Accelerator Functional Unit (AFU) Information Brief
## Contents

1 Accelerator Functional Unit (AFU) Information Brief....................................................... 3

1.1 About this Document................................................................. 3
  1.1.1 Intended Audience............................................................ 3
  1.1.2 Conventions................................................................. 3
  1.1.3 Related Documentation.................................................. 3
  1.1.4 Acronyms........................................................................ 4

1.2 Introduction.................................................................................... 4
  1.2.1 Introduction to the Acceleration Stack........................................ 4

1.3 Custom AFU Development.............................................................. 6
  1.3.1 AFU Integration within the Acceleration Stack......................... 6
  1.3.2 Utilizing Intel FPGA Basic Building Blocks.......................... 6

1.4 AFU Functional Verification............................................................. 7

1.5 AFU In-System Debug.................................................................. 7

1.6 Revision History.......................................................................... 8
1 Accelerator Functional Unit (AFU) Information Brief

1.1 About this Document

This document serves as a hardware developers guide for developing Accelerator Functional Unit (AFU) for the Acceleration Stack for Intel® Xeon® CPU with FPGAs product, hereafter referred to as the Acceleration Stack.

1.1.1 Intended Audience

The intended audience consists of FPGA RTL designers developing AFUs for the Acceleration Stack on the Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA hardware platform.

1.1.2 Conventions

Conventions used in this document include the following:

- # Indicates a command is to be entered as root.
- $ Indicates a command is to be entered as a user.
- This font Filenames, commands, and keywords are printed in this font. Long command lines are printed in this font. Although some very long command lines may wrap to the next line, the return is not considered part of the command – do not enter it.
- < variable_name > Indicates the placeholder text that appears between the angle brackets is to be replaced with an appropriate value. Do not enter the angle brackets.

1.1.3 Related Documentation

Table 1. Item Description

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acceleration Stack for Intel Xeon CPU with FPGAs AFU Simulation Environment User’s Guide</td>
<td>This document provides instructions on how to use the Intel Accelerator Functional Unit (AFU) Simulation Environment (ASE).</td>
</tr>
<tr>
<td>Intel Quartus® Prime Pro Edition Handbook Volume 3: Verification, Chapter 8 Design Debugging with the Signal Tap Logic Analyzer</td>
<td>This documentation describes Signal Tap and its use for general FPGA debug and provides a baseline reference for remote Signal Tap debug of AFUs.</td>
</tr>
</tbody>
</table>
1.1.4 Acronyms

Table 2. Acronyms Description

<table>
<thead>
<tr>
<th>Acronyms</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AFU</td>
<td>Accelerator Functional Unit (AFU)</td>
</tr>
<tr>
<td>FIM</td>
<td>FPGA Interface Manager (FIM)</td>
</tr>
<tr>
<td>FIU</td>
<td>FPGA Interface Unit (FIU)</td>
</tr>
<tr>
<td>API</td>
<td>Application Programming Interface</td>
</tr>
<tr>
<td>ASE</td>
<td>Intel Accelerator Functional Unit (AFU) Simulation Environment (ASE)</td>
</tr>
<tr>
<td>DMA</td>
<td>Direct Memory Access</td>
</tr>
<tr>
<td>DRAM</td>
<td>Dynamic Random-Access Memory</td>
</tr>
<tr>
<td>FIFO</td>
<td>First-In-First-Out</td>
</tr>
<tr>
<td>FPGA</td>
<td>Field Programmable Gate Array</td>
</tr>
<tr>
<td>PR</td>
<td>Partial Reconfiguration</td>
</tr>
<tr>
<td>OPAE</td>
<td>Open Programmable Acceleration Engine (OPAE)</td>
</tr>
<tr>
<td>Intel PAC with Intel Arria 10 GX FPGA</td>
<td>Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA</td>
</tr>
<tr>
<td>IP</td>
<td>Intellectual property</td>
</tr>
<tr>
<td>MMIO</td>
<td>Memory-Mapped Input/Output</td>
</tr>
<tr>
<td>NLB</td>
<td>Native Loopback Adapter</td>
</tr>
<tr>
<td>RD</td>
<td>Read</td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission Control Protocol</td>
</tr>
<tr>
<td>WR</td>
<td>Write</td>
</tr>
<tr>
<td>WrFence</td>
<td>Write Fence</td>
</tr>
<tr>
<td>WrInterface</td>
<td>Write Interface</td>
</tr>
</tbody>
</table>

1.2 Introduction

1.2.1 Introduction to the Acceleration Stack

The Acceleration Stack extends the performance and efficiency of an Intel Xeon processor by seamlessly offloading compute intensive tasks to accelerators in Intel FPGAs.

The Acceleration Stack supports multiple hardware platforms. This guide specifically covers the Intel Programmable Acceleration Card with Intel Arria 10 GX FPGA hardware platform.

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 AFUs to swap in and out of the PR region.

The Open Programmable Acceleration Engine (OPAE) software running on the Intel Xeon processor handles all the details of the reconfiguration process.
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.

**Figure 1. Overview of the Acceleration Stack**

1.2.1.1 Base Knowledge and Skills Prerequisites

This guide assumes the following FPGA logic design-related knowledge and skills:

- **Familiarity with** Partial Reconfiguration (PR) compilation flows, including the Intel Quartus Prime Pro Edition PR flow, concepts of physical and logical partitioning in the FPGA, module boundary best practices, and resource restrictions.

  The physical and logical partitioning of the FIM static region and the PR regions for AFUs has already been done. User AFUs conveniently plug-in to the structure defined by the Acceleration Stack with a well-structured set of standard interface signals to the FIM. This level of abstraction allows developers of AFUs to concentrate on their area of expertise in their end application space by minimizing time and effort on the PR flow itself. The Acceleration Stack provides helper scripts to automate the PR flow during compilation of the AFU RTL for generating a loadable AFU image for use by OPAE. The Acceleration Stack has already laid out the structure, and familiarity with PR flows is a plus for design of an AFU within this predetermined structure.

- **Familiarity and skill with** static timing closure, including familiarity and skill with the Timing Analyzer tool in Intel Quartus Prime Pro Edition, applying timing constraints, Synopsys* Design Constraints (.sdc) language and Tcl scripting, and design methods to close on timing critical paths.

- **Knowledge and skills with** industry standard RTL simulation tools supported by the Acceleration Stack.

- **Knowledge and skill with** the Signal Tap II Logic Analyzer tool in the Intel Quartus Prime Pro Edition software.
1.3 Custom AFU Development

1.3.1 AFU Integration within the Acceleration Stack

To facilitate dynamically loading AFUs, the Acceleration Stack utilizes a partial reconfiguration (PR) scheme. The base configuration contains one or more PR regions for loading AFUs and a static region that provides services and resources to loaded AFUs.

1.3.1.1 The Base Configuration

The base configuration includes the static region and one or more PR region partitions for loading AFUs from OPAE. The static region contains the FPGA Interface Manager (FIM), which provides services to AFUs loaded in PR regions that include a host connection through an abstracted memory mapped protocol over PCIe SR-IOV, a local pool of SDRAM memory, and clock and reset resources. The FIM also provides services to OPAE for dynamically loading AFUs and performing system management tasks like thermal management.

The base configuration is part of the Intel PAC with Intel Arria 10 GX FPGA hardware platform and is not modifiable.

The PR regions in the base configuration are empty – host applications must use OPAE to load AFUs into the PR regions.

The base configuration bitstream is included in the Acceleration Stack installation and initially configures the FPGA at power up from configuration flash residing on the Intel PAC with Intel Arria 10 GX FPGA.

1.3.1.2 The AFU PR Regions

Host software uses OPAE utilities and APIs to load an AFU into a PR region in the base configuration by reference to a loadable AFU image. A loadable AFU image is the combination of an AFU PR bitstream and associated AFU metadata. The AFU PR bitstream is the output from Intel Quartus Prime Pro Edition PR compilation of your AFU RTL design with the base configuration design database provided in the Acceleration Stack installation. The AFU metadata is used to provide OPAE with information on AFU characteristics and operational parameters and is defined in a separate JSON file. An OPAE utility generates the loadable AFU image from the AFU PR bitstream and AFU metadata. It is possible to have several variations of loadable AFU images for a given AFU revision by combining its PR bitstream with unique metadata using an OPAE utility.

The rest of this chapter describes how to design an AFU within the platform and services provided by the FIM, compile an AFU PR bitstream compatible with the base configuration, and generate a loadable AFU image for use by OPAE.

1.3.2 Utilizing Intel FPGA Basic Building Blocks

Intel FPGA Basic Building Blocks are Intel-provided reference designs of common functions and infrastructure blocks that users can instantiate in their AFU.
1.4 AFU Functional Verification

The Intel Accelerator Functional Unit (AFU) Simulation Environment (ASE) supports functional verification of AFU RTL code using host application C code developed for the OPAE API without the need for Intel PAC with Intel Arria 10 GX FPGA hardware. The ASE virtualizes the AFU’s physical link with the host, models certain aspects of the OPAE host memory model, and supports communication between the OPAE host application and supported RTL simulation tools used to emulate the AFU running on actual Intel PAC with Intel Arria 10 GX FPGA hardware.

ASE is useful for verifying your AFU’s interoperability with the rest of the Acceleration Stack using a quick, iterative functional debug environment to minimize time spent in subsequent portions of the AFU development flow that involve more time-intensive steps like PAR and timing closure. ASE also enables a more cost-efficient development environment by removing the dependency on Intel PAC with Intel Arria 10 GX FPGA hardware for early functional debug of AFU interoperability within the Acceleration Stack.

1.5 AFU In-System Debug

The Acceleration Stack provides a remote Signal Tap(1) facility. Use remote Signal Tap to debug an AFU in-system. The Signal Tap II Logic Analyzer included in Intel Quartus Prime Pro Edition allows you to trigger on AFU signal events and capture traces of signals in your AFU design. The remote capability allows for control of trigger conditions and upload of captured signal traces from a networked workstation running the Signal Tap GUI.

Conventional (non-remote) Signal Tap uses the physical FPGA JTAG interface and a Intel FPGA Download Cable II cable to bridge the Intel Quartus Prime Pro Edition Signal Tap application running on a host system with the Signal Tap controller instances embedded in the FPGA logic. With Remote Signal Tap, you can achieve the same result without physically connecting to JTAG, which enables signal-level, in-system debug of AFUs deployed in servers where physical access is limited.

For more information about Signal Tap, refer to the "Related Documentation" section.

In addition to Signal Tap, the remote debug facility in OPAE supports the following in-system debug tools included with Intel Quartus Prime Pro Edition:

- In-system sources and probes
- In-system Memory Content Editor
- Signal Probe
- System Console

Related Links
Related Documentation on page 3

(1) Signal Tap is an in-system logic analyzer that you can use to debug FPGA logic.
## 1.6 Revision History

<table>
<thead>
<tr>
<th>Document Version</th>
<th>Changes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2017.10.02</td>
<td>Initial Release.</td>
</tr>
</tbody>
</table>