:

```
cp a10_gx_pac_ias_1_1_pv_eth.patch $OPAE_PLATFORM_ROOT/hw/
```

10. Change to the `$OPAE_PLATFORM_ROOT/hw` directory:

```
cd $OPAE_PLATFORM_ROOT/hw/
```

11. Install the patch:

```
patch -s -p0 < a10_gx_pac_ias_1_1_pv_eth.patch
```

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

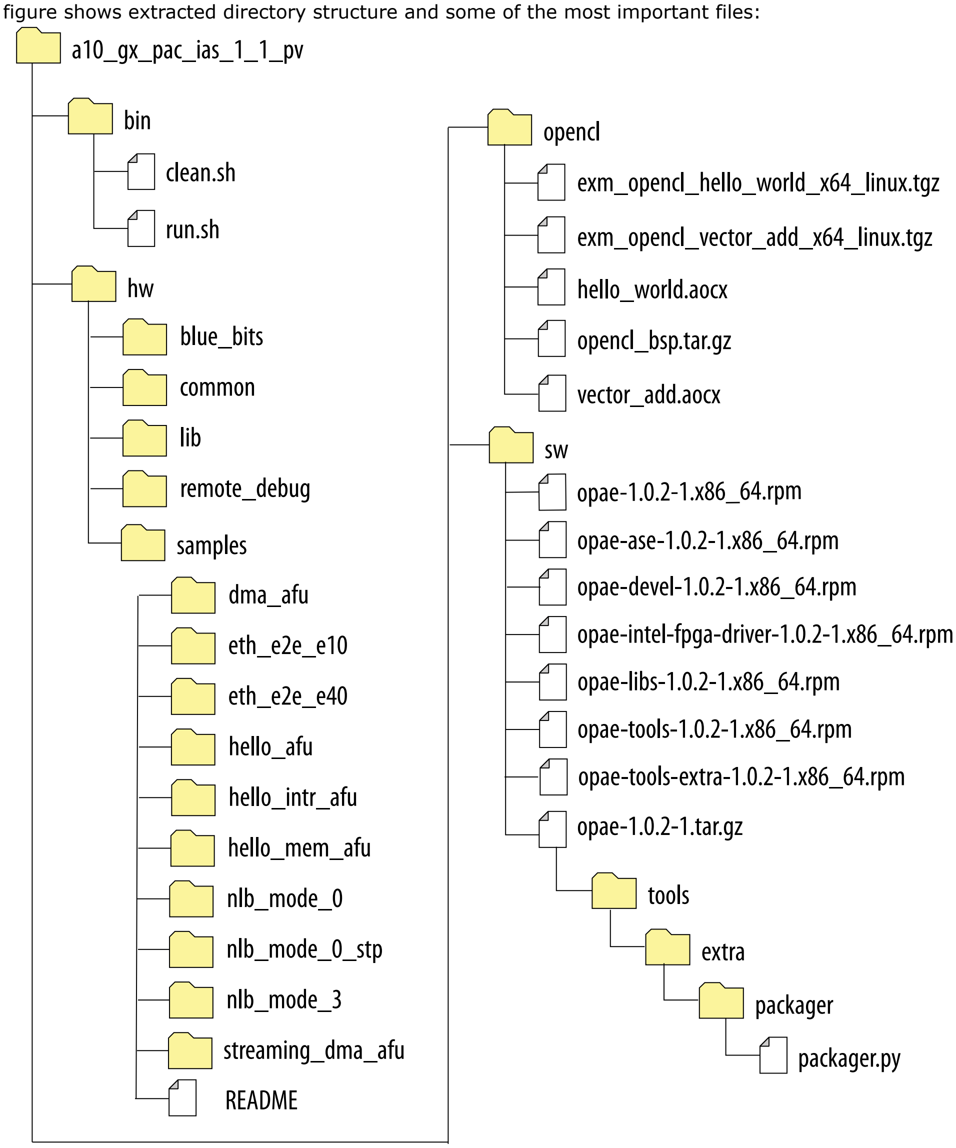
The `init_env.sh` defines `OPAE_PLATFORM_ROOT` environment variable. `OPAE_PLATFORM_ROOT` points to the extracted `a10_gx_pac_ias*` release directory. Depending on your previous choice, `a10_gx_pac_ias*` is available in one of the following directories:

- 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: 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 manual steps to install OPAE as detailed in the section *Installing the OPAE Software Package.*

Figure 3. Intel PAC with Intel Arria 10 GX FPGA 1.1 Production Directory Structure

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



Related Information

Installing the OPAE Software Package on page 13

Send Feedback

3. Installing the OPAE Software Package

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

Note: Skip this section if you have already installed OPAE by answering **Yes** when prompted, *Do you wish to install the OPAE?* while Installing the Acceleration Stack package on the host machine.

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)
- The installation provides two options for the OPAE framework:
  - The RPM option installs the OPAE framework using pre-built binaries. This option installs the following:
    - Binaries in `/usr/bin`
    - Libraries in `/usr/lib64`
    - Headers in `/usr/include`
  - The build-from-source option builds and installs the OPAE framework from the source. This option installs the following:
    - Binaries in `<custom_dir>/bin`
    - Libraries in `<custom_dir>/lib64`
    - Headers in `<custom_dir>/include`

3.1. Installing the OPAE Framework from Prebuilt Binaries (RPM)

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 hwloc-devel libpng12 rsync
```

Note: This command only installs 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. Update dynamic linker run-time bindings:

```
sudo ldconfig
```

4. Check the Linux kernel installation:

```
lsmod | grep fpga
```

Sample output:

```
intel_fpga_pac_hssi    18107  0
intel_fpga_afu         31735  0
intel_fpga_fme         52380  0
fpga_mgr_mod           14693  1 intel_fpga_fme
intel_fpga_pci         26519  2
intel_fpga_afu,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*
/user/lib64	opae-libs64* opae-ase*

5. Verify your library installation:

```
ls /usr/lib64
```

Sample output:

```
/usr/lib64/libopae-c++-utils.so.1.0.2
/usr/lib64/libopae-c.so.1.0.2
/usr/lib64/libopae-c-ase.so.1.0.2
/usr/lib64/libopae-c++.so.1.0.2
/usr/lib64/libopae-c++-nlb.so.1.0.2
/usr/lib64/libopae-c.so -> libopae-c.so.1
/usr/lib64/libopae-c.so.1 -> libopae-c.so.1.0.2
/usr/lib64/libopae-c-ase.so.1 -> libopae-c-ase.so.1.0.2
/usr/lib64/libopae-c-ase.so -> libopae-c-ase.so.1
/usr/lib64/libopae-c++-nlb.so.1 -> libopae-c++-nlb.so.1.0.2
```

```
/usr/lib64/libopae-c++-nlb.so -> libopae-c++-nlb.so.1
/usr/lib64/libopae-c++-utils.so -> libopae-c++-utils.so.1
/usr/lib64/libopae-c++-utils.so.1 -> libopae-c++-utils.so.1.0.2
/usr/lib64/libopae-c++.so.1 -> libopae-c++.so.1.0.2
/usr/lib64/libopae-c++.so -> libopae-c++.so.1
```

3.2. (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:
   ```
   sudo yum remove opae*-intel-fpga*.x86
   ```
   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. 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=ON -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. 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>`.

e. 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
```

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

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

4. Identifying and Updating the FIM

Each Acceleration Stack Release requires a different version of the FIM. Run the `fpgainfo` tool to identify the FIM currently loaded.

```
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
Device Id:           0x09C4
Fim Version:         1.1.3
Ports Num:           1
Socket Id:           0
Bitstream Id:        0x113000200000177
Bitstream Metadata:  0x18043013
Pr Interface Id:     9926ab6d-6c92-5a68-aabc-a7d84c545738
Object Id:           251658240
```

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

Acceleration Stack Version	FIM Version (PR Interface ID)	OPAE Version
1.1 Production	9926ab6d-6c92-5a68-aabc-a7d84c545738	1.0.2
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
1.0 Beta	3d949b98-7b30-5a9ab296-4530a780a3f9	0.13.0
1.0 Alpha	d4a76277-07da-528db623-8b9301feaffe	0.11.0

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
1.0 Beta or later	1. Follow the instructions in *Updating the BMC Configuration and Firmware*. 2. Follow the instructions in *Updating Flash using the `fpgaflash` tool*.
Unknown or 1.0 Alpha and earlier	1. Follow the instructions in *Updating the BMC Configuration and Firmware*. 2. Then follow the instructions in *Updating Flash using the Intel Quartus Prime Programmer*.

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

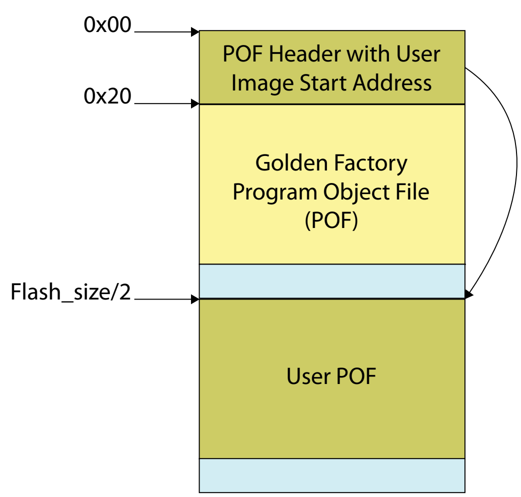
- Updating Flash Using the Intel Quartus Prime Programmer on page 32
- Updating the Board Management Controller (BMC) Configuration and Firmware on page 34
- Updating Flash using the fpgaflash Tool on page 18
- Open Programmable Acceleration Engine (OPAE) Tools Guide (fpgaflash)

4.1. Updating Flash using the fpgaflash Tool

Skip this section if you have the latest FIM which corresponds to the release.

The flash has two partitions. One partition stores the factory or golden image. The other partition stores the user image. If the user image is corrupt, the 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 flash memory:

1. Type the command:

```
sudo fpgaflash user $OPAE_PLATFORM_ROOT/hw/blue_bits/dcp_1_1.rpd
```


Send Feedback

Expected Output:

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

The flash erase, write, and verify process takes several minutes to complete. If you have multiple Intel PAC cards installed, you can specify the bus, device, and function (BDF) for the card to update using the following command. To find the BDF for your card, type the following command:

```
lspci | grep 09c[45]
```

Sample output:

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

In this example the BDF = 04:00.0:

```
sudo fpgaflash user $OPAE_PLATFORM_ROOT/hw/blue_bits/*.rpd 04:00.0
```

Note: If a catastrophic system event such as a power loss occurs during the flash programming process, first retry the `fpgaflash` command after the system recovers. If enumeration fails to find the Intel PAC, warm restart the system and retry the `fpgaflash` command. If the issue persists, follow the procedure in the *Update Flash using Intel Quartus Prime Programmer* to restore the FPGA configuration flash.

2. After you update the flash, power cycle host machine. A simple restart 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, follow these instructions:

a. Restart to verify that PCIe enumeration has assigned a PCIe bus.
b. If restarting does not resolve this problem, follow the steps in the *Updating the Flash using Intel Quartus Prime Programmer* to update the flash using Programmer.

Related Information

Updating Flash Using the Intel Quartus Prime Programmer on page 32

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.
   ```

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
   ```

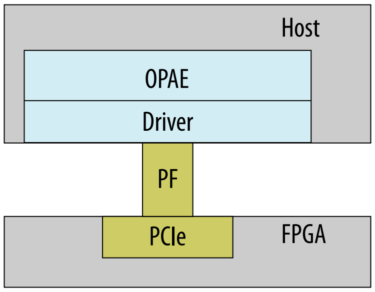
Related Information

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 5. 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 1.1 pv 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
```

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

   a. Refer to Table 5 on page 17 to determine the required *<FIM version>*.

   b. 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": "9926ab6d-6c92-5a68-aabc-a7d84c545738",
        "afu-top-interface": {
            "class": "ccip_std_afu"
        },
        "magic-no": 488605312,
        "power": 0,
        "accelerator-clusters": [
            {
                "total-contexts": 1,
                "name": "hello_afu",
                "accelerator-type-uuid": "850adcc2-6ceb-4b22-9722-
d43375b61c66"
            }
        ]
    }
}
```

   The `interface-uuid` should match the FIM version (PR interface ID) you found in Table 5 on page 17.

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:

```
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
```

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

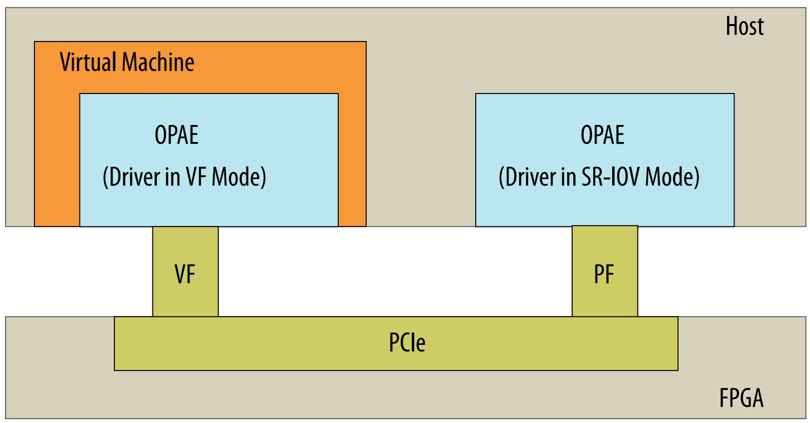
- Identifying and Updating the FIM on page 17
- Installing the OPAE Framework from Prebuilt Binaries (RPM) on page 13

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 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 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 7
- Installing the OPAE Software Package on page 13

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.

1. Run the following commands to export the required paths:

```
export port_path=$(find /sys/class/fpga/intel-fpga-dev.* \
 -maxdepth 1 -follow -iname intel-fpga-port.0)
export link_path=$(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 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 13
- Running the Hello FPGA Example on page 22
- Documentation Available for the Intel Acceleration Stack for Intel Xeon CPU with FPGAs 1.1 Production Release on page 37
- Native Loopback Accelerator Functional Unit (AFU) User Guide

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-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
```

8. FPGA Device Access Permission

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/.

To grant access to accelerators to regular (non-root) users, provide regular user read and write access to /dev/intel/fpga-port.* 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
```

9. 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 way to do this depends on your Linux distribution.

Use the `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
```

10. 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`.

11. 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
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 Flash Using the Intel Quartus Prime Programmer

Note: Use these instructions to update the FIM image in flash memory if the current flash is empty or stores a pre-alpha FIM.

This update requires the following hardware and drivers:

- A micro USB cable.
- The Intel FPGA Download Cable driver. Refer to the the Intel FPGA Download Cable (formerly USB-Blaster) Driver for Linux for more information about installing this driver.
- The BittWorks II Toolkit-Lite software. Refer to the *BMC firmware* information on the Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA web page for information on getting the BittWorks II Tooklit-Lite software.

1. Change to the `blue_bits` directory and export the path to your Intel Quartus Prime Pro Edition:

```
cd $OPAE_PLATFORM_ROOT/hw/blue_bits/
export QUARTUS_HOME=<path to quartus installation>
```

2. Find the Root Port and Endpoint of the Intel PAC with Intel Arria 10 GX FPGA card:

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

Example output 1, showing that the Root Port is `d7:0.0` and the Endpoint is `d8:0.0` :

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

Example output 2, showing that the Root Port is `0:1.0` and the Endpoint is `3:0.0`:

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

Example output 3, showing that the Root Port is `85:2.0` and the Endpoint is `86:0.0` and.:

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

Note: No output indicates a PCIe device enumeration failure and that flash is not programmed. This procedure allows for skipping steps 2 on page 32 and 4 on page 33 and directly programming the FIM image in 3 on page 33.

#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
```

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
```

3. Run the following Intel Quartus Prime Programmer command:

```
sudo $QUARTUS_HOME/bin/quartus_pgm -m JTAG -o 'pvbi;dcp_1_1.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_1.jic Info (213045): Using programming cable
"A10SA4 [1-1.4.3.1]" Info (213011): Using programming file
dcp_1_1.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
```

4. To unmask uncorrectable errors and mask correctable errors, run the following commands:

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
```

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
```

5. Power cycle the host machine.

Note: After this Intel PAC with Intel Arria 10 GX FPGA update, 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.

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

This update requires the following hardware and drivers:

- A micro USB cable.
- The Intel FPGA Download Cable driver. Refer to the the Intel FPGA Download Cable (formerly USB-Blaster) Driver for Linux for more information about installing this driver.
- Obtain the BittWorks II Toolkit-Lite Software (version: Release 2017.4 Enterprise Linux 7(64-bit)) and BMC firmware (26822) by registering at BittWorks II Toolkit Lite for Intel PAC and selecting Intel PAC.

1. To determine if the Bittware tools are already installed, run the following command :

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

2. If the Bittworks II tools are installed, run the following command to determine the version:

```
bwconfig --version
```

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

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

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

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

5. To run Bittware tools without having to provide a complete path, run the following command:

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

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

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

Sample Output:

```
BMC flags set to 81
```

Note: If you receive this error message:

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

a. bwconfig --remove=0

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

b. bwconfig --scan=usb

Sample output when the scan detects Intel PAC with Intel Arria 10 GX FPGA: 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"
```

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

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

8. Verify firmware by typing the following command:

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

Expected output:

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

If the firmware version is < 26815, upgrade to 26815 or greater by following the steps here:

9. To update to the latest firmware version, run:

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

Note: You might see the following error message: ERROR: Upgrade failed. You may safely this ignore the message.

10. Power cycle host machine. A simple restart is insufficient.

11. Verify firmware is 0x68c6 (26822).

Sample output:

```
bwmonitor --dev=0 --version
BwMonitor (cli) version 2017.4.233.31840
BMCLIB version : 2017.4.233.31840
MCU Firmware version : 0x68c6
```

Related Information

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

C. Documentation Available for the Intel Acceleration Stack for Intel Xeon CPU with FPGAs 1.1 Production Release

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

Table 7. Documentation Available for the Intel Acceleration Stack for Intel Xeon CPU with FPGAs 1.1 Production 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 Core Cache Interface (CCI-P) Reference Manual	Intel Acceleration Stack for Intel Xeon CPU with FPGAs Core Cache Interface (CCI-P) Reference Manual
Intel Acceleration Stack for Intel Xeon CPU with FPGAs 1.1 Release Notes	Intel Acceleration Stack for Intel Xeon CPU with FPGAs 1.1 Release Notes
Intel Acceleration Stack for Intel Xeon CPU with FPGAs 1.1 Errata	Intel Acceleration Stack for Intel Xeon CPU with FPGAs 1.1 Errata