1. Intel® Accelerator Functional Unit (AFU) Simulation Environment (ASE) ......................... 3
   1.1. System Requirements ........................................................................................................... 4
   1.2. Setting Up the ASE Environment .......................................................................................... 5
   1.3. Installing the OPAE Software ............................................................................................. 7
   1.4. Simulating the hello_afu AFU using the ASE (Client-Server Mode) .................................... 7
       1.4.1. About the Simulation Log Files ..................................................................................... 12
   1.5. Simulating the hello_afu AFU using the ASE (Regression Mode) ....................................... 12
   1.6. Simulating your Custom AFU ............................................................................................... 13
   1.7. Troubleshooting .................................................................................................................... 15
1. **Intel® Accelerator Functional Unit (AFU) Simulation Environment (ASE)**

The Intel® Accelerator Functional Unit (AFU) Simulation Environment (ASE) is a hardware software co-simulation environment for the Intel Xeon FPGA development kit based on the Direct Programming Interface (DPI).

The ASE provides a transactional model for the Core Cache Interface (CCI-P) protocol and a memory model for the FPGA-attached local memory. The CCI-P transactional model verifies CCI-P protocol correctness. The local memory model provides a simulation model for the dual-memory banks on the Intel Acceleration Stack for Intel Xeon® CPU with FPGAs daughtercard.

The ASE also validates Accelerator Functional Unit (AFU) compliance to the following protocols and APIs:

- CCI-P protocol specification
- Avalon® Memory Mapped (Avalon-MM) Interface Specification
- Open Programmable Acceleration Engine

This document describes how to simulate a sample AFU using the ASE environment. Refer to *Intel Accelerator Functional Unit (AFU) Simulation Environment (ASE) User Guide* for comprehensive details on ASE capabilities and internal architecture.

### Table 1. 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>Intel 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 GX FPGA</td>
<td>PCIe* accelerator card with an Intel Arria 10 FPGA. Contains a FPGA Interface Manager (FIM) that pairs with an Intel Xeon processor over PCIe bus.</td>
</tr>
<tr>
<td>Intel Xeon Scalable Platform 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 Ultra Path Interconnect (UPI).</td>
</tr>
</tbody>
</table>

**Related Information**

*Intel Accelerator Functional Unit (AFU) Simulation Environment (ASE) User Guide*
1.1. System Requirements

- 64-bit Linux operating system. ASE has been tested on CentOS 7.4 with Linux kernel 3.10
- Simulator:
  - 64-bit Synopsys* VCS-MX-2016.06-SP2-1 RTL Simulator
  - 64-bit Mentor Graphics* Modelsim SE Simulator (Version 10.5c)
  - 64-bit Mentor Graphics QuestaSim Simulator (Version 10.5c)
- C-Compiler: gcc 4.7.0 and above
- cmake: version 2.8.12
- GLIBC: version 2.17 or above
- Python: version 2.7 or above
- Intel Quartus® Prime Pro Edition (17.0.0 version)
1.2. Setting Up the ASE Environment

Refer the Intel Acceleration Stack Quick Start Guide for Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA to install and setup the hardware and software components of the Acceleration Stack. Your system must meet the requirements mentioned in the Getting Started section of the Intel Acceleration Stack Quick Start Guide for Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA.

Set the following environment variables before running ASE:

1. For VCS:

```bash
$ export VCS_HOME=<path to VCS installation directory>
$ export PATH=$VCS_HOME/bin:$PATH
```

Figure 1. Installation Directory Structure (VCS)

Make sure your system has a valid VCS license.

2. For Modelsim SE/QuestaSim:

```bash
$ export MTI_HOME=<path to Modelsim installation directory>
$ export PATH=$MTI_HOME/linux_x86_64/:$MTI_HOME/bin/:$PATH
```

Figure 2. Installation Directory Structure (Modelsim/Questa)

Make sure your system has a valid Modelsim SE/QuestaSim license.

3. Ensure that the Intel Quartus Prime Pro Edition has been exported:

```bash
$ echo $QUARTUS_HOME
```

Figure 3. Installation Directory Structure (Intel Quartus Prime)

Make sure your system has a valid Intel Quartus Prime license.

4. Extract `a10_gx_pac_ias_l_0_prq.tar.gz`
5. Create an environment variable for the extracted `a10_gx_pac_ias_1_0_prq` directory:

```bash
$ export DCP_LOC=<path to extracted a10_gx_pac_ias_1_0_prq directory>
```

**Related Information**

Intel Acceleration Stack Quick Start Guide for Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA
1.3. Installing the OPAE Software

Follow these instructions to install the OPAE libraries for ASE. You can skip the steps 1 to 6 if you have already installed OPAE libraries, binaries, include, and ASE libraries as described in Intel Acceleration Stack Quick Start Guide for Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA.

1. cd $DCP_LOC/sw
2. Install shared libraries at location /usr/lib to link your application:
   $ sudo yum install opae-0.13.1-1.x86_64-libs.rpm
3. Install the OPAE header at location /usr/include:
   $ sudo yum install opae-0.13.1-1.x86_64-devel.rpm
4. Install the OPAE provided tools at location /usr/bin (for example: fpgaconf, fpgainfo). For more information regarding the tools, refer to the OPAE tools document.
   $ sudo yum install opae-0.13.1-1.x86_64-tools.rpm
5. Install the ASE related shared libraries at location /usr/lib:
   $ sudo yum install opae-0.13.1-1.x86_64-ase.rpm
6. Verify the installed libraries files.
   $ ls /usr/lib
   Expected result:
   libopae-c-ase.so libopae-c.so libopae-c.so.0 ...

   Extract the OPAE source, the ASE directory within $OPAE_LOC has the required files to run the simulation.

7. $ tar xvzf opae-src-0.13.1.tar.gz
8. $ cd opae-0.13.1
9. $ export OPAE_LOC=`pwd`

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

1.4. Simulating the hello_afu AFU using the ASE (Client-Server Mode)

Client-server mode is used for interactive development and debug. Use the setup_sim.sh and run_app.sh scripts, located in $DCP_LOC/hw/common/scripts, to set up the simulation and run the application. You may run the application multiple times once the simulator is up and running.
1. Change directory to:

```bash
$ cd $DCP_LOC/hw/common/scripts
```

2. General command syntax:

```bash
$ sh setup_sim.sh -a <afu dir> -s <vcs|modelsim|questa> -b <opae base dir> [-r <rtl simulation dir>] [-m <EMIF_MODEL_BASIC|EMIF_MODEL_ADVANCED> memory model]
```

To run the simulation using Modelsim:

```bash
$ sh setup_sim.sh -a $DCP_LOC/hw/samples/hello_afu -s modelsim -b $OPAE_LOC
```

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
<th>Legal Values</th>
<th>Default</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>-a</td>
<td>Path to AFU source</td>
<td>Example: $DCP_LOC/hw/samples/hello_afu</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>-s</td>
<td>Simulator type</td>
<td>vcs, modelsim, questa</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>-b</td>
<td>$OPAE_LOC</td>
<td>$OPAE_LOC</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>-r</td>
<td>Optional directory to build the simulation. If specified, AFU source and ASE work directory is copied here. If not, simulation is built in $OPAE_LOC/rtl_sim</td>
<td>$OPAE_LOC</td>
<td>-</td>
<td>No</td>
</tr>
<tr>
<td>-m</td>
<td>Local Memory Model: selects the simulation model for FPGA private memory. Supported values are EMIF_MODEL_BASIC and EMIF_MODEL_ADVANCED. EMIF_MODEL_BASIC uses a simple system-verilog array to model dual banks of DRAM. EMIF_MODEL_ADVANCED uses an advanced cycle accurate model of the EMIF memory controller. EMIF_MODEL_BASIC is recommended for faster simulations.</td>
<td>EMIF_MODEL_BASIC, EMIF_MODEL_ADVANCED</td>
<td>EMIF_MODEL_BASIC</td>
<td>No</td>
</tr>
</tbody>
</table>
Sample output:

**Figure 5. Messages from Simulation Environment Initialization**

3. Open a new terminal to run simulation of your application using the `$run_app.sh` script.

4. Set `$DCP_LOC` and `$OPAE_LOC` environment variables in new terminal.

5. Change directory to:

   ```bash
   $ cd $DCP_LOC/hw/common/scripts
   ```

6. General command syntax:

   ```bash
   $ sh run_app.sh -a <afu dir> -b <opae base dir> [-i <opae install path>] [-r <rtl simulation dir>]
   ```

To run the simulation using Modelsim:

```bash
$ sh run_app.sh -a $DCP_LOC/hw/samples/hello_afu -b $OPAE_LOC
```
Table 3. Flag Descriptions

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
<th>Legal Values</th>
<th>Default</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>-a</td>
<td>Path to AFU source</td>
<td>Example: $DCP_LOC/hw/samples/hello_afu</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>-b</td>
<td>$OPAE_LOC</td>
<td>$OPAE_LOC</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>-r</td>
<td>Optional directory to build the simulation. If specified, AFU source, log files (including waveforms), and ASE work directory are copied here. If not, simulation is built in $OPAE_LOC/rtl_sim. <strong>Note:</strong> If you specified -r in <code>setup_sim.sh</code>, you must specify -r in <code>run_app.sh</code></td>
<td></td>
<td>$OPAE_LOC/rtl_sim</td>
<td>No</td>
</tr>
<tr>
<td>-i</td>
<td>Optional path to OPAE installation. You must specify the install path if you don’t use the RPM flow. If you are using the RPM flow, the install path is not required.</td>
<td>&lt;custom opae_loc directory&gt;</td>
<td>-</td>
<td>No</td>
</tr>
</tbody>
</table>

(1) If you built OPAE from the source instructed in Building the OPAE Software section of Intel Acceleration Stack Quick Start Guide, provide the installation path. For example: `/home/john/opaeinstall`
Sample output:

**Figure 6. Print Statements from the hello_afu Simulation**

```plaintext
[APP] Starting UMsg watcher ... SUCCESS
[APP] MMIO Read : tid = 0x000, offset = 0x8
[APP] MMIO Read Resp : tid = 0x000, data = 9722d43375b61c66
[APP] MMIO Read : tid = 0x001, offset = 0x10
[APP] MMIO Read Resp : tid = 0x001, data = 850adcc26ceb4b22
Running Test
[APP] Issuing Soft Reset...
[APP] MMIO Read : tid = 0x002, offset = 0x0
[APP] MMIO Read Resp : tid = 0x002, data = 1000010000000000
AFU DFF REG = 1000010000000000
[APP] MMIO Read : tid = 0x003, offset = 0x8
[APP] MMIO Read Resp : tid = 0x003, data = 9722d43375b61c66
AFU ID LO = 9722d43375b61c66
[APP] MMIO Read : tid = 0x004, offset = 0x10
[APP] MMIO Read Resp : tid = 0x004, data = 850adcc26ceb4b22
AFU ID HI = 850adcc26ceb4b22
[APP] MMIO Read : tid = 0x005, offset = 0x18
[APP] MMIO Read Resp : tid = 0x005, data = 0
AFU NEXT = 00000000
[APP] MMIO Read : tid = 0x006, offset = 0x20
[APP] MMIO Read Resp : tid = 0x006, data = 0
AFU RESERVED = 00000000
[APP] MMIO Read : tid = 0x007, offset = 0x80
[APP] MMIO Read Resp : tid = 0x007, data = 0
Reading Scratch Register (Byte Offset=00000080) = 00000000
MMIO write to Scratch Register (Byte Offset=00000080) = 123456789abcdef
[APP] MMIO Write : tid = 0x008, offset = 0x80, data = 0x123456789abcdef
[APP] MMIO Read : tid = 0x009, offset = 0x80
[APP] MMIO Read Resp : tid = 0x009, data = 123456789abcdef
Reading Scratch Register (Byte Offset=00000080) = 123456789abcdef
Setting Scratch Register (Byte Offset=00000080) = 00000000
[APP] MMIO Write : tid = 0x00a, offset = 0x80, data = 0x0
[APP] MMIO Read : tid = 0x00b, offset = 0x80
[APP] MMIO Read Resp : tid = 0x00b, data = 0
Reading Scratch Register (Byte Offset=00000080) = 00000000
Done Running Test
[APP] Closing Watcher threads
[APP] Deallocating UMAS
[APP] Deallocating memory /umas.1076182902209867 ...SUCCESS
[APP] Deallocating MMIO map
[APP] Deallocating memory /mmio.1076182902209867 ...SUCCESS
[APP] Deinitializing simulation session
[APP] Session ended
Took 877,761,729 nsec
```

7. Type CTRL-C from the Simulation Environment Initialization terminal window to close the simulation. You can run the application multiple times while the simulator is still running.

8. You can stop the simulation using command `$ pkill ase_simv`.

**Related Information**

Intel Acceleration Stack Quick Start Guide for Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA
1.4.1. About the Simulation Log Files

- Log files are available in the directory specified in the -r switch used in `setup_sim.sh`.
- To view waveform database, use:
  
  ```
  $ cd <simulation work directory location>
  $ make wave
  ```
- A listing of CCIP transactions is provided in `./work/ccip_transactions.tsv`.

1.5. Simulating the hello_afu AFU using the ASE (Regression Mode)

Use the `regress.sh` script, located in `$DCP_LOC/hw/common/scripts`, to execute the simulator and the application exactly once.

1. $ cd $DCP_LOC/hw/common/scripts
2. Usage:

   ```
   regress.sh -a <afu dir> -s <vcs|modelsim|questa> -b <opae base dir> [-i <opae install path>] [-r <rtl simulation dir>] [-m <EMIF_MODEL_BASIC|EMIF_MODEL_ADVANCED> memory model]
   ```

Table 4. Table Descriptions

<table>
<thead>
<tr>
<th>Flag</th>
<th>Description</th>
<th>Legal Values</th>
<th>Default</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>-a</td>
<td>Path to AFU source</td>
<td>Example: <code>$DCP_LOC/hw/samples/hello_afu</code></td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>-s</td>
<td>Simulator type</td>
<td>vcs, modelsim, questa</td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>-b</td>
<td><code>$OPAE_LOC</code></td>
<td><code>$OPAE_LOC</code></td>
<td>-</td>
<td>Yes</td>
</tr>
<tr>
<td>-r</td>
<td>Optional directory to build the simulation.</td>
<td>If specified, AFU source and ASE work directory is copied here. If not, simulation is built in <code>$OPAE_LOC/rtl_sim</code></td>
<td>-</td>
<td>No</td>
</tr>
<tr>
<td>-m</td>
<td>Local Memory Model: selects the simulation model for FPGA private memory. Supported values are EMIF_MODEL_BASIC and EMIF_MODEL_ADVANCED. EMIF_MODEL_BASIC uses a simple system-Verilog array to model dual banks of DRAM. EMIF_MODEL_ADVANCED uses an advanced cycle accurate model of the EMIF memory controller. EMIF_MODEL_BASIC is recommended for faster simulations.</td>
<td>EMIF_MODEL_BASIC, EMIF_MODEL_ADVANCED</td>
<td>EMIF_MODEL_BASIC</td>
<td>No</td>
</tr>
<tr>
<td>-i</td>
<td>Optional path to OPAE installation(2). You must specify the install path if you don't use the RPM flow. If you are using the RPM flow, the install path is not required.</td>
<td><code>&lt;custom opae_loc directory&gt;</code> Example: <code>/home/john/opaeinstall</code></td>
<td>-</td>
<td>No</td>
</tr>
</tbody>
</table>
Follow these steps from both Client-Server and Regression mode to simulate other AFUs listed below:

**Table 5. AFU Examples**

<table>
<thead>
<tr>
<th>AFU</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nlb_mode_0</td>
<td>nlb_mode_0 demonstrates the memory copy test. The software application is located at $DCP_LOC/sw/opae-&lt;release_number&gt;/sample/hello_fpga.c</td>
</tr>
<tr>
<td>hello_intr_afu</td>
<td>hello_intr_afu demonstrates the user interrupt feature in ASE.</td>
</tr>
<tr>
<td>dma_afu (only Client-Server mode)</td>
<td>dma_afu demonstrates a DMA Basic Building Block for host to FPGA, FPGA to host and FPGA to FPGA memory transfers. When simulating this AFU, the buffer size used for DMA transfer is intentionally kept small to keep the simulation time reasonable.</td>
</tr>
</tbody>
</table>

**Related Information**
- Simulating the hello_afu AFU using the ASE (Client-Server Mode) on page 7
- Simulating the hello_afu AFU using the ASE (Regression Mode) on page 12
- Native Loopback Accelerator Functional Unit (AFU) User Guide
- DMA Accelerator Functional Unit (AFU) User Guide

**1.6. Simulating your Custom AFU**

**Standard Simulation Scripts**

To simulate your custom AFU, Intel recommends that you copy an existing AFU from the release package and modify it as necessary.

**Figure 7. Sample Directory Structure for Standard Simulation Scripts**

```
<afu_name> (Example: $DCP_LOC/hw/samples/dma_afu)
├── _bin
├── _hw
└── _rtl (RTL source files)
```

ASE automatically scans and includes RTL source files under `<afu_name>/hw/rtl` during simulation.

**Custom Simulation Scripts**

Custom simulation scripts can be placed under `<afu_name>/hw/sim`. You need custom simulation scripts if you don’t want ASE to include every RTL source file placed under `<afu_name>/hw/rtl`. You may need custom simulation scripts if your AFU includes Platform Designer systems.

---

(2) If you built OPAE from the source instructed in *Building the OPAE Software* section of *Intel Acceleration Stack Quick Start Guide*, provide the installation path. For example: `/home/john/opaeinstall`
Figure 8. Sample Directory Structure for Custom Simulation Scripts

```
<afu name> (Example: $DCP_LOC/hw/samples/dma_afu)
  └┬─────|──────┐
     │     │     │
     │     │     │
  →   └┬──┐    │    ┌──┐
     │   │    │    │    │
     └─┬─┘    │    │    │
          └─┬─┘    │    │
             └─┬─┘    │    │
                  └─┬─┘    │
                      └─┬─┘
                           │
                           │
                           │
                           │
                           │
```

Intel recommends you to follow $OPAE_LOC/samples/dma_afu if you want to use custom simulation scripts.

Intel recommends to run the simulation using either setup_sim.sh, run_app.sh or regress.sh under $DCP_LOC/hw/common/scripts. If your AFU uses a custom simulation script, the standard simulation script will automatically find and invoke it. The custom simulation scripts must reside at <path to afu>/hw/sim, if you want the standard simulation script to automatically find and invoke them.

The sim_common.sh under $DCP_LOC/hw/common/scripts provides a handy collection of bash functions for ASE setup. For more information about ASE scripts, refer to Intel Accelerator Functional Unit (AFU) Simulation Environment (ASE) User Guide. If the AFU consists of one or more Platform Designer systems, use the simulation scripts available under dma_afu as a reference.

If you choose to write custom simulation scripts instead of using the bash routines in sim_common.sh, you must ensure to define the following text macros for the simulation to work properly:

```
+define INCLUDE_DDR4
+define DDR_ADDR_WIDTH=26
```

You can add these text macros in the ase_sources.mk file.

- **Example (VCS):**
  ```
echo "SNPS_VLOGAN_OPT+= +define INCLUDE_DDR4 +define DDR_ADDR_WIDTH=26" >> ase_sources.mk
  ```

- **Example (Modelsim/Questa):**
  ```
echo "MENT_VLOG_OPT += +define INCLUDE_DDR4 +define DDR_ADDR_WIDTH=26" >> ase_sources.mk
  ```

For Modelsim/Questa, you must suppress the following error message codes:

- 3485
- 3584

Add the error codes to suppress ase_sources.mk file using the following:

```
echo "MENT_VLOG_OPT += -suppress 3485,3584" >> ase_sources.mk
echo "MENT_VSIM_OPT += -suppress 3485,3584" >> ase_sources.mk
```

**Note:** The ASE memory model supports configurable address width for FPGA local memory. Supported address widths are 26 bits (8G) and 27 bits (16G). Default configuration is 26 bits (8G). You can customize the address width using the parameter DDR_ADDR_WIDTH.
For example:

- To configure the ASE memory model to 16G configuration in VCS:
  ```
  echo "SNPS_VLOGAN_OPT +define+DDR_ADDR_WIDTH=27" >> ase_sources.mk
  ```

- To configure the ASE memory model to 16G configuration in Modelsim/Questa:
  ```
  echo "MENT_VLOG_OPT += +define+DDR_ADDR_WIDTH=27" >> ase_sources.mk
  ```

**Related Information**

Intel Accelerator Functional Unit (AFU) Simulation Environment (ASE) User Guide

1.7. Troubleshooting

Mitigate the following errors with the provided solution.

**Error messages:**

- # [SIM] An ASE instance is probably still running in current directory !
- # [SIM] Check for PID 28816
- # [SIM] Simulation will exit... you may use a SIGKILL to kill the simulation process.
- # [SIM] Also check if .ase_ready.pid file is removed before proceeding.

**Solution:**

1. Kill zombie simulation processes and temporary files left behind by failed simulation processes or crashes by issuing command `pkill ase_simv`
2. Delete .ase_ready.pid file from the directory specified in the `-r` switch of the `sim_setup.sh` or `regress.sh` command. Default directory is `$OPAE_LOC/rtl_sim`.


<table>
<thead>
<tr>
<th>Document Version</th>
<th>Intel Acceleration Stack Version</th>
<th>Changes</th>
</tr>
</thead>
</table>
| 2018.04.11       | 1.0 Production (supported with Intel Quartus Prime Pro Edition 17.0) | Updated the System Requirements section, directory structure and corresponding filenames.  
  Editorial enhancements. |
| 2017.12.22       | 1.0 Beta (supported with Intel Quartus Prime Pro Edition 17.0) | Updated information in:  
  - Installing the Software  
  - Simulating the `hello_afu` AFU using the ASE (Client-Server Mode)  
  - Simulating the `hello_afu` AFU using the ASE (Regression Mode)  
  - Simulating your Custom AFU |
| 2017.10.02       | 1.0 Alpha (supported with Intel Quartus Prime Pro Edition 17.0) | Added topics: |

*continued*
<table>
<thead>
<tr>
<th>Document Version</th>
<th>Intel Acceleration Stack Version</th>
<th>Changes</th>
</tr>
</thead>
</table>
|                   |                                  | • *Simulating the hello_afu AFU using the ASE (Regression Mode)*  
|                   |                                  | • *Simulating the hello_intr_afu*  
|                   |                                  | • *Simulating the dma_afu*  
|                   |                                  | • *Simulating your Custom AFU*  
|                   |                                  | • *Troubleshooting*  |