

**FDAS PCIe Hard IP Macro Design Specification for Intel Agilex F Implementation**

FDAS\_PCIE\_HARD\_IP\_MACRO\_DS Revision 3 Draft C

Classification:

Document type:

Date:  2023-03-02

Status:

© Copyright 2023 SKA Observatory.

  This work is licensed under a [Creative Commons Attribution 4.0 International License](http://creativecommons.org/licenses/by/4.0/)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| *Role* | *Name* | *Designation* | *Affiliation* | *Signature* | *Date* |
| Author | Martin Droog | FPGA Design Engineer | SKAO |  | 02/03/2023 |
| Owner | Ben Stappers | Head of SKA PSS Team | SKAO |  |  |
| Approver | Ben Stappers | Head of SKA PSS Team | SKAO |  |  |
| Released by | Lina Levin Preston | SKA PSS Team Scrum Master | SKAO |  |  |

|  |  |  |
| --- | --- | --- |
| Document History | | |
| Issue | Date | Comments |
| Issue 2 Draft A | 05/04/2022 | First Issue for review.  Comments from Prabu Thiagaraj and Atul Ghalame, and responses from Martin Droog:-  1) Pinouts for PCIe and DDR are given in the respective documents. It will be helpful to add the reference board schematics source (possibly a link to the intel schematics for the board or some file containing the pinouts). Once we have the board, we can perhaps upload relevant intel documents etc., with these details so that they will all be in one place.  **M. Droog Response: The Intel Agilex F Board schematics shall be referenced in this document.**  2) References to the PCIe documentation etc. may be added to the document..  **M. Droog Response: The Intel Agilex PCIe User Guide reference shall be added to this document.** |
| Issue 2 Draft B | 28/04/2022 | * Intel Agilex F Board schematics reference added * Intel Agilex PCIe User Guide reference added |
| Issue 2 Draft C | 23/04/2022 | Trial builds of the FDAS FPGA showed that logic was missing from the completed build, notably the DSP blocks in HSUM and the FFT Filter store RAMs in CONV. Investigations with different builds showed that the problem centred on the PCIe Hard IP Macro’s interface supporting the MC configuration of the FDAS FPGA. The build in PI13 that was used for the performance study had a Transparent Avalon Block added to this port in Intel Quartus Prime Platform Designer, and this block had been removed in PI14. The reason for removal was that it was not known how the Intel Quartus Prime software will select the 22 address bits needed for the PCIF module from the 37 bits the PCIe Hard IP Macro (the upper 15 bits contain Physical and Virtual Machine numbers which FDAS is not interested in).  However the removal of this Transparent Avalon Block appears to be the root cause of the missing logic in the trial FDAS build.  Until this is understood the Transparent Avalon Block for the MC configuration interface shall be restored to allow builds to progress. The updated PCIF module will not be affected by this change as its response[1:0] and write\_response\_valid output signals will simply not used. |
| Issue 2 Draft D | 26/05/2022 | Following on from Issue 2 Draft C the Avalon Transparent block for the MC configuration interface now supports the write\_response\_valid and response[2:0] signals. |
| Issue 2 Draft E | 28/06/2022 | Minor typo corrections.  Add step to rename the interfaces of the Transparent Avalon Block for DMA.  Updated PCIe Hard IP Macro ports for Intel Quartus Prime Ver 22.2. |
| Issue 2 Draft F | 05/07/2022 | Further updates to PCIe Hard IP Macro ports for Intel Quartus Prime Ver 22.2. |
| Issue 2 Draft G | 13/07/2022 | Further updates to PCIe Hard IP Macro configuration for Intel Quartus Prime Ver 22.2. |
| Issue 2 Draft H | 14/07/2022 | RESET\_RELEASE IP added to drive ninit\_done |
| Issue 3 Draft A | 25/10/2022 | Increased the number of DMA access interfaces to DDR SDRAM from two to four. |
| Issue 3 Draft B | 24/01/2023 | * Removed Burstcount from the Transparent Avalon Block supporting the PIO interface. * PIO Address Space should be 32Mbyte – 25bits, not 4Mbyte – 22bits * All Transparent Avalon Blocks set to WORD addressing on inputs and outputs. * Changes to improve PCIe bandwidth:- * Transparent Avalon block for the H2D DMA modified to support a wait\_request\_allowance of 16 to match that of the PCIe Hard IP Macro. The DDRIF2 module needs to support this to be compatible. * Transparent Avalon block for the D2H DMA modified to support 32 pending reads to match that of the PCIe Hard IP Macro. The DDRIF2 module already supports this. |
| Issue 3 Draft C | 02/03/2023 | Updated for the Intel Quartus Prime Version 22.4 IP |

**Table of Contents**

1 Introduction 5

2 Place in the System 6

3 Functional Specification 8

3.1 PCIe Hard IP Macro Settings in PLATFORM DESIGNER 11

4 Design Requirement Tags 24

5 Interface Specification 27

6 MCI Memory Mapped Interface 33

7 Design Parameterisation 34

8 F Series Development Card External PCIe Pinout 35

9 References 37

10 Abbreviations and Acronyms 38

# Introduction

This document captures the requirements for the Intel Agilex PCIe Hard IP macro used in the FDAS FPGA.

The PCIe Hard IP macro is required to:-

* Support the configuration of the FDAS FPGA via the PCIe interface, and to provide status information back to the PC/Computer via the PCIe interface. This shall use single 32-bit word PCIe Transaction Layer Packets (TLPs).
* Support the transfer of freq-bin samples from the PC/ Computer to the DDRIF2 module so that they can be stored in external DDR SDRAM. This shall use a DMA (Direct Memory Access) transfer technique since a large amount of data (222 freq-bin samples, each of 64-bits) has to be transferred in as short a time as possible.
* Support the reading of the FOP (Filter Output Plane) via three DDRIF2 modules from three external DDR SDRAMs to the PC/Computer for diagnostic support. This shall use a DMA transfer technique since a large amount of data (85 x 222 samples, each of 64-bits) has to be transferred.

The Agilex PCIe Hard IP macro shall be created using the Intel/ Altera Quartus Prime Platform Designer Version 21.3 environment.

The PCIe Hard IP macro shall include an internal DMA controller to support the DMA transfers to/from the DDRIF2 modules.

The number of PCIe lanes shall be determined by the host system. In the case of the FDAS implementation the number of PCIe lanes shall be fixed at 16 using the “Gen 4” generation.

The System Clock “CLK\_SYS” shall be 350MHz.

The clock frequency “CLK\_PCIE” for the PCIe internal interfaces and the width of the DMA data bus is dependent on the PCIe Generation and the number of PCIe lanes. The selection for FDAS is highlighted in the table below:-

|  |  |  |  |
| --- | --- | --- | --- |
| PCIe Lanes | PCIe Generation | CLK\_PCIE | Internal DMA Data Bus Width |
| x8 | Gen 4 | 250MHz | 256 bits |
| x16 | Gen 4 | 350MHz | 512 bits |

Table ‑ : PCIe Clock Frequency

For the FDAS implementation in Intel Agilex F the PCIe interface shall be Gen 4 x 16 lanes and hence the CLK\_PCIE frequency shall be 350MHz with internal DMA data bus width of 512 bits.

This document relates to the Intel Quartus Prime Version 21.3 based flow for IP generation. The Intel Agilex PCIe with DMA is described in an Intel document [Ref: Agilex PCIe User Guide].

# Place in the System

The Intel Agilex PCIe Hard IP Macro’s place in the system is shown highlighted in the figure below:-

**DDRIF2 #2**

**DDRIF2 #2**

PCIe

**CLD**

**CTRL**

FDAS Control

CLD\_DONE

**CONV**

data[63:0]

ready

**HSUM**

**FDAS**

CONV\_DONE CONV\_FFT\_REDAY

valid

**PCIE HARD IP MACRO (INTEL IP)**

**MCI\_TOP**

CLD\_DM\_TRIGGER

CLD\_ENABLE

CLD\_PAGE[31:0]

CONV\_DM\_TRIGGER

CONV\_ENABLE

CONV\_PAGE[31:0]

IFFT\_LOOP\_NUM[5:0]

HSUM\_DM\_TRIGGER

HSUM\_ENABLE

HSUM\_PAGE[31:0]

HSUM\_DONE

**PCIF**

ADDR

DATA[511:0]

**DDRIF2 #1**

**DDRIF2 #0, #2 and #3 acting in unison to provide a 1536-bit data interface to CONV and HSUM**

ADDR

3xDATA[5111:0]

3xDATA[511:0]

ADDR

ADDR

DATA[511:0]

*Note: The PCIe Hard Macro can read and write to allExternal DDR memories for diagnostic purposes. Not shown in this figure for clarity.*

*Note CLD, CONV and HSUM are designed with paging of the DDR memory for a future implementation. This allows different regions of the DDR SDRAM memory to be used to store data if desired.*

**FDAS\_DDR\_CONTROLLER\_CALIBRATION\_HPS (INTEL IP)**

**provides the DDR Controllers for DDRIF2#0 and DDRIF#1**

**FDAS\_DDR\_CONTROLLER\_CALIBRATION (INTEL IP)**

**provides the DDR Controllers for DDRIF2#2 and DDRIF#3**

DMA Transfer

**External 4 Gibi-Byte**

**DDR4 SDRAM Memory #1**

**External 4 Gibi-Byte**

**DDR4 SDRAM Memory #0. #2 and #3**

MC

Interface

Via

PCIe

*Note: In this implementation CLD\_PAGE, CONV\_PAGE and HSUM\_PAGE are fixed (static) in this implementation. However with a future implementation with more DDR Interfaces to CONV/HSUM a Paging technique shall enable increased processing performance.*

MC Bus

OVERLAP\_SIZE[9:0]

FOP\_SAMPLE\_NUM[22:0]

**MSIX**

MSI-X

Figure 2‑ : Intel Agilex PCIe Hard IP Macro Location in FDAS

The figure below shows the connectivity of the PCIe Hard IP Macro in more detail.

PCIE

Hard IP

Macro

CLD

Module

CONV

Module

HSUM

Module

DDRIF2 #1 Module containing clock boundary FIFOs

DDR 4 SDRAM#1

**FDAS**

512-bit @ 350MHz access from PCIe to each DDRIF2

512-bit @ 350MHz

1536-bit @ 350MHz

1536-bit @ 350MHz

64-bit @ 1333MHz

C

A

L

DDR\_

CONT

HPS

Ch 1

FDAS\_DDR\_

CONTROLLER\_HPS

CALIBRATION

(INTEL IP) #1

DDR\_

CONT

Ch 0

DDRIF2 #0 Module containing clock boundary FIFOs

DDR 4 SDRAM#0

64-bit @ 1333MHz

DDRIF2 #2 Module containing clock boundary FIFOs

DDR 4 SDRAM#2

64-bit @ 1333MHz

C

A

L

DDR\_

CONT

Ch 2

FDAS\_DDR\_

CONTROLLER\_

CALIBRATION

(INTEL IP) #1

DDR\_

CONT

Ch 3

DDRIF2 #3 Module containing clock boundary FIFOs

DDR 4 SDRAM#3

64-bit @ 1333MHz

“FIFO\_READY”, “FIFO\_FULL” and “DATA\_AVAIL” signals transferred between DDRIF2 modules for CONV/ HSUM data paths

512 bits to/from each DDRIF2

Figure ‑ : DDRIF2 Connectivity in FDAS

# Functional Specification

h2ddmWrite\_o

h2ddm\_address\_o[63:0]

h2ddm\_writedata\_o[511:0]

h2ddm\_burstcount\_o[3:0]

h2dmm\_byteenable\_o[31:0]]

h2dmm\_waitrequest\_i

d2hdm\_read\_o

d2hdm\_address\_o[63:0]

d2hdm\_byteenable\_o[31:0]

d2hdm\_burstcount\_o[3:0]

d2mdm\_readdata\_i[511:0]

d2hdm\_waitrequest\_i

d2hdm\_readdatavalid\_i

d2hdm\_response\_i[1:0]

rx\_pio\_read\_o

rx\_pio\_write\_o

rx\_pio\_address\_o[36:0]

rx\_pio\_burstcount\_o[3:0]

rx\_io\_byteenable\_o[7:0]

rx\_pio\_writedata\_o[63:0]

rx\_pio\_readdata\_i[63:0]

rx\_pio\_readdatavalid\_i

rx\_pio\_waitrequest\_i

rx\_pio\_response\_i[1:0]

rx\_pio\_writeresponsevalid\_i

usr\_event\_msix\_ready\_o

usr\_even\_msix\_valid\_i

usr\_event\_msix\_data\_i[15:0]

**Agilex**

**PCIe Hard IP with**

**DMA Controller and Avalon MM Bridge**

app\_clk

refclk0

refclk1

pin\_perst

rx\_out0[15:0:0]

rx\_in0[15:0:0]

Reset

Serial

Data

(PCIe)

**DDRIF2 #0**

**MCI\_TOP**

**PCIF**

**FDAS**

Avalon MM Rx Master

Module Interface

Host to Device Avalon

MM Master Interface

(512-bit data)

Device to Host

DMA Avalon

MM Master Interface

(512 -bit data)

**DDRIF2 #1**

refclk0

**TAB**

**TAB**

**TAB**

**TAB**

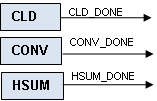
Notes:-

* TAB = Transparent Avalon MM Bridge. These are required to provide a destination for the Memory Mapped interfaces in the Intel Quartus Prime Platform Designer environment
* For Clarity the Clock/Reset Bridges are not shown

**PCIE\_HIP\_FDAS**

refclk1

**MSIX**



Config

User MSI-X

Interface

**TAB**

**RESET\_**

**RELEASE**

**(Intel IP)**

ninit\_done

app\_rst\_n

**DDRIF2 #2**

**DDRIF2 #3**

**TAB**

**TAB**

**TAB**

**TAB**

Figure ‑: Intel Agilex PCIe Hard Ip Macro Top Level Connectivity in FDAS

The above figure shows the main connectivity of the Intel Agilex PCIe Hard IP macro module within the FDAS FPGA.

The PCIe Hard IP macro uses the 350MHz CLK\_PCIE, hence signals which are transferred between the PCIe Hard IP macro and modules within FDAS shall require meta-stability protection.

The following interfaces of the Intel Agilex PCIe Hard IP macro connect to the PCIF module:-

* **Avalon MM Rx Master Module Interface (rx\_pio)**: This interface supports transfers initiated by the Computer/PC over the PCIe. This is used when the FDAS Micro Configuration register values are programmed, for example with the convolution filter values. The Computer/PC can initiate a write to the FDAS configuration registers or request a read of the FDAS configuration registers via this interface. The interface is compliant the Avalon MM protocol, with Address /Data buses and with the Hard PCIe macro performing the TLP encapsulation and de-encapsulation. Only the lower 32 bits of the data buses are used by the PCIF module. This Interface uses the BAR2 Base Address Register.

The Transparent Avalon MM Bridge provides an “Agent” for the PCIe Hard IP Macro to connect to within the Platform Designer environment. This avoids logic being stripped out in the build.

The following interfaces of the Intel Agilex PCIe Hard IP macro connect to the MSIX module:-

* **User MSI-X Interface (usr\_event\_msix)**: This interface supports the sending of MSI-X (Extended Message Signalled Interrupts) by the FDAS FPGA to the Host PC via the PCIe Interface.

The PCIe Hard IP Macro contains the MSI-X “Interrupt Vector Table” that is used by both the DMA engine and the User Interrupts to signal to the PC/Computer that an event (i.e. DMA or User) has occurred. The table has four entries per DMA channel (also called Queue), with a maximum of 512 DMA channels, with each entry containing an Interrupt vector configured by the PC/Computer. For each DMA channel (Queue) two of the entries are for DMA operation Interrupts and two are for User Interrupts. The lower two bits of the address to the Interrupt vector table select the Interrupt:-

Address[1:0] Interrupt

“00” Host to Device DMA Interrupt

“01” Host to Device User MSI-X

“10” Device to Host DMA Interrupt

“11” Device to Host User MSI-X

The User MSI-X interface allows the desired entry to be selected via a 16-bit value user\_event\_msix\_data[15:0] (rsvd[3:0],msix\_queue\_dir, msix\_queue\_num\_i[10:0]).

msix\_queue\_dir = 1 for Host to Device and msix\_queue\_dir = 0 for Device to Host. msix\_queue\_dir is effectively selecting the bottom two bits of the Interrupt Vector Table (i.e. “01” or “11”) and msix\_queue\_num\_i[10:0] is selecting the DMA channel.

The following interfaces of the Intel Agilex PCIe Hard IP macro connect to the DDRIF2 modules:-

* **Host to Device Avalon MM Master Interface (h2ddm):** Via Transparent Avalon –MM Bridges (“TAB”) to allow address decoding and provide an “Agent” in Platform Designer for the PCIe Hard IP Macro to connect to. This interface supports the DMA transfer of data from the PC/Computer to the DDRIF2 modules via the PCIe Interface using a DMA technique.
* **Device to Host Avalon MM Master Interface (d2hdm):**  **):** Via Transparent Avalon –MM Bridges (“TAB”) to allow address decoding and provide an “Agent” in Platform Designer for the PCIe Hard IP Macro to connect to. This interface supports the DMA transfer of data from the DDRIF2 modules to the PC/Computer via the PCIe Interface using a DMA technique.

The PCIe Hard IP Macro contains a DMA engine to allow data to be transferred between the PC/Computer and the external DDR SDRAMs via the DDRIF2 modules. The DMA engine is programmed with the transactions to perform by the PC/Computer and uses BAR0 Base Address Register.

## PCIe Hard IP Macro Settings in PLATFORM DESIGNER

Table 3‑1 below shows the steps to generate the Intel Agilex F PCIe Hard IP Macro.

| Step | Description |
| --- | --- |
| 1 | Run up Quartus Prime |
| 2 | Top Bar Menu:- File > New Project Wizard   * Select a working directory for the project * Define a name for the project: **PCIE\_HIP\_FDAS** (for Version 21.3 of Quartus Prime) * Empty Project |
| 3 | Top Bar Menu:- Assignments > Device   * Select Family: -Agilex (F-series/I-series) * Select Device:- All   From the list select the desired part: Agilex FPGA: AGFB014R24A2E2VR0 |
| 4 | Top Bar Menu:- Tools > PLATFORM DESIGNER |
| 5 | Create simple Transparent Avalon Bridge components (TAB) that can be used to export signals from the PCIe Hard IP Macro to the FDAS core without having to instantiate any of the FDAS core components in the PLATFORM DESIGNER environment:-  To support the connection to the PCIF module:-  Create the **mm\_transparent\_pio\_no\_burst.vhd** entity file containing:-  entity mm\_transparent\_no\_burst\_pio is  generic (  DATA\_WIDTH : integer := 32;  BYTE\_SIZE : integer := 8;  ADDRESS\_WIDTH : integer := 20;  RESPONSE\_WIDTH : integer := 2  );  port (  clk : in std\_logic := 'X'; -- clk  m0\_waitrequest : in std\_logic := 'X'; -- waitrequest  m0\_readdata : in std\_logic\_vector(DATA\_WIDTH-1 downto 0) := (others => 'X'); -- readdata  m0\_readdatavalid : in std\_logic := 'X'; -- readdatavalid  m0\_writeresponsevalid : in std\_logic := 'X'; -- write response valid  m0\_response : in std\_logic\_vector(RESPONSE\_WIDTH -1 downto 0) := (others => 'X'); -- response  m0\_writedata : out std\_logic\_vector(DATA\_WIDTH-1 downto 0); -- writedata  m0\_address : out std\_logic\_vector(ADDRESS\_WIDTH-1 downto 0); -- address  m0\_write : out std\_logic; -- write  m0\_read : out std\_logic; -- read  m0\_byteenable : out std\_logic\_vector(DATA\_WIDTH/BYTE\_SIZE-1 downto 0); -- byteenable  reset : in std\_logic := 'X'; -- reset  s0\_waitrequest : out std\_logic; -- waitrequest  s0\_readdata : out std\_logic\_vector(DATA\_WIDTH-1 downto 0); -- readdata  s0\_readdatavalid : out std\_logic; -- readdatavalid  s0\_writeresponsevalid : out std\_logic; -- write response valid  s0\_response : out std\_logic\_vector(RESPONSE\_WIDTH -1 downto 0); -- response  s0\_writedata : in std\_logic\_vector(DATA\_WIDTH-1 downto 0) := (others => 'X'); -- writedata  s0\_address : in std\_logic\_vector(ADDRESS\_WIDTH-1 downto 0) := (others => 'X'); -- address  s0\_write : in std\_logic := 'X'; -- write  s0\_read : in std\_logic := 'X'; -- read  s0\_byteenable : in std\_logic\_vector(DATA\_WIDTH/BYTE\_SIZE-1 downto 0) := (others => 'X') -- byteenable  );  -- Declarations  end entity mm\_transparent\_no\_burst\_pio;  Create the **mm\_transparent\_no\_burst\_pio\_synth.vhd** architecture file containing:-  architecture synth of mm\_transparent\_no\_burst\_pio is  begin  m0\_writedata <= s0\_writedata;  m0\_address <= s0\_address;  m0\_write <= s0\_write;  m0\_read <= s0\_read;  m0\_byteenable <= s0\_byteenable;  s0\_waitrequest <= m0\_waitrequest;  s0\_readdata <= m0\_readdata;  s0\_readdatavalid <= m0\_readdatavalid;  s0\_writeresponsevalid <= m0\_writeresponsevalid;  s0\_response <= s0\_response;    end architecture synth; -- of mm\_transparent\_no\_burst\_pio  To support the connection to the DDRIF2 modules to read data from the DDR4 SDRAM memory:-  Create the **mm\_rd\_transparent.vhd** entity file containing:-  entity mm\_rd\_transparent is  generic (  DATA\_WIDTH : integer := 256;  BYTE\_SIZE : integer := 8;  ADDRESS\_WIDTH : integer := 32;  BURSTCOUNT\_WIDTH : integer := 4  );  port (  clk : in std\_logic := 'X'; -- clk  m0\_waitrequest : in std\_logic := 'X'; -- waitrequest  m0\_readdata : in std\_logic\_vector(DATA\_WIDTH-1 downto 0) := (others => 'X'); -- readdata  m0\_readdatavalid : in std\_logic := 'X'; -- readdatavalid  m0\_response : in std\_logic\_vector(1 downto 0) := (others => 'X'); -- response  m0\_burstcount : out std\_logic\_vector(BURSTCOUNT\_WIDTH-1 downto 0); -- burstcount  m0\_address : out std\_logic\_vector(ADDRESS\_WIDTH-1 downto 0); -- address  m0\_read : out std\_logic; -- read  reset : in std\_logic := 'X'; -- reset  s0\_waitrequest : out std\_logic; -- waitrequest  s0\_readdata : out std\_logic\_vector(DATA\_WIDTH-1 downto 0); -- readdata  s0\_readdatavalid : out std\_logic; -- readdatavalid  s0\_response : out std\_logic\_vector(1 downto 0); -- response  s0\_burstcount : in std\_logic\_vector(BURSTCOUNT\_WIDTH-1 downto 0) := (others => 'X'); -- burstcount  s0\_address : in std\_logic\_vector(ADDRESS\_WIDTH-1 downto 0) := (others => 'X'); -- address  s0\_read : in std\_logic := 'X' -- read  );  end entity mm\_rd\_transparent;  Create the **mm\_rd\_transparent\_synth.vhd** architecture file containing:-  architecture synth of mm\_rd\_transparent is  begin  m0\_burstcount <= s0\_burstcount;  m0\_address <= s0\_address;  m0\_read <= s0\_read;  s0\_waitrequest <= m0\_waitrequest;  s0\_readdata <= m0\_readdata;  s0\_readdatavalid <= m0\_readdatavalid;  s0\_response <= m0\_response;  end architecture synth; -- of mm\_rd\_transparent  To support the connection to the DDRIF2 modules to write data to the DDR4 SDRAM memory:-  Create the **mm\_wr\_transparent.vhd** entity file containing:-  entity mm\_wr\_transparent is  generic (  DATA\_WIDTH : integer := 256;  BYTE\_SIZE : integer := 8;  ADDRESS\_WIDTH : integer := 32;  BURSTCOUNT\_WIDTH : integer := 4  );  port (  clk : in std\_logic := 'X'; -- clk  m0\_waitrequest : in std\_logic := 'X'; -- waitrequest  m0\_burstcount : out std\_logic\_vector(BURSTCOUNT\_WIDTH-1 downto 0); -- burstcount  m0\_writedata : out std\_logic\_vector(DATA\_WIDTH-1 downto 0); -- writedata  m0\_address : out std\_logic\_vector(ADDRESS\_WIDTH-1 downto 0); -- address  m0\_write : out std\_logic; -- write  m0\_byteenable : out std\_logic\_vector(DATA\_WIDTH/BYTE\_SIZE-1 downto 0); -- byteenable  reset : in std\_logic := 'X'; -- reset  s0\_waitrequest : out std\_logic; -- waitrequest  s0\_burstcount : in std\_logic\_vector(BURSTCOUNT\_WIDTH-1 downto 0) := (others => 'X'); -- burstcount  s0\_writedata : in std\_logic\_vector(DATA\_WIDTH-1 downto 0) := (others => 'X'); -- writedata  s0\_address : in std\_logic\_vector(ADDRESS\_WIDTH-1 downto 0) := (others => 'X'); -- address  s0\_write : in std\_logic := 'X'; -- write  s0\_byteenable : in std\_logic\_vector(DATA\_WIDTH/BYTE\_SIZE-1 downto 0) := (others => 'X') -- byteenable  );  end entity mm\_wr\_transparent;  Create the **mm\_wr\_transparent\_synth.vhd** architecture file containing:-  architecture synth of mm\_wr\_transparent is  begin  m0\_burstcount <= s0\_burstcount;  m0\_writedata <= s0\_writedata;  m0\_address <= s0\_address;  m0\_write <= s0\_write;  m0\_byteenable <= s0\_byteenable;  s0\_waitrequest <= m0\_waitrequest;    end architecture synth; -- of mm\_wr\_transparent  In the PLATFORM DESIGNER environment select File > New Component to create the required Transparent Avalon Bridge components using the VHDL files above in the “Files” tab for both the Synthesis file and the VHDL simulation files (can use “Copy from synthesis files” to save the effort of browsing for the same files again). . These components can be named:-  **“mm\_transparent\_no\_burst\_pio** “to support connection to the PCIF module  **“mm\_rd\_transparent”** to support connection to DDRIF2 modules to read data from the DDR4 SDRAMs  **“mm\_wr\_transparent”** to support connection to DDRIF2 modules to write to the DDR4 SDRAMs  For each new component in the “Files” tab Analyse the files. Initially many errors may be reported.  Then in the “Signal & Interfaces” tab:-   * Click on the signal group in the “Name” panel and ensure the “s0” signals are for an Avalon Memory mapped Agent and the “m0” signals are for an Avalon memory mapped Host. * Rename the signal group for the s0 signals as “s0” and the signal group for the m0 signals as “m0” * Click on each signal group in the “Name” panel and ensure it has an associated reset * Click on each signal in the “Name” panel and ensure it has the correct signal type and direction * Click on each signal group in the “Name” panel ensure in the drop down settings to the far right that any pending read/write transactions are set to clear any errors and that the addressing is set to WORD on “s0” and “m0” ports. * For the **“mm\_wr\_transparent”** ensure the WaitRequest Allowance is set to 16 on the “”s0” and “m0” ports to match that of the PCIe Macro. * For the **“mm\_rd\_transparent”** ensure the Maximum Pending Read Transactions reads are set to 32 to match that of the PCIe Macro.   Note that when creating these components in PLATFORM DESIGNER the addressing must be word based (i.e. each address increment is for a 512-bit data word for the DMA interface and a 64-bit data word for the PIO interface). |
| 6 | In the PLATFORM DESIGNER environment IP Catalog window  Select “Interface Protocols” > PCI Express > Multi Channel DMA P-Tile for PCI Express  Press “+ADD” to add to the design.  This will put the “**intel\_pcie\_ptile\_mcdma\_0**” component in the “system contents” window of the PLATFORM DESIGNER environment. |
| 7 | In the PLATFORM DESIGNER environment IP Catalog window  Select “Basic Functions> Bridges and Adaptors” > reset > reset bridge  Press “+ADD” to add to the design.  This will put the “**reset\_bridge\_0**” component in the “system contents” window of the PLATFORM DESIGNER environment.  *This is needed because the reset needs to be passed to the Modules within FDAS (DDRIF2, PCIF)* |
| 8 | In the PLATFORM DESIGNER environment IP Catalog window  Select “Basic Functions> Bridges and Adaptors” > clock > clock bridge  Press “+ADD” to add to the design.  This will put the “**clock\_bridge\_0**” component in the “system contents” window of the PLATFORM DESIGNER environment.  *This is needed because the 350MHz PCIe clock needs to be passed to the Modules within FDAS (DDRIF2,PCI)* |
| 9 | In the PLATFORM DESIGNER environment IP Catalog window  Select the **“mm\_transparent\_no\_burst\_pio”** component  Press “+ADD” to add to the design.  Change ‘HDL entity name’ to “**rxm\_bar2\_0**”  Set the parameters as follows:  DATA\_WIDTH=64  BYTE\_SIZE=8  ADDRESS\_WIDTH=22  RESPONSE\_WIDTH = 2  Press “Finish”.  This will put the “**mm\_transparent\_no\_burst\_pio** ” component in the “system contents” window of the PLATFORM DESIGNER environment.  This component is essentially a transparent Avalon-MM bridge to provide a memory mapped destination within Intel Quartus Prime Platform Designer for the Intel PCIe Hard IP Macro (the logic seems to be stripped away if this is not provided).  **Not that the data width has to be set to 64 bits as this is the width of the data bus out of the PCIe Hard Macro (unlike for Arria 10 which had a 32-bit interface. This means that the bitmap addressing will need to take account of this (address step will be 8, with only the lower 32 bits of the data bus being used).** |
| 10 | In the PLATFORM DESIGNER environment IP Catalog window  Select the **“mm\_wr\_transparent”** component  Press “+ADD” to add to the design.  Change ‘HDL entity name’ to “**dma\_rd\_0**”  Set the parameters as follows:  DATA\_WIDTH=512  BYTE\_SIZE=8  ADDRESS\_WIDTH=26  BURSTCOUNT\_WIDTH=4  Press “Finish”.  This will put the “**mm\_wr\_transparent** ” component in the “system contents” window of the PLATFORM DESIGNER environment.  This component is essentially a transparent Avalon-MM bridge to provide a memory mapped destination within Intel Quartus Prime Platform Designer for the Intel PCIe Hard IP Macro (the logic seems to be stripped away if this is not provided). |
| 11 | In the PLATFORM DESIGNER environment IP Catalog window  Select the **“mm\_rd\_transparent”** component  Press “+ADD” to add to the design.  Change ‘HDL entity name’ to “**dma\_wr\_0**”  Set the parameters as follows:  DATA\_WIDTH=512  BYTE\_SIZE=8  ADDRESS\_WIDTH=26  BURSTCOUNT\_WIDTH=4  Press “Finish”.  This will put the “**mm\_rd\_transparent** ” component in the “system contents” window of the PLATFORM DESIGNER environment.  This component is essentially a transparent Avalon-MM bridge to provide a memory mapped destination within Intel Quartus Prime Platform Designer for the Intel PCIe Hard IP Macro (the logic seems to be stripped away if this is not provided). |
| 12 | In the PLATFORM DESIGNER environment IP Catalog window  Select the **“mm\_wr\_transparent”** component  Press “+ADD” to add to the design.  Change ‘HDL entity name’ to “**dma\_rd\_1**”  Set the parameters as follows:  DATA\_WIDTH=512  BYTE\_SIZE=8  ADDRESS\_WIDTH=26  BURSTCOUNT\_WIDTH=4  Press “Finish”.  This will put the “**mm\_wr\_transparent** ” component in the “system contents” window of the PLATFORM DESIGNER environment.  This component is essentially a transparent Avalon-MM bridge to provide a memory mapped destination within Intel Quartus Prime Platform Designer for the Intel PCIe Hard IP Macro (the logic seems to be stripped away if this is not provided). |
| 13 | In the PLATFORM DESIGNER environment IP Catalog window  Select the **“mm\_rd\_transparent”** component  Press “+ADD” to add to the design.  Change ‘HDL entity name’ to “**dma\_wr\_1**”  Set the parameters as follows:  DATA\_WIDTH=512  BYTE\_SIZE=8  ADDRESS\_WIDTH=26  BURSTCOUNT\_WIDTH=4  Press “Finish”.  This will put the “**mm\_rd\_transparent** ” component in the “system contents” window of the PLATFORM DESIGNER environment.  This component is essentially a transparent Avalon-MM bridge to provide a memory mapped destination within Intel Quartus Prime Platform Designer for the Intel PCIe Hard IP Macro (the logic seems to be stripped away if this is not provided). |
| 14 | In the PLATFORM DESIGNER environment IP Catalog window  Select the **“mm\_wr\_transparent”** component  Press “+ADD” to add to the design.  Change ‘HDL entity name’ to “**dma\_rd\_2**”  Set the parameters as follows:  DATA\_WIDTH=512  BYTE\_SIZE=8  ADDRESS\_WIDTH=26  BURSTCOUNT\_WIDTH=4  Press “Finish”.  This will put the “**mm\_wr\_transparent** ” component in the “system contents” window of the PLATFORM DESIGNER environment.  This component is essentially a transparent Avalon-MM bridge to provide a memory mapped destination within Intel Quartus Prime Platform Designer for the Intel PCIe Hard IP Macro (the logic seems to be stripped away if this is not provided). |
| 15 | In the PLATFORM DESIGNER environment IP Catalog window  Select the **“mm\_rd\_transparent”** component  Press “+ADD” to add to the design.  Change ‘HDL entity name’ to “**dma\_wr\_2**”  Set the parameters as follows:  DATA\_WIDTH=512  BYTE\_SIZE=8  ADDRESS\_WIDTH=26  BURSTCOUNT\_WIDTH=4  Press “Finish”.  This will put the “**mm\_rd\_transparent** ” component in the “system contents” window of the PLATFORM DESIGNER environment.  This component is essentially a transparent Avalon-MM bridge to provide a memory mapped destination within Intel Quartus Prime Platform Designer for the Intel PCIe Hard IP Macro (the logic seems to be stripped away if this is not provided). |
| 16 | In the PLATFORM DESIGNER environment IP Catalog window  Select the **“mm\_wr\_transparent”** component  Press “+ADD” to add to the design.  Change ‘HDL entity name’ to “**dma\_rd\_3**”  Set the parameters as follows:  DATA\_WIDTH=512  BYTE\_SIZE=8  ADDRESS\_WIDTH=26  BURSTCOUNT\_WIDTH=4  Press “Finish”.  This will put the “**mm\_wr\_transparent** ” component in the “system contents” window of the PLATFORM DESIGNER environment.  This component is essentially a transparent Avalon-MM bridge to provide a memory mapped destination within Intel Quartus Prime Platform Designer for the Intel PCIe Hard IP Macro (the logic seems to be stripped away if this is not provided). |
| 17 | In the PLATFORM DESIGNER environment IP Catalog window  Select the **“mm\_rd\_transparent”** component  Press “+ADD” to add to the design.  Change ‘HDL entity name’ to “**dma\_wr\_3**”  Set the parameters as follows:  DATA\_WIDTH=512  BYTE\_SIZE=8  ADDRESS\_WIDTH=26  BURSTCOUNT\_WIDTH=4  Press “Finish”.  This will put the “**mm\_rd\_transparent** ” component in the “system contents” window of the PLATFORM DESIGNER environment.  This component is essentially a transparent Avalon-MM bridge to provide a memory mapped destination within Intel Quartus Prime Platform Designer for the Intel PCIe Hard IP Macro (the logic seems to be stripped away if this is not provided). |
| 18 | **Intel\_pcie\_ptile\_mcdma0\_0 Parameter Configuration**  Double click on “**Intel\_pcie\_ptile\_mcdma0\_0**” in the “system contents” window so that the parameters can be seen in another window. In this parameters window enter the desired values for the various parameters.  **Design Environment:** System  **Top-Level Settings Tab**  **Hard IP Mode** **:**Gen 4x16, Interface: 512-bit (Gen 4 PCIe, 16 lane , 512 Bit interface to FDAS logic @ 350MHz)  **Port Mode:** Native Endpoint  **Enable PHY Reconfiguration:** OFF  **Enable Ptile Debug Toolkit:** OFF  **PLD Clock Frequency:** 350MHz  **Enable SRIS Mode:** OFF  **P-Time Sim Mode:** OFF  **Enable CVP (Intel VSEC):** ON  **PCIe0 Settings Tab**  **MCDMA Settings Sub-Tab**  **BAR2 Address Width:** 32 MBytes – 25 bits *(this is a byte address range, however the 3 bottom bits are automatically dropped by Platform Designer as it recognises the mm\_transparent\_no\_burst\_pio block has a Word address input)*  **User Mode:** Multi channel DMA  **Interface Type:** AVMM  **Number of Ports:** 1 (greyed out)  **Enable User-MSIX:** ON  **Enable User-FLR:** OFF  **D2H Prefetch Channels:** 8 (greyed out)  **Maximum Descriptor Fetch:** 16 (greyed out)  **Enable Metadata:** OFF (greyed out)  **Enable Configuration Intercept Interface:** OFF  **Enable 10-bit tag support:** OFF  **Enable address byte aligned transfer:** OFF (greyed out)  **Enable pld\_warm\_rst\_rdy\_and link\_rq\_rst\_n interface to top level**: OFF  **PCIe0 PCI Express/ PCI Capabilities Sub-Tab**  **PCIe0 Device Sub-Sub-Tab**  **Maximum payload size supported:** 512 bytes  **Support Extended Tag Field:** ON (greyed out)  **PCIe0 Multifunction and SR-IOV Systems Settings:**  **Enable multiple physical functions:** OFF  **Enable SR\_IOV support:** OFF  **PF0**  **Number of DMA channels allocated to PF0:** 4  **PCIe0 Link Sub-Sub-Tab**  **Link Port number (Root port only):** 1  **Slot clock configuration:** ON  **PCIe0 MSI-X Sub-Sub Tab**  **Enable MSI-X:** ON (greyed out)  **Table Size:** 15 (greyed out)  **Table Offset:** 0x0000\_0000\_0002\_0000 (greyed out)  **Table Bar Indicator:** 0 (greyed out)  **Pending bit array (PBA) offset**: 0x0000\_0000\_0003\_0000 (greyed out)  **PBA BAR Indicator:** 0 (greyed out)  **PCIe0 DEV SER Sub-Sub Tab**  **Enable Device Serial Number Capability:** OFF  **Device Serial Number (DW1) :** 0x0000\_0000\_0000\_0000 (greyed out)  **Device Serial Number (DW2) :** 0x0000\_0000\_0000\_0000 (greyed out)  **PCIe0 PRS Sub-Sub Tab**  **PF0 Enable PRS:** OFF  **PCIe0 ATS Sub-Sub Tab**  **Enable Address Translation Services (ATS):** OFF  **PCIe0 TPH Sub-Sub Tab**  **Enable TLP Processing Hints (TPH):** OFF  **PCIe0 Configuration, Debug and Extension Options Sub-Tab**  **Gen 3 Requested equalization far-end TX preset vector:** 0x00000004  **Gen 3 Requested equalization far-end TX preset vector:** 0x00000270  **Enable Rx Buffer Limit Ports:** ON (greyed out)  **Enable HIP Reconfig Interface:** OFF  **PCIe0 Device Identification Registers Sub-Tab**  **Vendor ID:** 0x00001172  **Device ID:** 0x00000000  **Revision ID:** 0x00000001  **Class Code:** 0x00ff0000  **Subsystem Vendor ID:** 0x00000000  **Subsystem Device ID:** 0x00000000 |
| 19 | **clock\_bridge\_0 Parameter Configuration**  Double click on “**clock\_bridge\_0**” in the “system contents” window so that the parameters can be seen in another window. In this parameters window enter the desired values for the various parameters.  **Parameters**  **Explicit Clock Rate :** 350000000Hz  **Number of Clock Outputs:**1 |
| 20 | **reset\_bridge\_0 Parameter Configuration**  Double click on “**reset\_bridge\_0**” in the “system contents” window so that the parameters can be seen in another window. In this parameters window enter the desired values for the various parameters.  **Parameters**  **Active Low Reset :** ON  **Synchronous Edges:**Deassert  **Number of Reset Outputs:** 1  **Use Reset Request Signal:** OFF  **Use Synchronous Resets:** OFF |
| 21 | **In the PLATFORM DESIGNER environment “System Contents” window**  Remove the unwanted clock generator IP and reset IP that may be present. |
| 22 | **In the PLATFORM DESIGNER environment “System Contents” window**  For **intel\_pcie\_ptile\_mcdma\_0:**  Export the **refclk0** and **refclk1** inputs to pins by double-clicking on the Export column. This will need to connect to a 100MHz clock, either via a PLL or an external FPGA pin.  For **intel\_pcie\_ptile\_mcdma\_0:**  Export **usr\_msix** pins by double-clicking on the Export column. This will allow the MSI-X interrupts to be controlled FDAS.  Connect the **intel\_pcie\_ptile\_mcdma\_0 “app\_clk”** 350MHz output to the following:-   * **clock\_bridge\_0 “in\_clk”** port * **reset\_bridge\_0, “clk”** port * **rxm\_bar2\_0 “clock”** port * **dma\_wr\_0 “clock”** port * **dma\_wr\_1 “clock”** port * **dma\_wr\_2 “clock”** port * **dma\_wr\_3 “clock”** port * **dma\_rd\_0 “clock”**port * **dma\_rd\_1 “clock”**port * **dma\_rd\_2“clock”**port * **dma\_rd\_3 “clock”**port   Export the **intel\_pcie\_ptile\_mcdma\_0**  “**hip\_serial”** interface t to pins by double-clicking on the Export column. This is the serial PCIe interface to the pins of the FPGA.  Connect the **intel\_pcie\_ptile\_mcdma\_0** “**app\_nreset\_status”** output to the following:-   * **reset\_bridge\_0, “in\_reset”** port * **rxm\_bar2\_0 “reset”** port * **dma\_wr\_0 “reset”** port * **dma\_wr\_1 “reset”** port * **dma\_wr\_2 “reset”** port * **dma\_wr\_3 “reset”** port * **dma\_rd\_0 “reset”** port * **dma\_rd\_1 “reset”** port * **dma\_rd\_2 “reset”** port * **dma\_rd\_3 “reset”** port   Export the **intel\_pcie\_ptile\_mcdma\_0** “**pin\_perst”** , **“ninit\_done**” & “**p0\_pld”**  pins by double-clicking on the Export column.  Connect the **intel\_pcie\_ptile\_mcdma\_0** “**rx\_pio\_master** interface to the following:-   * **rxm\_bar2\_0 “S0”** port   Connect the **intel\_pcie\_ptile\_mcdma\_0** “**h2ddm\_master”** interface to the following:-   * **dma\_rd\_0 “S0”** port * **dma\_rd\_1 “S0”** port * **dma\_rd\_2 “S0”** port * **dma\_rd\_3 “S0”** port   This will allow the “Read DMA Avalon MM Master Interface” to be connected to the DDRIF2 modules to support the DMA transfer of data from the PC/Computer to external DDR4 SDRAM.  Connect the **intel\_pcie\_ptile\_mcdma\_0** “**d2hdm\_master”** interface to the following:-   * **dma\_wr\_0 “S0”** port * **dma\_wr\_1 “S0”** port * **dma\_wr\_2 “S0”** port * **dma\_wr\_3 “S0”** port   This will allow the “Write DMA Avalon MM Master Interface” to be connected to the DDRIF2 modules to support the DMA transfer of the information from external DDR4 SDRAM memories to the PC/Computer for diagnostic observation. |
| 23 | **In the PLATFORM DESIGNER environment “System Contents” window**  For **rxm\_bar2\_0:**  Export the **rxm\_bar2\_0** “**M0”** interface to pins by double-clicking on the Export column. This is to allow the connection to PCIF module. |
| 24 | **In the PLATFORM DESIGNER environment “System Contents” window**  For **dma\_rd\_0:**  Export the **dma\_rd\_0** “**M0”** interface to pins by double-clicking on the Export column. This is to allow the connection to write data to DDR4 SDRAM #1. Rename the exported port to **rd\_dma\_0**. |
| 25 | **In the PLATFORM DESIGNER environment “System Contents” window**  For **dma\_wr\_0:**  Export the **dma\_wr\_0** “**M0”** interface to pins by double-clicking on the Export column. This is to allow the connection to read data from DDR4 SDRAM #1. Rename the exported port to **wr\_dma\_0**. |
| 26 | **In the PLATFORM DESIGNER environment “System Contents” window**  For **dma\_rd\_1:**  Export the **dma\_rd\_1** “**M0”** interface to pins by double-clicking on the Export column. This is to allow the connection to write data to DDR4 SDRAM #2. Rename the exported port to **rd\_dma\_1**. |
| 27 | **In the PLATFORM DESIGNER environment “System Contents” window**  For **dma\_wr\_1:**  Export the **dma\_wr\_1** “**M0”** interface to pins by double-clicking on the Export column. This is to allow the connection to read data from DDR4 SDRAM #2. Rename the exported port to **wr\_dma\_1**. |
| 28 | **In the PLATFORM DESIGNER environment “System Contents” window**  For **dma\_rd\_2:**  Export the **dma\_rd\_2** “**M0”** interface to pins by double-clicking on the Export column. This is to allow the connection to write data to DDR4 SDRAM #1. Rename the exported port to **rd\_dma\_2**. |
| 29 | **In the PLATFORM DESIGNER environment “System Contents” window**  For **dma\_wr\_2:**  Export the **dma\_wr\_2** “**M0”** interface to pins by double-clicking on the Export column. This is to allow the connection to read data from DDR4 SDRAM #1. Rename the exported port to **wr\_dma\_2**. |
| 30 | **In the PLATFORM DESIGNER environment “System Contents” window**  For **dma\_rd\_3:**  Export the **dma\_rd\_3** “**M0”** interface to pins by double-clicking on the Export column. This is to allow the connection to write data to DDR4 SDRAM #2. Rename the exported port to **rd\_dma\_3**. |
| 31 | **In the PLATFORM DESIGNER environment “System Contents” window**  For **dma\_wr\_3:**  Export the **dma\_wr\_3** “**M0”** interface to pins by double-clicking on the Export column. This is to allow the connection to read data from DDR4 SDRAM #2. Rename the exported port to **wr\_dma\_3**. |
| 32 | **In the PLATFORM DESIGNER environment “System Contents” window**  For the **clock\_bridge\_0:**  Export the output clock to a pin by double-clicking on the Export column and rename to “**out\_clk”**.  This is the 350MHz clock for the FDAS core logic processing signals from the PCIE\_HARD\_IP\_MACRO. |
| 33 | **In the PLATFORM DESIGNER environment “System Contents” window**  For the **reset\_bridge\_0:**  Export the output reset to a pin by double-clicking on the Export column and rename to “**out\_reset”.**  This is the reset for the FDAS core logic on the 350MHz domain. |
| 34 | **In the PLATFORM DESIGNER environment “System Contents” window**  For the **rxm\_bar2\_0** ensure:   * **“S0”** interface has the address range 0x0 to 0x1FF,FFFF (It should be noted that FDAS will only use the lower 32 data bits of every address location)   For the **dma\_wr\_0** and **dma\_rd\_0** ensure:   * **“S0”** interface has the address range 0x2,0000,0000 to 0x2,FFFF,FFFF   For the **dma\_wr\_1** and **dma\_rd\_1** ensure:   * **“S0”** interface has the address range 0x4,0000,0000 to 0x4,FFFF,FFFF   For the **dma\_wr\_2** and **dma\_rd\_2** ensure:   * **“S0”** interface has the address range 0x6,0000,0000 to 0x6,FFFF,FFFF   For the **dma\_wr\_3** and **dma\_rd\_3** ensure:   * **“S0”** interface has the address range 0x8,0000,0000 to 0x8,FFFF,FFFF   Note that the above address values in the ranges are byte addresses (not word addresses) |
| 35 | Click the “Generate HDL” button to generate the HDL  Ensure the IP design and the simulation language is VHDL. |

Table 3‑1 : PCIe Hard IP Macro Settings in PLATFORM DESIGNER

The completed design in Platform Designer should be as shown in Figure 3‑2:-

Graphical user interface, table

Description automatically generated

Table

Description automatically generated

Figure ‑ : Intel Agilex F PCIe Hard Macro in Quartus Prime Platform Designer

# Design Requirement Tags

The relevant design requirement tags from the FDAS Design Specification Document for the PCIe Hard IP macro.

| **Design Requirement Tag** | **Description** | **Comment** |
| --- | --- | --- |
| FDAS.DATAIN:010/A | The data presented to FDAS shall be the result of an FFT with 222 (4,194,304) freq-bins. | The 222 freq-bin samples, each of 64-bit shall be passed over the PCIe as a DMA transfer and written into DDR4 SDRAM via the DDRIF2 module.  The PC/Computer Software shall be responsible for describing the DMA transfer to ensure the information goes into the correct page of external DDR4 SDRAM memory. |
| FDAS.DATAIN:020/A | Each freq-bin shall contain a complex number with real and imaginary value. |
| FDAS.DATAIN:030/A | Each element of the complex number shall be represented in IEEE 754 single precision format (see Sec 2.7 of the FDRFA spec). Hence the complex number consists of two IEEE 754 values. |
| FDAS.DATAIN:040/A | The FFT sequence shall be stored by FDAS in external DDR SDRAM memory, with all values static for a known period of time. |
| FDAS.DATAIN:060/A | The external memory containing the input data shall have two pages, to allow one page to be written to whilst the other page is being accessed by the FDAS core processing. |
| FDAS.DATAIN:070/A | For a given page the allocation of freq-bins to memory locations shall be known and fixed, thus the memory address can be used to identify the freq-bin number and the real and imaginary value of each freq-bin. |
| FDAS.MC:010/A | FDAS shall be configured, monitored and controlled by a host computer via a Monitor and Control (MC) interface. This interface shall be little-endian, with byte assignments within a 32-bit word | The MC interface shall be provided by the PCIe Interface via the PCIF module. Each 32-bit value shall be provided by a PCIe TLP with a 32-bit data field. |
| FDAS.MC:020/A | The MC Interface shall be supported by the PCIe interface to the FPGA | The MC interface shall be provided by the PCIe Interface via the PCIF module. Each 32-bit value shall be provided by a PCIe TLP with a 32-bit data field. |
| FDAS.MC:030/A | The MC register map for the FDAS function is shown in the table below:-   | MC register | | --- | | **Module: FDAS\_CTRL** | |  | |  | | MC\_MAN\_OVERRIDE\_CFG | | MAN\_CLD\_DM\_TRIGGER\_CFG | | MAN\_CONV\_DM\_TRIGGER\_CFG | | MAN\_HSUM\_DM\_TRIGGER\_CFG | | MAN\_CLD\_ENABLE\_CFG | | MAN\_CLD\_PAUSE\_EN\_CFG | | MAN\_CLD\_PAUSE\_RST\_CFG | | MAN\_CLD\_PAUSE\_CNT[31:0] | | MAN\_CONV\_ENABLE\_CFG | | MAN\_CONV\_PAUSE\_EN\_CFG | | MAN\_CONV\_PAUSE\_RST\_CFG | | MAN\_CONV\_PAUSE\_CNT[31:0] | | MAN\_HSUM\_ENABLE\_CFG | | MAN\_HSUM\_PAUSE\_EN\_CFG | | MAN\_HSUM\_PAUSE\_RST\_CFG | | MAN\_HSUM\_PAUSE\_CNT[31:0] | |  | |  | |  | |  | | **Module CONV** | | MC\_OVERLAP\_SIZE\_CFG[9:0] | | MC\_FC\_CFG{1:42}{1:1024}[63:0] | |  | |  | | **Module HSUM** | | MC\_HSUM\_A\_SET\_CFG | | MC\_B\_START\_CFG{1:2}[21:0] | | MC\_B\_STOP\_CFG{1:2}[21:0] | | MC\_P\_EN\_CFG{1:2}{1:3}[4:0] | | MC\_H\_CFG{1:2}[2:0] | | MC\_A\_CFG{1:2}[3:0] | | MC\_HPSEL\_CFG{1:2}{1:3}{1:8}{1:11}{1:21}[6:0] | | MC\_M\_CFG[31:0] | | MC\_T\_CFG{1:2}{1:21}{1:8}[31:0] | | MC\_THRESH\_SET\_CFG[21:0] | | MC\_TRESHOLD\_FILTER\_EN\_CFG | | MC\_T\_RESULTS{1:2}{1:2}{1:25}{1:8}{1:4}[31:0] | | MC\_T\_EXCEED{1:8}[31:0] | | MC\_S\_EXCEED{1:8}[28:0] | | MC\_P\_EXCEED{1:8}[6:0] | | MC\_DM\_CNT[31:0] | | MC\_DM\_CNT\_RESET\_CFG | | The MC interface shall be provided by the PCIe Interface via the PCIF module. Shortened names of the MC registers shall be used:-   | MC register | | --- | | **Module: CTRL** | | DM\_TRIG | | PAGE | | MAN\_OVERRIDE | | MAN\_CLD\_ TRIG | | MAN\_CONV\_ TRIG | | MAN\_HSUM\_ TRIG | | MAN\_CLD\_EN | | MAN\_CLD\_PAUSE\_EN | | MAN\_CLD\_PAUSE\_RST | | MAN\_CLD\_PAUSE\_CNT[31:0] | | MAN\_CONV\_EN | | MAN\_CONV\_PAUSE\_EN | | MAN\_CONV\_PAUSE\_RST | | MAN\_CONV\_PAUSE\_CNT[31:0] | | MAN\_HSUM\_EN | | MAN\_HSUM\_PAUSE\_EN | | MAN\_HSUM\_PAUSE\_RST | | MAN\_HSUM\_PAUSE\_CNT[31:0] | | LATCHED\_CLD\_DONE | | LATCHED\_CONV\_DONE | | LATCHED\_HSUM\_DONE | |  | | **Module CONV** | | OVERLAP\_SIZE [9:0] | | FILTER\_{1:42}\_COEFFICEINTS{1:2047}[31:0] | | INSERT\_COUNT\_EN | | FFT\_{1:1024}\_[31:0] | | **Module HSUM** | | HSUM\_A\_SET | | B\_START\_{1:2}[21:0] | | B\_STOP\_{1:2}[21:0] | | HSUM\_{1:3}\_P\_{1:2}[4:0] | | NUM\_H\_ {1:2}[2:0] | | NUM\_A\_ {1:2}[3:0] | | HSUM\_{1:3}\_HPSEL\_{0:7}[6:0] | | M\_VALUE[31:0] | | HSUM\_ {1:3]\_HARM\_{0:7]\_T[31:0] | | B\_THRESH\_SET [21:0] | | TRESH\_FILTER\_EN | | HARM\_{0:7}\_T\_RESULTS[31:0] | | HARM\_{0:7}\_T\_EXCEED[31:0] | | HARM\_{0:7}\_S\_EXCEED[28:0] | | HARM\_{0:7}\_P\_EXCEED[6:0] | | MC\_DM\_CNT[31:0] | | DM\_CNT\_RESET | |
| FDAS.DIAGNOSTIC:010/A | It shall be possible, via the PCIe, for the MC Interface to have read/write access to any location of the external DDR memory. Ideally block accesses shall be possible so that a region of the DDR external memory can be accessed via the PCIe with a minimum number of read/write commands. | The PCIe Interface has to be able to access all the external DDR4 SDRAM locations. |
| FDAS.DIAGNOSTIC:040/A | It shall be possible, via the PCIe, for the MC Interface to have read/write access to results of the initial FFT within the convolution process. |

Table ‑ : PCIe Hard IP Macro Design Requirement Tags

# Interface Specification

See the PLATFORM DESIGNER environment for the Interfaces. The exported interfaces to allow connection to the PCIF, MSIX and DDRIF2 modules are detailed below:-

| Intel Quartus Prime Platform Designer Signal Name  (Intel PCIe Macro Port name in brackets) | FDAS PCIe Hard IP Macro VHDL Wrapper Signal Name | Direction | Clock Domain | Description |
| --- | --- | --- | --- | --- |
|  | **PCIe Interface** |  |  |  |
| intel\_pcie\_ptile\_mcdma\_0\_hip\_serial\_tx\_p\_out[15:0]  (tx\_out[15:0])  intel\_pcie\_ptile\_mcdma\_0\_hip\_serial\_tx\_n\_out[15:0] | TX\_OUT[15:0] (true)  TX\_OUT[15:0]\_N (comp) | OUT | CLK\_REF | Transmit PCIe data to the PC/Computer. (16 lanes) |
| intel\_pcie\_ptile\_mcdma\_0\_hip\_serial\_rx\_p\_in[15:0]  (rx\_in[15:0])  intel\_pcie\_ptile\_mcdma\_0\_hip\_serial\_rx\_n\_in[15:0] | RX\_IN[15:0] (true)  RX\_IN[15:0]\_N (comp) | IN | CLK\_REF | Receive PCIe data from the PC/Computer. (16 lanes) |
|  |  |  |  |  |
|  | **PCIE Hard IP Macro Read DMA Avalon MM Master Interface to DDRIF2 modules (via Avalon Bridge)**  **(Host to Device)** |  |  |  |
| rd\_dma\_0\_write  (h2ddm\_write\_o)  rd\_dma\_1\_write  (h2ddm\_write\_o)  rd\_dma\_2\_write  (h2ddm\_write\_o)  rd\_dma\_3\_write  (h2ddm\_write\_o) | RD\_DMA\_0\_WRITE\_O  (to DDRIF2#1)  RD\_DMA\_1\_WRITE\_O  (to DDRIF2#2)  RD\_DMA\_2\_WRITE\_O  (to DDRIF2#3)  RD\_DMA\_3\_WRITE\_O  (to DDRIF2#4) | OUT | CLK\_PCIE | When asserted, indicates that the Read DMA module is ready to write read completion data to a memory component in the Avalon-MM address space.  This Interface is used by the Hard IP Macro DMA Controller to pass the DMA data read from the PC/Computer and write it to the DDRIF2 modules. |
| rd\_dma\_0\_address[25:0]  (h2ddm\_address\_o[63:0])  rd\_dma\_1\_address[25:0]  (h2ddm\_address\_o[63:0])  rd\_dma\_2\_address[25:0]  (h2ddm\_address\_o[63:0])  rd\_dma\_3\_address[25:0]  (h2ddm\_address\_o[63:0])  Note only 26 bits of the 64 bit address are used | RD\_DMA\_0\_ADDRESS\_O[25:0]  (to DDRIF2#1)  RD\_DMA\_1\_ADDRESS\_O[25:0]  (to DDRIF2#2)  RD\_DMA\_2\_ADDRESS\_O[25:0]  (to DDRIF2#3)  RD\_DMA\_3\_ADDRESS\_O[25:0]  (to DDRIF2#4)  **Note the Address is a WORD address** | OUT | CLK\_PCIE | Specifies the write address in the Avalon-MM address space for the read completion data.  This address is either the address of a DDR location |
| rd\_dma\_0\_writedata[511:0]  (h2ddm\_writedata\_o[511:0])  rd\_dma\_1\_writedata[511:0]  (h2ddm\_writedata\_o[511:0])  rd\_dma\_2\_writedata[511:0]  (h2ddm\_writedata\_o[511:0])  rd\_dma\_3\_writedata[511:0]  (h2ddm\_writedata\_o[511:0]) | RD\_DMA\_0\_WRITEDATA\_O[511:0]  (to DDRIF2#1)  RD\_DMA\_1\_WRITEDATA\_O[511:0]  (to DDRIF2#2)  RD\_DMA\_2\_WRITEDATA\_O[511:0]  (to DDRIF2#3)  RD\_DMA\_3\_WRITEDATA\_O[511:0]  (to DDRIF2#4) | OUT | CLK\_PCIE | The read completion data to be written to the Avalon-MM address space.  This Interface is used by the Hard IP Macro DMA Controller to pass the DMA data read from the PC/Computer and write it to the DDRIF2 modules. |
| rd\_dma\_0\_burstcount[3:0]  (h2ddm\_burstcount\_o[3:0])  rd\_dma\_1\_burstcount[3:0]  (h2ddm\_burstcount\_o[3:0])  rd\_dma\_2\_burstcount[3:0]  (h2ddm\_burstcount\_o[3:0])  rd\_dma\_3\_burstcount[3:0]  (h2ddm\_burstcount\_o[3:0]) | RD\_DMA\_0\_BURSTCOUNT\_O [3:0]  (to DDRIF2#1)  RD\_DMA\_1\_BURSTCOUNT\_O [3:0]  (to DDRIF2#2)  RD\_DMA\_2\_BURSTCOUNT\_O [3:0]  (to DDRIF2#3)  RD\_DMA\_3\_BURSTCOUNT\_O [3:0]  (to DDRIF2#4) | OUT | CLK\_PCIE | Specifies the burst count in 512-bit words for the DMA data read from the PC/Computer and written to the DDRIF2 modules. This bus is 4 bits for the 512-bit interface |
| rd\_dma\_0\_byteenable[63:0]  (h2ddm\_byteenable\_o[63:0])  rd\_dma\_1\_byteenable[63:0]  (h2ddm\_byteenable\_o[63:0])  rd\_dma\_2\_byteenable[63:0]  (h2ddm\_byteenable\_o[63:0])  rd\_dma\_3\_byteenable[63:0]  (h2ddm\_byteenable\_o[63:0]) | RD\_DMA\_0\_BYTEENABLE\_O[63:0]  (to DDRIF2#1)  RD\_DMA\_1\_BYTEENABLE\_O[63:0]  (to DDRIF2#2)  RD\_DMA\_2\_BYTEENABLE\_O[63:0]  (to DDRIF2#3)  RD\_DMA\_3\_BYTEENABLE\_O[63:0]  (to DDRIF2#4) | OUT | CLK\_PCIE | Specifies which bytes of a word are valid for the DMA data read from the PC/Computer and written to the DDRIF2 modules. |
| rd\_dma\_0\_waitrequest  (h2ddm\_waitrequest\_i)  rd\_dma\_1\_waitrequest  (h2ddm\_waitrequest\_i)  rd\_dma\_2\_waitrequest  (h2ddm\_waitrequest\_i)  rd\_dma\_3\_waitrequest  (h2ddm\_waitrequest\_i) | RD\_DMA\_0\_WAITREQUEST\_I  (from DDRIF2#1)  RD\_DMA\_1WAITREQUEST\_I  (from DDRIF2#2)  RD\_DMA\_2\_WAITREQUEST\_I  (from DDRIF2#3)  RD\_DMA\_3\_WAITREQUEST\_I  (from DDRIF2#4) | IN | CLK\_PCIE | When asserted, indicates that the DDRIF2 module is not ready to receive data from the PC/Computer |
|  | **PCIE Hard IP Macro Write DMA Avalon MM Master Interface to DDRIF2 modules (via Avalon Bridge)**  **(Device to Host)** |  |  |  |
| wr\_dma\_0\_read  (d2hdm\_read\_o)  wr\_dma\_1\_read  (d2hdm\_read\_o)  wr\_dma\_2\_read  (d2hdm\_read\_o)  wr\_dma\_3\_read  (d2hdm\_read\_o) | WR\_DMA\_0\_READ\_O  (to DDRIF2#1)  WR\_DMA\_1\_READ\_O  (to DDRIF2#2)  WR\_DMA\_2\_READ\_O  (to DDRIF2#3)  WR\_DMA\_3\_READ\_O  (to DDRIF2#4) | OUT | CLK\_PCIE | When asserted, indicates that the Write DMA module is reading data from a memory component in the Avalon-MM address space to write to the PCIe address space.  A DMA transfer from the DDRIF2 module to the PC/Computer is performed. |
| wr\_dma\_0\_response[1:0]  (d2hdm\_response\_i[1:0])  wr\_dma\_1\_response  (d2hdm\_response\_i[1:0])  wr\_dma\_2\_response[1:0]  (d2hdm\_response\_i[1:0])  wr\_dma\_3\_response  (d2hdm\_response\_i[1:0]) | WR\_DMA\_0\_RESPONSE\_I[1:0]  (for DDRIF2#1)  WR\_DMA\_0\_RESPONSE\_I[1:0]  (for DDRIF2#1)  WR\_DMA\_2\_RESPONSE\_I[1:0]  (for DDRIF2#3)  WR\_DMA\_3\_RESPONSE\_I[1:0]  (for DDRIF2#4) | IN | CLK\_PCIE | Response code for the DMA access to read data from the DDR memory. This should be hardcoded to “00” = OK for the Intel Agilex PCIe Hard IP Macro. |
| wr\_dma\_0\_address[25:0]  (d2hdm\_address\_o[63:0])  wr\_dma\_1\_address[25:0]  (d2hdm\_address\_o[63:0])  wr\_dma\_2\_address[25:0]  (d2hdm\_address\_o[63:0])  wr\_dma\_3\_address[25:0]  (d2hdm\_address\_o[63:0])  Note only 26 bits of the 64 bit address are used | WR\_DMA\_0\_ADDRESS\_O[25:0]  (to DDRIF2#1)  WR\_DMA\_1\_ADDRESS\_O[25:0]  (to DDRIF2#2)  WR\_DMA\_2\_ADDRESS\_O[25:0]  (to DDRIF2#3)  WR\_DMA\_3\_ADDRESS\_O[25:0]  (to DDRIF2#4)  **Note the Address is a WORD address** | OUT | CLK\_PCIE | Specifies the address for the data to be read from a memory component in the Avalon-MM address space.  This is the address of a DDR location. |
| wr\_dma\_0\_readdata[511:0]  (d2hdm\_readdata\_i[511:0])  wr\_dma\_1\_readdata[511:0]  (d2hdm\_readdata\_i[511:0])  wr\_dma\_3\_readdata[511:0]  (d2hdm\_readdata\_i[511:0])  wr\_dma\_4\_readdata[511:0]  (d2hdm\_readdata\_i[511:0]) | WR\_DMA\_0\_READDATA\_I[511:0]  (from DDRIF2#1)  WR\_DMA\_1\_READDATA\_I[511:0]  (from DDRIF2#2)  WR\_DMA\_2\_READDATA\_I[511:0]  (from DDRIF2#3)  WR\_DMA\_3\_READDATA\_I[511:0]  (from DDRIF2#4) | IN | CLK\_PCIE | Specifies the completion data that will be written to the PCIe address space by the Write DMA module.  This is the data that is DMA transferred from the DDR memory to the PC/Computer. |
| wr\_dma\_0\_burstcount[3:0]  (d2hdm\_burstcount\_o[3:0])  wr\_dma\_1\_burstcount[3:0]  (d2hdm\_burstcount\_o[3:0])  wr\_dma\_2\_burstcount[3:0]  (d2hdm\_burstcount\_o[3:0])  wr\_dma\_3\_burstcount[3:0]  (d2hdm\_burstcount\_o[3:0]) | WR\_DMA\_0\_BURSTCOUNT\_O[3:0]  (to DDRIF2#1)  WR\_DMA\_1\_BURSTCOUNT\_O[3:0]  (to DDRIF2#2)  WR\_DMA\_2\_BURSTCOUNT\_O[3:0]  (to DDRIF2#3)  WR\_DMA\_3\_BURSTCOUNT\_O[3:0]  (to DDRIF2#4) | OUT | CLK\_PCIE | Specifies the burst count in 512 bit words for the data to be read from DDR memory and passed to the PC/Computer. This bus is 4 bits for the 512-bit interface. |
| wr\_dma\_0\_waitrequest  (d2hdm\_waitrequest\_i)  wr\_dma\_1\_waitrequest  (d2hdm\_waitrequest\_i)  wr\_dma\_2\_waitrequest  (d2hdm\_waitrequest\_i)  wr\_dma\_3\_waitrequest  (d2hdm\_waitrequest\_i) | WR\_DMA\_0\_WAITREQUEST\_I  (from DDRIF2#1)  WR\_DMA\_1\_WAITREQUEST\_I  (from DDRIF2#2)  WR\_DMA\_2\_WAITREQUEST\_I  (from DDRIF2#3)  WR\_DMA\_3\_WAITREQUEST\_I  (from DDRIF2#4) | IN | CLK\_PCIE | When asserted, indicates that the memory is not ready to be read.  The DDRIF2 module shall assert this if it is not ready to supply data to the PC/Computer. |
| wr\_dma\_0\_readdatavalid  (d2hdm\_readdatavalid\_i)  wr\_dma\_1\_readdatavalid  (d2hdm\_readdatavalid\_i)  wr\_dma\_2\_readdatavalid  (d2hdm\_readdatavalid\_i)  wr\_dma\_3\_readdatavalid  (d2hdm\_readdatavalid\_i) | WR\_DMA\_0\_READDATAVALID\_I  (from DDRIF2#1)  WR\_DMA\_1\_READDATAVALID\_I  (from DDRIF2#2)  WR\_DMA\_2\_READDATAVALID\_I  (from DDRIF2#3)  WR\_DMA\_3\_READDATAVALID\_I  (from DDRIF2#4) | IN | CLK\_PCIE | When asserted, indicates that WR\_DMA\_\*\_  READDATA\_I is valid.  The DDRIF2 module shall assert this signal when the data is valid. |
|  |  |  |  |  |
|  | **PCIE Hard IP Macro RX Master Interface to PCIF module** |  |  |  |
| rxm\_bar2\_0\_m0\_read  (rx\_pio\_read\_o) | RXM\_READ\_O | OUT | CLK\_PCIE | Read Request (to read from FDAS) Indication Issued by the PC/Computer.  ‘1’ = Read Request. |
| rxm\_bar2\_0\_m0\_write  (rx\_pio\_write\_o) | RXM\_WRITE\_O | OUT | CLK\_PCIE | Write Request (to write to FDAS) Indication Issued by the PC/Computer.  ‘1’ = Write Request. |
| rxm\_bar2\_0\_m0\_address[21:0]  (22 bit slice of rx\_pio\_address\_o[36:0])  Note only the lower 22 bits of the address are used. The upper 15 bits contain Virtual and Physical machine numbers which are not used by FDAS. | RXM\_ADDRESS\_O [21:0]  **Note the Address is a WORD address** | IN | CLK\_PCIE | FDAS Address location that the PC/Computer wishes to read from or write to. |
| rxm\_bar2\_0\_m0\_byteenable [7:0]  (rx\_pio\_byteenable\_o[7:0])  Note only the lower 4 byte enable bits are used as the PCIF module only supports a 32-bit data bus. | RXM\_BYTEENABLE\_O [3:0] | IN | CLK\_PCIE | Enables for the 32-bit word from the PC/Computer when a write to FDAS is requested. |
| rxm\_bar2\_0\_m0\_writedata [63:0]  (rx\_pio\_writedata[63:0])  Note only the lower 32 bits of the write data are used by the PCIF module. | RXM\_WRITEDATA\_O [31:0] | IN | CLK\_PCIE | Data from the PC/Computer to be written to FDAS. |
| rxm\_bar2\_0\_m0\_readdata [63:0]  (rx\_pio\_readdata[63:0])  Note only the lower 32 bits of read data are provided by the PCIF module. The upper 32 bits are hard coded to 0. | RXM\_READDATA\_I [31:0] | IN | CLK\_PCIE | Data from the selected Address of FDAS to be sent to the PC/ Computer. |
| rxm\_bar2\_0\_m0\_readdatavalid (rx\_pio\_readdatavalid\_i) | RXM\_READDATAVALID\_I | IN | CLK\_PCIE | Indication that the Read data from the selected Address of FDAS is valid.  ‘1’ = Read Data is Valid |
| rxm\_bar2\_0\_m0\_waitrequest  (rx\_pio\_waitrequest\_i) | RXM\_WAITREQUEST\_I | IN | CLK\_PCIE | Request by PCIF to indicate that it is not ready to respond to the Read/Write Request from the PC/Computer. |
| rxm\_bar2\_0\_m0\_writeresponsevalid  (rx\_pio\_writeresponsevalid\_i) | RXM\_WRITE\_RESPONSE\_  VALID\_I | IN | CLK\_PCIE | Write Response Valid signal from PCIF to indicate the write access has completed |
| rxm\_bar2\_0\_m0\_response  (rx\_pio\_response\_i[1:0]) | RXM\_RESPONSE\_I[1:0] | IN | CLK\_PCIE | Write Response signal from PCIF. Set to “00” |
|  |  |  |  |  |
|  | **PCIE Hard IP Macro MSI-X Interface to MSIX module** |  |  |  |
| intel\_pcie\_ptile\_mcdma\_0\_usr\_msix\_ready  (usr\_event\_msix\_ready\_o) | USER\_MSIX\_READY\_O | OUT | CLK\_PCIE | PCIe Hard IP Macro is ready to receive a user interrupt vector table number |
| intel\_pcie\_ptile\_mcdma\_0\_usr\_msix\_data[15:0]  (usr\_event\_msix\_data\_i[15:0]) | USER\_MSIX\_DATA\_I[15:0]  Bits[15:12] = Reserved  Bit[11] = msix\_queue\_dir (0 = D2H, 1= H2D)  Bits[10:0]: DMA Channel number | IN | CLK\_PCIE | Interrupt Vector Table Channel Number and Direction from the MSIX module. This is an address to the vector table which is loaded with Interrupt vector information configured by the PC/Computer |
| intel\_pcie\_ptile\_mcdma\_0\_usr\_msix\_valid  (usr\_event\_msix\_valid\_i) | USER\_MSIX\_VALID\_I | IN | CLK\_PCIE | Indication that USER\_MSIX\_DATA\_I is valid. |
|  |  |  |  |  |
|  | **Global Clock/Resets** |  |  |  |
| intel\_pcie\_ptile\_mcdma\_0\_refclk0\_clk  (refclk0)  intel\_pcie\_ptile\_mcdma\_0\_refclk1\_clk  (refclk1) | CLK\_REF\_I\_0  CLK\_REF\_I\_1 | IN | - | 100MHz Reference Clocks to the PCIe Hard IP Macro. |
| clk\_out\_clk  (app\_clk) | CLK\_PCIE\_O | OUT | - | 350MHz clock from the PCIe Hard IP macro to the Application Logic |
| intel\_pcie\_ptile\_mcdma\_0\_pin\_perst\_reset\_n (pin\_perst) | PIN\_PERST\_I | IN | - | Active Low reset from the pin of the device |
| reset\_out\_reset\_n  (app\_rst\_n) | RST\_PCIE\_N\_O | OUT | - | Resets DMA Soft blocks and user logic. RST\_PCIE\_N\_O is asserted when software writes to SW\_RESET register bit[0] |
| intel\_pcie\_ptile\_mcdma\_0\_ninit\_done\_reset (ninit\_done) | NINIT\_DONE\_I | IN | - | Active Low signal. A “1” indicates that the FPGA device is not yet fully configured.  A "0" indicates the device has been  configured and is in normal operating  mode.  To use the ninit\_done input, instantiate  the Reset Release Intel FPGA IP in your  design and use its ninit\_done output.  The Reset Release IP is required in  Intel Stratix 10 design. It holds the  Multi Channel DMA for PCI Express IP  in reset until the FPGA is fully  configured and has entered user mode. |

Table ‑ : Intel Agilex F PCIe Hard IP Macro Pinlist

# MCI Memory Mapped Interface

The PCIe Hard IP macro has Type 0 configuration space and DMA Descriptor configuration. This shall be detailed in a later release of this document when the Intel Agilex PCIe driver software has been evaluated.

# Design Parameterisation

The PCIe Hard IP Macro module does not have any generic parameterisation.

# F Series Development Card External PCIe Pinout

The target Agilex F device chosen for the translation is:-

AGFB014R24A2E2VR0

This device is fitted to the Intel Agilex F-Series Development Board [Ref: Agilex Development Board].

The required PCIe Pinout for this Intel Agilex F-Series Development Board is described below. The 16 PCIe lanes, 100MHz reference clocks and reset are connected to the following FPGA pins:-

PIN\_BU58 -to PIN\_PERST\_N

PIN\_AJ48 -to CLK\_PCIE\_REF\_0 *(REF\_CLK\_0)*

PIN\_AH49 -to CLK\_PCIE\_REF\_0\_N

PIN\_AE48 -to CLK\_PCIE\_REF\_1 *(REF\_CLK\_1)*

PIN\_AD49 -to CLK\_PCIE\_REF\_1\_N

PIN\_BP55 -to TX\_OUT0

PIN\_BR56 -to TX\_OUT0\_N

PIN\_BN52 -to TX\_OUT1

PIN\_BM53 -to TX\_OUT1\_N

PIN\_BK55 -to TX\_OUT2

PIN\_BL56 -to TX\_OUT2\_N

PIN\_BJ52 -to TX\_OUT3

PIN\_BH53 -to TX\_OUT3\_N

PIN\_BF55 -to TX\_OUT4

PIN\_BG56 -to TX\_OUT4\_N

PIN\_BE52 -to TX\_OUT5

PIN\_BD53 -to TX\_OUT5\_N

PIN\_BB55 -to TX\_OUT6

PIN\_BC56 -to TX\_OUT6\_N

PIN\_BA52 -to TX\_OUT7

PIN\_AY53 -to TX\_OUT7\_N

PIN\_AV55 -to TX\_OUT8

PIN\_AW56 -to TX\_OUT8\_N

PIN\_AU52 -to TX\_OUT9

PIN\_AT53 -to TX\_OUT9\_N

PIN\_AP55 -to TX\_OUT10

PIN\_AR56 -to TX\_OUT10\_N

PIN\_AN52 -to TX\_OUT11

PIN\_AM53 -to TX\_OUT11\_N

PIN\_AK55 -to TX\_OUT12

PIN\_AL56 -to TX\_OUT12\_N

PIN\_AJ52 -to TX\_OUT13

PIN\_AH53 -to TX\_OUT13\_N

PIN\_AF55 -to TX\_OUT14

PIN\_AG56 -to TX\_OUT14\_N

PIN\_AE52 -to TX\_OUT15

PIN\_AD53 -to TX\_OUT15\_N

PIN\_BP61 -to RX\_IN0

PIN\_BR62 -to RX\_IN0\_N

PIN\_BN58 -to RX\_IN1

PIN\_BM59 -to RX\_IN1\_N

PIN\_BK61 -to RX\_IN2

PIN\_BL62 -to RX\_IN2\_N

PIN\_BJ58 -to RX\_IN3

PIN\_BH59 -to RX\_IN3\_N

PIN\_BF61 -to RX\_IN4

PIN\_BG62 -to RX\_IN4\_N

PIN\_BE58 -to RX\_IN5

PIN\_BD59 -to RX\_IN5\_N

PIN\_BB61 -to RX\_IN6

PIN\_BC62 -to RX\_IN6\_N

PIN\_BA58 -to RX\_IN7

PIN\_AY59 -to RX\_IN7\_N

PIN\_AV61 -to RX\_IN8

PIN\_AW62 -to RX\_IN8\_N

PIN\_AU58 -to RX\_IN9

PIN\_AT59 -to RX\_IN9\_N

PIN\_AP61 -to RX\_IN10

PIN\_AR62 -to RX\_IN10\_N

PIN\_AN58 -to RX\_IN11

PIN\_AM59 -to RX\_IN11\_N

PIN\_AK61 -to RX\_IN12

PIN\_AL62 -to RX\_IN12\_N

PIN\_AJ58 -to RX\_IN13

PIN\_AH59 -to RX\_IN13\_N

PIN\_AF61 -to RX\_IN14

PIN\_AG62 -to RX\_IN14\_N

PIN\_AE58 -to RX\_IN15

PIN\_AD59 -to RX\_IN15\_N

# References

|  |  |  |
| --- | --- | --- |
| **Bookmark** | **Reference** | **Description** |
| [Ref: Agilex PCIe User Guide] | Multi Channel DMA Intel FPGA IP for PCI Express User Guide UG-20297 | 2021.10.29 | Intel PCIe interface with DMA User Guide for the Stratix 10 and Agilex FPGA families using the Intel Quartus Prime software version 21.3 |
| [Ref: Agilex Development Board] | AgilexF\_FPGA\_DK\_2V\_ES.pdf containing:-  AgileX F-Series FPGA Dev Kit\_Enpirion schematics  Document Number: K57065-001-A  Date: Thursday, May 07, 2020 | Intel Agilex F Development Board schematics. This board is fitted with an Agilex AGFB014R24A2E2VR0 FPGA. The board supports a PCIe interface and four DDR4 SDRAM interfaces. |

# Abbreviations and Acronyms

|  |  |
| --- | --- |
| DMA | Direct Memory Access |
| DDR | Double Data Rate |
| FDAS | Fourier Domain Acceleration Search |
| FFT | Fast Fourier Transform |
| FOP | Filter Output Plane |
| FPGA | Field Programmable Gate Array |
| IEEE | Institute of Electrical Engineers |
| IP | Intellectual Property |
| MC | Micro Controller Interface |
| MSI-X | Extended Message Signalled Interrupt |
| PCIe | Peripheral Component Interconnect Express |
| SDRAM | Synchronous Dynamic RAM |
| TLP | Transaction Layer Packet |