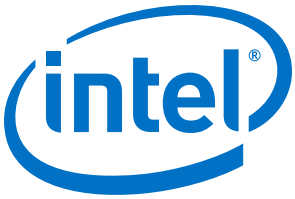


Intel Accelerator Functional Unit (AFU) Simulation Environment (ASE) Quick Start User Guide

Updated for Intel® Acceleration Stack for Intel® Xeon® CPU with FPGAs: **1.1 Production**

Subscribe
Send Feedback

UG-20165 | 2018.08.06
Latest document on the web: **PDF** | **HTML**

Contents

1. Intel Accelerator Functional Unit (AFU) Simulation Environment (ASE) Quick Start User Guide....3
1.1. System Requirements....4
1.2. Setting Up the Environment....5
1.3. Migrating Simulation from Acceleration Stack Version 1.0 to Version 1.1....6
1.4. Simulating `hello_afu` in Client-Server Mode....6
1.4.1. Simulation in Client-Server Mode....7
1.5. Simulating an AFU in Regression Mode....8
1.5.1. Command Line Flags for regress.sh....9
1.5.2. Further Documentation for Regression Mode....10
1.6. AFU Examples....11
1.7. Troubleshooting....11
1.8. Document Revision History for Intel Accelerator Functional Unit (AFU) Simulation Environment (ASE) Quick Start User Guide....12

1. Intel Accelerator Functional Unit (AFU) Simulation Environment (ASE) Quick Start User Guide

The Intel® Accelerator Functional Unit (AFU) Simulation Environment (ASE) is a hardware and software co-simulation environment for the Intel Xeon® Processor with Integrated FPGA.

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 local memory model provides a simulation model for the dual-memory banks on the Intel Programmable Acceleration Card with Intel Arria® 10 GX FPGA.

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 the *Intel Accelerator Functional Unit (AFU) Simulation Environment (ASE) User Guide* for comprehensive details on ASE capabilities and internal architecture.

Note: This document applies to the Intel Acceleration Stack for Intel Xeon CPU with FPGAs version 1.1. For information about version 1.0, refer to the *Intel Accelerator Functional Unit (AFU) Simulation Environment (ASE) Quick Start User Guide for Intel Acceleration Stack 1.0*.

Table 1. Acceleration Stack for Intel Xeon CPU with FPGAs Glossary

Term	Abbreviation	Description
Intel Acceleration Stack for Intel Xeon CPU with FPGAs	Acceleration Stack	A collection of software, firmware and tools that provides performance-optimized connectivity between an Intel FPGA and an Intel Xeon processor.
Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA	Intel PAC with Intel Arria 10 GX FPGA	PCIe* accelerator card with an Intel Arria 10 FPGA. Contains an FPGA Interface Manager (FIM) that pairs with an Intel Xeon processor over PCIe bus.
Intel Xeon Scalable Platform with Integrated FPGA	Integrated FPGA Platform	Intel Xeon plus FPGA platform with the Intel Xeon and an FPGA in a single package and sharing a coherent cache of memory via Ultra Path Interconnect (UPI).

Related Information

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

1.1. System Requirements

System requirements for ASE version 1.1 are as follows:

- 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 or above
- CMake: version 2.8.12 or above
- GNU C Library: version 2.17 or above
- Python: version 2.7
- Intel Quartus® Prime Pro Edition (17.1.1 version)

1.2. Setting Up the Environment

You must set up your simulation environment and install the OPAE software before running the ASE.

1. Set the following environment variables for your simulation software:
   - **For VCS:**

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

     The VCS installation directory structure is as follows:

     ```
     admin  bin  etc     gnu  include      linux    mmc       suse32  vcfca    vgcommon
     amd64  doc  flexlm  gui  install.log  linux64  packages  suse64  verific  vms
     ```

     Make sure your system has a valid VCS license.
   - **For Modelsim SE/QuestaSim:**

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

     The Modelsim/Questa installation directory structure is as follows:

     ```
     avm          gcc32               ieee      LICENSE       mpich2     perl_src              sv_std    uvm-1.1d     vhdlopt_lib
     bin          gcc-4.3.3-linux     ieee_env  linux         _msidata   RELEASE_NOTES         synopsys  uvm-1.2      vhdl_src
     cov_src      gcc-4.3.3-linux_x86_64  ieeepure  linux_x86_64  osver   RELEASE_NOTES.html    tcl       uvmc-2.3.1   vital1995
     docs         gcc-4.5.0-linux     include   mc2_lib       osvvm      RELEASE_NOTES.txt     tcl.fs    uvm_reg-1.1  vital2000
     drill_src    gcc-4.5.0-linux_x86_64  infact  mgc_ams       ovm-2.1.1  rnm                   upf_lib   vco          vital2.2b
     examples     gcc-4.7.4-linux     keyring   modelsim.ini  ovm-2.1.2  std                   upf_src   verilog      vm_src
     floatfixlib  gcc-4.7.4-linux_x86_64  lib  modelsim_lib  pa_lib     std_developerskit     uvm-1.1c  verilog_src  vovl_src
     ```

     Make sure your system has a valid Modelsim SE/QuestaSim license.
   - **For Intel Quartus Prime Pro Edition:**

     ```
     $ export QUARTUS_HOME=<path to Intel Quartus Prime Pro Edition
     installation directory>
     ```

     The Intel Quartus Prime installation directory structure is as follows:

     ```
     adm  common  drivers  dsp_builder  extlibs32  linux64  qdesigns    sopc_builder
     bin  cusp    dspba    eda          libraries  lmf      readme.txt  version.txt
     ```

2. Export:

   ```
   $ export LM_LICENSE_FILE=<Quartus Prime License>
   ```

3. Extract `a10_gx_pac_ias_1_1_pv_dev_installer.tar.gz` as follows:

   ```
   $ mkdir a10_gx_pac_ias_1_1_pv_dev_installer
   $ cd a10_gx_pac_ias_1_1_pv_dev_installer
   $ tar xf <installer_path>/a10_gx_pac_ias_1_1_pv_dev_installer.tar.gz
   $ export OPAE_PLATFORM_ROOT=$PWD
   ```

4. Install OPAE libraries, binaries, include files, and ASE libraries as described in the *Installing the OPAE Software* chapter of the *Intel Acceleration Stack 1.1 Quick Start Guide for Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA*.

Your environment must be set up correctly to configure and build an AFU. In particular, you must ensure that the OPAE SDK is properly installed. OPAE SDK scripts must be on PATH and include files and libraries must be available to the C compiler. In addition, you must ensure that the `OPAE_PLATFORM_ROOT` environment variable is set.

To ensure that the OPAE SDK and ASE are properly installed, in a shell, confirm that the `afu_sim_setup` program is found on the PATH.

Related Information

Installing the OPAE Software
Chapter of the *Intel Acceleration Stack 1.1 Quick Start Guide for Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA*

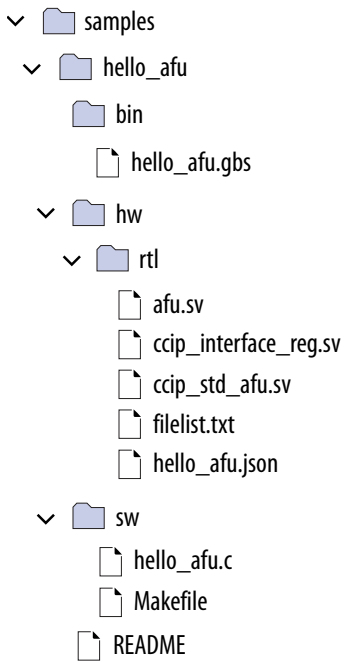
1.3. Migrating Simulation from Acceleration Stack Version 1.0 to Version 1.1

Acceleration Stack v. 1.1 provides the `afu_sim_setup` tool, which supports the inclusion of network interfaces, and better connects the files used for simulation and synthesis. Wherever possible, Intel recommends using `afu_sim_setup` rather than the `setup_sim.sh` and `run_app.sh` scripts.

1.4. Simulating `hello_afu` in Client-Server Mode

The hello_afu example is a simple AFU template that demonstrates the primary CCI-P interface. The RTL satisfies the minimum requirements of an AFU, responding to memory-mapped I/O reads to return the device feature header and the AFU's UUID.

Figure 1. hello_afu Directory Tree



Note: This document uses *<AFU example>* to refer to an example design directory, such as `hello_afu` in the figure above.

The software demonstrates the minimum requirements to attach to an FPGA using OPAE. The RTL demonstrates the minimum requirements to satisfy the OPAE driver and the `hello_afu` example software.

RTL simulation and synthesis are controlled by `filelist.txt`.

To successfully configure and build the AFU samples, your environment must be set up correctly, as described in *Setting Up the Environment*.

Related Information

- Intel Accelerator Functional Unit (AFU) Simulation Environment (ASE) User Guide
- Setting Up the Environment on page 5
- Installing the OPAE Software
  Chapter of the *Intel Acceleration Stack 1.1 Quick Start Guide for Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA*
- Developing AFUs with the SDK
  Chapter of the Accelerator Functional Unit (AFU) Developer's Guide

1.4.1. Simulation in Client-Server Mode

The following example flow introduces the basic ASE scripts. All examples can be simulated with the ASE, except eth-e2e_e10 and eth-e2e_e10.

Simulation requires two software processes: one for RTL simulation and the other to run the connected software. To construct an RTL simulation environment, execute the following in `$OPAE_PLATFORM_ROOT/hw/samples/hello_afu`:

```
$ afu_sim_setup --source hw/rtl/filelist.txt build_sim
```

This command constructs an ASE environment in the `build_sim` subdirectory.

To build and execute the simulator:

```
$ cd build_sim
$ make
$ make sim
```

The simulator prints a message that it is ready for simulation. It also prints a message prompting you to set the `ASE_WORKDIR` environment variable.

Open another shell for software simulation. To build and run the software in the new shell:

```
$ cd a10_gx_pac_ias_1_1_pv_dev_installer
$ export OPAE_PLATFORM_ROOT=$PWD
$ export ASE_WORKDIR=$OPAE_PLATFORM_ROOT/hw/samples/hello_afu/build_sim/work
$ cd $OPAE_PLATFORM_ROOT/hw/samples/hello_afu/sw
$ make clean
$ make USE_ASE=1
$ ./hello_afu
```

Note: The specific pathname for `ASE_WORKDIR` may vary. Use the pathname provided by the simulator prompt.

The software and simulator run, log transactions and exit.

1.4.1.1. Simulation Log Files

The waveform, CCI-P transactions, and simulation log files are stored in the simulation work directory.

To view the waveform database, perform the following steps:

1. Go to the directory in which you executed the `make sim` command.
2. Type:

```
$ make wave
```

A listing of CCI-P transactions is provided in `./work/ccip_transactions.tsv`.

Log files are available in the directory specified in the simulator `build_sim/work` directory.

1.4.1.2. Design Declarations

RTL sources are specified in `$OPAE_PLATFORM_ROOT/hw/samples/<AFU example>/hw/rtl/filelist.txt`, where *<AFU example>* is the example directory as shown in the figure above. In addition to SystemVerilog/VHDL, the AFU's JavaScript Object Notation (`.json`) file is also declared there. The AFU `.json` describes the interfaces required by the AFU. It also holds a UUID to identify the AFU when it is loaded on an FPGA.

The AFU declares that it expects the `ccip_std_afu` top-level interface by setting `afu-top-interface` to `ccip_std_afu` in `hw/rtl/hello_afu.json`. This is the base CCI-P interface with clocks, reset and CCI-P Rx/Tx structures. Other interface options are described in more advanced examples.

The AFU UUID is declared only once: in the `.json` file. The RTL loads the UUID from `afu_json_info.vh`, which is generated automatically by the OPAE scripts. The software loads the UUID from `afu_json_info.h`, which is generated by `sw/Makefile`.

1.4.1.3. Troubleshooting Client-Server Simulation

If the `afu_sim_setup` command fails, confirm that:

- `afu_sim_setup` is on your PATH. It should be in `/usr/bin`, or in `<opae install path>` if you built OPAE from source files.
- You have Python version 2.7 or higher installed.

If you are unable to build and execute the simulator, it is likely that your RTL simulation tool is not installed properly.

When you attempt to build and run the software, if you see an "Error enumerating AFCs" message, you omitted setting USE_ASE=1 on the make command line. The software is searching for a physical FPGA device. To recover, repeat the steps from the `make clean` command.

1.5. Simulating an AFU in Regression Mode

For most AFUs, the `afu_sim_setup` flow and the regression flow are functionally equivalent. You might choose to use the regression flow for one of the following reasons:

- The regression flow triggers both the simulator process and the software process using a single command.
- The regression flow compiles Platform Designer components from Platform Designer source files and includes the generated RTL in the RTL simulation.

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

1. `$ cd $OPAE_PLATFORM_ROOT/hw/common/scripts`
2. Usage:

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

Related Information

- Intel Accelerator Functional Unit (AFU) Simulation Environment (ASE) User Guide
- Simulating hello_afu in Client-Server Mode on page 6

1.5.1. Command Line Flags for regress.sh

Table 2. Flag Descriptions

Flag	Description	Legal Values	Default	Required
`-a`	Path to AFU source[1]	*<valid pathname>*	-	Yes
`-r`	Creates a directory to build the simulation.	*<valid pathname>*	-	Yes
`-s`	Simulator type	`vcs`, `modelsim`, or `questa`	VCS if installed; otherwise, Modelsim or QuestaSim	No
`-p`	Target platform name	Directory containing an OPAE installation	Board specified by `OPAE_PLATFORM_ROOT`	No
`-v`	AFU configuration variant	Depends on AFU selected (`-a` flag).	Depends on AFU selected (`-a` flag).	No
`-i`	Path to OPAE installation	*<valid pathname>*	-	No
`-m`	Local memory model	`EMIF_MODEL_BASIC`, `EMIF_MODEL_ADVANCED`	`EMIF_MODEL_BASIC`	No
`-b`	Path to OPAE source	*<valid pathname>*	-	No (except for NLB)

Target Platform Name -p

The target platform name is used by the Platform Interface Manager (PIM).

`-p` defaults to `discrete` if `OPAE_PLATFORM_ROOT` is undefined or does not specify a valid OPAE release. `discrete` models a generic PCIe card with two banks of local memory

(1) Example: `$OPAE_PLATFORM_ROOT/hw/samples/hello_afu`

Note: Intel recommends setting OPAE_PLATFORM_ROOT as described in *Setting Up the Environment*.

AFU Configuration Variant -v

`regress.sh` looks in the `hw/rtl/filelist*.txt` file to determine the configuration variant. When you specify the variant as `-v <variant>`, `regress.sh` looks in `hw/rtl/filelist_<variant>.txt` for the AFU RTL source files.

The default value of `-v` depends on AFU selected (`-a` flag):

- Native loopback (NLB) AFU: RTL files listed in `hw/rtl/filelist_mode_3.txt`
- Other AFUs: RTL files listed in `hw/rtl/filelist.txt`

Path to OPAE Installation -i

You must specify the install path if you do not use the RPM flow. If you are using the RPM flow, the install path is not required. (2)

Local Memory Model -m

`-m` selects the simulation model for FPGA private memory.

- `EMIF_MODEL_BASIC` uses a simple System Verilog array to model dual banks of DRAM.

  Note: Intel recommends using `EMIF_MODEL_BASIC` for a faster simulation.
- `EMIF_MODEL_ADVANCED` uses an advanced cycle-accurate model of the EMIF memory controller.

Path to OPAE Source -b

Note: This flag is required when simulating the NLB AFU in the regression flow. `-b` is used to locate the application source relative to the OPAE source for NLB.

Related Information

Setting Up the Environment on page 5

1.5.2. Further Documentation for Regression Mode

Related Information

- Native Loopback Accelerator Functional Unit (AFU) User Guide
- Streaming DMA Accelerator Functional Unit (AFU) User Guide
- Developing AFUs with the SDK
  Chapter of the Accelerator Functional Unit (AFU) Developer's Guide

(2) If you built OPAE from the source as instructed in *Building the OPAE Software* section of *Intel Acceleration Stack 1.1 Quick Start Guide for Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA*, provide the installation path. For example:`/home/john/opaeinstall`

1.6. AFU Examples

Table 3. AFU Examples

Each AFU example includes a detailed README file, providing an operational description and notes on how to simulate the design. For a fuller understanding of the simulation process, review the README file in each AFU example.

AFU	Description
`hello_mem_afu`	`hello_mem_afu` demonstrates an AFU that builds a simple state machine to access memory. The state machine is capable of several access patterns to local memory directly attached to FPGA pins, such as DDR4 DIMMs. This memory is distinct from the host memory accessed over CCI-P. The `hello_mem_afu` controller state machine is managed by the host through MMIO requests to CSRs.
`hello_intr_afu`	`hello_intr_afu` demonstrates the user interrupt feature in ASE.
`dma_afu` (3)	`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.

`nlb_mode_0`	`nlb_mode_0` is a CCI-P system demonstrating the memory copy test. The software application is located at `$OPAE_PLATFORM_ROOT/sw/opae-<release_number>/sample/hello_fpga.c`
	Note: To simulate the NLB AFU in the regression flow, you must specify the path to the OPAE source, using the `-b` flag, as follows:
	`$ sh regress.sh -a <afu dir> -r rtl_sim -s < vcs|modelsim|questa > [-i <opae install path>] -b <path to opae source dir>`

1.7. Troubleshooting

If the following error appears during simulation, correct it by following the steps below.

Error Message

```
# [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. Type `pkill ase_simv` to kill zombie simulation processes and remove any temporary files (left behind by failed simulation processes or crashes).
2. Delete the `.ase_ready.pid` file, found in the `$ASE_WORKDIR` directory.

(3) Available only in Client-Server mode

1.8. Document Revision History for Intel Accelerator Functional Unit (AFU) Simulation Environment (ASE) Quick Start User Guide

Document Version	Intel Acceleration Stack Version	Changes
2018.08.06	1.1 Production (supported with Intel Quartus Prime Pro Edition 17.1.1)	Initial production release