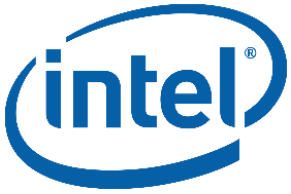

BDW + FPGA

Beta Release 5.0.2
Asynchronous CCI-P Shim

21-Jul-16
Document Version 1.0

Notices and Disclaimers

No license (express or implied, by estoppel or otherwise) to any intellectual property rights is granted by this document.

Intel disclaims all express and implied warranties, including without limitation, the implied warranties of merchantability, fitness for a particular purpose, and non-infringement, as well as any warranty arising from course of performance, course of dealing, or usage in trade.

This document contains information on products, services and/or processes in development. All information provided here is subject to change without notice. Contact your Intel representative to obtain the latest forecast, schedule, specifications and roadmaps.

The products and services described may contain defects or errors known as errata which may cause deviations from published specifications. Current characterized errata are available on request.

Copies of documents which have an order number and are referenced in this document may be obtained by calling 1-800-548-4725 or by visiting **www.intel.com/design/literature.htm**.

Intel, the Intel logo, Xeon, and Intel QuickAssist Technology are trademarks of Intel Corporation in the U.S. and/or other countries.

*Other names and brands may be claimed as the property of others.

Updates

This document belongs to the group of documents provided for the BDW + FPGA product.

Identify the latest copy by the date printed in the footer on each page.

Questions and Feedback

Intel solicits and appreciates feedback. Input should be provided through Intel® Premier Support (IPS). Customers need to ensure IPS access by working with their respective Account Manager/ FAE.

Revision History

Date	Revision Doc	Modifications
15-JUN-16	5.0.2 v1.0	Initial

Contents

Figures

Tables

About this Document

This document is a description of the CCI-P compliant Asynchronous clock-crossing shim

Intended Audience

RTL developers, AFU designers using the BDW + FPGA platform

Conventions

Conventions used in this document include the following:

#	preceding a command indicates the command is to be entered as root.
$	indicates a command is to be entered as a user.
`This font` `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

Related Documentation

Item	Description
BDW + FPGA Beta Release 5.0.2 Read This First	This document summarizes the available documentation and provides hints about how users might navigate through it.
BDW + FPGA Beta Release 5.0.2 Release Notes	This document lists the key features, limitations, changes from the previous release, and possible future changes.
BDW + FPGA Beta Release 5.0.2 Software Installation Guide	This document lists the software prerequisites needed by the AAL SDK and provides instructions on how to install the AAL SDK
BDW + FPGA Beta Release 5.0.2 AFU Simulation Environment User's Guide	This document provides instructions on how to use the Accelerated Functional Unit (AFU) Simulation Environment (ASE).
BDW + FPGA Beta Release 5.0.2 Software Architecture Guide	This document presents the rationale behind AAL and the concepts upon which AAL is based.
BDW + FPGA Beta Release 5.0.2 Programmer's Guide	This document shows how the concepts described in BDW + FPGA Beta Release 5.0.2 Software Architecture Guide can be implemented in code. It does not assume the existence of an AFU.
BDW + FPGA Beta Release 5.0.2 Core Cache Interface (CCI-P) Specification	This document describes the Core Cache Interface (CCI-P) specification which is the interface between the Accelerated Function Unit (AFU) and the .FPGA Interface Unit.
BDW + FPGA Beta Release 5.0.2 Sample Programs Guide	This document describes how to run the sample programs provided with Release 5.0.2 of the BDW + FPGA Accelerator Abstraction Layer and the BDW + FPGA platform.
BDW + FPGA Beta Release 5.0.2 How to Build, Load, and Debug Bitstreams	This document describes how to rebuild one of the provided bitstreams, load it, and debug it. It also describes how to modify this procedure for a custom bitstream.

Glossary

Acronym	Expansion	Description
AAL	Accelerator Abstraction Layer	A set of runtime and software development tools that facilitate the deployment of systems consisting of a collection of non-uniform, asymmetric compute resources. The AAL SDK is the AAL Software Development Kit.
AFU	Accelerated Function Unit	Hardware Accelerator implemented in FPGA logic that accelerates or intends to accelerate an application kernel
ASE	AFU Simulation Environment	The AFU Simulation Environment (ASE) provides a consistent transaction level hardware interface and software API that allows users to develop production-quality AFU RTL and software host applications that can then be run on the real FPGA system without modification.
CA	Caching Agent	A Caching Agent (CA) makes read and write requests to the coherent memory in the system. It is also responsible for servicing snoops generated by other agents in the system.
CCI-P	Core Cache Interface	CCI-P is the hardware-side signaling interface between the Accelerated Function Unit (AFU) and the FPGA Interface Unit (FIU).
DPI	Direct Programming Interface	A set of features in SystemVerilog that allows export/import of parameters to/from a C function
FPGA	Field Programmable Gate Array	http://en.wikipedia.org/wiki/Fpga
IPC	Inter-Process Communication	Refers to constructs in Linux-like shared memory (**`/dev/shm`**) and message queues (**`/dev/mqueue`**); these are leveraged for ASE core functionality.
NLB	Native Loopback Adapter	A sample AFU

1 Introduction

Asynchronous CCI-P Shim is a Basic Building Block (BBB) that can be used to attach Accelerator Functional Units (AFUs) that do not operate at the CCI-P Interface's clock (pClk) of 400 Mhz. It can be considered to be a CCI-P compliant clock-crossing bridge.

This building block has been tested with Quartus 15.1.2 patch 220 and the 5.0..2 Beta System Release.

Throughout this document, the Asynchronous CCI-P module is be referred to as **`ccip_async_shim`**

This BBB package consists of three directories:

`HW`	Contains RTL files
`par`	Contains place-and-route constraints (SDC) files
`Samples`	Contains samples on how to use **`ccip_async_shim`** in an AFU design.

2 Design

The **`ccip_async_shim`** consists of a number of Asynchronous FIFOs instantiated from Altera's IP library as shown in Table 1. This Dual Clock FIFO component is called **`dcfifo`**. Also, refer to Figure 1.

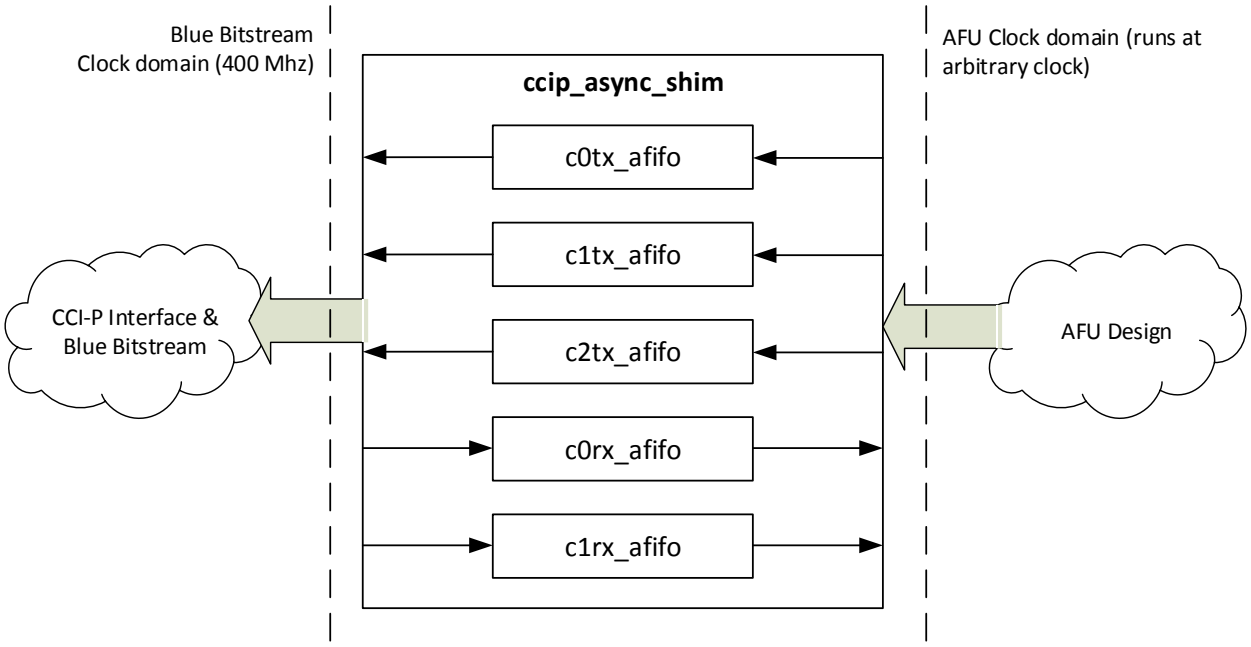

Figure 1: The Async Shim Resides between the AFU and CCI-P

Table 1:Description of Async CCI-P Instantiations

Instantiation	Operation	Description
C0tx_afifo	Read Requests	Write @ **afu_clk** Read @ **bb_clk** Carries Read Requests (header, valid bit) Back pressure signal (**C0TxAlmFull**) is connected to **c0tx_afifo** Write count Note: Read Responses for Multi-line requests may generate multiple 1, 2, or 4 responses.
C1tx_afifo	Write Requests Write Fence Request Interrupt Request	Write @ **afu_clk** Read @ **bb_clk** Carries Header bit, Valid bit and data Back pressure signal (**C1TxAlmFull**) is connected to **c1tx_afifo** Write count
C2tx_afifo	MMIO Read Response	Write @ **afu_clk** Read @ **bb_clk**. Takes in MMIO Header, valid bit and MMIO read data No backpressure. This is not designed for
C0rx_afifo	MMIO Write Request MMIO Read Request Read Responses	Write @ **bb_clk** Read @ **afu_clk**. No backpressure on this channel Carries Data, valid bits and header Note that: Read responses may be multi-line and AFIFO must be deep enough to accommodate for this.
C1rx_afifo	Write Response Write Fence Response Interrupt Response	Write @ **bb_clk** Read @ **afu_clk**. No backpressure Carries Header and Valid bit

3 Configuration and Usage

Note	CCI-P Asynchronous Shim (**ccip_async_shim**) must be sized appropriately before use. Failure to do so might cause transactions to be dropped. Customers must analyze the production and consumption behavior of the AFU before sizing the BBB to requirements

ccip_async_shim exposes the following switches for sizing as shown in Table 2.

Table 2: Verilog Switches

Verilog Switch	Description	Default
C0TX_DEPTH_RADIX	Controls depth of **c0tx_afifo**, Depth = $2^{C0TX\_DEPTH\_RADIX}$	8 (Depth = 256)
C1TX_DEPTH_RADIX	Controls depth of **c1tx_afifo**, Depth = $2^{C1TX\_DEPTH\_RADIX}$	8 (Depth = 256)
C2TX_DEPTH_RADIX	Controls depth of **c2tx_afifo**, Depth = $2^{C2TX\_DEPTH\_RADIX}$	8 (Depth = 256)
C0RX_DEPTH_RADIX	Controls depth of **c0rx_afifo**, Depth = $2^{C0RX\_DEPTH\_RADIX}$	10 (Depth = 1024)
C1RX_DEPTH_RADIX	Controls depth of **c1rx_afifo**, Depth = $2^{C1RX\_DEPTH\_RADIX}$	10 (Depth = 1024)

3.1.1 Other Debug/Pipelining Switches

- DEBUG_ENABLE: Enables an array of counters that maintain a running tally of incoming and outgoing CCI-P transactions.
  - When in simulation, these counters may be observed in VCS/Mentor Waveform viewers
  - When running In-System, these counters may be connected to Signaltap and a running tally can be observed there.
- ENABLE_EXTRA_PIPELINE: Enables an extra pipeline stage in the read path of the FIFO, potentially allowing for some timing improvement at the expense of adding extra latency.

3.1.2 Error Vector:

- An error vector **async_shim_error** (5 wires long, one each for each Altera dcfifo) checks for some common conditions that may cause malfunctions. Please note, this does not detect all possible error conditions. Note that this error vector cannot be read from software and is readable only through Signaltap or Simulation.

3.1.3 Important Design Considerations:

- If Read Requests are expected to be multi-line, care must be taken to allocate enough depth to c0rx_afifo. This is because each read request can generate 1, 2 or 4 read responses. The default setting for c0rx_afifo is 4-times c0tx_afifo. Although, customers must make their own design decisions by analyzing production and consumption rates of the AFU.
- It is possible to simulate BBBs, as they may be considered as components of a green-bitstream in ASE. For certain error conditions, an assertion warning statement may be generated for some erroneous conditions.

- When building bitstreams with **`ccip_async_shim`**, the Quartus project must include the SDC file available inside the **`par`** directory. A sample QSF addition file and import settings are also included as a reference.
- Also, see samples supplied along with the BBB to see how the module can be instantiated and integrated with AFU designs.