Intel Acceleration Stack Quick Start Guide

Intel FPGA Programmable Acceleration Card D5005

Updated for Intel® Acceleration Stack for Intel® Xeon® CPU with FPGAs: **2.0.1**

Subscribe
Send Feedback

UG-20202 | 2021.07.09
Latest document on the web: PDF | HTML

Contents

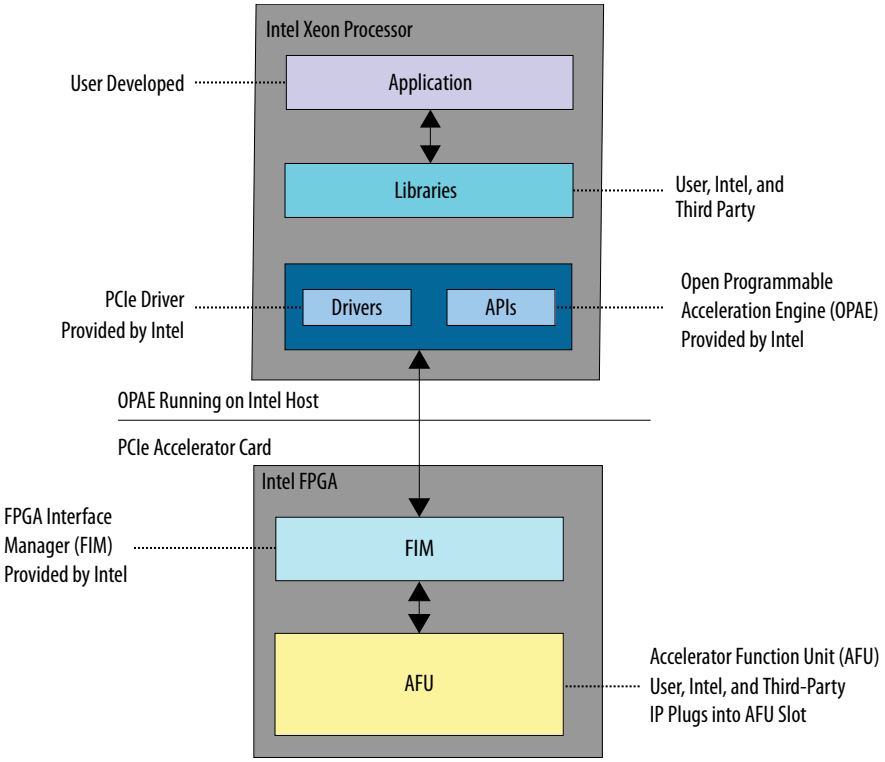
1. About this Document

This document provides a brief introduction to the Intel® FPGA Programmable Acceleration Card (Intel FPGA PAC) D5005. This guide provides the instructions for the following procedures:

- Installing the Open Programmable Acceleration Engine (OPAE) on the host Intel Xeon® Processor to manage and access the Intel FPGA PAC.
- Configuring and flashing the FPGA image and Board Management Controller images.
- Running the example `hello_afu` in a virtualized and non-virtualized environment.
- Handling Graceful Thermal Shutdown

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

Figure 1. Intel FPGA PAC Platform Hardware and Software Overview



2. System Requirements and Release Installation

Note: Ensure that your system meets the following requirements before you proceed to install the Intel FPGA PAC D5005. The Intel FPGA PAC D5005 may not function properly if your system does not meet these requirements.

Table 1. System Requirements for the Intel FPGA PAC D5005

Hardware and Software Components	Description
Main board	PCI Express* 3.0 compliant motherboard with at least one dual-width x16 PCIe* slot available for card installation
Server	Supported Server List
Board power supply	Auxiliary Power (12V)
Operating system	Red Hat* Enterprise Linux* (RHEL) version 7.6 Kernel 3.10.0-957
Internet connection	Required to install the OPAE driver and receive updates

2.1. Unpacking the Intel FPGA PAC

Box Contents

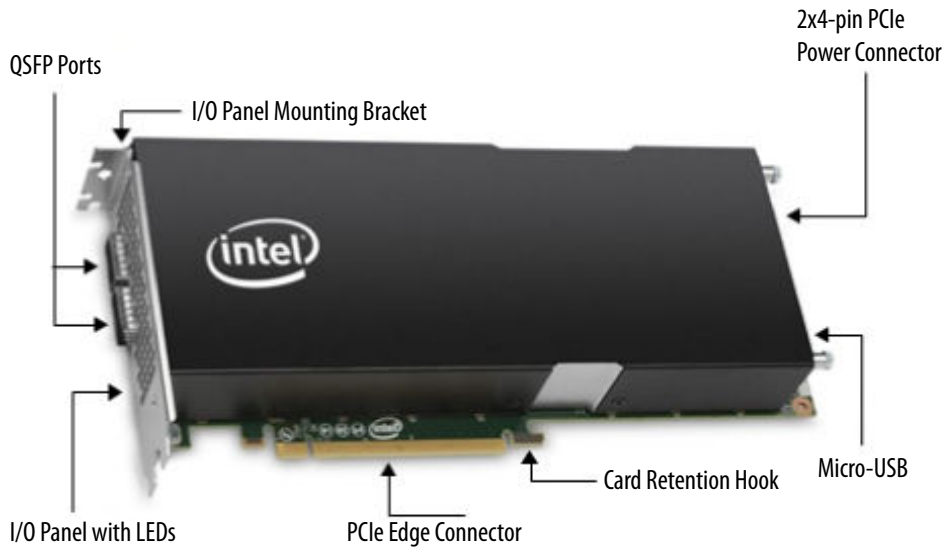
- Intel FPGA PAC D5005
- Extension mounting bracket
- Mounting screws

Note: The extension mounting bracket and mounting screws may already be installed.

Intel FPGA PAC D5005 External Hardware Features

- PCIe 2x4-pin auxiliary power connector
- PCIe card edge connector
- Two quad small form factor pluggable (QSFP) interfaces
- Micro-USB
- Card extender mounting bracket
- Card extender mounting screws

Figure 2. Intel FPGA PAC D5005



Related Information

Intel FPGA PAC D5005 Specifications

2.2. Precautions for Hardware Installation

You must open the server chassis to install the Intel FPGA PAC. Follow all safety precautions and the electrostatic discharge (ESD) guidelines provided to avoid damaging the server or the Intel FPGA PAC.

Warning: To avoid electric shock, power down your server and unplug it from the power outlet before opening the server chassis.

ESD Guidelines

Electronics components on the Intel FPGA PAC and server are sensitive to ESD. To avoid damaging the Intel FPGA PAC and server, follow these ESD prevention guidelines:

- Wear a grounded ESD strap during the Intel FPGA PAC installation.
- Leave the Intel FPGA PAC in its ESD-safe packaging until you are ready to install the card.
- During installation, handle the Intel FPGA PAC only by the edge of the board.
- Never touch any exposed circuitry, edge connectors, or printed circuits on the Intel FPGA PAC or server.
- Do not put the Intel FPGA PAC on any metal surface during installation.
- If you must put the Intel FPGA PAC down, put the card in the ESD-safe packaging.

2.3. Hardware Installation

Follow these instructions to install the Intel FPGA PAC in your server.

1. Open your server chassis.
2. Identify an available PCIe slot with enough clearance to house the Intel FPGA PAC. For slots that can receive a full-length PCIe card, Intel recommends that you use the provided extension mounting bracket to secure the card edge for additional support.
3. Remove any I/O panel covers for the slot you are using.
4. Install the Intel FPGA PAC in the PCIe slot by inserting the PCIe x16 edge connector and ensuring the card retention hook is properly engaged.
5. Use the two screws provided to secure the I/O panel bracket to the server chassis. If you are using the extension bracket, secure the extension bracket to the server.
6. Connect the 2 x 4-pin 12 V auxiliary power cable from the server to the matching 2 x 4-pin power connector on the Intel FPGA PAC.
7. Reinstall the chassis cover.

2.4. Setting Up the Host Machine Software

2.4.1. Installing Red Hat Enterprise Linux version 7.6

1. Enable the following options in the BIOS:
   - SR-IOV
   - Virtualization Technology
2. Select the following options and packages during initial installation:
   — Base Environment: Server with GUI
   — Add ons:
      — Development Tools
      — Compatibility Libraries
      — Virtualization Tools
      — Virtualization Hypervisor
3. Purchase a RHEL subscription from the Red Hat* store. This step is required to fetch and install packages from the RHEL repository.
4. Execute the registration steps on your server to register with RHEL.
5. To attach the license:

```
subscription-manager attach --auto
```

6. Additionally, enable the following repository:

```
subscription-manager repos --enable rhel-7-server-rpms \
rhel-7-server-optional-rpms
```

   At this point the installed kernel is 3.10. You can install the additional packages listed in the step 2 above after the initial installation of RHEL 7.6 by running:

```
$ sudo yum groupinstall <package_name>
```

   The Python3 package is required to install one of the OPAE rpm.

```
$ sudo yum install -y python36 python36-pip
```

2.4.2. Installing the Intel Acceleration Stack

To install the Acceleration Stack, select either the **Acceleration Stack for Runtime** or the **Acceleration Stack for Development**. The following table explains the differences between the two versions of the Acceleration Stack.

	Acceleration Stack for Runtime	Acceleration Stack for Development
Purpose	• Deployment of production-ready solutions. • Software development of runtime host application.	• RTL development of an AFU using the Intel Quartus® Prime Pro Edition and the Acceleration Stack.
OPAE Software Development Kit (SDK)	OPAE SDK version 1.1.4-8	OPAE SDK version 1.1.4-8
Intel Quartus Prime Pro Edition	Not included or required	Included: Intel Quartus Prime Pro Edition 19.2 with related interface licenses (such as SR-IOV).
Download Information	Getting Started webpage	

Note: You must install Python3 using the following command before installing the **Acceleration Stack for Runtime** or the **Acceleration Stack for Development**.

```
sudo yum install python3
```

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

The Python3 package is required to install one of the OPAE rpm.

```
$ sudo yum install -y python36 python36-pip
```

Follow these instructions to extract the release package:

1. Extract the archive file:

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

2. Change to install directory:

   ```
   cd *rte_installer
   ```

3. Run `./setup.sh` and follow the prompts:

   ```
   ./setup.sh
   ```

4. You are prompted with the following questions:
   a. Do you wish to continue to install the software (Acceleration Stack)?(1)

Option	Description
Select Yes (y)	*If you want to continue using an unsupported platform.*
Select No (n)	*To set up the correct environment, refer to* Installing Red Hat Enterprise Linux version 7.6*.*

   b. Do you wish to install the OPAE?

(1) This question is only asked if your OS and kernel version doesn't match the system requirements.

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

c. Do you wish to install the OPAE PACSign package?

Note: This installation requires you have to admin and network access.

Option	Description
Select Yes (y)	*If you plan on signing any bitstreams.*
Select No (n)	*If you do not intent to sign any bitstream. To manually install the OPAE PACSign package later, follow the steps listed in section* Installing the PACSign*.*

5. Accept the license.
6. When you receive an installation directory prompt for your Intel Acceleration Stack release, you can specify an install directory. Otherwise, the installer uses the default directory at `/home/<username>/intelrtestack`.
7. A second prompt displays to install the Intel OpenCL* RTE package. The default location for this installation is: `/opt/opencl_rte`.
   a. If you select OPAE as an installation option, you will be prompted for (y/d/N) permission to install the package `opae-intel-fpga-driver`.
8. Run the initialization script to set the required environment variables and OPAE_PLATFORM_ROOT:

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

Note: Intel recommends that you run this initialization script after every login. You can also add this script to your shell initialization so that it automatically runs at every login.

9. The OPAE_PLATFORM_ROOT environment variable is used as a relative path to executables and files throughout this document. The following command prints OPAE_PLATFORM_ROOT.

```
echo $OPAE_PLATFORM_ROOT
```

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

The Python3 package is required to install one of the OPAE rpm.

```
$ sudo yum install -y python36 python36-pip
```

Use this installation for Accelerator Functional Unit (AFU) development and compilation.

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 questions:
   a. Do you wish to continue to install the software (Acceleration Stack)?[2]

Option	Description
Select Yes (y)	*If you want to continue using an unsupported platform.*
Select No (n)	*To set up the correct environment, refer to* Installing Red Hat Enterprise Linux version 7.6*.*

   b. Do you wish to install the OPAE?

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

   c. Do you wish to install the OPAE PACSign package?

   Note: This installation requires you have to admin and network access.

Option	Description
Select Yes (y)	*If you plan on signing any bitstreams.*
Select No (n)	*If you do not intent to sign any bitstream. To manually install the OPAE PACSign package later, follow the steps listed in section* Installing the PACSign*.*

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 the Intel Acceleration Stack.
7. A second prompt displays to install the Intel OpenCL SDK package and Intel Quartus Prime Pro Edition software. The default location for this installation is `/opt/intelFPGA_pro`.
   a. If you select OPAE as an installation option, you will be prompted for (y/d/N) permission to install the package `opae-intel-fpga-driver`.
   b. You need to enter the sudo password to install the OpenCL SDK package if you are not installing as an administrator.
8. Run the initialization script to set the required environment variables, QUARTUS_HOME, OPAE_PLATFORM_ROOT, and other OpenCL variables.

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

   Note: Intel recommends that you run this initialization script after every login. You can also add this script to your shell initialization so that it automatically runs at every login.

[2] This question is only asked if your OS and kernel version doesn't match the system requirements.

9. The OPAE_PLATFORM_ROOT environment variable is used as a relative path to executables and files throughout the document. The following command prints OPAE_PLATFORM_ROOT.

```
echo $OPAE_PLATFORM_ROOT
```

Related Information

Installing the OPAE Software Package on page 13

2.4.3. Verifying the Installation

Ensure the Intel Acceleration Stack installed correctly.

1. Check the driver installation.

```
lsmod | grep fpga
```

Expected output:

```
ifpga_sec_mgr          13757  1 intel_max10
intel_fpga_pac_iopll    13722  0
intel_fpga_pac_hssi    24389  0
intel_fpga_afu         36165  0
intel_fpga_fme         71639  0
fpga_mgr_mod           14812  1 intel_fpga_fme
intel_fpga_pci         26500  2 intel_fpga_afu,intel_fpga_fme
```

2. Verify the OPAE library installation.

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

Expected output:

```
opae-tools-1.1.4-8.x86_64
opae-devel-1.1.4-8.x86_64
opae-tools-extra-1.1.4-8.x86_64
opae-intel-fpga-driver-2.0.2-1.x86_64
opae.pac_sign-1.0.2-1.x86_64
opae-libs-1.1.4-8.x86_64
opae-one-time-update-d5005-2.0.1-5.noarch
opae-1.1.4-8.x86_64
opae.admin-1.0.2-1.noarch
opae-ase-1.1.4-8.x86_64
opae-super-rsu-d5005-2.0.1-6.noarch
```

Note: The output may differ if you choose to manually install the OPAE.

3. Installing the OPAE Software Package

Note: You can skip this section if you have already installed OPAE by answering *Yes* when the `setup.sh` script prompted you with the question: *Do you wish to install the OPAE?*

The host must have Internet connectivity to retrieve software packages. The installation steps require sudo or root privileges on your host. The following commands show installation as root.

Some packages require you to enable the EPEL repository. You enable the repository by installing the `epel-release-latest` package with the following command:

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

or you may seek assistance from your system administrator.

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 python2-pip tbb-devel
```

```
$ sudo pip install intelhex
```

```
$ sudo yum install -y python36 python36-pip
```

This command only installs missing packages.

3.1. Unpacking and Installing the Intel FPGA Driver and the OPAE Software Development Kit (SDK)

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
```

```
sudo yum remove opae*.noarch
```

   b. Install the Intel FPGA driver, OPAE SDK, and PACSign:

```
cd $OPAE_PLATFORM_ROOT/sw
```

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

2. Check the driver installation:

```
lsmod | grep fpga
```

Expected output:

```
ifpga_sec_mgr          13757  1 intel_max10
intel_fpga_pac_iopll    13722  0
intel_fpga_pac_hssi    24389  0
intel_fpga_afu         36165  0
intel_fpga_fme         71639  0
fpga_mgr_mod           14812  1 intel_fpga_fme
intel_fpga_pci         26500  2 intel_fpga_afu,intel_fpga_fme
```

3. Verify the OPAE library installation:

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

Expected output:

```
opae-tools-1.1.4-8.x86_64
opae-devel-1.1.4-8.x86_64
opae-tools-extra-1.1.4-8.x86_64
opae-intel-fpga-driver-2.0.2-1.x86_64
opae.pac_sign-1.0.2-1.x86_64
opae-libs-1.1.4-8.x86_64
opae-one-time-update-d5005-2.0.1-5.noarch
opae-1.1.4-8.x86_64
opae.admin-1.0.2-1.noarch
opae-ase-1.1.4-8.x86_64
opae-super-rsu-d5005-2.0.1-6.noarch
```

3.2. Installing the PACSign

Note: If you have performed the steps from section *Unpacking and Installing the Intel FPGA Driver and the OPAE Software Development Kit (SDK)*, this step is redundant.

Install the OPAE PACSign package:

```
$ cd $OPAE_PLATFORM_ROOT/sw
```

```
$ sudo yum install opae.pac_sign-1.0.2-1.x86_64.rpm
```

4. Identify the FPGA Interface Manager (FIM) and BMC Firmware Version

Each Acceleration Stack release has a unique FIM version. Use the `fpgainfo` command to identify the FIM (PR interface) and BMC firmware version by running the following command:

```
sudo fpgainfo fme
```

Sample output:

```
Board Management Controller, microcontroller FW version 2.0.12, RTL version
2.0.6
//****** FME ******//
Object Id                        : 0xED00000
PCIe s:b:d:f                     : 0000:AF:00:0
Device Id                        : 0x0B2B
Ports Num                        : 01
Bitstream Id                     : 0x202000200000237
Bitstream Version                : 2.0.2
Pr Interface Id                  : 9346116d-a52d-5ca8-b06a-a9a389ef7c8d
MAC address                      : 64:4c:36:f:44:1f
```

The Intel FPGA PAC D5005 enumerates as PCIe device 8086:0b2b. To identify the PCIe Bus: Device: Function, type:

```
lspci | grep 0b2b
AF:00.0 Processing accelerators: Intel Corporation Device 0b2b (rev 01)
```

This output indicates:

- Bus: 0xAF
- Device 0x00
- Function 0x0

Table 2. Correspondence Between Acceleration Stack, FIM, and OPAE Versions

Note: BMC Firmware Versions beginning with the number 2 that are downloaded to the Intel FPGA PAC indicate the board is is capable of secure bitstream authentication.

Acceleration Stack Version	FIM Version (PR Interface ID)	OPAE Version	BMC Firmware Version	BMC MAX10 Version
2.0.1	9346116d-a52d-5ca8-b06a-a9a389ef7c8d	1.1.4-8	2.0.12	2.0.6
2.0.1 (Beta)	8db6b54c-930e-5976-a03b-09f3c913aa95	1.1.4-8	2.0.10	2.0.4
2.0.1 (Pre-Beta)	3faf8a54-b32c-5f83-8f99-5c7c3ef9ce51	1.1.4-6	2.0.6	2.0.2
2.0	bfac4d85-1ee8-56fe-8c95-865ce1bbaa2d	1.1.4-3	1.0.12	1.0.15
2.0 (Pre-Beta)	a9f2d0f3-b398-57b0-b34f-d226bf364fee	1.1.4-1	1.0.6	1.0.6

If your FIM and BMC firmware version correspond to the most recent version for Acceleration Stack 2.0.1, then proceed to the next chapter, *Running FPGA Diagnostics*. If your FIM or BMC is an older version, follow the steps in the next section: *Updating the FIM and BMC*.

Related Information

Running FPGA Diagnostics on page 18

4.1. Updating the FIM and BMC

The following table provides instructions to update the FIM based on the release currently running on the Intel FPGA PAC D5005.

Table 3. FIM Update Method

Current Acceleration Stack Version	FIM Version	Update Method
2.0.1 beta	8db6b54c-930e-5976-a03b-09f3c913aa95	1. Load the temporary MAX10 image to allow successful flash update: `$ sudo fpgasupdate /usr/share/opae/d5005/one-time-update\ /darby_max10*.bin` 2. Power cycle. 3. Use the `fpgaotsu` tool, provided in the release package, as described in the following section: *Updating the FIM and BMC using the fpgaotsu*.
2.0.1 pre-beta	3faf8a54-b32c-5f83-8f99-5c7c3ef9ce51	
2.0	bfac4d85-1ee8-56fe-8c95-865ce1bbaa2d	Use the fpgaotsu tool provided with the release as described in the following section: *Updating the FIM and BMC using the fpgaotsu*.
2.0 pre-beta	a9f2d0f3-b398-57b0-b34f-d226bf364fee	1. Follow the *Installing the Intel Acceleration Stack Runtime Package on Host Machine* and *Updating the FIM and BMC using the fpgaflash Tool* in the Intel Acceleration Stack Quick Start Guide for Intel FPGA PAC D5005 2.0 version. Following these steps upgrades your FIM from v2.0 pre-beta to v2.0. 2. Follow the instructions in this document to reinstall "Intel Acceleration Stack Runtime package" or "Intel Acceleration Stack development package" followed by the steps in the *Upgrading the FIM and BMC to Secure State* section.

4.1.1. Updating the FIM and BMC using the fpgaotsu

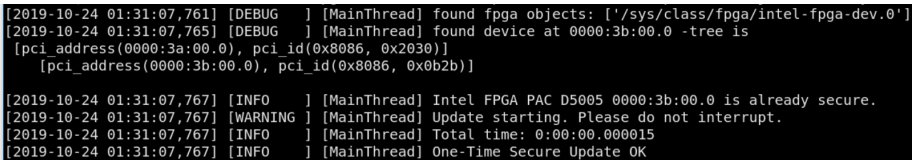
To update your FIM and BMC, complete the following steps. All files required for the update are located at `/usr/share/opae/d5005/one-time-update/`.

1. Run the FPGA one-time secure update command: `fpgaotsu`

```
sudo fpgaotsu /usr/share/opae/d5005/one-time-update/otsu.json
```

Note: This command can take up to 40 minutes to complete. Stop any service accessing the FPGA such as `pacd` before performing the update.

Figure 3. Sample Output



If a power failure or power cycle occurs during the upgrade process, perform these operations based on the state of the card:

BMC Firmware and Intel MAX® 10 Version	Action to Recover
1.x.x (Non-secure state)	Repeat step 1 again.
2.x.x (Partial secure state)	a. `sudo super-rsu /usr/share/opae/d5005/super-rsu/rsu-d5005.json` b. Check exit code of command: `echo $?` • 0=Success • non-zero value=Failed update

2. Power cycle the server for the updates to take effect.

   Use the following command to confirm a successful completion of the update:

   ```
   sudo fpgaotsu /usr/share/opae/d5005/one-time-update/otsu.json \
   --verify --log-level debug
   ```

3. Confirm that the output matches the desired FIM and BMC version found in section Identifying the FIM and BMC Firmware Version.

   Note: If you have programmed the root entry hash into the Intel FPGA PAC, it will be erased during the FIM upgrade to version 2.0.1.

5. Running FPGA Diagnostics

This section presents instructions on how to run FPGA diagnostics using the `fpgadiag` and `fpgabist` utility. You can run the diagnostic test, provided with the Acceleration Stack Runtime/Development package, to ensure the board interface components (PCIe + SDRAM) are operating as expected. The `fpgabist` tool accepts `dma_afu` AFU and `fpgadiag` tool accepts `nlb_mode_3` AFU.

Note: If a flash is programmed with a root entry hash, you must ensure that the AFUs are signed with an appropriate root key and code signing key before running the FPGA diagnostics. For more information on signing, refer to the Security User Guide: Intel FPGA Programmable Acceleration Card D5005.

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"
```

   Note: The above configuration is for a single card system. For multiple cards, set the number of 2 MB hugepages to 20*<*number_of_cards*>.

2. Configure and run diagnostics with the `nlb_mode_3` AFU image.

```
sudo fpgasupdate $OPAE_PLATFORM_ROOT/hw/samples/nlb_mode_3/bin/\
nlb_mode_3_unsigned.gbs
```

   The `fpgasupdate` tool accepts PCIe B:D:F if you need to target a specific Intel FPGA PAC in multi-card systems.

   Replace the PCIe B:D:F value with appropriate values in the following commands:

```
$ fpgadiag -B 0x82 -D 0x00 -F 0x00 --mode=read --read-vc=vh0 \
--write-vc=vh0 --multi-cl=4 --begin=1024 --end=1024 --timeout-sec=5 --cont
```

```
Cachelines Read_Count Write_Count Cache_Rd_Hit Cache_Wr_Hit Cache_Rd_Miss
      1024  888206752           0            0            0             0

Cache_Wr_Miss   Eviction 'Clocks(@250 MHz)'   Rd_Bandwidth   Wr_Bandwidth
            0          0       1250013605      11.369 GB/s     0.000 GB/s

VH0_Rd_Count VH0_Wr_Count VH1_Rd_Count VH1_Wr_Count VL0_Rd_Count
   888206756            1            0            0            0

VL0_Wr_Count
          0
```

```
$ fpgadiag -B 0x82 -D 0x00 -F 0x00 --mode=write --read-vc=vh0 \
--write-vc=vh0 --multi-cl=4 --begin=1024 --end=1024 --timeout-sec=5 --cont
```

```
Cachelines Read_Count Write_Count Cache_Rd_Hit Cache_Wr_Hit Cache_Rd_Miss
      1024          0  1000011208            0            0             0

Cache_Wr_Miss   Eviction 'Clocks(@250 MHz)'   Rd_Bandwidth   Wr_Bandwidth
            0          0       1250014578       0.000 GB/s    12.800 GB/s
```

```
VH0_Rd_Count VH0_Wr_Count VH1_Rd_Count VH1_Wr_Count VL0_Rd_Count
           0   1000011209            0            0            0

VL0_Wr_Count
           0
```

```
$ fpgadiag -B 0x82 -D 0x00 -F 0x00 --mode=trput --read-vc=vh0 \
--write-vc=vh0 --multi-cl=4 --begin=1024 --end=1024 --timeout-sec=5 --cont
```

```
Cachelines Read_Count Write_Count Cache_Rd_Hit Cache_Wr_Hit Cache_Rd_Miss
      1024  812096892   836718216            0            0             0

Cache_Wr_Miss   Eviction 'Clocks(@250 MHz)'   Rd_Bandwidth   Wr_Bandwidth
            0          0       1250012621      10.395 GB/s    10.710 GB/s

VH0_Rd_Count VH0_Wr_Count VH1_Rd_Count VH1_Wr_Count VL0_Rd_Count
   812096892    836718217            0            0            0

VL0_Wr_Count
           0
```

For more information, refer to the fpgadiag tool.

3. Configure two 1 GB hugepages for DMA AFU diagnostics.

```
sudo sh -c "echo 2 > /sys/kernel/mm/hugepages/hugepages-\
1048576kB/nr_hugepages"
```

Note: The above configuration is for a single card system. For multiple cards, set the number of 1 GB hugepages to 2*<*number_of_cards*>.

4. Configure and run diagnostics with the DMA AFU image.

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

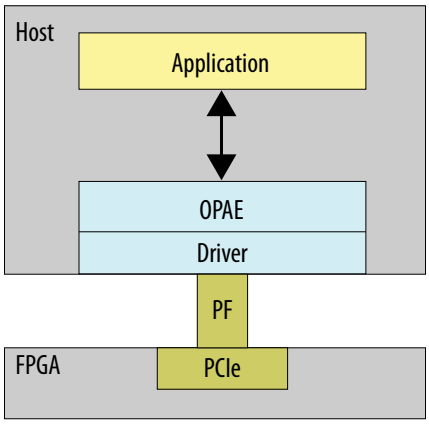
Sample output:

```
Running mode: dma_afu
Attempting Partial Reconfiguration:
Reading bitstream
Looking for slot
Found slot
Programming bitstream
Writing bitstream
Done
Running fpga_dma_test test...

PASS! Bandwidth = 12708 MB/s
Finished Executing DMA Tests

Built-in Self-Test Completed.
```

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 or SR-IOV. The host links to the FPGA with a single PCIe physical function (PF).

Figure 4. OPAE Driver in Non-Virtualized Mode



6.1. Loading an AFU Image into the FPGA

You can utilize the `fpgasupdate` utility to load an AFU image. In Acceleration Stack 2.0.1 and later versions, the Intel FPGA PAC must be programmed with AFU images that have been prepended with mandatory headers. These headers are applied by the PACSign tool. For more information on the PACSign tool, please refer to the Security User Guide: Intel FPGA Programmable Acceleration Card D5005.

The samples included with Acceleration Stack have been processed by PACSign and the AFU binary files are located at:

```
$OPAE_PLATFORM_ROOT/hw/samples/<AFU Name>/bin/*_unsigned.gbs
```

If the Intel FPGA PAC is programmed with a root entry hash following the steps in the *Security User Guide: Intel FPGA Programmable Acceleration Card D5005*, then the provided AFU bitstreams (for example: `hello_afu_unsigned.gbs`) must be signed using PACSign with the appropriate root and code signing keys before you can successfully program the signed AFU bitstream.

```
sudo fpgasupdate <AFU image>
```

The `fpgasupdate` tool can program an unsigned AFU bitstream provided that there is no root entry hash programmed into the flash.

The `fpgasupdate` tool also accepts PCIe Bus:Device:Function (BDF) as an additional optional argument if multiple cards are connected to the server. Use the help text (`-h`) to see how additional arguments must be passed. For example: `sudo fpgasupdate -h`.

To identify the BDF run the following command:

```
lspci | grep 0b2b
```

Sample output:

```
37:00.0 Processing accelerators: Intel Corporation Device 0b2b (rev 01)
```

In the Sample Output, the PCIe Bus is 0x37, the Device is 0x00, and the Function is 0x0.

6.2. Running the Hello FPGA Example

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

1. Run the following command to load the `hello_fpga` sample host application:

   ```
   sudo fpgasupdate $OPAE_PLATFORM_ROOT/hw/samples/nlb_mode_0/bin/\
   nlb_mode_0_unsigned.gbs
   ```

   Note: The `fpgasupdate` tool accepts PCIE B:D.F (for example: 04:00.0) as an argument

2. Configure the system hugepages 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"
   ```

   Note: The above configuration is for a single card system. For multiple cards, set the number of 2 MB hugepages to 20*<*number_of_cards*>.

3. Change to the source code directory for `hello_fpga` located at `$OPAE_PLATFORM_ROOT/sw/opae*/samples/hello_fpga.c`:

   ```
   cd $OPAE_PLATFORM_ROOT/sw
   ```

   a. Extract the `tar` file:

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

   b. Change to the OPAE directory:

      ```
      cd opae*
      ```

4. 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
   ```

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

```
sudo ./hello_fpga
```

Sample output:

```
Running Test
Running on Bus 0x3b
Done Running Test
```

The Acceleration Stack 2.0.1 Release includes the following working AFU images in the `$OPAE_PLATFORM_ROOT/hw/samples` directory:

- `dma_afu/bin/dma_afu_unsigned.gbs`
- `streaming_dma_afu/bin/streaming_dma_afu_unsigned.gbs`
- `hello_afu/bin/hello_afu_unsigned.gbs`
- `hello_intr_afu/bin/hello_intr_afu_unsigned.gbs`
- `hello_mem_afu/bin/hello_mem_afu_unsigned.gbs`
- `nlb_mode_0/bin/nlb_mode_0_unsigned.gbs`
- `nlb_mode_3/bin/nlb_mode_3_unsigned.gbs`
- `nlb_mode_0_stp/bin/nlb_mode_0_stp_unsigned.gbs`
- `hssi_prbs/bin/hssi_prbs_unsigned.gbs`
- `byte_enable/bin/byte_enable_afu_unsigned.gbs`

Note: For `fpgasupdate` to successfully program the bitstreams, all AFU bitstreams built through the AFU development process must be processed by PACSign with or without signing keys to add appropriate headers. The bitstreams provided within the sample AFUs (`$OPAE_PLATFORM_ROOT/hw/<sample afu>/bin/*_unsigned.gbs`) are processed with PACSign to add an appropriate header but they are not signed with any key.

The code below shows an example AFU host code compilation and execution.

```
sudo fpgasupdate $OPAE_PLATFORM_ROOT/hw/samples/<afu>/bin/*_unsigned.gbs
```

```
cd $OPAE_PLATFORM_ROOT/hw/samples/<afu>/sw
```

```
make
```

```
sudo ./<executable>
```

For sample AFU `nlb_mode_0` and `nlb_mode_3`, the pre-compiled executables are copied to location `/usr/bin`.

To execute AFU sample `nlb_mode_0`, type the following commands:

```
sudo fpgasupdate \
$OPAE_PLATFORM_ROOT/hw/samples/nlb_mode_0/bin/*_unsigned.gbs
```

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

```
sudo nlb0
```

To execute sample `nlb_mode_3`, type the following commands:

```
sudo fpgasupdate \
$OPAE_PLATFORM_ROOT/hw/samples/nlb_mode_3/bin/*_unsigned.gbs
```

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

```
sudo nlb3
```

Refer to the README file available under each <example AFU> directory for additional information.

Related Information

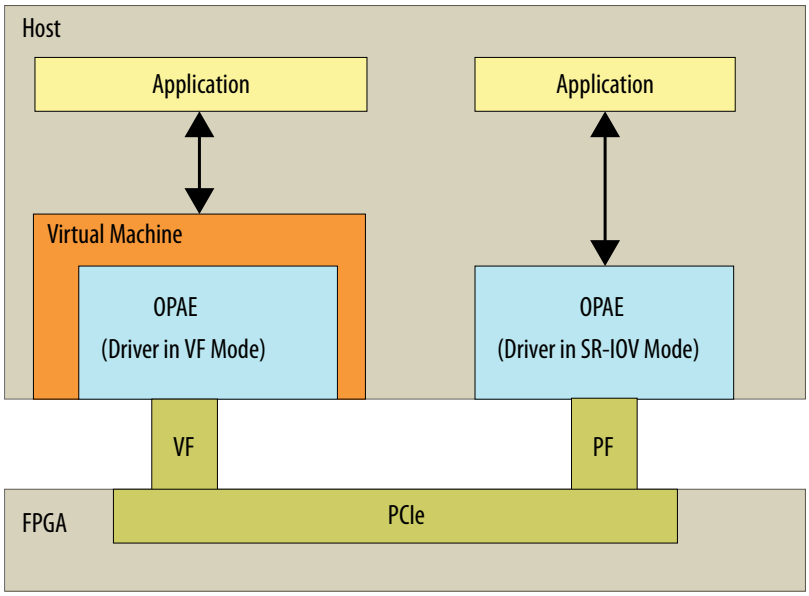
- DMA Accelerator Functional Unit (AFU) User Guide
  For information on how to compile and execute the `dma_afu`.
- Streaming DMA Accelerator Functional Unit (AFU) User Guide
  For information on how to compile and execute the `streaming_dma_afu`.

7. Running the OPAE in a Virtualized Environment

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

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

Figure 5. OPAE Driver in SR-IOV Mode



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 on a board with no root entry hash programmed.

```
sudo fpgasupdate \
$OPAE_PLATFORM_ROOT/hw/samples/nlb_mode_0/bin/nlb_mode_0_unsigned.gbs
```

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 `GRUB_CMDLINE_LINUX` entry by updating the GRUB configuration file.
2. GRUB reads its configuration from either the `/boot/grub2/grub.cfg` file on traditional BIOS-based machines or from the `/boot/efi/EFI/redhat/grub.cfg` file on UEFI machines. Depending on your system, execute one of the instructions below:
   - BIOS based machine:

     ```
     grub2-mkconfig -o /boot/grub2/grub.cfg
     ```

   - UEFI based machine:

     ```
     grub2-mkconfig -o /boot/efi/EFI/redhat/grub.cfg
     ```

3. Restart to apply the new GRUB configuration file.
4. To verify the GRUB update, run the following command:

   ```
   cat /proc/cmdline
   ```

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

   Sample output:

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

7.2. Configuring the VF Port on the Host

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

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

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

Sample output:

```
/sys/class/fpga/intel-fpga-dev.0 -> ../../devices/
pci0000:36/0000:36:00.0/0000:37:00.0/fpga/intel-fpga-dev.0
/sys/class/fpga/intel-fpga-dev.1 -> ../../devices/pci0000:ae/
0000:ae:00.0/0000:af:00.0/fpga/intel-fpga-dev.1
```

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

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

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

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

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

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

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

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

```
lspci -nn | grep :0b2[bc]
```

Sample output:

```
04:00.0 Processing accelerators [1200]: Intel Corporation Device [8086:0b2b]
04:00.1 Processing accelerators [1200]: Intel Corporation Device [8086:0b2c]
```

`lspci` shows an additional Device number, 0b2c. This is the VF Device you assign to a VM. The original Bus and Device numbers for the PF remain 0b2b.

Note that the Domain:Bus:Device.Function (BDF) notation for the VF device is: 000:04:00.1

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 ID and Device ID for the VF Device:

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

Expected output:

```
04:00.1 1200: 8086:0b2c
```

8. Bind the VF to the `vfio-pci` driver: Use the Vendor ID and Device ID from previous step.

```
sudo sh -c "echo 8086 0b2c > \
/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 0b2c. On the virtual machine, install the Intel FPGA Driver and OPAE Software. Refer to *Installing the Release on the Host* 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_PLATFORM_ROOT/sw/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
```

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

1. Uninstall the driver on the VM:

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

2. Detach the VF from the VM.

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

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

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

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

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

```
export pci_path=/sys/class/fpga/intel-fpga-dev.*/device
```

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

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

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

6. Assign the port to the PF using the `fpgaport` tool:

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

8. Intel Acceleration Stack Quick Start Guide: Intel FPGA PAC D5005 Archives

Acceleration Stack Version	User Guide
2.0 (supported with Intel Quartus Prime Pro Edition 18.1.2 version)	Intel Acceleration Stack Quick Start Guide: Intel FPGA Programmable Acceleration Card D5005

9. Document Revision History for Intel Acceleration Stack Quick Start Guide: Intel FPGA PAC D5005

Document Version	Intel Acceleration Stack Version	Changes
2021.07.09	2.0.1 (supported with Intel Quartus Prime Pro Edition 19.2 version)	Added a Troubleshooting FAQ: *Why does the PCIe not detect the Intel FPGA PAC D5005 card?*
2020.09.09	2.0.1 (supported with Intel Quartus Prime Pro Edition 19.2 version)	Added a Troubleshooting FAQ: *What action do I take if the Intel FPGA PAC D5005 does not show up on the PCIe bus?*
2019.12.26	2.0.1 (supported with Intel Quartus Prime Pro Edition 19.2 version)	Added a note about installing Python3 in section: *Installing the Intel Acceleration Stack*,
2019.12.18	2.0.1 (supported with Intel Quartus Prime Pro Edition 19.2 version)	• Updated steps in section: *Updating the FIM and BMC using the fpgaotsu*. • Added a new section: *Intel Acceleration Stack Quick Start Guide: Intel FPGA PAC D5005 Archives*.
2019.11.04	2.0.1 (supported with Intel Quartus Prime Pro Edition 19.2 version)	Updated the following sections: • *Installing the Intel Acceleration Sack Runtime Package on the Host Machine* • *Installing the Intel Acceleration Stack Development Package on the Host Machine* • *Verifying the Installation* • *Identify the FPGA Interface Manager (FIM) and BMC Firmware Version* • *Updating the FIM and BMC* • *Running the OPAE in a Non-Virtualized Environment* • *Running the Hello FPGA Example* • *Running the OPAE in a Virtualized Environment*
2019.08.05	2.0 (supported with Intel Quartus Prime Pro Edition 18.1.2 version)	Initial release.

A. Handling Graceful Thermal Shutdown

Note:

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

The Intel FPGA PAC Daemon (`pacd`) is a program that can be used to help protect the server from crashing due to hardware reaching an upper or lower non-recoverable sensor threshold. `pacd` is capable of monitoring any of the 46 sensors reported by Board Management Controller. `pacd` can be run standalone, as a daemon, or as a *systemd* service. When the *OPAE tools extra package* is installed, `pacd` gets placed in the OPAE binaries directory (default: */usr/bin*) along with configuration and service files – `pacd.conf` and `pacd.service`, respectively.

On startup, `pacd` sets its thresholds to the BMC's sensor threshold values which are readjusted such that the threshold range is expanded up or down by 5%. This is to pass on the Graceful Thermal Shutdown responsibility to `pacd`.

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

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

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

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

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

   Edit entry:

   ```
   DefaultGBSOptions=-n /home/<username>/intelrtestack/\
   a10_gx_pac_ias_1_2_pv/hw/samples/nlb_mode_3/bin/nlb_mode_3_unsigned.gbs
   ```

   Edit entry:

   ```
   DefaultGBSOptions=-n /home/<username>/intelrtestack/\
   d5005_pac_ias_2_0_1*/hw/samples/nlb_mode_3/bin/nlb_mode_3_unsigned.gbs
   ```

Note: Optional settings include:

- PCIe address (For example: -S, -B, -D, -F), `pacd` monitors all Intel FPGA PACs matching the PCIe address components specified. For example, if you specify -B 5 only, all Intel FPGA PACs on PCIe bus 5 becomes monitored.
- Sensor Threshold—The thresholds are global, so specifying -T 11:95.0:93.0 monitors sensor 11 on all selected Intel FPGA PACs. When the value exceeds 95.0, it causes the default bitstream specified with -n in pacd.conf to be programmed (PR). The sensor is considered triggereduntil its value drops below 93.0.
- You can specify a cool down period by:
  a. Changing the `CooldownInterval=<time period>`
  b. Setting `-c <time period>` for `ThresholdOptions` in `pacd.conf` or

  If both are set, then the `-c <time period>` for `ThresholdOptions` takes precedence.
- The Sensor Number can be found by running this command:

```
sudo fpgainfo bmc
```

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

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

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

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

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

Edit entry:

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

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

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

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

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

```
systemctl daemon-reload
```

```
systemctl start pacd.service
```

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

```
systemctl enable pacd.service
```

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

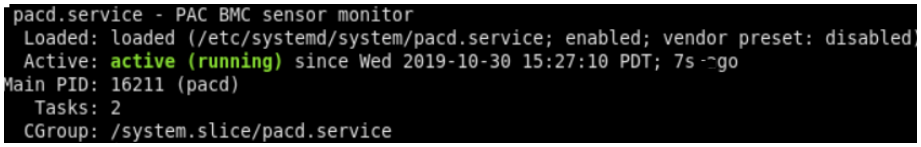
For a full list of `systemctl` commands, run the following command:

```
systemctl -h
```

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

```
systemctl status pacd.service
```

Sample Output:



8. To stop the service:

```
systemctl stop pacd.service
```

B. FPGA Device Access Permission

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

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

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

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

Here are the commands to run:

```
sudo chmod 666 /dev/intel-fpga-fme.*
```

```
sudo chmod 666 /dev/intel-fpga-port.*
```

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

C. Memlock Limit

Depending on the requirements of your host application, you may also want to increase the maximum amount of memory that a user process can lock. This applies when the host application allocate memory using `fpgaPrepareBuffer` OPAE calls. The exact method may vary with your Linux distribution.

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

```
ulimit -l
```

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

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

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

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

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

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

D. Troubleshooting Frequently Asked Questions (FAQ)

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

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

1. Refer to Identify the FPGA Interface Manager (FIM) and BMC Firmware Version on page 15 to determine the required *<FIM version>*.
2. To verify that the AFU is compatible with the FIM version, run the following command:

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

For example, for `nlb_mode_0_signed.gbs` run the following command:

```
packager gbs-info --gbs=$OPAE_PLATFORM_ROOT/hw/samples/nlb_mode_0/bin/\
nlb_mode_0_unsigned.gbs
```

Sample output:

```
{
    "afu-image": {
        "interface-uuid": "9346116d-a52d-5ca8-b06a-a9a389ef7c8d",
        "afu-top-interface": {
            "class": "ccip_std_afu_avalon_mm",
            "module-ports": [
                {
                    "params": {
                        "clock": "pClk"
                    },
                    "optional": true,
                    "class": "local-memory"
                }
            ]
        },
        "magic-no": 488605312,
        "power": 0,
        "accelerator-clusters": [
            {
                "total-contexts": 1,
                "name": "nlb_mode_0",
                "accelerator-type-uuid": "d8424dc4-a4a3-c413-
f89e-433683f9040b"
            }
        ]
    },
    "version": 1,
    "signature": {
        "root_hash": "",
        "CSK": null,
        "root_pub_key-Y": "0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00",
        "root_pub_key-X": "0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
```

```
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00",
        "CSK_pub_key-Y": "0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00",
        "CSK_pub_key-X": "0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0x00
0x00 0x00 0x00 0x00 0x00 0x00 0x00",
        "root_key": null
    }
}
```

The interface-uuid should match the FIM version (PR interface ID) you found in Identify the FPGA Interface Manager (FIM) and BMC Firmware Version on page 15.

D.2. Which environment variables are required?

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

```
source /<installation directory>/intelrtestack/init_env.sh
```

D.3. What actions do I take if I see the error message "Error enumerating resources: no driver available"?

1. Validate that your card is detected by PCIe.

   ```
   lspci | grep 0b2b
   ```

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

D.4. Command `lsmod | grep fpga` shows no output after installing the OPAE driver. How to successfully install the OPAE driver?

1. Check the kernel version running on the server:

   ```
   $ uname -a
   ```

   Sample output:

   ```
   Linux test_machine 3.10.0-957.el7.x86_64
   ```
2. List the kernel source on the system:

   ```
   $ ls -l /usr/src/kernels/
   ```

   Sample output:

   ```
   drwxr-xr-x. 22 root root 4096 Jun 21 13:05 3.10.0-957.el7.x86_64
   ```
3. List the installed kernel header:

   ```
   $ rpm -qa | grep kernel-header

   kernel-headers-3.10.0-957.el7.x86_64
   ```

If the kernel source and headers do not match the kernel version running on the server, there can be issues with installing the OPAE driver.

To fix this issue:

1. Remove the incompatible kernel source and kernel header.
2. Install the correct kernel source:

```
sudo yum install kernel-headers-`uname -r`
```

3. Install the correct kernel headers:

```
sudo yum install kernel-headers-`uname -r`
```

4. Remove and re-install the OPAE driver:

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

```
cd $OPAE_PLATFORM_ROOT/sw/
```

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

D.5. Command `rpm -qa | grep opae` does not return the installed opae rpm package. How to successfully install the packages?

Refer to section: Installing the OPAE Software Package.

D.6. What action do I take if the Intel FPGA PAC D5005 does not show up on the PCIe bus?

Generally, you may observe this issue in the following scenerio:

After inserting the Intel FPGA PAC D5005 in the server and rebooting, the Intel FPGA PAC D5005 appears to be powered and the activity leds are on. But the Intel FPGA PAC D5005 does not show up on the PCIe bus (no 0b2b device).

Root Cause:

An improper insertion of the Intel FPGA PAC D5005 in the PCIe slot of the server.

Corrective Action:

Remove the Intel FPGA PAC D5005 from the server and follow the instructions for Hardware Installation on page 7 to install the Intel FPGA PAC D5005 in your server.

D.7. Why does the PCIe not detect the Intel FPGA PAC D5005 card?

Most common causes are:

- Improper card insertion
- AUX power not connected
- SR-IOV and/or Virtualization Technology not enabled in server BIOS
- Insufficient cooling (card shut down)

Ensure to take appropriate corrective action for these common causes.