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**

Subscribe
Send Feedback

UG-20202 | 2019.08.05
Latest document on the web: PDF | HTML

Contents

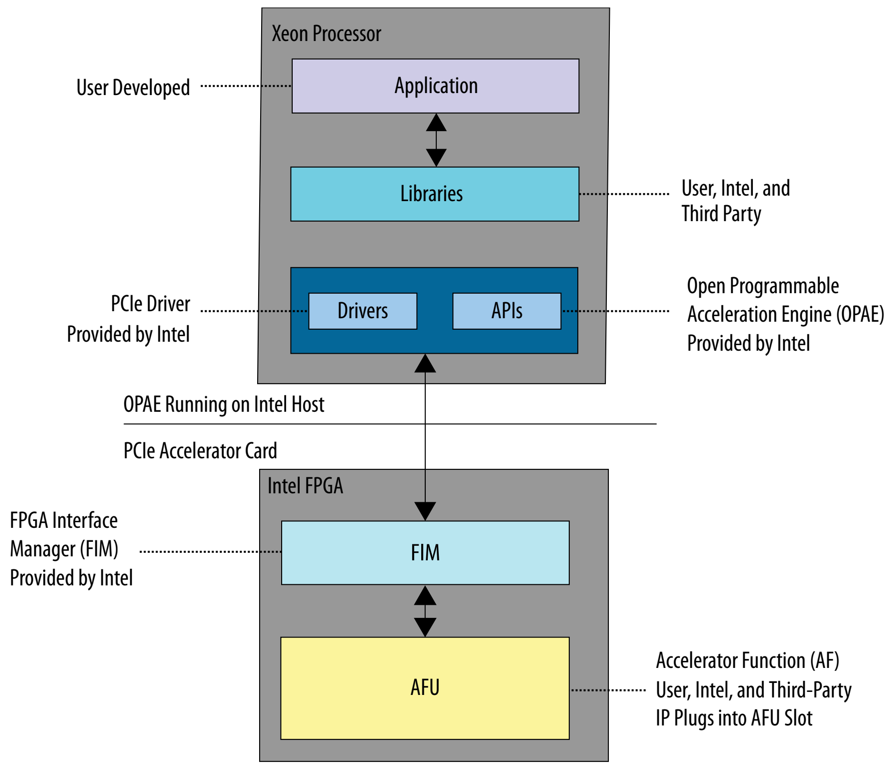
1. Introduction to the Intel FPGA Programmable Acceleration Card D5005

This guide provides a brief introduction to the 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 non-virtualized environment.

The Intel FPGA PAC D5005 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.

Figure 1. Overview of the Intel FPGA PAC Platform Hardware and Software


Xeon Processor
User Developed
Application
Libraries
User, Intel, and Third Party
PCIe Driver Provided by Intel
Drivers
APIs
Open Programmable Acceleration Engine (OPAE) Provided by Intel
OPAE Running on Intel Host
PCIe Accelerator Card
Intel FPGA
FPGA Interface Manager (FIM) Provided by Intel
FIM
AFU
Accelerator Function (AF) User, Intel, and Third-Party IP Plugs into AFU Slot

Related Information

- Developing AFUs with the OPAE SDK
- Intel FPGA SDK for OpenCL Pro Edition Programming Guide

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 will 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
System power supply	Auxiliary Power (12V)[1]
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

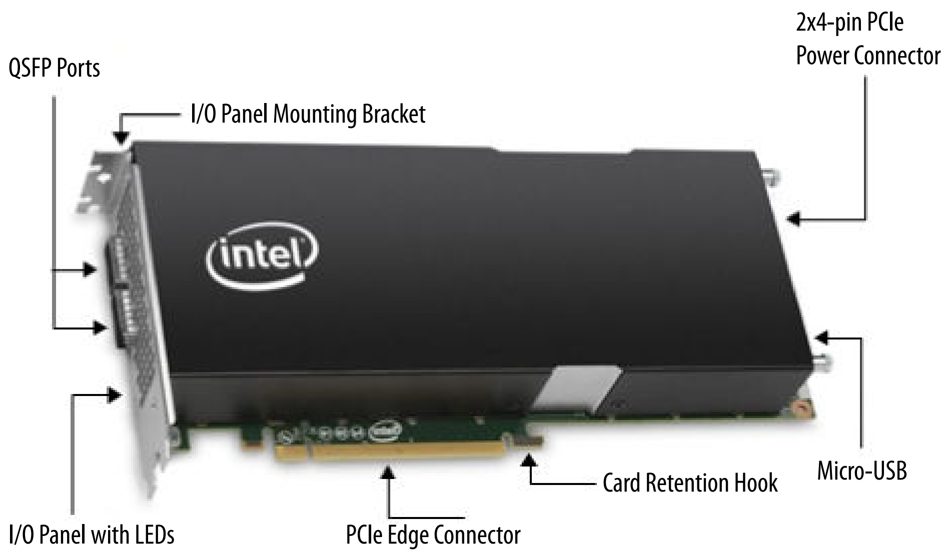
- Intel FPGA PAC D5005
- Extension mounting bracket
- Mounting screws

Intel FPGA PAC 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

(1) Refer to the Supported Server List for power cable details.

Figure 2. Intel FPGA PAC D5005



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. Buy RHEL subscription from Red Hat* store. This step is required to fetch and install packages from 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-optional-rpms
   ```
   At this point the installed kernel is 3.10. You can install these packages after initial installation of the RHEL 7.6 by running:
   ```
   $ sudo yum groupinstall <package_name>
   ```
   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	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 1.1.4-3	OPAE SDK version 1.1.4-3
Intel Quartus Prime Pro Edition	Not Included or required	Included: Intel Quartus Prime Pro Edition 18.1.2 with related interface licenses (SR-IOV and Low Latency 10 Gbps Ethernet MAC/PHY).

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

Follow these instructions to extract the release package and upgrade the kernel:

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 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>/intelrtestack`.
7. Run the initialization script to set the required environment variables and OPAE_PLATFORM_ROOT:

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

8. Run the script `setup_fim_and_bmc.sh` to update the fpga image and board management controller (BMC). For detailed instructions, refer to Identify the FPGA Interface Manager (FIM) and BMC Firmware Version on page 13.
   The $OPAE_PLATFORM_ROOT is used as a relative path to executables and files throughout the document. The following command prints the environment variable OPAE_PLATFORM_ROOT.

   ```
   echo $OPAE_PLATFORM_ROOT
   ```

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

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 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 Software and the Intel SDK for OpenCL*.
7. Run the initialization script to set the required environment variables, QUARTUS_HOME, OPAE_PLATFORM_ROOT, and other OpenCL variables.
   ```
   source /home/<username>/inteldevstack/init_env.sh
   ```
   Note: To avoid having to setup the environment variables after every reboot, save the exported environment variables to your shell initialization script.
8. Run the script `setup_fim_and_bmc.sh` to update the fpga image and board management controller (BMC). For detailed instructions, refer to Identify the FPGA Interface Manager (FIM) and BMC Firmware Version on page 13.
   The $OPAE_PLATFORM_ROOT is used as a relative path to executables and files throughout the document. The following command prints the environment variable OPAE_PLATFORM_ROOT.
   ```
   echo $OPAE_PLATFORM_ROOT
   ```

Related Information

Installing the OPAE Software Package on page 11

2.4.4. Verifying the Installation

Ensure the Intel Acceleration Stack installed correctly.

1. Check the driver installation.

   ```
   lsmod | grep fpga
   ```

   Sample output:

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

2. Verify the OPAE library installation.

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

   Sample output:

   ```
   opae-tools-extra-1.1.4-3.x86_64
   opae-intel-fpga-driver-1.1.4-3.x86_64
   opae-libs-1.1.4-3.x86_64
   opae-tools-1.1.4-3.x86_64
   opae-ase-1.1.4-3.x86_64
   opae-devel-1.1.4-3.x86_64
   ```

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 steps show installation as root.

1. 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
```

   This command only installs missing packages.

   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.

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
```

   b. Install the Intel FPGA driver and OPAE SDK:

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

2. Check the driver installation:

```
lsmod | grep fpga
```

Sample output:

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

3. Verify the OPAE library installation:

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

Sample output:

```
opae-tools-extra-1.1.4-3.x86_64
opae-intel-fpga-driver-1.1.4-3.x86_64
opae-libs-1.1.4-3.x86_64
opae-tools-1.1.4-3.x86_64
opae-ase-1.1.4-3.x86_64
opae-devel-1.1.4-3.x86_64
```

4. Edit the `init_env.sh` file to include `setup_permission.sh` script:

```
$   cd $OPAE_PLATFORM_ROOT/
// Add the following command at the end of file init_env.sh
$ vim init_env.sh
if ls /dev/intel-fpga-* 1> /dev/null 2>&1; then
source $AOCL_BOARD_PACKAGE_ROOT/linux64/libexec/setup_permissions.sh
fi
$  source $OPAE_PLATFORM_ROOT/init_env.sh
```

The `setup_permission.sh` script must be executed after every reboot. For convenience, Intel recommends you to include it as part of the `init_env.sh` script.

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 1.0.12, RTL version
1.0.15, PCB info 'D'
Last Power Down Cause: unavailable
Last Reset Cause: unavailable (can't open)
//****** FME ******//
Object Id                          : 0xF100001
PCIe s:b:d:f                       : 0000:AF:00:0
Device Id                          : 0x0B2B
Socket Id                          : 0x00
Ports Num                          : 01
Bitstream Id                       : 0x203000200000325
Bitstream Version                  : 2.0.339
Pr Interface Id                    : bfac4d85-1ee8-56fe-8c95-865ce1bbaa2d
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

Acceleration Stack Version	FIM Version (PR Interface ID)	OPAE Version	BMC Firmware Version	BMC MAX10 Version
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, 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 using the `fpagflash` Tool* .

Related Information

Running FPGA Diagnostics on page 15

4.1. Updating the FIM and BMC using the `fpgaflash` Tool

1. Run the script:

```
cd $OPAE_PLATFORM_ROOT
./setup_fim_and_bmc.sh -b <bus id> -d <device id> -f <function> -p
$OPAE_PLATFORM_ROOT 2>&1 | tee flash_update.log
```

   For example:

```
./setup_fim_and_bmc -b D8 -d 0 -f 0 -p $OPAE_PLATFORM_ROOT 2>&1 | tee
flash_update.log
```

   - This script prompts you to enter BMC file location. Download the provided BMC package: `d5005_pac_ias_2_0_bmc.tar.gz`. The tarball contains
     - BMC Firmware (FW) version: `max10_darby_revd_v1.0.12.hex`
     - BMC MAX10 RTL version: `max10_darby_revd_v1.0.15_cfm1_auto.rpd`
   - The script execution takes few mins.
2. Power cycle the Intel FPGA PAC.
3. Rerun `sudo fpgainfo fme` to verify the FIM and BMC upgrade was successful.

For any future updates, use the following commands to update:

- FIM:

```
sudo fpgaflash user $OPAE_PLATFORM_ROOT/hw/blue_bits/*.bin
```

- BMC FW:

```
sudo fpgaflash bmc_fw *.hex
```

- Intel MAX® 10 BMC:

```
sudo fpgaflash bmc_img *.rpd
```

5. Running FPGA Diagnostics

This section presents instructions on how to run the FPGA diagnostics by using the `fpgabist` utility. You can run the diagnostic test, provided with the Acceleration Stack Runtime/Development package, to ensure the board interface components (PCIe + DMA) are working according to the use model. 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"
```

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 `nlb_mode_3` AFU image.

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

The `fpgabist` tool accepts PCIe -B, -D, -F as arguments on multi card systems. Sample output:

```
Running fpgadiag trput test...

Cachelines Read_Count Write_Count Cache_Rd_Hit Cache_Wr_Hit Cache_Rd_Miss
      1024  670267380   670269108            0            0             0

Cache_Wr_Miss   Eviction 'Clocks(@250 MHz)'   Rd_Bandwidth   Wr_Bandwidth
            0          0       1250189534     12.762 GB/s     12.77 GB/s

VH0_Rd_Count VH0_Wr_Count VH1_Rd_Count VH1_Wr_Count
   670267380    670269109            0            0

VL0_Rd_Count VL0_Wr_Count
           0            0

Finished Executing NLB (FPGA DIAG)Tests

Built-in Self-Test Completed.
```

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

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

4. Configure and run diagnostics with DMA AFU image.

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

Sample output:

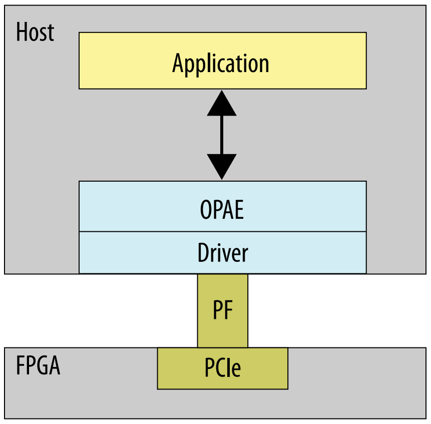
```
Running mode: dma_afu
Attempting Partial Reconfiguration:
Reading bitstream
```

```
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 3. OPAE Driver in Non-Virtualized Mode



6.1. Loading the AFU Image into the FPGA

Use the `fpgaconf` utility to load the AFU image.

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

The tool also accepts PCIe Bus:Device:Function (BDF) as an additional optional arguments if multiple cards are connected to the server. Use the help text (`-h`) for the OPAE tools to see how additional arguments must be passed. For example: `sudo fpgaconf -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 fpgaconf $OPAE_PLATFORM_ROOT/hw/samples/nlb_mode_0/bin/\
   nlb_mode_0.gbs
   ```

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

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

3. 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
   ```

4. Extract the `tar` file:

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

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:

   ```
   sudo ./hello_fpga
   ```

   Sample output:

   ```
   Running Test
   Done Running Test
   ```

The Acceleration Stack 2.0 Release includes the following working AFU images in the `$OPAE_PLATFORM_ROOT/hw/samples` directory:

- `dma_afu/bin/dma_afu.gbs`
- `dma_afu/bin/streaming_dma_afu.gbs`
- `hello_afu/bin/hello_afu.gbs`
- `hello_intr_afu/bin/hello_intr_afu.gbs`
- `hello_mem_afu/bin/hello_mem_afu.gbs`
- `nlb_mode_0/bin/nlb_mode_0.gbs`
- `nlb_mode_3/bin/nlb_mode_3.gbs`
- `eth_e2e_e10/bin/`

The code below shows an example `hello_afu` compile and execute.

```
cd $OPAE_PLATFORM_ROOT/hw/samples
sudo fpgaconf $OPAE_PLATFORM_ROOT/hw/samples/hello_afu/bin/*.gbs
cd $OPAE_PLATFORM_ROOT/hw/samples/<example AFU>/sw
make
sudo ./<exe>
```

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

```
sudo fpgaconf\
$OPAE_PLATFORM_ROOT/hw/samples/nlb_mode_0/bin/*.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 fpgaconf\
$OPAE_PLATFORM_ROOT/hw/samples/nlb_mode_3/bin/*.gbs
sudo sh -c "echo 20 > /sys/kernel/mm/hugepages/\
hugepages-2048kB/nr_hugepages"
sudo nlb3
```

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

Related Information

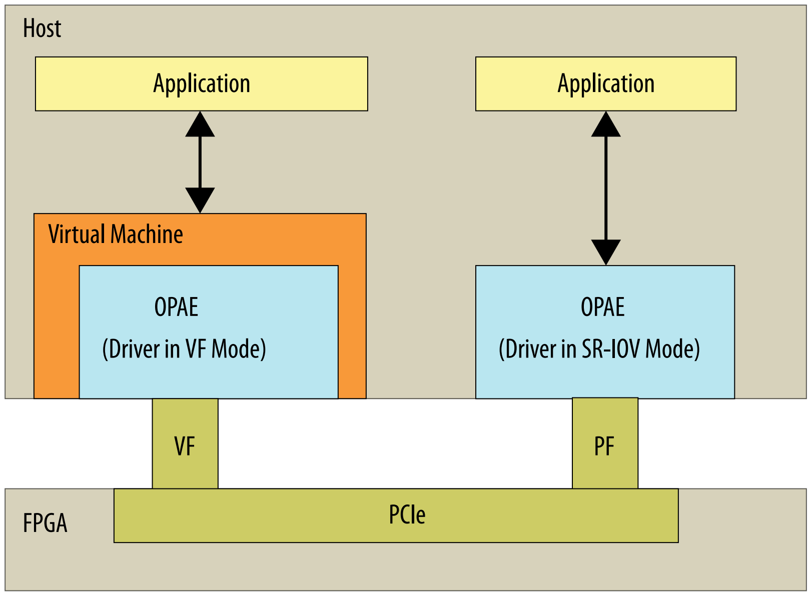
- DMA Accelerator Functional Unit (AFU) User Guide
  For information on how to compile and execute the `dma_afu`.
- Streaming DMA Accelerator Functional Unit (AFU) User Guide
  For information on how to compile and execute the `streaming_dma_afu`.

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 4. 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.

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

Sample 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*
```

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:

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

2. Detach the VF from the VM.

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

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

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

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

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

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

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

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

6. Assign the port to PF using fpgaport tool:

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

8. Document Revision History for Intel Acceleration Stack Quick Start Guide

Document Version	Intel Acceleration Stack Version	Changes
2019.08.05	2.0 (supported with Intel Quartus Prime Pro Edition 18.1.2 version)	• Added the server and connector information in table: *System Requirements for the Intel FPGA PAC D5005*. • Added new steps in section: — *Installing Red Hat Enterprise Linux version 7.6* — *Installing the Intel Acceleration Stack Runtime Package on the Host Machine* — *Installing the Intel Acceleration Stack Development Package on the Host Machine* — *Updating the FIM and BMC using the fpgaflash Tool* • Updated OPAE package version number
2019.02.01	2.0 Pre Beta (supported with Intel Quartus Prime Pro Edition 18.1 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.
- Refer to the OPAE pacd documentation for more details on using pacd, including considerations that may lead to an unexpected system reboot.

The Intel FPGA PAC Daemon (pacd) is a program that can be used to help protect the user’s server from crashing due to hardware reaching an upper non-recoverable or lower non-recoverable sensor threshold. pacd is capable of monitoring any of the 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 a configuration and service file – pacd.conf and pacd.service, respectively.

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

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

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

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

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

   Edit entry:

   ```
   DefaultGBSOptions=-n /home/<username>/intelrtestack/\
   d5005_pac_ias_2_0_b*/hw/samples/nlb_mode_3/bin/nlb_mode_3.gbs
   ```

Note: Optional settings include:

- PCIe address (For example: -S, -B, -D, -F), `pacd` monitors all Intel 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 triggered (and no PR is performed) until its value drops below 93.0
- Specify `-c <time period>` for `ThresholdOptions` in `pacd.conf` to vary the cool down period or change `CooldownInterval=<time period>`.
- The Sensor Number can be found by running this command:

  ```
  sudo fpgainfo bmc
  ```

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

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

   RHEL:

   ```
   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:
   RHEL:

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

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

   RHEL:

   ```
   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).

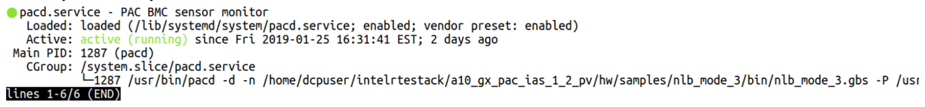
For a full list of `systemctl` commands, run the following command:

```
systemctl -h
```

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

```
systemctl status pacd.service
```

Sample Output:



```
●pacd.service - PAC BMC sensor monitor
   Loaded: loaded (/lib/systemd/system/pacd.service; enabled; vendor preset: enabled)
   Active: active (running) since Fri 2019-01-25 16:31:41 EST; 2 days ago
 Main PID: 1287 (pacd)
   CGroup: /system.slice/pacd.service
           └─1287 /usr/bin/pacd -d -n /home/dcpuser/intelrtestack/a10_gx_pac_ias_1_2_pv/hw/samples/nlb_mode_3/bin/nlb_mode_3.gbs -P /usr
lines 1-6/6 (END)
```

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

8. To stop the service:

```
systemctl stop pacd.service
```

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

B. Troubleshooting Frequently Asked Questions (FAQ)

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

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

1. Refer to Identify the FPGA Interface Manager (FIM) and BMC Firmware Version on page 13 to determine the required <*FIM version*>.
2. To verify that the AFU is compatible with the FIM version, run the following command:

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

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

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

Sample output:

```
{
    "version": 1,
    "afu-image": {
        "interface-uuid": "8e5d8914-e35d-5e92-ab36-c9082a2b1934",
        "afu-top-interface": {
            "class": "ccip_std_afu"
        },
        "magic-no": 488605312,
        "power": 0,
        "accelerator-clusters": [
            {
                "total-contexts": 1,
                "name": "nlb_400",
                "accelerator-type-uuid": "d8424dc4-a4a3-c413-
f89e-433683f9040b"
            }
        ]
    }
}
```

The interface-uuid should match the FIM version (PR interface ID) you found in Identify the FPGA Interface Manager (FIM) and BMC Firmware Version on page 13.

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

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

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

```
fpgaflash --help
```

B.3. Which environment variables are required?

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

```
source init_env.sh
```

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

1. Validate that your card is detected by PCIe.
   ```
   lspci | grep 0b2b
   ```
   If it is not detected, remove the card and then plug it back in again.
2. Reinstall OPAE by following steps listed the *Installing the OPAE Software* section.