CoDeveloper Framework User Guide

Solarflare AOE User Guide Version 1.0.2

Impulse Accelerated Technologies, Inc.

www.ImpulseAccelerated.com

1.0 Table of Contents

1.0 TABLE OF CONTENTS 2
2.0 OVERVIEW 3
2.1. Hardware Description 3
2.1.1. AOE Brief Overview 3
2.1.2. User Framework 4
3.0 BEFORE GETTING STARTED: READ THIS FIRST 5
3.1. Known limitations: 5
3.2. Required Software Tools and Licensing: 5
3.3. Additional Required Files 5
3.4. Required Hardware 6
4.0 DEVELOPMENT SYSTEM SETUP 7
4.1. Development System Setup (Linux) 7
4.1.1. FDK Installation 7
4.1.2. Running Quartus on Linux 7
4.2. Development System Setup (Windows) 7
4.2.1. Install Impulse CoDeveloper v3.70.e.7 or newer 7
4.2.2. Install Solarflare AOE PSP 7
4.2.3. Running Quartus on Windows 7
5.0 PASSTHROUGH: HOW TO BUILD THE FPGA BINARY TUTORIAL 8
5.1. Project File Setup 9
5.2. Generate the Core System 9
5.3. Build the FPGA Binary 10
5.4. Program the FPGA Binary 11
5.4.1. Program the FPGA Binary Using the Programming Cable 11
5.5. Run and Test the FPGA Binary 12

2.0 Overview

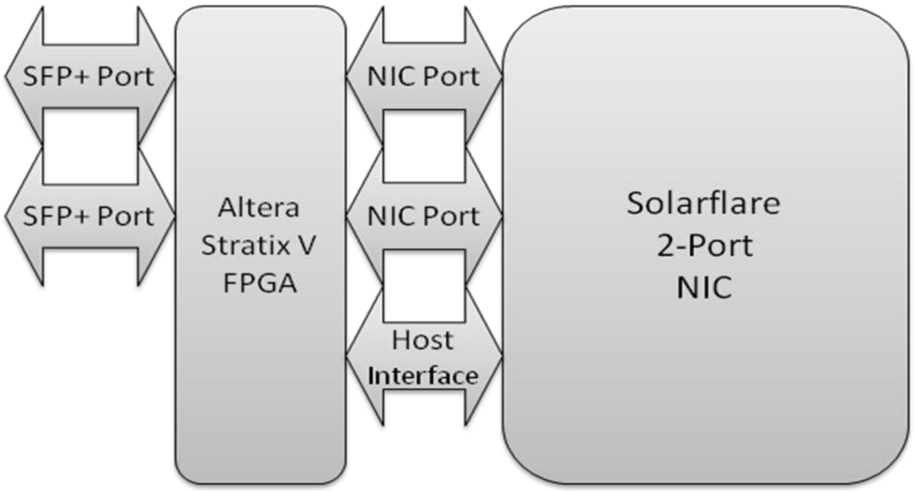
This user guide covers the installation of the CoDeveloper Platform Support Package (PSP) and framework support for the Solarflare Application Offload Engine (AOE) low-latency programmable 10G NIC. Highlights include:

- C-to-FPGA using Impulse C to create hardware modules
- Simplified integration environment using Altera Qsys to connect Impulse C modules directly to 10Gbps Ethernet MACs within the AOE's Stratix V FPGA
- Support for host<->FPGA communication when exporting software via Impulse C API libraries integrating to underlying FDK driver and libraries

2.1. Hardware Description

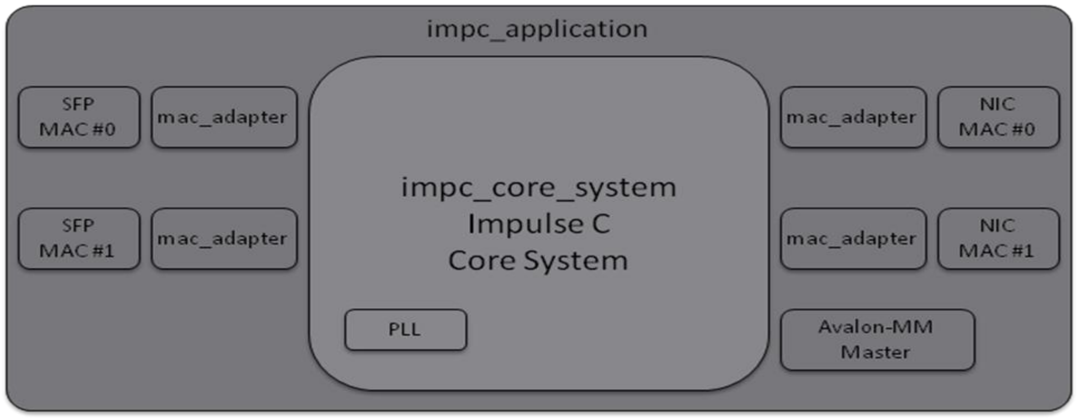
2.1.1. AOE Brief Overview

Below is a simplified view of the Solarflare AOE NIC containing an FPGA. The NIC is configured as a 2-port 10Gbps NIC with a Stratix V FPGA placed in between the NIC and the two external SFP+ ports. A host interface for register access between the NIC and FPGA is also provided. The default "passthrough" FPGA binary connects the SFP+ ports directly to the NIC ports of the AOE effectively making the entire AOE behave as a standard 2-port NIC. Please see the Solarflare AOE and FDK developer's guides and documentation for more details.



2.1.2. User Framework

Below is an overview of the layers that make up the user framework for the FPGA within the AOE.



The outer “impc_application” layer has been created to help simplify the user environment to basic MAC interfaces, a single clock, and access via the Avalon-MM Master. The “mac_adapter” blocks provide both clock domain crossing between the user clock and MAC clocks as well as widen the data streams to 64-bits. Not explicitly shown is the FDK provided basic framework for the AOE referred to as “board_services”, this is what the “impc_application” layer uses to connect to the SFP and NIC MACs and the Avlaon-MM Master.

3.0 Before Getting Started: Read This First

Before getting started, please ensure that you have obtained and installed all the necessary software tools, additional files, and hardware as described below.

3.1. Known limitations:

Below are the current limitations of the provided framework and Solarflare AOE PSP:

1. Current support is for FDK version 1.0 only
2. AOE configured with a Stratix V A5 FPGA
3. 10Gbps Ethernet support only on all NIC and SFP interfaces
4. CPU host communication is purely register based and using software polling
5. Solarflare AOE PSP is tested with CoDeveloper running under Windows only
6. Access via co_memory to the onboard DDR not supported yet

3.2. Required Software Tools and Licensing:

- Impulse CoDeveloper v3.70.e.8 or newer available from: http://www.impulseaccelerated.com/ReleaseFiles/
    - License is either provided as part of a kit or available from Impulse by contacting info@impulsec.com
- Solarflare FDK and subsequent Altera IP licenses that come with it
- Subscription Edition of Quartus v12.1 Service Pack 1 for Windows or Linux (64-bit only)
    - Software and license is either provided as part of a kit or available from your local Altera representative
- Windows (32 or 64-bit) for Impulse CoDeveloper and optional Microsoft Visual Studio
- Windows or Linux (RHEL/CentOS or SUSE) 64-bit for running Quartus
- (optional)Microsoft Visual Studio C++ 2010 Express or full-version for use with CoDeveloper plugin

3.3. Additional Required Files

Upon request by supported users, access to all framework files supporting the Solarflare AOE which include Quartus and Qsys base project files, PSP, and documentation are provided via a customized link on the Impulse website. Please contact support@impulsec.com to request access.

3.4. Required Hardware

The following hardware is required for development:

- Solarflare AOE NIC and an appropriate number of 10Gbps Ethernet SFP+ modules and cables
- Optional, but highly recommended, is an Altera Byte Blaster USB cable and the AOE cable adapter from Solarflare.
- Development system(s):
    - Windows 32 or 64-bit OS for running CoDeveloper
        - >20GB free disk space recommended
        - >2GB RAM recommended
    - Windows or Linux (RHEL/CentOS or SUSE) 64-bit version for running the Quartus II tools:
        - Minimum of 16GB RAM, 20GB recommended
        - >100GB recommended free disk space
        - Minimum of 4 CPUs recommended

4.0 Development System Setup

The recommended development system is for the user to run CoDeveloper under Windows as either the standalone IDE or as the plugin into MS Visual Studio 2010 C++ with the rest of development, including running Quartus, run under Linux. A target system containing the FPGA platform is also recommended to be separate from the development system running any tools

4.1. Development System Setup (Linux)

4.1.1. FDK Installation

The Solarflare FDK needs to be installed on the target system as well as the system Quartus will be run on. Please see the Solarflare supplied documentation for installation.

4.1.2. Running Quartus on Linux

4.1.2.1. Install Quartus II Subscription Edition

1. Please see vendor supplied documentation for installation.
2. Please see vendor supplied documentation for licensing.
3. To ensure that the 64-bit version of the Altera tools are always used, set the environment variable: QUARTUS_64BIT=1

4.2. Development System Setup (Windows)

4.2.1. Install Impulse CoDeveloper v3.70.e.7 or newer

1. Download the latest version 3.70.x installer and installation notes from: http://www.impulseaccelerated.com/ReleaseFiles/

4.2.2. Install Solarflare AOE PSP

The provided “<date>_SolarflareAOE_PSP.zip” contains all the necessary PSP files and is installed by unzipping it to the CoDeveloper installation directory typically “C:\Impulse\CoDeveloper3”.

4.2.3. Running Quartus on Windows

Copying the necessary files from the FDK to run Quartus under Windows should be possible, however is untested.

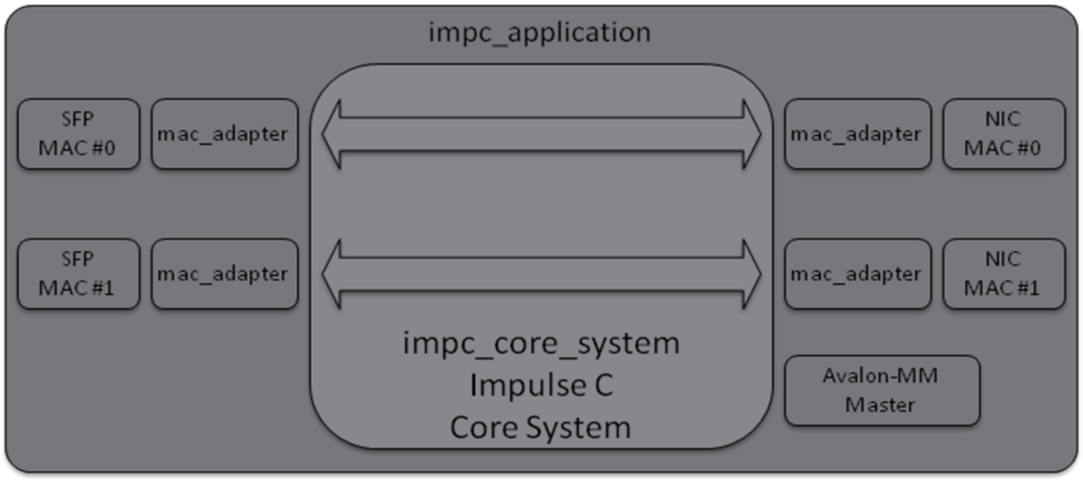
4.2.3.1. Install Quartus II Subscription Edition

1. Please see vendor supplied documentation for installation.
2. Please see vendor supplied documentation for licensing.

5.0 Passthrough: How to Build the FPGA Binary Tutorial

The base Quartus and Qsys project files provided are referred to as “Passthrough” and serves as an example of how to build the FPGA binary ensuring correct system configuration and that all necessary files are present to build the FPGA. The Passthrough example also provides a baseline to ensure basic operation of the AOE before adding a user application.

Operation of the Passthrough is shown below where each SFP #0-1 is simply connected to the corresponding NIC #0-1. This effectively makes the FPGA appear as a “wire” and the AOE appears as a basic 2-port NIC.



5.1. Project File Setup

As part of the additional files for developing on the Solarflare AOE you will have received a link to download the file "<release date>_ sf_impc_base_project.zip". The .zip file is to be unzipped into the directory on the development machine that Quartus will be run and must be located in the same sub-directory within the FDK installation where "example_project" exists. Once unzipped the following directory structure will be present:

```
<fdk install dir>\sf_impc_base_project
```

NOTE: Ensure there are no spaces (' ') in the directory path chosen to avoid potential path issues with any of the tools.

The 'sf_impc_base_project' directory contains the Quartus project and is the only directory intended to be modified by the end user. The 'sf_impc_base_project' directory may be copied and renamed for different builds of the FPGA, however must always remain within the <fdk install dir> because relative paths are used to reference files in the other directories present.

5.2. Generate the Core System

Start Quartus II 64-bit normally for the operating system being used and open the Quartus project file 'sf_impc_base_project\sf_impc_base_project.qpf'. To generate the Core System in Qsys:

1. Start Qsys from Tools->Qsys
2. When the "Open" dialog appears, select the "impc_core_system.qsys" file and click the "Open" button. Once opened, no errors must be present.
3. Now generate the HDL files for the Core System by clicking on the top-right "Generation" tab, and then clicking the lower-left "Generate" button.
4. At this point the impc_core_system HDL has been generated for the Qsys system into the same named directory and no errors must be present.

Notes:

- To preserve the pass-through logic, do not make any changes to the Qsys system at this time. How to insert an Impulse C module will be described in another tutorial.
- These steps must be repeated each time a change to the 'impc_core_system.qsys' Qsys project is modified.

5.3. Build the FPGA Binary

Start compilation of the FPGA from within Quartus by selecting Processing->"Start Compilation". Viewing the "Tasks" window will show the progress of building the FPGA through the "Analysis & Synthesis", "Fitter", "Assembler", and "TimeQuest Timing Analysis" phases. No errors should be reported at any time during compilation.

When complete, the FPGA binary will appear as 'sf_impc_base_project.sof' ready to be programmed into the FPGA using the Byte Blaster cable.

Notes:

- If an error does appear, it will typically appear during the first ~10 minutes of the "Analysis & Synthesis" phase and indicates a file was not found correctly. Please verify all the files were unzipped correctly into the correct location and try again.

5.4. Program the FPGA Binary

The FPGA binary may be programmed into the AOE in one of two ways: Command line utility to program the flash and then rebooting the system (see AOE documentation for more details) or using an Altera Byte Blaster programming cable which is recommended and described here.

5.4.1. Program the FPGA Binary Using the Programming Cable

Programming the FPGA on the AOE may be done using the Altera Byte Blaster USB cable as follows:

1) Connect the Byte Blaster USB cable from the host running Quartus to the AOE. The JTAG connector is located at the top of the card and requires an adapter to plug into the AOE.
2) Start the programmer from Quartus using Tools->Programmer
    a. "Hardware Setup" should already show the USB Byte Blaster cable
    b. "Auto Detect" will identify the FPGA, specify "5SGXMA5K" when prompted for the specific device found.
    c. Select the FPGA "5SGXMA5K" device:
        i. Right-click and select "Change File"
        ii. Browse to select either the 'sf_impc_base_project.sof' file
        iii. Check the box for "Program/Configure"
    d. Program device by clicking the "Start" button.
    e. Save programmer configuration using File->Save

5.5. Run and Test the FPGA Binary

With the FPGA programmed with the base pass-through project, ports #0-1 of the AOE will behave as a standard 10Gbps Ethernet ports. To test:

1) Insert a 10G SFP+ module into both SFP+ ports and loop the two ports to each other or connect directly to a network
2) If not already, start the Solarflare driver for the AOE following the steps in the AOE FDK guide which are summarized here:
    a. Change to the FDK software directory: cd <fdk install dir>
    b. Set FDK_PATH via the command: export FDK_PATH $(pwd)
    c. If AOE libraries have not been installed into the normal system path (e.g. /usr/lib64), then set LD_LIBRARY_PATH to point to where built .so's exists using the command: export LD_LIBRARY_PATH=$FDK_PATH/libs/$(uname -m):$LD_LIBRARY_PATH
    d. If not already done yet, build and load the drivers from the 'openonload' installation with AOE driver. Summary of steps from the openonload installation directory and AFTER the AOE driver has been installed there too:
        i. Build (done only once): sudo ./scripts/onload_build
        ii. Load: sudo ./scripts/onload_tool reload
    e. Configure Ethernet ports as appropriate
        i. On the test system they appeared as "eth4" and "eth5" and were minimally configured using the commands:
            1. sudo ifconfig eth4 192.168.4.62
            2. sudo ifconfig eth5 192.168.5.62

    NOTE: Steps above will need to be repeated each time the host is rebooted
3) Ping and other traffic will flow in and out through both ports of the AOE normally as if they were connected directly to the network. Using ping, Wireshark, and/or other network commands and tools should be used at this time to verify that both ports are passing traffic as expected.