# Contents

1 Introduction to the Acceleration Stack for Intel® Xeon® CPU with FPGAs ...................... 3  
   1.1 Acronym List for Intel Acceleration Stack Quick Start Guide for Intel Programmable  
       Acceleration Card with Intel Arria 10 GX FPGA....................................................... 6  
   1.2 Acceleration Glossary........................................................................................................ 7  
   1.3 Differences Between the Hardware Platforms Supported by the Acceleration Stack...... 7  

2 Getting Started.................................................................................................................. 9  
   2.1 Workstation Requirements................................................................................................. 9  
   2.2 Installing Required OS Packages and Components While Installing CentOS 7.4............. 10  
   2.3 Installing the Intel Quartus Prime Pro Edition Software............................................... 11  
   2.4 Extracting the Intel PAC with Arria 10 Package............................................................. 11  
   2.5 Installing the Intel PAC with Arria 10 Card In the Host Machine.................................. 12  

3 Update the Board Management Controller (BMC) Configuration and Firmware............. 14  

4 Update Flash with Beta FPGA Interface Manager (FIM) Image using Intel Quartus  
   Prime Programmer.............................................................................................................. 16  

5 Installing the OPAE Software Package............................................................................. 17  
   5.1 Installing the Intel FPGA Driver..................................................................................... 17  
   5.2 Installing and Building the OPAE Software.................................................................. 18  
       5.2.1 Installing the OPAE Software............................................................................... 18  
       5.2.2 Building the OPAE Software................................................................................. 18  

6 Running FPGA Diagnostics............................................................................................... 20  

7 Using the OPAE in a Non-Virtualized Environment ........................................................ 21  
   7.1 Loading the AFU Image into the FPGA.......................................................................... 21  
   7.2 OPAE Sample Application Programs .......................................................................... 22  
       7.2.1 Running the Hello FPGA Example........................................................................ 22  

8 Running the OPAE in a Virtualized Environment ............................................................ 24  
   8.1 Updating Settings Required for VFs............................................................................... 25  
   8.2 Configuring the VF Port on the Host............................................................................. 25  
   8.3 Running the Hello FPGA Example on Virtual Machine.................................................. 26  
       8.3.1 Disconnecting the VF from the VM and Reconnecting to the PF......................... 27  

A Additional Flag Settings to AFU Makefiles....................................................................... 28  

B Update Flash with Beta FPGA Interface Manager (FIM) Image using fpgaflash Tool....29  

C Documentation Available for the Acceleration Stack for Intel Xeon CPU with FPGAs  
   1.0 Beta Release.................................................................................................................... 31  

D Document Revision History for the Intel Acceleration Stack Quick Start Guide for  
   Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA............................ 32
1 Introduction to the Acceleration Stack for Intel® Xeon® CPU with FPGAs

Note: This document is the preliminary version and subject to change.

Acceleration Stack for Intel® Xeon® CPU with FPGAs is a collection of software, firmware, and tools that allows software developers to leverage the power of Intel FPGAs by being able to run custom software on the Xeon processor that offloads computationally intensive tasks to the Intel Arria® 10 FPGA freeing cores within the Xeon for other processing tasks.

This guide specifically covers the Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA hardware platform, which connects a PCIe accelerator card with an Intel Arria 10 GX FPGA to a motherboard with a Xeon processor.

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

To take advantage of the flexibility of the FPGA, you can reconfigure a special, partial reconfiguration (PR) region of the Intel Arria 10 GX FPGA at run time. You can design multiple Accelerator Functional Units (AFUs) to swap in and out of this PR region. For more information, refer to the "Overview of the Acceleration Stack for Intel Xeon CPU with FPGAs Platform Hardware and Software" figure. The Open Programmable Acceleration Engine (OPAE) software running on the Intel Xeon processor handles all the details of the reconfiguration process.

For more information about the OPAE software, refer to the OPAE documents in the dcp_1_0_beta_docs.zip file, which is provided with your download.

Reconfiguration is one of many utilities that the OPAE provides. The OPAE also provides libraries, drivers, and sample programs that you use when developing AFUs.

To facilitate dynamically loading AFUs, the Acceleration Stack includes the following two components:
• The FPGA Interface Manager (FIM), which provides a framework for loaded AFUs within the Intel Xeon Processor with Integrated FPGA, referred to as Integrated FPGA Platform throughout this document, contains the FPGA logic to support the accelerators, including the PCIe IP core, the CCI-P fabric, the on-board DDR memory interfaces, and the FPGA Management Engine (FME). The FIM also includes the PR regions for loading AFUs. At power up, the Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA, referred to as Intel PAC with Arria 10 throughout this document, Intel PAC with Arria 10 is configured from on-board FPGA configuration flash, containing the FIM bitstream image. The PR regions are initially empty until OPAE is used to load AFUs. The framework provided by the FIM cannot be altered. The current release of the Acceleration Stack for the Intel PAC with Arria 10 supports a single PR region in the FIM.

• Loadable AFU images, which are dynamically loaded by OPAE into the PR regions in the FIM. The Acceleration Stack supports creating AFU images with multiple design flows, including a low level RTL flow to high level design flows such as OpenCL. An AFU image is the combination of the AFU’s PR region bitstream generated from the supported design flows and metadata used to provide OPAE information on the AFU’s characteristics and operational parameters used at load time. The current release of the Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA supports dynamically swapping AFU images in a single PR region per installed Intel PAC with Arria 10.

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

Figure 2 on page 5 provides additional information about the AFU.
Figure 2. **Arria 10 with a Single AFU PR Region**

![Diagram](image)

The AFU connects to the Xeon processor through the CCI-P interface and then the PCIe link. The Intel PAC with Arria 10 platform uses a simplified version of the CCI-P interface. Refer to the *Intel Xeon with FPGA IP Core Cache Interface (CCI-P) Specification* for more information about the CCI-P interface. This specification is available in the dcp_1_0_beta_docs.zip file that accompanies this guide.

The AFU also connects to two banks of private DDR4 memory. Each DDR4 memory bank interface is a standard Avalon® Memory-Mapped (Avalon-MM) interface. Refer to *Avalon-MM Interface Specifications* for more information about this interface.

The Intel PAC with Arria 10 features with a single QSFP+ network port, and two banks of DDR4. This accelerator card includes up to 8 gigabytes (GB) of DDR4 x72 memory with error correction code (ECC). The current release of the Acceleration Stack does not support the QSFP+ interface.

For the Acceleration Stack for Intel Xeon CPU with FPGAs 1.0 release, servers can scale performance by installing Intel PAC with Arria 10 accelerator cards in all available PCIe slots.

*Note:* Acceleration using multiple accelerator cards is currently not available unless it has been validated with the Beta release. You can use this feature when it is available with the 1.0 release.

**Related Links**

*Avalon Memory-Mapped Interfaces*
## 1.1 Acronym List for Intel Acceleration Stack Quick Start Guide for Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFU</td>
<td>Accelerator Functional Unit</td>
</tr>
<tr>
<td>ASE</td>
<td>AFU Simulation Environment</td>
</tr>
<tr>
<td>CCI-P</td>
<td>Core Cache Interface</td>
</tr>
<tr>
<td>FIM</td>
<td>FPGA Interface Manager</td>
</tr>
<tr>
<td>FME</td>
<td>FPGA Management Engine</td>
</tr>
<tr>
<td>Intel Xeon Processor with Integrated FPGA</td>
<td>Integrated FPGA Platform</td>
</tr>
<tr>
<td>IOMMU</td>
<td>Input-Output Memory Management Unit</td>
</tr>
<tr>
<td>OPAE</td>
<td>Open Programmable Acceleration Engine (OPAE)</td>
</tr>
<tr>
<td>PR</td>
<td>Partial Reconfiguration</td>
</tr>
<tr>
<td>RAS</td>
<td>Reliability, Assessability and Serviceability (RAS)</td>
</tr>
<tr>
<td>RBF</td>
<td>Raw Binary File</td>
</tr>
<tr>
<td>Xeon + FPGA</td>
<td>Xeon + FPGA</td>
</tr>
</tbody>
</table>
1.2 Acceleration Glossary

Table 2. Acceleration Stack for Intel Xeon CPU with FPGAs Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acceleration Stack for Intel Xeon CPU with FPGAs</td>
<td>Acceleration Stack</td>
<td>A collection of software, firmware and tools that provides performance-optimized connectivity between an Intel FPGA and an Intel Xeon processor.</td>
</tr>
<tr>
<td>Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA</td>
<td>Intel PAC with Arria 10</td>
<td>PCIe* accelerator card with an Intel Arria 10 FPGA. Programmable Acceleration Card is abbreviated PAC. Contains a FPGA Interface Manager (FIM) that pairs with an Intel Xeon processor over PCIe bus.</td>
</tr>
<tr>
<td>Intel Xeon Processor with Integrated FPGA</td>
<td>Integrated FPGA Platform</td>
<td>Intel Xeon plus FPGA platform with the Intel Xeon and an FPGA in a single package and sharing a coherent view of memory via Quick Path Interconnect (QPI).</td>
</tr>
</tbody>
</table>

1.3 Differences Between the Hardware Platforms Supported by the Acceleration Stack

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

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

<table>
<thead>
<tr>
<th>Feature</th>
<th>Intel Xeon Processor with Integrated FPGA</th>
<th>Intel PAC with Arria 10</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cached Memory Link</td>
<td>Yes</td>
<td>No</td>
<td>The Intel Xeon Processor with Integrated FPGA platform includes a low latency coherent link to host memory. The Intel PAC with Arria 10 platform does not include this link; consequently, UMsg is not supported.</td>
</tr>
<tr>
<td>PCIe Interfaces</td>
<td>2, Gen3x8</td>
<td>1, Gen3x8</td>
<td>The Intel Xeon Processor with Integrated FPGA platform has two PCIe links: • The primary link • A secondary link for extra bandwidth</td>
</tr>
<tr>
<td>Physical attachment</td>
<td>Internal on-die package</td>
<td>PCIe accelerator card</td>
<td>The shares the CCI-P interface for AFUs. This interface abstracts the physical link differences between the Intel Xeon Processor with Integrated FPGA and platforms.</td>
</tr>
</tbody>
</table>

continued...
<table>
<thead>
<tr>
<th>Feature</th>
<th>Intel Xeon Processor with Integrated FPGA</th>
<th>Intel PAC with Arria 10</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Network</td>
<td>10/40/100 Gbps Ethernet (GbE)</td>
<td>Not Supported*(1)</td>
<td>None.</td>
</tr>
<tr>
<td>Quick Path Interconnect (QPI)</td>
<td>Supported</td>
<td>Not supported</td>
<td>The shares the CCI-P interface for AFUs. This interface abstracts the physical link differences between the Intel Xeon Processor with Integrated FPGA and platforms.</td>
</tr>
<tr>
<td>SMBus</td>
<td>Sideband Management Supported</td>
<td>Not Applicable</td>
<td>None.</td>
</tr>
<tr>
<td>Thermal shutdown</td>
<td>Supported</td>
<td>Not Supported</td>
<td>None.</td>
</tr>
<tr>
<td>Remote JTAG</td>
<td>Supported</td>
<td>Not Supported</td>
<td>None.</td>
</tr>
<tr>
<td>Power/thermal AFU stalling</td>
<td>Supported</td>
<td>Not Supported</td>
<td>None.</td>
</tr>
<tr>
<td>Reliability, Assessability and Serviceability (RAS)</td>
<td>Supported</td>
<td>Not Supported</td>
<td>None.</td>
</tr>
<tr>
<td>Performance analysis, performance counters, Vtune integration</td>
<td>Supported</td>
<td>Not Supported</td>
<td>None.</td>
</tr>
<tr>
<td>Telemetry - temperature and power reporting to Control and Status Registers (CSRs)</td>
<td>Supported</td>
<td>Not Supported</td>
<td>None.</td>
</tr>
<tr>
<td>Private Memory</td>
<td>Not Supported</td>
<td>2 banks of DDR4 SDRAM</td>
<td>None.</td>
</tr>
<tr>
<td>CCI-P Access Modes</td>
<td>UPI, PCIe0, PCIe1</td>
<td>PCIe0</td>
<td>The Intel Xeon Processor with Integrated FPGA platform has a secondary link for extra bandwidth.</td>
</tr>
</tbody>
</table>

*(1) Networking connectivity planned for a future release.*
2 Getting Started

2.1 Workstation Requirements

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

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

The following servers have been tested for this release:

- Dell R640
- Dell R740

The best known server configuration with PAC card must include:

- Intel Xeon processor
- A PCI Express* x16 Slot
- RAM: 128 GB
- Hard Disk Drive: 2 terabytes (TB)
- BIOS Revision: 3.91
- RHEL 7.4 or Cent OS 7.4

*Other names and brands may be claimed as the property of others.
2.2 Installing Required OS Packages and Components While Installing CentOS 7.4

Install the software and select the following options and packages:

- Development and Creative Workstation
- Additional Development
- Compatibility Libraries
- Development Tools
- Platform Development
- Python
- Virtualization Hypervisor

Table 4. Useful Linux Commands

The following Linux commands provide information about your system.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sudo dmidecode -t bios</code></td>
<td>Lists BIOS information, including revision</td>
</tr>
<tr>
<td><code>cat /proc/cpuinfo</code></td>
<td>Lists CPU information</td>
</tr>
<tr>
<td><code>cat /etc/redhat-release</code></td>
<td>Lists CentOS version information</td>
</tr>
<tr>
<td><code>cat /proc/version</code></td>
<td>Lists Linux kernel version</td>
</tr>
</tbody>
</table>

The following commands list the exact versions the BIOS and processor used in testing for this release.

```
$ sudo dmidecode -t bios
# dmidecode 3.0
Scanning /dev/mem for entry point.
SM BIOS 2.7 present.
Handle 0x0000, DMI type 0, 24 bytes
BIOS Information
  Vendor: Hewlett-Packard
  Version: J61 v03.91
  Release Date: 10/17/2016
  Address: 0xF0000
  Runtime Size: 64 kB
  ROM Size: 16384 kB

$ cat /proc/cpuinfo
processor: 0
  vendor_id: GenuineIntel
  cpu family: 6
  model: 62
  model name: Intel(R) Xeon(R) CPU E5-1650 v2 @ 3.50GHz
  stepping: 4

$ cat /proc/version
  Linux version 3.10.0-514.21.1.el7.x86_64 (builder@kbuilder.dev.centos.org) #1 SMP Thu May 25 17:04:51 UTC 2017
  GCC version 4.8.5 20150623 (Red Hat 4.8.5-11) (GCC) #1 SMP Thu May 25 17:04:51 UTC 2017
$ cat /etc/redhat-release
  CentOS Linux release 7.3.1611 (Core)
```
2.3 Installing the Intel Quartus Prime Pro Edition Software

Install the Intel Quartus Prime Pro Edition 17.0.0 software to create, debug, and simulate Accelerator Functions. Download this software from the Intel FPGA Download Center.

Note: Only Intel Quartus Prime Pro Edition 17.0.0 is supported.

Refer to Intel FPGA Requirements for the necessary RAM and disk space to install and run the Intel Quartus Prime Pro Edition 17.0 software. Because the Intel PAC with Arria 10 platform only requires support for Intel Arria 10 devices, the disk space requirement is only 25 GB.

You must have an IP-PCIE/SRIOV license to generate the AFU image. Refer to Use the Self-Service Licensing Center to get and manage a license.

2.4 Extracting the Intel PAC with Arria 10 Package

The archive file, dcp_1_0_beta.tar.gz, includes the files for the Intel PAC with Arria 10 1.0 Beta Release.

Complete the following steps to extract the archive file and define an environment variable named DCP_LOC pointing to the extracted release location. Be sure to replace <path_to_download_location> with the directory path to the release archive on your workstation.

1. $ mkdir dcp_1.0_beta
2. $ cd dcp_1.0_beta
3. $ export DCP_LOC=`pwd`
4. $ tar xf <path_to_download_location>/dcp_1_0_beta.tar.gz
2.5 Installing the Intel PAC with Arria 10 Card In the Host Machine

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

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

1. Enable the following options in the BIOS:
2. Plug the Intel PAC with Arria 10 card into the x16 slot on the motherboard.

3. A Micro USB cable is required:
   - To update firmware on the card.
   - If programming of flash over PCIe (fpgaflash tool) fails, flash has to be programmed over Jtag (Intel Quartus Prime Programmer).

   Note: Future upgrades to flash over PCIe (fpgaflash tool) does not require USB cable.
3 Update the Board Management Controller (BMC) Configuration and Firmware

1. Loading FPGA Interface Manager (FIM) as a pre-requisite requires using the Bittware tool along with a USB-Blaster cable to update the BMC configuration and firmware.

2. To obtain the firmware and tools for updating the BMC firmware, refer to the BMC firmware information.

3. Determine if the Bittware tools are already installed and their version by running the command:

   ```sh
   $ bwconfig --version
   ```

4. If the tool is installed, and the reported version is not 2017.4, remove the currently installed tools. The command varies with the release, but it typically is:

   ```sh
   $ sudo yum remove bw2tk
   ```

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

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

6. Check if the BMC configuration flag is set to 0x81:

   ```sh
   $ bwmonitor --dev=0 --type=Bmc --flags --read
   BMC flags set to 80
   Sun Nov 26 17:25:01 2017
   Board Management Controller A10SA4 dev 0
   (1) Microcontroller
   ```

Note: If you receive error:

   ```sh
   ERROR Item not found: could not open device 0
   ```

   a. `bwconfig --remove=0`
   
   b. `bwconfig --scan=usb`

   Sample output when the Intel Programmable Acceleration Card is detected:

   ```sh
   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 Programmable Acceleration Card with Intel Arria 10 GX FPGA to the BittWorks Toolkit managed device list:

   ```sh
   $ bwconfig --add=usb --result=0
   Sample output:
   Scanning for devices
   ```
7. If flag is not already set to 0x81, set the flag by running command:

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

8. To update to the latest firmware version: 26810, run:

```bash
$ bwmonitor --dev=0 --type=bmc --write=1 --file=a10sa4-26810-fw.hex
```

9. Verify firmware updated

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

BwMonitor (cli) version 2017.4.233.30111
BMCLIB version : 2017.4.233.30111
MCU Firmware version : 0x68ba
Wed Nov 29 20:45:05 2017


**Related Links**

Intel® Programmable Acceleration Card with Intel Arria® 10 GX FPGA
4 Update Flash with Beta FPGA Interface Manager (FIM) Image using Intel Quartus Prime Programmer

Note: A Micro USB cable is required.

1. cd $DCP_LOC/hw/blue_bits/
2. export QUARTUS_HOME=<path to quartus installation>
3. $QUARTUS_HOME/bin/quartus_pgm -m JTAG -o 'pvbi;dcp_1_0.jic'
4. Power cycle the host machine.

Note: Now that Intel PAC with Arria 10 has been updated to beta, 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 Beta FPGA Interface Manager (FIM) Image using fpgaflash Tool " section

Related Links
- Update the Board Management Controller (BMC) Configuration and Firmware on page 14
- Update Flash with Beta FPGA Interface Manager (FIM) Image using Intel Quartus Prime Programmer on page 16
- Installing the OPAE Software Package on page 17
- Update Flash with Beta FPGA Interface Manager (FIM) Image using fpgaflash Tool on page 29
5 Installing the OPAE Software Package

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

After completing the OPAE software installation,

- Intel FPGA Driver is installed and loaded
- OPAE source is available at $OPAE_LOC
- RPM flow will install OPAE binaries at /usr/bin
- RPM flow will install OPAE libraries at /usr/lib
- RPM flow will install OPAE headers at /usr/include

Related Links
Installing Required OS Packages and Components While Installing CentOS 7.4 on page 10

5.1 Installing the Intel FPGA Driver

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

Complete the following steps to install the Intel FPGA driver:

1. Remove any previous version of the OPAE FPGA driver by running the command: $ sudo yum remove opae-intel-fpga-drv.x86_64.
2. Install the Extra Packages for Enterprise Linux (EPEL): $ sudo yum install epel-release
3. Install the Intel FPGA kernel drivers:
   $ cd $DCP_LOC/sw
   $ sudo yum install $DCP_LOC/sw/opae-intel-fpga-drv-0.13.0.x86_64.rpm
4. Check the Linux kernel installation: lsmod | grep fpga
   Sample output:
<table>
<thead>
<tr>
<th>Module</th>
<th>Size</th>
<th>Prio</th>
</tr>
</thead>
<tbody>
<tr>
<td>intel_fpga_fme</td>
<td>51462</td>
<td>0</td>
</tr>
<tr>
<td>intel_fpga_afu</td>
<td>31735</td>
<td>0</td>
</tr>
<tr>
<td>fpga_mgr_mod</td>
<td>14693</td>
<td>1</td>
</tr>
<tr>
<td>intel_fpga_pci</td>
<td>25804</td>
<td>2</td>
</tr>
</tbody>
</table>

*Other names and brands may be claimed as the property of others.
5.2 Installing and Building the OPAE Software

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
```

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

5.2.1 Installing the OPAE Software

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

Complete the following steps to install the OPAE software:

1. **Install shared libraries at location** /usr/lib, **required for user applications to link against**:
   ```
sudo yum install opae-0.13.0-1.x86_64-libs.rpm
   
   **Note:**
   
   For more information about tools, refer to the OPAE tools document.
   ```

2. **Install the OPAE header at location** /usr/include:
   ```
sudo yum install opae-0.13.0-1.x86_64-devel.rpm
   
   **Note:**
   
   For more information about tools, refer to the OPAE tools document.
   ```

3. **Install the OPAE provided tools at location** /usr/bin (For example: fpgaconf and fpgainfo):
   ```
sudo yum install opae-0.13.0-1.x86_64-tools.rpm
   
   **Note:**
   
   For more information about tools, refer to the OPAE tools document.
   ```

4. **Install the ASE related shared libraries at location** /usr/lib:
   ```
sudo yum install opae-0.13.0-1.x86_64-ase.rpm
   
   **Note:**
   
   For more information about tools, refer to the OPAE tools document.
   ```

5. `$ sudo ldconfig`

5.2.2 Building the OPAE Software

If you prefer building the OPAE Software from source (NOT REQUIRED), you must configure the Open Programmable Acceleration Engine (OPAE) software package using the `cmake` and `make` commands.
Complete the following steps to build the OPAE software:

1. $ tar xf opae-src-0.13.0.tar.gz
2. $ cd opae-0.13.0-1
3. $ mkdir build && cd build
4. Configure the OPAE software package and set the version to 0.13.0 to match the driver version number:
   ```
   cmake .. -DBUILD_ASE=ON -DCMAKE_INSTALL_PREFIX=<path to install directory>
   (For example: cmake .. -DBUILD_ASE=ON -DCMAKE_INSTALL_PREFIX=/home/john/opaeinstall)
   ```
   By default, if you followed the rpm install flow, the binaries, libraries and include files get installed under `/usr/`.
   
   **Note:** You may get an error because the `cmake` command cannot find the git repository. You can safely ignore this error message. The git repository is not required to successfully build the OPAE software.

5. $ make install
   This `make` command builds the following packages:
   - Executables in `<path to install directory>`
   - Libraries in `<path to install directory>`

6. $ make doc
   The headers are in `<path to install directory>/include`
   This `make` command makes the Doxygen documentation in `<path to install directory>`.

7. Identify the loaded FPGA Interface Manager image:

<table>
<thead>
<tr>
<th>Beta FIM ID</th>
<th>3d949b987b305a9ab2964530a780a3f9</th>
</tr>
</thead>
</table>

Run the `fpgainfo` tool. For more information about the tool refer to the **OPAE Tools Guide**.

```bash
sudo fpgainfo fme
```

**Sample Output:**

```plaintext
//***** FME *****//
Class Path : /sys/class/fpga/intel-fpga-dev.0/intel-fpga-fme.0
Device Path : /sys/devices/pci0000:00/0000:00:03.0/0000:00:03.0/0000:00:04:00.0/
  fpga/intel-fpga-dev.0/intel-fpga-fme.0
Bus : 0x04
Device : 0x00
Function : 0x00
Version : 0
Ports Num : 1
Socket Id : 0
Pr Interface Id : 3d949b98-7b30-5a9a-b296-4530a780a3f9
Object Id : 0xf500000 FPGADEVICE
```
6 Running FPGA Diagnostics

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

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

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

2. Run the fpgabist test with the NLB_3 AFU image to run the "fpga_dma_test":

```bash
sudo fpgabist $DCP_LOC/hw/samples/nlb_mode_3/bin/nlb_mode_3.gbs
```

Sample output:

```
Cachelines Read_Count Write_Count Cache_Rd_Hit Cache_Wr_Hit Cache_Rd_Miss Cache_Wr_Miss Eviction 'Clocks (@400 MHz)' Rd_Bandwidth Wr_Bandwidth
0 0 0 0 0 0 1200074611 6.160 GB/s 6.263 GB/s
6.160 GB/s 6.263 GB/s
VH0_Rd_Count VH0_Wr_Count VH1_Rd_Count VH1_Wr_Count VL0_Rd_Count VL0_Wr_Count
2887494796 2935937473 0 0 0
```

3. Run the fpgabist test with the DMA AFU image to run the "DMA" mode.

```bash
sudo fpgabist $DCP_LOC/hw/samples/dma_afu/bin/dma_afu.gbs
```

Sample output:

```
Running test in HW mode
Buffer Verification Success!
Running DDR sweep test
Allocated test buffer
Fill test buffer
DDR Sweep Host to FPGA
Measured bandwidth = 7809.031447 Megabytes/sec
Clear buffer
DDR Sweep FPGA to Host
Measured bandwidth = 8421.504502 Megabytes/sec
Verifying buffer...
Buffer Verification Success!
 Finished Executing DMA Tests
```

Related Links

OPAE FPGA Tools - fpgabist

For more information about the fpgabist utility
7 Using the OPAE in a Non-Virtualized Environment

This section shows OPAE examples running directly on the BareMetal operating system without a virtual machine or Single Root I/O Virtualization (SR-IOV).

Figure 4. OPAE Driver in Non-Virtualized Mode

7.1 Loading the AFU Image into the FPGA

Use the fpgaconf utility, located at fpgaconf to load the loadable AFU image. The AFU image's filename is the only parameter:

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

The Acceleration Stack for Intel Xeon CPU with FPGAs 1.0 Beta installation includes the following AFU images in the $DCP_LOC/hw/samples directory:

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

Note: You must have a IP-PCIE/SRIOV license to generate the AFU PR bitstream that is part of the loadable AFU image. For more information, refer to the "Intel FPGA Software Licensing Support" webpage on the Intel FPGA website to get a license.

Related Links

Intel FPGA Software Licensing Support
7.2 OPAE Sample Application Programs

7.2.1 Running the Hello FPGA Example

The hello_fpga sample host application uses the OPAE library to test the hardware in native loopback mode (NLB). Load the Intel PAC with Arria 10 with the nlb_mode_0 AFU image to run this example.

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

1. $ sudo fpgaconf $DCP_LOC/hw/samples/nlb_mode_0/bin/nlb_mode_0.gbs
   
   Note: If you see an "Error enumerating FPGAs: not found" message, ensure that your FPGA Interface Manager version is compatible with your AFU image.

2. Configure the system hugepage to allocate 20, 2-MB hugepages that this utility requires. This command requires root privileges:
   
   $ sudo sh -c "echo 20 > /sys/kernel/mm/hugepages/hugepages-\n   2048kB/nr_hugepages"

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

3. $ cd $DCP_LOC/sw
4. $ tar xf $DCP_LOC/sw/opae-src-0.13.0.tar.gz
5. $ cd opae-0.13.0-1
6. $ export OPAE_LOC=`pwd`
7. Type the following:
   
   $ sudo gcc -o hello_fpga -std=gnu99 -rdynamic -ljson-c -luuid -lpthread -lopae-c -lm -Wl,-path -lopae-c -z noexecstack -z relro -z now -fstack-protector -FPIE -FPIC -pie -O2 -D_FORTIFY_SOURCE=2 -Wformat -Wformat-security -I <path to opae install>/include -L<path to opae install>/lib $OPAE_LOC/samples/hello_fpga.c

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

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

   As a sudo user:

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

8. $ sudo ./hello_fpga

Sample output:

Running Test
Done Running Test

For more information about the hello_fpga example refer to the following files:
Source code located at $OPAE_LOC/samples/hello_fpga.c
AFU register descriptions located in the docs directory: docs/nlb-csrdescription.pdf

Related Links
Documentation Available for the Acceleration Stack for Intel Xeon CPU with FPGAs 1.0 Beta Release on page 31
8 Running the OPAE in a Virtualized Environment

In SR-IOV mode, a host processor uses a physical function (PF) to access management functions. A virtual machine (VM) uses a virtual function (VF) to access the AFU. PR is not available in this 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. Partial reconfiguration (PR) is not available on a virtualized environment. 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. This is the AFU image required to run the example, Running the Hello FPGA Example in a Virtualized Environment, that you can run after setting up the virtualized environment.

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

Related Links

- Getting Started on page 9
- Installing the OPAE Software Package on page 17
8.1 Updating Settings Required for VFs

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

1. Add `intel_iommu=on` to the kernel command line by updating the grub configuration.
2. Reboot to apply the new grub configuration file.
3. To verify the grub update, run the following command: `$ cat /proc/cmdline`.
   
   Sample output below shows `intel_iommu=on` on the kernel command line.

```
Sample output:
BOOT_IMAGE=/vmlinuz-3.10.0-514.21.1.el7.x86_64
root=/dev/mapper/cl_jarrod--z620--lab-root ro
intel_iommu=on
crashkernel=auto
rd.lvm.lv=cl_jarrod-z620-lab/root
rd.lvm.lv=cl_jarrod-z620-lab/swap
rhgb quiet
```

8.2 Configuring the VF Port on the Host

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

1. `$ export port_path=$(find /sys/class/fpga/intel-fpga-dev.* -maxdepth 1 -follow -iname intel-fpga-port.0)`
2. `$ export link_path=$(readlink -m /$port_path/../)`
3. `$ export pci_path=$link_path/../../`
4. Release the port controlled by the PF using the fpgaport tool:
   
   ```
   $ sudo fpgaport release /dev/intel-fpga-fme.0 0
   ```
5. Enable SR-IOV and VFs. Each VF has 1 AFU Port.
   
   ```
   $ sudo sh -c "echo 1 > $pci_path/sriov_numvfs"
   ```
6. `$ lspci -nn | grep :09c[45]`
   
   Sample output:

   ```
   04:00.0 Processing accelerators [1200]: Intel Corporation Device [8086:09c4]
   04:00.1 Processing accelerators [1200]: Intel Corporation Device [8086:09c5]
   ```
   
   The VF is enabled. lspci shows an additional device number, 09c5. This is the VF device you assign to a VM. The original bus and device numbers for the PF remains 09c4.
   
   Note that the Domain:Bus:Device.Function (BDF) notation for the VF device is: 000:04:00.1
7. Load the vfio-pci driver: `$ sudo modprobe vfio-pci`
8. Unbind the VF device from its driver:
   
   ```
   $ sudo sh -c "echo 0000:04:00.1 > \\n/sys/bus/pci/devices/0000:04:00.1/driver/unbind"
   ```
9. Find the vendor and device ID for the VF device: `$ lspci -n -s 04:00.1`
10. Bind the VF to the vfio-pci driver:
   $ sudo sh -c "echo 8086 09c5 > /sys/bus/pci/drivers/vfio-pci/new_id"

8.3 Running the Hello FPGA Example on Virtual Machine

This section assumes that the Virtual Machine (VM) is setup and connected to the virtual function (VF) device with id 09c5. On the virtual machine, install the Intel FPGA Driver and OPAE Software. Refer to Installing the OPAE Software Package section for instructions.

To test the operation of the NLB mode 0 AFU in a virtualized environment.

1. Configure the system hugepage to allocate 20, 2 MB hugepages that this utility requires. This command requires root privileges:
   $ sudo sh -c "echo 20 > /sys/kernel/mm/hugepages/hugepages-2048kB/nr_hugepages"

2. $ tar xf $DCP_LOC/sw/opae-src-0.13.0.tar.gz
3. $ cd opae-0.13.0-1
4. $ export OPAE_LOC=`pwd`
5. Type the following command:
6. $ sudo ./hello_fpga

Sample output:
   Running Test
   Done Running Test

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

- Source code located at $OPAE_LOC/samples/hello_fpga.c
- AFU register descriptions located in the following directory: docs/nlb-csr-description.pdf

Related Links

- Installing the OPAE Software Package on page 17
- Running the Hello FPGA Example on page 22
- Documentation Available for the Acceleration Stack for Intel Xeon CPU with FPGAs 1.0 Beta Release on page 31
8.3.1 Disconnecting the VF from the VM and Reconnecting to the PF

1. Uninstall the driver on the VM: $ yum remove opae-intel-fpga-driv.x86_64
2. Detach the VF from the VM. On the host machine, unbind the VF PCI device from the vfio-pci driver:
   $ sudo sh -c "echo -n 0000:04:00.1 > /sys/bus/pci/drivers/vfio-pci/unbind"
3. Bind the VF to the intel-fpga driver:
   $ sudo sh -c "echo -n 0000:04:00.1 > /sys/bus/pci/drivers/intel-fpga-pci/bind"
4. Set to 0 VFs and disable SR-IOV:
   $ sudo sh -c "echo 0 > $pci_path/sriov_numvfs"
5. Assign the port to PF using fpgaport tool:
   sudo fpgaport assign /dev/intel-fpga-fme.0 0
A Additional Flag Settings to AFU Makefiles

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

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

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

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

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

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

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

# format string vulnerabilities
CFLAGS+=-Wformat -Wformat-security
```
B Update Flash with Beta FPGA Interface Manager (FIM) Image using `fpgaflash` Tool

**Note:** You must have already followed the steps provided in the "Update the Board Management Controller (BMC) Configuration and Firmware" and "Update Flash with Beta FPGA Interface Manager (FIM) Image using Intel Quartus Prime Programmer" sections.

**Note:** This tool allows you to update the user partition of the flash required for future updates to flash (post Beta).

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

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

**Figure 6. Flash Layout**

- To update the user POF, use command:

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

  **Expected Output:**

  Flash successfully verified
Note: If you have multiple Intel PAC with Arria 10 cards installed, you could choose to flash a specific card by specifying the PCIe bus:device.function, for example:

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

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

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

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

Sample output:

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

If this command does not return one or more devices in its output as shown above, then
a. Reboot to see if PCIe enumeration has assigned a bus.
b. If reboot does not help, follow the steps in the "Update Flash with Beta FPGA Interface Manager (FIM) Image using Intel Quartus Prime Programmer" section to update the flash using Intel Quartus Prime Programmer.

Related Links

- Installing the OPAE Software on page 18
C Documentation Available for the Acceleration Stack for Intel Xeon CPU with FPGAs 1.0 Beta Release

The following documents are included in the dcp_1.0_beta_docs.zip file.

<table>
<thead>
<tr>
<th>Document</th>
<th>Directory Path and Filename</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open Programmable Acceleration Engine (OPAE) C API Programming Guide</td>
<td>GitHub Link</td>
</tr>
<tr>
<td>Open Programmable Acceleration Engine (OPAE) Linux Device Driver Architecture Guide</td>
<td>GitHub Link</td>
</tr>
<tr>
<td>Open Programmable Acceleration Engine (OPAE) Tools Guide</td>
<td>GitHub Link</td>
</tr>
<tr>
<td>Intel Accelerator Functional Unit (AFU) Simulation Environment (ASE) User Guide</td>
<td>GitHub Link</td>
</tr>
<tr>
<td>Accelerator Functional Unit AFU Developers Guide</td>
<td>docs/afu-developers-guide.pdf</td>
</tr>
</tbody>
</table>
## Document Revision History for the Intel Acceleration Stack Quick Start Guide for Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA

<table>
<thead>
<tr>
<th>Document Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2018.01.19</td>
<td>The following updates were made in this version:</td>
</tr>
<tr>
<td></td>
<td>• In the &quot;Building the OPAE Software&quot; section:</td>
</tr>
<tr>
<td></td>
<td>— Added the &quot;make install&quot; step</td>
</tr>
<tr>
<td></td>
<td>— Added the missing dependency Doxygen.</td>
</tr>
<tr>
<td></td>
<td>• Corrected the hugepages setting to run <code>fpgabist</code> example in the &quot;Running FPGA Diagnostics&quot; section.</td>
</tr>
<tr>
<td></td>
<td>• In the &quot;Update the Board Management Controller (BMC) Configuration and Firmware&quot; section:</td>
</tr>
<tr>
<td></td>
<td>— Added steps for when an error is received.</td>
</tr>
<tr>
<td></td>
<td>— Fixed a typo in the command in step 7.</td>
</tr>
<tr>
<td></td>
<td>• In the &quot;Update Flash with Beta FPGA Interface Manager (FIM) Image using Intel Quartus Prime Programmer&quot; section:</td>
</tr>
<tr>
<td></td>
<td>— Corrected typo in step 3.</td>
</tr>
<tr>
<td></td>
<td>— Replaced Intel FPGA Download Cable with &quot;Micro USB cable.</td>
</tr>
<tr>
<td></td>
<td>— Added steps for when the OPAE is built from source instead of rpm install in the &quot;Running the Hello FPGA Example&quot;.</td>
</tr>
</tbody>
</table>

| 2017.12.22       | • Beta version |
|                  | • Updated populated memory on accelerator card in "Introduction to the Acceleration Stack for Intel Xeon CPU with FPGAs" section. |
|                  | • Removed Alpha information from the "Update Flash with Beta FPGA Interface Manager (FIM) Image using Intel Quartus Prime Pro Edition Programmer" section. |
|                  | • Removed the "Installing the BitWare Linux Toolkit" section. |
|                  | • "Flash the Card" and "Running FPGA Diagnostics" are no longer sub-sections of the "Getting Started" section. They have been promoted to main sections listed under the "Installing the OPAE Software" section. |
|                  | • Modified the FIM ID and Sample Output code in the "Building the OPAE Software" section. |
|                  | • Updated the $ sudo gcc command in the "Running the Hello FPGA Example" section. |
|                  | • Updated the $ sudo gcc command in the "Running the Hello FPGA Example on Virtual Machine" section. |
|                  | • Added "Additional Flag Settings to AFU Makefiles" appendix. |
|                  | • Updated "Documentation Available for the 1.0 Beta Release" section. |

| 2017.11.10       | Preliminary version |

| 2017.10.02       | Made the following changes to the user guide: |
|                  | • Replaced the old nomenclature with the new and marketing-approved version. |
|                  | • Corrected minor errors and typos. |
|                  | • Updated the DCP 1.0 Beta Directory Structure figure. |
|                  | • Updated command lines to reflect the current version of the software. |
|                  | • Removed "Building the ifpga Tool" section. |
|                  | • Added "Running FPGA Diagnostics" section. |

*Other names and brands may be claimed as the property of others.*
<table>
<thead>
<tr>
<th>Document Version</th>
<th>Changes</th>
</tr>
</thead>
</table>
| August 2017      | Corrected minor errors in the following topics in this user guide:  
|                  | • Flashing the Card  
|                  | • Configuring the VF Port on the Host  
|                  | • Loading the Accelerator Function into the FPGA  
|                  | • Running the Hello FPGA Example |
| July 2017        | Made the following changes to the user guide:  
|                  | • Corrected the `lspci` command in the Flashing the Card procedure.  
|                  | • Corrected Step 1 and Step 7 in Building the OPAE Software procedure. Also, reordered Steps 3-5.  
|                  | • Fixed minor errors and typos. |
| June 2017        | Made the following changes:  
|                  | • Replaced the OPAE software with the OPAE software. The OPAE utilities to program the AF and test AFUs are different than those that were available with the earlier OPAE software package.  
|                  | • Removed the Developing AFUs chapter. A new AFU Developers Guide is now available as a separate document in `dcp_1.0_beta_docs.zip`.  
|                  | • Expanded the information covering virtualized environments.  
|                  | • Removed the Using the Packager chapter. A new Packager User Guide is now available as a separate document in `dcp_1.0_beta_docs.zip`.  
|                  | • Renamed Standard Mode Non-Virtualized Mode. |
| April 2017       | Made the following changes:  
|                  | • Added support for multiple card configurations.  
|                  | • Added support for the Inspur FPGA Accelerator-F10A card.  
|                  | • Changed to CentOS 7.2 for all development activities. Separate host and development machines are no longer required.  
|                  | • Added figures illustrating OPAE driver in standard and SR-IOV modes.  
|                  | • Renamed the scripts to compile an AFU. Moved the scripts to a subdirectory of the build directory.  
|                  | • Corrected minor errors and typos. |
| February 2017    | Initial release of DCP_0.5 |