

**FDAS Design Description for Implementation in the Intel Agilex F Development Board With Three DDR Interfaces Used**

FDAS \_DS Revision 4 Draft E

Classification: UNRESTRICTED

Document type: DTE

Date: 2024-01-03

Status:

© Copyright 2024 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 |  | 03/01/2024 |
| 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 4 Draft A | 05/05/2023 | First Issue of the FDAS Implementation specification for the Intel Agilex F Development board with three DDR SDRAM Interfaces. |
| Issue 4 Draft B | 10/05/2023 | Comment 1:  AT4-997 : Ben Stappers  It might be helpful to indicate how the sums in the yellow boxes are calculated and why it is a conditional, just so one doesn't need to check the equation (In the inserted FDAS\_PROCESSING\_TIME spreadsheet)  Response: The spreadsheet has been updated to explain the conditional calculations. |
| Issue 4 Draft C | 17/05/2023 | Re-build to correct a bug in the read address from HSUM to DDRIF2 #2 (typo error in signal name) |
| Issue 4 Draft D | 05/08/2023 | 1. The DDRIF2 resets now reset the DDR clock domain circuits as well as the system clock circuits as the DDR Controller resets take up to one second to occur. This makes repeated use of the DDR Controller resets to reset the DDRIF2 DDR Clock domain circuits impractical. 2. The CTRL module now reports the number of CONV and HSUM DDR SDRAM accesses for diagnostic purposes. 3. MCI\_TOP module reset descriptions clarified |
| Issue 4 Draft E | 03/01/2024 | * The PCIE Reset now has separate synchronisers for each DDRIF2 module and the MSIX module. * Added the Switch Positions for the Agilex Development Board (Appendix C) * Added Pulsar Acceleration Equations (Appendix D). * Added Convolution Filter Template Description (Appendix E) * Added FDAS Configuration information (Appendix F). * Added extra descriptions to the PCIe overview in Appendix A * Memory Map addresses now for 64 bit words as this is what is required for FDAS in Agilex. * Added CvP Download Desription to Appendix B * Added DDR Base Address to the Memory Map |

**Table of Contents**

1 Introduction 6

2 FDAS Functional Description 7

2.1 FDAS Implementation Overview 9

2.1.1 FDAS Design Scaling 10

2.1.2 Theoretical DM Processing Time 15

2.1.3 Processing Flow for this FDAS Implementation 23

2.1.4 Programming Guide 26

2.1.5 Design Requirement Tag Compliance 32

3 FDAS Clocking Architecture 40

4 FDAS Reset Architecture 42

5 FDAS FPGA Pinout 44

6 External DDR4 SDRAM Memory Organisation 58

6.1 Input Samples for a DM 59

6.1.1 Point A: DM Samples: PC/Computer <> PCIe Hard IP Macro 59

6.1.2 Point B: DM Samples: PCIe Hard IP Macro <> DDRIF2 60

6.1.3 Point C: DM Samples: DDRIF2 <> DDR CONTROLLER / CLD 60

6.1.4 Point D: DM Samples: DDR CONTROLLER <> SDRAM 61

6.2 Convolution Results for a DM (FOP) 62

6.2.1 Point F: FOP: DDRIF2 <> CONV/HSUM/DDR CONTROLLER 63

6.2.2 Point H: FOP: DDR CONTROLLER <> SDRAM 64

6.2.3 Point J: FOP: DDR CONTROLLER <> SDRAM 66

6.2.4 Point K: FOP: DDRIF2 <> PCIe Hard IP Macro 67

6.2.5 Point L: FOP: PCIe Hard IP Macro <> PC/Computer 69

7 FDAS MCI Memory mapped Interface 75

7.1 FDAS DDR SDRAM Memory Map 75

7.2 FDAS Configuration Memory Map 75

7.2.1 Address Decode 75

7.2.2 Module MCI\_TOP 76

7.2.3 Module CTRL 80

7.2.4 Module MSIX 88

7.2.5 Module CONV 90

7.2.6 Module HSUM 93

8 FDAS Parameterisation 105

9 FDAS FPGA Resource Utilisation 106

10 FDAS FPGA Timing Analysis 107

11 FDAS FPGA Power Estimate 108

12 FDAS Measured Processing Time 109

13 FDAS Performance Comparisons 110

13.1 Comparison Summary 111

14 Intel Agilex F Card Measured Power Consumption 112

14.1 Power Characteristic with FDAS Inactive 112

14.2 Power Characteristic with FDAS Operating Continuously 113

15 Appendix A: Overview of PCIe in Agilex F 114

15.1 PCIe Physical Protocol Layers 114

15.2 PCIe Configuration Space 117

15.2.1 Base Address Register 120

15.2.2 PCIe Bus Enumeration 121

15.3 Altera Agilex F Avalon-MM DMA 122

15.3.1 Agilex PCIe with Avalon-MM Multi Channel DMA Engine 122

16 APPENDIX B: Configuration Via PCIe (CvP) 134

16.1 Creating the CvP Image using Intel Quartus Prime 135

16.2 Downloading the FDAS Design Using CvP 142

16.2.1 Downloading the Periphery Image 142

16.2.2 Downloading the Core Image 143

17 APPENDIX C: Agilex Development Board Switch Positions 145

18 APPENDIX D: Accelerating Pulsar Equations 150

18.1 Accelerating Pulsar Equations 150

18.2 Derivation of Equations 150

18.3 Determining the Acceleration from the FOP 153

19 APPENDIX E: CONV Filter Templates 155

19.1 Template Generation 155

19.2 Template Alignment, Padding and Fourier Transform 166

20 Appendix F: FDAS Configuration 167

21 Appendix G: PowerPoint Presentation of FDAS 168

22 References 169

23 Abbreviations and Acronyms 170

# Introduction

This document provides a description of the FDAS design for implementation in the Intel Agilex Development Board with three DDR SDRAM interfaces used.[Ref: Agilex Development Board].

The architectural concept for the FDAS FPGA has been previously captured in the “FDAS Requirements and Architecture Specification” [FDAS\_RA].

The FDAS Design supports:-

* Two DDR Interfaces connected to the CONV and HSUM modules, operating in unison to provide increased access bandwidth to the FOP stored in two external SDRAM sticks. The CLD module is connected to a single DDR Interface which is connected to an SDRAM stick containing the incoming observation data.
* An “Enhanced summing tree” supporting up to 12 harmonics
* MSI-X Interrupts to indicate when core processing modules have finished processing a DM.
* Configuration via PCIe (CvP): This requires a “Peripheral Image” (.jic) to be downloaded to Flash, allowing the “Core Image” (.rbf) to be downloaded via PCIe (See Appendix B for details of file generation and programming)

For each module within FDAS a module design specification has been created to describe the module design and operation.

The target card is the Intel Agilex F Development Board fitted with:-

* 1-off Intel Agilex AGFB014R24B2E2V FPGA device.
* 3-off DDR4 Interface slots each fitted with 9-off Micron MTA9ASF1G72PZ-2G9E1UI devices (8-bit data bus, 1GibiByte capacity). Each DDR4 interface supports:-
  + 72-bit interface consisting of a 64-bit data interface and 8-bit ECC (Error Correcting Code) interface. The ECC feature is not used in FDAS as it reduces performance.
  + 2400 Mega-Transfers per second.
  + 8 Gibi-Bytes memory capacity.

The Intel Agilex card has a 16-lane Gen 4 PCIe Interface allowing direct connection to the PCIe bus in a PC. Each lane supports 2 Gbyte/sec, hence 32 Gbyte/sec overall.

This implementation has been built using the Intel Quartus Prime tool version 22.4.

# FDAS Functional Description

The Architecture for this implementation of FDAS is shown in the figure below:-

**DDRIF2 #2**

PCIe

**CLD**

**CTRL**

FDAS Control

CLD\_DONE

**CONV**

data[63:0]

ready

**HSUM**

**FDAS**

CONV\_DONE CONV\_FFT\_READYY

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 #0**

**DDRIF2 #2 and #3 acting in unison to provide a 1024-bit data interface to CONV and HSUM**

ADDR

2xDATA[5111:0]

2xDATA[511:0]

ADDR

ADDR

DATA[511:0]

*Note: The PCIe Hard Macro can read and write to all External 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**

**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 #0**

**External 4 Gibi-Byte**

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

MC

Interface

Via

PCIe

*Note: In this implementation CLD\_PAGE, CONV\_PAGE and HSUM\_PAGE are fixed (static) in this implementation.*

MC Bus

OVERLAP\_SIZE[9:0]

FOP\_SAMPLE\_NUM[22:0]

**MSIX**

MSI-X

Figure ‑ : FDAS Implementation

The figure below shows the detailed connectivity of the DDR Controller modules the FDAS design in more detail.

Figure ‑: Detailed DDR Controller Connectivity

PCIE

Hard IP

Macro

CLD

Module

CONV

Module

HSUM

Module

DDRIF2 #0 Module containing clock boundary FIFOs

DDR 4 SDRAM#0

**FDAS**

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

512-bit @ 350MHz

1024-bit @ 350MHz

1024-bit @ 350MHz

64-bit @ 1200MHz

C

A

L

DDR\_

CONT

Ch 0

FDAS\_DDR\_

CONTROLLER\_HPS

CALIBRATION

(INTEL IP)

DDR\_

CONT

HPS

Ch 1

DDRIF2 #2 Module containing clock boundary FIFOs

DDR 4 SDRAM#2

64-bit @ 1200MHz

C

A

L

DDR\_

CONT

Ch 2

FDAS\_DDR\_

CONTROLLER\_

CALIBRATION

(INTEL IP)

DDR\_

CONT

Ch 3

DDRIF2 #3 Module containing clock boundary FIFOs

DDR 4 SDRAM#3

64-bit @ 12000MHz

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

512 bits to/from each DDRIF2

It should be noted that the paging of the DDR4 SDRAM via CLD\_PAGE[31:0], CONV\_PAGE[31:0] and HSUM\_PAGE[31:0] are not required in this implementation and hence they are fixed at a value of 0 by the CTRL module. However the CLD, CONV and HSUM modules are ready to support paging if and when it is required.

## FDAS Implementation Overview

As stated in Sec 1 the main differences between the FDAS architecture described in the “FDAS Requirements and Architecture Specification” [FDAS\_RA] and this FDAS implementation are the simplification of the DDR4 external memory interfaces.

However the main processing modules (CLD, CONV and HSUM) are designed such that they can be used in designs with increased DDR4 external memory bandwidth, such as the design described in this document.

### FDAS Design Scaling

It was realised during the creation of the “FDAS Requirements and Architecture Specification” [FDAS\_RA] that the resource constraints of the target FPGA device and fitting efficiency impact on the performance of FDAS. To provide a flexible design methodology to de-risk the development and support future implementations in larger FPGA devices VHDL generic parameterisation has been used to scale the size of the CLD, CONV and HSUM modules.

In addition parameters configurable by the host PC/Computer via the PCIe and MC interface allow the performance of FDAS to be altered as desired during its operation.

The list of main generics and MC configurable parameters that affect the performance of FDAS are listed in the tables below:-

#### Generic Parameters Affecting Performance

| Generic | Description | Range |
| --- | --- | --- |
| ddr\_g | CONV and HSUM modules each have a generic to indicate how many DDR4 External Memory interfaces the module has access to. Each DDR4 External Memory interface data bus is 512-bits wide internally in the FPGA. Hence for two DDR4 interfaces the data bus is 1024-bits and for three DDR4 interfaces the data bus is 1536-bits wide. An increase in the data bus width increases the bandwidth to the external memory and thus reduces the processing times. | 1, 2 or 3 |
| fft\_g | The “fft\_g” generic set the number of points within the FFTs and IFFTs of the CONV module.  The greater the number of points within an FFT/IFFT the more DSP blocks and internal memory blocks that are required for the FFT.  The number of points within the FFT affects the maximum filter length that can be supported in the Overlap-Save convolution method.  Ideally the number of points in the FFT should be set to the minimum required to support the maximum filter length, thus allowing the FPGA available resource to support the maximum number of IFFTs in parallel.  (The “fft\_abits\_g” generic is set in sympathy with “fft\_g” to define the number of bits for the internal counters). | 256, 512, 1024 |
| ifft\_g | This generic set the number of instances of the CONV\_IFFT sub-module within the CONV module. Each CONV\_IFFT sub-module contains two IFFTs and generates information for two rows of the FOP (equivalent +ve and -ve acceleration convolutions) on each processing iteration loop.  The greater the number of CONV\_IFFT sub-module instances the more rows of the FOP that can be simultaneously generated, and thus the fewer the number of processing iteration loops that are required to generate the complete FOP. | 1 to 42 |
| ifft\_loop\_g | The “ifft\_loop\_g” generic sets the number of times the CONV\_IFFT sub-modules are re-used in a loop to generate the FOP.  This generic along with the ifft\_g generic determines the number of FOP rows generated via the convolution with the filters.  The number of rows within the FOP are given by:-  1 + 2 x (ifft\_g + ifft\_loop\_g­)  (The “ifft\_loop\_g” generic is set in sympathy with “ifft\_g” to define the number of bits for the internal counters). | 1 to 42 |
| summer\_g | This generic sets the number of instances of the SUMMER sub-module within the HSUM module.  This generic determines the number of DSP blocks required to support the Harmonic Summing process.  Ideally the number of SUMMERs is set so that the summing time is of the same order as the time to obtain the FOP data from DDR4 SDRAM. This ensures most efficient use of logic and minimises processing times. | 1, 2, 3 |
| harmonic\_g | The maximum number of harmonics that the HSUM module is capable of supporting.  Currently for the Enhanced Summing Tree this generic is limited to 12. | 8 to 16 |

Table 2‑1: FDAS Generic Parameters

#### Performance affecting Memory Map Configuration Parameters

| MC Configuration  Parameter | Description | Range |
| --- | --- | --- |
| FOP\_SAMPLE\_NUM[22:0] | The FOP\_SAMPLE\_NUM configuration parameter defines the number of samples for the CLD and CONV modules to process to generate the FOP.  This configuration parameter sets the number of columns of the FOP. | 0 to 222 |
| IFFT\_LOOP\_NUM[5:0] | This configuration parameter defines the number of times the CONV\_IFFT sub-modules are actually looped through.  This configuration parameter sets the number of rows of the FOP.  The maximum number of FOP rows that the FPGA can physically support is given by:-  1 + 2 x (ifft\_g + ifft\_loop\_g­)  However this configuration parameter allows a fewer number of FOP rows to be generated for a particular DM if desired. | 1 to 42 |
| OVERLAP\_SIZE[9:0] | This configuration parameter sets the size of the overlap for the Overlap-Save convolution method.  The size of the overlap is related to the length of the convolution filters and determines how quickly the sample can be processed. | 0 to 1023 |
| A\_SET | The configuration parameter sets the number of analysis runs the HSUM module performs on the FOP.  Either one or two analysis runs can be performed. If two runs are performed the time to process the increases. | 0 or 1 |
| B\_START {1:2}[22:0]  B\_STOP {1:2}[22:0] | These configuration parameters set the range of FOP columns that a seed\_f0 is tested for by the HSUM module. The greater the range the greater the HSUM module processing time.  Two sets of values for two analysis runs. | 0 to 222 |
| FOP\_ROW{1:2}[7:0] | This is the number of FOP rows read by the HSUM module from the DDR SDRAM. The greater FOP\_ROW the greater the time to read the necessary FOP information from DDR SDRAM.  Two sets of values for two analysis runs. | 1 to 85 |
| P\_EN{1:2}{1:3}[4:0] | This configuration parameter sets the number of Rows within a FOP column that a seed\_f0 is tested for. The greater the number of rows the greater the HSUM processing time. The maximum is 21 FOP rows.  There are two sets of configuration to support up to two runs set by A\_SET, and is a set per SUMMER sub-module instance within the HSUM module ( max. of three).  Config value of 0 = 1 seed\_f0 | 0 to 20 |
| H{1:2}[3:0] | This configuration parameter sets the number harmonics tested for. The greater the number of harmonics the greater the HSUM module processing time since the SUMMER sub-module operates in a pipeline fashion with the summation results of one harmonic level processed by the summing of the next harmonic level on the next clock cycle. The maximum supported is currently 12 harmonics.  There are two sets of configuration to support up to two runs set by A\_SET.  Config value of 0 = 1 harmonic (i.e. the fundamental) | 0 to 15 |
| A{1:2}[3:0] | This sets the number Acceleration Ambiguity Slopes tested for. The greater the number of slopes the greater the HSUM module processing time. The maximum 11 slopes.  There are two sets of configuration to support up to two runs set by A\_SET.  Config value of 0 = 1 slope | 0 to 10 |

Table 2‑2: FDAS Memory Map Configuration Parameters

### Theoretical DM Processing Time

The equations in the following sub-sections are based on those from the original FDAS Requirements and Architecture Specification [FDAS\_RA] with some refinements.

A spreadsheet has been created to calculate the predicted processing time and is embedded in this document.



#### Predicted DM Processing Time in Agilex

This is the predicted time for an accelerated pulsar search.

For the predicted DM Processing Time the following Generic Parameterisation shall be used as shown in Table 2‑3 below:-

| Generic | Range |
| --- | --- |
| ddr\_g | 2 |
| fft\_g | 1024 |
| ifft\_g | 14 |
| ifft\_loop\_g | 3 |
| summer\_g | 1 |
| harmonic\_g | 8 |

Table 2‑3 : Generic Parameterisation for Predicted DM Processing Time

For the predicted DM Processing Time the following Memory Map Configuration shall be set as shown in Table 2‑4 below:-

| MC Configuration  Parameter | Description | value |
| --- | --- | --- |
| FOP\_SAMPLE\_NUM[22:0] | The FOP\_SAMPLE\_NUM configuration parameter defines the number of samples for the CLD and CONV modules to process to generate the FOP.  This configuration parameter sets the number of columns of the FOP.  This is calculated as follows:-  Desired number of observation samples to process = 222  OVERLAP\_SIZE = 420  New samples processed every 1024 point FFT  = 1024 – 420 = 604.  Number of FFTs to perform =  INT(222/ 604) = 6944  Samples processed = 604 \* 6944  = 4194176 | 4,194,176 (This is as close to 222 as possible taking into account the overlap size of 420) |
| IFFT\_LOOP\_NUM[5:0] | This configuration parameter defines the number of times the CONV\_IFFT sub-modules are actually looped through.  This configuration parameter sets the number of rows of the FOP. The maximum number of FOP rows that the FPGA can physically support is given by:-  1 + 2 x (ifft\_g + ifft\_loop\_g­)  However, this configuration parameter allows a fewer number of FOP rows to be generated for a particular DM if desired. | 3 |
| OVERLAP\_SIZE[9:0] | This configuration parameter sets the size of the overlap for the Overlap-Save convolution method.  The size of the overlap is related to the length of the convolution filters and determines how quickly the sample can be processed. | 420 |
| A\_SET | The configuration parameter sets the number of analysis runs the HSUM module performs on the FOP.  Either one or two analysis runs can be performed. If two runs are performed the time to process the DM increases.  (Setting of 0 = 1 Analysis Run) | 0 only one analysis run |
| B\_START {1:2}[22:0]  B\_STOP {1:2}[22:0] | These configuration parameters set the range of FOP columns that a seed\_f0 is tested for by the HSUM module. The greater the range the greater the HSUM module processing time.  Two sets of values for two analysis runs. | B\_STOP  -BSTART  =262,144 |
| FOP\_ROW{1:2}[7:0] | This is the number of FOP rows read by the HSUM module from the DDR SDRAM. The greater FOP\_ROW the greater the time to read the necessary FOP information from DDR SDRAM.  Two sets of values for two analysis runs. | 85 |
| P\_EN{1:2}{1:3}[4:0] | This configuration parameter sets the number of Rows within a FOP column that a seed\_f0 is tested for. The greater the number of rows the greater the HSUM processing time. The maximum is 21 FOP rows.  There are two sets of configuration to support up to two runs set by A\_SET and is a set per SUMMER sub-module instance within the HSUM module ( max. of three).  (setting of 0 = 1 row) | 20 |
| H{1:2}[3:0] | This configuration parameter sets the number Harmonics tested for. The greater the number of harmonics the greater the HSUM module processing time since the SUMMER sub-module operates in a pipeline fashion with the summation results of one harmonic level processed by the summing of the next harmonic level on the next clock cycle.  There are two sets of configuration to support up to two runs set by A\_SET.  (setting of 0 = 1 harmonic). | 7 |
| A{1:2}[3:0] | This sets the number Acceleration Ambiguity Slopes tested for. The greater the number of slopes the greater the HSUM module processing time. The maximum 11 slopes.  There are two sets of configuration to support up to two runs set by A\_SET.  (setting of 0 = 1 slope) | 10 |

Table 2‑4: FDAS Memory Map Configuration for Predicted DM Processing Time

#### Accelerated Search Theoretical Convolution and FOP Storage Time

##### Theoretical Convolution Processing Time

The chosen architecture pads the 42-off post transformed filters to 1024 coefficients and the FFTs are 1024 points. Hence the only four variables for this function are:-

* The system clock frequency (**f**) in Hz.
* The number of DSP blocks available, since this sets the No. of IFFT instances available (**N\_IFFT**)
* The size of the overlap **OVERLAP\_SIZE** in the “overlap-save” convolution.
* The number of freq-bins (**CFB**) convoluted to form the FOP.

For this analysis 222 samples for the convolution will be used.

The Number of back-to-back convolution segments (CS), the number of cycles to process a segment (CC) and the overall convolution time (CT) are given by the equations:-

The time for the CONV module to perform its processing is defined in the following three equations.

Number of back-to-back 1024-point convolution segments to perform (CS):-

|  |
| --- |
| **CS = (CFB)/(1024 –OVERLAP\_SIZE + 1)** **………eq1** |

Where CFB is the number of freq-bin sample to process.

Number of cycles for each 1024-point convolution (CC):-

**CC = (pipeline cycles for one convolution/total number of convolutions)**

**+ [(cycles for FFT/IFFT streaming) x (No. of Filters / No. of CONV\_IFFT functions available due to limited DSP)]**

|  |
| --- |
| **CC = (2048/CS) + [(1024) x (42/N\_IFFT)]** **………eq2** |

Convolution time (CT), where f = system clock frequency

|  |
| --- |
| **CT = (CS x CC) x (1/f) ………eq3** |

Hence in the current implementation assuming 4,194,176 samples to ensure an integral number of FFTs are processed with an overlap of 420 bins:

CS = (4,194,176)/(1024 – 420 + 1) = 6932.52

CC = (2048/6932.52) + [(1024) x (42/14)] = 3072.15 cycles

CT = (6932.52 x 3071.15) x (1/350 x 106) = **60.85ms**

##### Theoretical Convolution Results (FOP) > DDR4 External Memory Time

The overall DDR access time (DTCONV) is given by:-

DTCONV =

[(32-bit x 85 FOP Rows) x (No. of freq-bins convoluted) / (FOP word efficiency)]

[(No. of DDR Interfaces) x (DDR Bandwidth) x (DDR Efficiency)]

|  |
| --- |
| **DTCONV =**  **(32 x 85 x CFB / FE) ]**  **[ddr\_g x DB x DE] ………eq4** |

In this case with three DDR interfaces the FOP Word efficiency “FE” = 0.89. (i.e. out of every 3072 bit locations read from the DDR4 SDRAM 320 bits are unused).

In this case ddr\_g =2

The DDR bandwidth “DB” at 1200.0MHz = 2 clock edges x 64 bit x 1200.0MHz

= 153.6 Gbit/s

The DDR Efficiency “DE” = 0.8 (i.e. 80% efficient)

Hence in the current implementation assuming 4 Million samples:

DTCONV = [(32 x 85 x 4,194,176/0.89] / [2 x 153.6 x 109 x 0.8]

DTCONV = **52.16ms**

*Note for Interest:-*

*There is actually a hard limit on the DDR writing time, even if the DDR efficiency was 100%. This is based on the fact that internal to the FDAS design the data bus is 512-bits @ 350MHz per DDR interface. For three DDR interfaces with the above requirements, with each FOP Column requiring 2-off 1536-bit transfers* and 4,*194,176 FOP Columns the total write time would be:-*

*Write Time = (4,194,176 x 2/ (350x 106)*

*= 23.9ms*

##### Accelerated Search Theoretical Overall Convolution & FOP Storage Time

Since the Convolution processing and storage of convolution results (i.e. the FOP) occur simultaneously the overall time is limited by the convolution.

Hence the theoretical time to form the FOP in DDR4 External Memory = **60.85ms**

#### Accelerated Search Theoretical Harmonic Summing Time

##### Theoretical Reading FOP Columns from DDR4 External Memory Time

The overall DDR access time (DTHSUM) is given by:-

DTHSUM = (32-bit x 85 FOP Rows x No. of “seed\_f0” FOP col x No. of FOP harmonic cols / FOP word efficiency)]

[(No. of DDR Interfaces) x (DDR Bandwidth) x (DDR Efficiency)]

|  |
| --- |
| **DT = [ (32 x 85 x { B\_STOP – B\_START } x HC / FE)]**  **[ddr\_g x DB x DE] ………eq5** |

In this case with three DDR interfaces the FOP Word efficiency “FE” = 0.89. (i.e. out of every 3072 bit locations read from the DDR4 SDRAM 320 bits are unused).

The number of FOP columns “HC” to read for eight harmonics is set to 40 columns.

|  |  |
| --- | --- |
| Harmonic No. | Number of FOP Columns to read |
| 1 (i.e. fundamental tone) | 1 |
| 2 | 3 |
| 3 | 3 |
| 4 | 5 |
| 5 | 5 |
| 6 | 7 |
| 7 | 7 |
| 8 | 9 |

The number of FOP columns “B\_STOP – B\_START” in which a seed\_f0 may be present is set to 262,144.

Ddr\_g =2

The DDR bandwidth “DB” at 1200.0MHz = 2 clock edges x 64 bit x 1200.0MHz

= 153.6 Gbit/s

The DDR Efficiency “DE” = 0.8 (i.e. 80% efficient)

Hence in the current implementation:

DTHSUM = [(32 x 85 x {262144} x 40 / 0.89)]/[2 x 153.6 x 109 x 0.8]

DTHSUM = **130.40ms**

*Note for Interest:-*

*There is actually a hard limit on the DDR reading time, even if the DDR efficiency was 100%. This is based on the fact that internal to the FDAS design the data bus is 512-bits @ 350MHz per DDR interface. For three DDR interfaces with the above requirements, with each FOP Column requiring 2-off 1536-bit transfers, for 40 FOP columns to be read per seed column and for 262,144 seed columns the total read time would be:-*

*Read Time = (262,144 x 40 x 2)/ (350 x 106)*

*= 59.9ms*

##### Theoretical Harmonic Summing Time

The overall Harmonic Summing processing time (ST) is given by:-

ST = [No. of “seed\_f0” FOP cols x

((No. of “seed\_f0” FOP rows x No. of Orbital Acceleration Ambiguity Slopes)

+ HPSEL RAM Read Stagger + Summer Tree Latency)]

[(No. of SUMMER Instances) x (system clock frequency)]

|  |
| --- |
| **ST = [{B\_STOP – B\_START} x ((P\_EN x A) + HRS + STL)]**  **[NS x f] ………eq6** |

The number of FOP rows “P\_EN” (of a FOP column) in which a seed\_f0 may be present is set to 21 FOP rows.

The number of Acceleration Ambiguity slopes “A” is set to 11.

The HRS (HPSEL RAM Read Stagger) is set to 8. The HPSEL RAM contains the configuration for the FOP row selection. The configured value matches the number of harmonics summed as the read request to the HPSEL RAM is delayed by one clock cycle per harmonic.

The STL (SUMMER Tree Latency) is set to 9. This is the number of cycles latency in the summer tree to process a seed\_f0. This is in addition to the HRS.

The number of FOP columns “B\_STOP – B\_START” in which a seed\_f0 may be present is 262,144.

The number of SUMMER sub-module Instances “NS” within the HSUM module is set to 1.

Hence in the current implementation:

ST = [{262144} x ((21 x 11) + 8 + 9)]/[1 x 350 x 106]

DTHSUM = **185.74ms**

##### Accelerated Search Theoretical Overall Total Harmonic Summing Time

Since the reading of the FOP columns from DDR4 External memory and the Harmonic Summing occur simultaneously the overall time is limited by the Harmonic Summing processing.

Hence the theoretical Harmonic Summing Time = **185.74ms**

#### Accelerated Search Theoretical Overall DM Processing Time

Assuming a requirement of:-

* 4 million samples (222)
* 1024-point FFT/IFFTs
* 85 FOP Rows (i.e. 85 filters)
* Seed\_f0 within the first 262,144 columns of the FOP
* Seed\_f0 in 21 rows of a FOP Column
* 8 Harmonic tones analysed.
* 11 Acceleration Ambiguity Slopes analysed.

With this implementation of FDAS is set to:-

* ddr\_g generic set to 2 (number of DDR4 Interfaces to CONV/HSUM).
* fft\_g generic set to 1024-point FFTs
* ifft\_g generic set to 14 CONV\_IFFT sub-modules in the CONV module
* ifft\_loop\_g generic set to 3 to generate 1 + 2x(3x14) = 85 FOP Rows
* summer\_g generic set to 1 SUMMER sub-module instance in the HSUM module
* harmonic\_g generic set to 8

Theoretical DM processing time = Theoretical Overall Convolution Time + Theoretical Overall Harmonic Summing Time

= 60.85 + 185.75

≈ **247ms**

### Processing Flow for this FDAS Implementation

Starting from the point where software has written the 222 freq-bin observation data for DM #N into the DDR4 external SDRAM Memory # 1 the normal sequence of events for the FDAS processing flow shall be:-

* External Software commands DM #N to be processed via a 0 > 1 transition of “DM\_TRIG” over the PCIe/MC interface.
* On receipt of the “DM\_TRIG”:-
  + The CTRL module automatically asserts a 0 > 1 transition on the CLD\_DM\_TRIGGER, CONV\_DM\_TIGGER:-
    - CLD and CONV modules shall start to process observation data for DM “N”, with the CLD module reading the observation data from DDR4 external SDRAM Memory # 0, and the CONV module writing the FOP information to DDR4 external SDRAM Memory #2 and #3.
* On completion of processing DM #N, the CLD and CONV modules shall assert 0 > 1 transition on the CLD\_DONE and CONV\_DONE signals respectively.
* On receipt of the CLD\_DONE and CONV\_DONE indications:-
  + The CTRL module automatically asserts a 0 > 1 transition on the HSUM\_DM\_TRIGGER:-
    - The HSUM module shall start to process the FOP for the DM “N”, reading the FOP information from DDR4 external SDRAM Memory #2 and #3
  + The PC/Computer detects the CONV\_DONE MSI-X interrupt via the PCIe, and writes the 222 freq-bin observation data for DM #N+1 into the DDR4 external SDRAM Memory # 0.
* On completion of processing DM #N, the HSUM module shall assert 0 > 1 transition on the HSUM\_DONE signal.
* On receipt of the HSUM\_DONE indication:-
  + The PC/Computer detects the HSUM\_DONE MSI-X interrupt via the PCIe, and on detecting that it is asserted, returns to the first step of this sequence (thus triggering the processing of DM #N+1).
  + The PC/Computer examines the Harmonic Summing results for DM #N via the PCIe/MC interface whilst DM #N+1 is being processed.

The operation is summarised in the table below:-

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Host PC/Computer | PCIe/MC Interface | FDAS FPGA | External DDR4 SDRAM Memory #0  222 freq-bin  sequence for the  DM already loaded in SDRAM memory | External DDR4 SDRAM Memories #2,#3 |
| DM\_Trigger  Host PC/Computer receives CONV\_DONE  MSI-X Interrupt and actions DMA transfer of observed data set for the next DM  CONV\_DONE  MSI-X Interrupt  Host PC/Computer receives  HSUM\_DONE  MSI-X Interrupt.  Reads Results for the Harmonically Summed DM whilst the next DM is being processed by FDAS. | DM\_TRIG  Read Results for the Harmonically Summed DM  HSUM\_DONE  MSI-X Interrupt  DMA Transfer of 222 freq-bin  Sequence for the next DM over  PCIe | Results for the Harmonically Summed DM available while the next DM is processed by FDAS  CLD\_DONE & CONV\_DONE initiates Harmonic Summing of the FOP for the DM stored in SDRAM #2,#3  DM\_TRIG  From PC/Computer initiates Convolution of the DM in SDRAM#0 with the FOP stored in SDRAM  #2, #3  222 freq-bin  sequence for the next  DM  HSUM\_DONE  CLD\_DONE  CONV\_DONE | 222 freq-bin  FOP for  The DM |  |

Table 2‑5 : Swim Lane diagram for this Implementation of FDAS

### Programming Guide

#### Programming via the MC Interface over PCIe

The MC Interface is accessed via BAR2 of the PCIe Interface. The user programs the desired base address offset into BAR2 of the Type 0 Configuration space of the FDAS FPGA PCIe. This base address is then added to the addresses detailed in Sec 7.

##### Convolution Programming Guide

The CONV module processes an integral number of FFTs, and hence it is necessary to set the number of samples that are processed to fit into an integral number of FFTs taking into account the overlap.

For example if we have an OVERLAP\_SIZE of 420 (decimal) and 1,024 point FFTs this means that 604 new samples are actually processed by each FFT.

Hence if we wish to process 222 = 4,194,304 samples we will need to perform 4,194,304/604 = 6944.211 FFTs.

If we choose to round this down we will need to process 6944 x 604 = 4,194,176 samples. In this case FOP\_SAMPLE\_NUM = 4,194,175 since the programmed value is one less than the number needed.

If we choose to round this up we will need to process 6945 x 604 = 4,194,780 samples. In this case FOP\_SAMPLE\_NUM = 4,194,779.

In this Implementation of FDAS the ifft\_g generic is set to 14, so that 28 rows of the FOP are generated per processing loop iteration of the CONV module (14 +ve acceleration and 14 -ve acceleration rows per loop). The actual number of processing loops performed is set by IFFT\_LOOP\_NUM (up to the maximum defined by the ifft\_loop\_g generic which is 3 in this case).

Total FOP Rows = 1 + (2 \*( ifft\_g\* IFFT\_LOOP\_NUM))

##### Harmonic Summing Programming Guide

###### “FOP\_COL\_OFFSET” Programming

The “FOP\_COL\_OFFSET[8:0]” register needs to be programmed into the HSUM module to indicate the FOP Column number which contains the first convolution result from CONV. This is due to the fact that the FIR Filter coefficients for the different filter lengths used by CONV have all been shifted by the necessary amount to ensure that a particular freq-bin position of the observation samples appears at the centre of each of the different length filters at the same point in time as the CONV processing progresses. In the current design the FIR filter for FOP Row p[0] (i.e. the centre row of the FOP) is effectively a single element filter, whereas the FIR filter for FOP Rows p[42]/p[-42] is a 421 element filter. Hence in order for an observation sample freq-bin to be at the centre of both filters simultaneously the filter for FOP Row p[0] has to be shifted by (421 -1)/ 2 = 210. This infers that the FOP column that contains the convolution result for the first observation sample freq-bin is Column 210, and this is the value that needs to be programmed into FOP\_COL\_OFFSET[8:0].

This configuration is common to all SUMMER sub-module instances in the HSUM module.

###### “B\_START” & “B\_STOP” Programming

The desired Harmonic Summing range of FOP columns containing a “seed\_f0” is programmed via the “B\_START” and “B\_STOP” registers. There are two sets of registers (B\_START\_1, B\_START\_2 & B\_STOP\_1, B\_STOP\_2) to cater for two analysis runs, each with a configurable range. The FOP\_COL\_OFFSET value is taken into account automatically and so if the user wishes to perform harmonic summing over the first 10 convolution result locations B\_START will be programmed to 0 and B\_STOP programmed to 9 (assuming FOP\_COL\_OFFSET = 210, the actual FOP columns analysed for a seed\_f0 will then be 210 to 219).

This configuration is common to all SUMMER sub-module instances in the HSUM module.

###### “H” Programming

The number of Harmonics to be summed is programmed via the “H” register. There are two registers (H\_1, H\_2) to cater for two analysis runs, each with a configurable number of harmonics summed. Reducing the number of harmonics to be summed reduces the time to read the required information from DDR SDRAM memory since less FOP columns are required, and this reduces the Summing time since the summed results do not have to progress as far through the summing tree.

This configuration is common to all SUMMER sub-module instances in the HSUM module.

.

###### “FOP\_ROW” programming

The number of FOP rows to be read from the DDR\_SDRAM is programmed via the “FOP\_ROW” register. There are two registers (FOP\_ROW\_1, FOP\_ROW\_2) to cater for two analysis runs, each with a configurable number of rows. The FOP rows are read from the centre (i.e p[0]) outwards to (p[-42]/p[-42]). To read just the centre Row FOP\_ROW should be programmed to 1. To read rows p[-1], p[0], p[1] FOP ROW should be programmed to 3. To read rows p[-42],p[-41]… p0],… p[41], p[42] FOP ROW should be programmed to 85. Reducing the number of FOP Rows read from DDR SDRAM reduces the time to read the data.

This configuration is common to all SUMMER sub-module instances in the HSUM module.

###### “P\_EN” Programming

The number of orbital accelerations in a FOP Column (i.e. the number of seed\_f0 locations in a FOP Column) is programmed via the “P\_EN” register. There are two registers (P\_EN \_1, P\_EN \_2) to cater for two analysis runs, each with a configurable number of orbital accelerations. The maximum number of orbital accelerations is 21. Reducing the number of orbital accelerations for a seed\_f0 reduces the summing time.

This configuration is per SUMMER sub-module instance in the HSUM module.

###### “A” Programming

The number of orbital acceleration ambiguity slopes analysed is programmed via the “A” register. There are two registers (A \_1, A \_2) to cater for two analysis runs, each with a configurable number of orbital acceleration ambiguity slopes. The maximum number of orbital acceleration ambiguity slopes is 11. Reducing the number of orbital acceleration ambiguity slopes for a seed\_f0 reduces the summing time.

This configuration is per SUMMER sub-module instance in the HSUM module.

###### “A\_SET” Programming

The number of analysis runs to be performed is programmed via the “A\_SET” register (0 = 1 run, 1 = 2 runs).

This configuration is common to all SUMMER sub-module instances in the HSUM module.

###### “THRESH\_SET” Programming

The HSUM module supports two sets of Harmonic Summing power thresholds, -one for low frequency periodicity candidates and one for high frequency periodicity candidates. The FOP seed\_f0 column at which the change in threshold levels occurs is programmed via the “THRESH\_SET” register.

This configuration is common to all SUMMER sub-module instances in the HSUM module.

###### “M” Programming

The harmonic summing normally uses power information from selected locations of the FOP, however a user programmed power level via the “M” register can be selected for summing instead. If M is set to a power level of 0, this is useful when a desired FOP Row exceeds the size of the FOP allowing a power value of 0 to be used in summing. This is also a useful feature for diagnostic testing.

This configuration is common to all SUMMER sub-module instances in the HSUM module.

###### “HPSEL” Programming

For each of up to eight harmonics there is a set of configuration registers to determine which FOP row should be selected for summation. The Harmonic Summation supports:-

* Seed\_f0 in any of up to 21 rows in a FOP Column (i.e. up to 21 different orbital accelerations).
* Up to 11 orbital acceleration ambiguity slopes per seed\_f0

Hence for an analysis run a total 21 x 11 = 231 registers are required for each harmonic, with each register programmed with a FOP Row number in the range 0x00 to 0x60, where 0x00 to 0x59 selects a FOP Row and 0x60 selects the M value instead of a FOP.

Two sets of HPSEL registers (i.e. 2 x 231) are required per harmonic to support two analysis runs, each with configurable FOP locations selected for summing.

This configuration is per SUMMER sub-module instance in the HSUM module.

###### “THRESHOLD” Programming

For each of up to eight harmonics there is a set of configuration registers containing the power level thresholds above which a summation will trigger the collection of a result. Each of up to 21 seed\_f0 rows has a configurable power threshold.

Two sets of THRESHOLD registers (i.e. 2 x 21) are required per harmonic to support two sets of thresholds with the swap point over occurring as the FOP column containing the seed\_f0 passes the “THRESH\_SET” FOP column.

This configuration is per SUMMER sub-module instance in the HSUM module.

###### “T\_FILTER\_EN” Programming

As threshold levels are crossed the pertinent information for the crossing point (i.e. harmonic, FOP Row, FOP Column and power level) is collected.

If T\_FILTER\_EN is programmed to 0, all the results for every crossing are stored (even if there are multiple crossings for a particular seed\_f0 location).

If T\_FILTER\_EN is programmed to 1, at the completion of summing a seed\_f0 location only the threshold crossing of the highest harmonic is stored to save memory.

This configuration is common to all SUMMER sub-module instances in the HSUM module.

#### Loading DM Samples via the PCIe Interface

The DM samples are loaded into the FDAS FPGA via DMA accesses over the PCIe Interface.

##### FDAS DMA Descriptor Controller setup

The FDAS DMA Descriptor Controller is accessed via BAR0 of the PCIe Interface. The user programs the desired base address offset into BAR0 of the Type 0 Configuration space of the FDAS FPGA PCIe. This base address is then added to the addresses detailed below.

##### DDR4 SDRAM Addresses for PCIe DMA Transfers

To perform a DMA transfer the PC/Computer Read/ Write (as appropriate) the Descriptor table is programmed with the information for the transfer. This includes the source and destination addresses for the data transfer in the PC/Computer and relevant DDR4 SDRAM as viewed via the PCIe. This information is then transferred to the FDAS DMA engine within the PCIe Hard IP Macro.

The Byte Address range of DDR4 SDRAM #0 (supporting the CLD module) as seen PCIe is 0x200000000 – 0x2FFFFFFFF.

The Byte Address range of DDR4 SDRAM #2 (supporting the CONV/ HSUM modules) as seen via the PCIe is 0x600000000 – 0x6FFFFFFFF.

The Byte Address range of DDR4 SDRAM #3 (supporting the CONV/ HSUM modules) as seen via the PCIe is 0x800000000 – 0x8FFFFFFFF.

### Design Requirement Tag Compliance

| **Design Requirement Tag** | **Description** | **Modules Supporting the Requirement** | **Compliance** |
| --- | --- | --- | --- |
| FDAS.DATAIN:010/A | The data presented to FDAS shall be the result of an FFT with 222 (4,194,304) freq-bins. | CLD  PCIe Hard IP Macro | **Yes.**  The actual number of samples processed is set by MC Configuration  FOP\_SAMPLE\_NUM |
| FDAS.DATAIN:020/A | Each freq-bin shall contain a complex number with real and imaginary value | CLD  PCIe Hard IP Macro | **Yes.** |
| FDAS.DATAIN:030/A | Each element of the complex number shall be represented in IEEE 754 single precision format. Hence the complex number consists of two IEEE 754 values. | CLD  PCIe Hard IP Macro | **Yes.** |
| 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. | CLD  DDRIF2  PCIe Hard IP Macro  DDR Controller | **Yes.** |
| FDAS.DATAIN:050/A | The input data to FDAS shall be statically available for 111ms. | CLD | **No.**  In this implementation of FDAS the data shall need to be static for more than 111ms. DONE signals shall indicate when new samples can be accepted. |
| 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. | CLD  DDRIF2  PCIe Hard IP Macro | **Partial.**  CLD, CONV and HSUM modules support paging via the DDR memory starting address for processing provided by CTRL. However this implementation does not need paging as the HSUM module processing starts when the CONV module has finished. |
| 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. | CLD  DDRIF2  PCIe Hard IP Macro | **Yes.** |
| FDAS.DATAIN:080/A | The input data to FDAS (i.e. the observed data) shall be stored in external DDR SDRAM memory with the information shared equally across the available SDRAM devices. The **internal** FDAS data bus width shall be dependent on the number of external DDR SDRAM memory interfaces. A generic “ddr\_g” shall indicate the number of external DDR interfaces. | CLD  DDRIF2  DDR Controller | **Partial.**  CLD, CONV and HSUM modules support up to three DDR interfaces via the ddr\_g generic.  However in this implementation with reduced complexity DDR interface only one External DDR SDRAM device shall be used to store the freq-bin samples, with a separate External DDR SDRAM device being used to store the FOP. |
| FDAS.DATAIN:090/A | External commands from Software via a Monitor and Control (MC) interface shall instruct FDAS to commence reading the FFT sequence from memory, and also which memory page to access | CTRL  DDRIF2 | **Yes.**  The Software commands shall instruct the CTRL module to commence processing a DM. The CTRL module shall ensure CLD, CONV and HSUM modules operate in the correct sequence. |
| FDAS.CONVOLUTION:010/A | The FDAS input data shall be a post complex-FFT sequence of 222 freq-bins in the frequency domain and shall be convolved with 84 different FIR filters. | CONV | **Yes.** |
| FDAS.CONVOLUTION:020/A | The convolution with the 84 FIR filters shall be conceptualised as 84 parallel activities, resulting in 84 post convolution sequences of 222 freq-bins in the frequency domain. | CONV | **Yes.** |
| FDAS.CONVOLUTION:025/A | For this implementation it is only necessary to write out the first 221 freq-bins, since the “seed\_f0” is constrained to the first 262,144 FOP columns and only the first eight harmonics shall be analysed. To support a future design supporting 222 freq-bins a generic “fop\_g” shall control whether 221 or 222 freq-bins are written out. | CONV | **Yes.**  The number of samples processed in configured via the MC value FOP\_SAMPLE\_NUM. |
| FDAS.CONVOLUTION:030/A | Each of the 84 convolutions shall occur in isolation to the other 83 convolutions. The outputs of each convolution are kept separate from the others. | CONV | **Yes.** |
| FDAS.CONVOLUTION:040/A | Each convolution of the 222 freq-bin input sequence can be performed by either:-   * Direct time domain convolution   OR   * The FFT overlap-add method,   OR   * The FFT overlap-save method   The current intention is to use the overlap-save method. | CONV | **Yes.**  Overlap-save method used. |
| FDAS.CONVOLUTION:050/A | Direct convolution and Fourier Transform convolutions can co-exist in FDAS to allow the most efficient method to be used dependent on the FIR filter length. | CONV | **Yes.**  The overlap-save method is used for all convolutions. |
| FDAS.CONVOLUTION:060/A | The filter coefficients will be supplied by software via the MC interface and written directly to internal FPGA memory. These coefficients may be re-configured from time-to-time, but for a particular convolution run the coefficients will be static. | CONV | **Yes.** |
| FDAS.CONVOLUTION:070/A | Each FIR filter coefficient will be a complex value with real and imaginary part in IEEE 754 format with 32-bit single precision real part and 32-bit single precision imaginary part. | CONV | **Yes.** |
| FDAS.CONVOLUTION:080/A | For a particular filter number the +ve and –ve filter convolution coefficients have the same length and absolute values, apart from the fact that the –ve filter convolution coefficients are a complex conjugate of the +ve filter convolution coefficients.  This infers that if the +ve and –ve filter convolutions for a particular filter length can be performed simultaneously, the same multiplication results can be used for both convolutions, with only sign changes for the additions/subtractions. | CONV | **Yes.** |
| FDAS.CONVOLUTION:090/A | Each FIR filter coefficient shall have the real and imaginary part supplied via the MC interface in IEEE 754 format (see sec 2.7 of “FDAS REQUIREMENTS AND ARCHITECTURE SPECIFICATION”). | CONV | **Yes.** |
| FDAS.CONVOLUTION:100/A | The FIR filter lengths shall be fixed with the values in the “Table 3-4 FIR filter lengths” of “FDAS REQUIREMENTS AND ARCHITECTURE SPECIFICATION”. | CONV | **Yes.** |
| FDAS.CONVOLUTION:110/A | Regardless of the convolution method, complex number multiplications are required:-  (a + ib) \* (c + id) = a**\***c **–** b**\***d + i(a**\***d **+** b**\***c)  Hence each complex multiplication requires four real multiplications and two real addition/subtractions. | CONV | **Yes.** |
| FDAS.CONVOLUTION:120/A | For each of the 84 post-convolution frequency domain sequences each of the 222 freq-bins shall contain the power level (phase information is no longer required). i.e.:-  Power = (real**\***real) **+** (imaginary **\*** imaginary)  Conversion to power requires two real multiplications and one real addition. | CONV | **Yes.** |
| FDAS.HARMONIC\_SUM:010/A | Each of the 84 post-convolution 222 freq-bin frequency domain sequences containing power information form part of the Filter Output Plane (FOP). | HSUM | **Yes.**  The FOP is generated by CONV to this format. |
| FDAS.HARMONIC\_SUM:020/A | The original FDAS input 222 freq-bin frequency domain sequence also forms part of the FOP, with the contents of each freq-bin having been converted from complex amplitude to a power value. | HSUM | **Yes.**  The FOP is generated by CONV to this format. |
| FDAS.HARMONIC\_SUM:030/A | The sequences are arranged in relation to the convolution filter number “p” (-42, -41.. 0…41, 42) and freq-bin “b” (1 …. 4,194,304) with the original FDAS input sequence occupying the central row (i.e. p =0) of the FOP. | HSUM | **Yes.**  The FOP is generated by CONV to this format. |
| FDAS.HARMONIC\_SUM:035/A | It shall be possible to analyse the same FOP twice (mainly to facilitate reprocessing of the central FOP row p =0) with different user configuration for the summing parameters.  An MC configured value “A\_SET” shall determine if one analysis-run or two analysis-runs are to be performed. The relevant MC registers have two sets of configuration values, as indicated by the {a\_set} field to support the two different analysis-runs (where “a\_set” takes the values 1 and 2).  The time gap between the end of one analysis-run and the start of the second analysis-run shall be as short as possible (ideally no time gap). | HSUM | **Intention is Yes.** |
| FDAS.HARMONIC\_SUM:040/A | Harmonic summing over a globally user configurable freq-bin (b) range “B\_START {a\_set}[21:0]” to “B\_STOP {a\_set}[21:0]” and a globally configurable number of Filter Convolutions (p) “P\_EN{a\_set}[4:0]” up to a maximum of 21 Filters shall be carried out. Two harmonic summing analysis-runs on the same FOP with different configuration (via {a\_set}) shall be supported. . | HSUM | **Intention is Yes.** |
| FDAS.HARMONIC\_SUM:050/A | A globally user configurable number of harmonics “H {a\_set}[2:0]” up to a maximum of eight harmonics (f0, 2xf0, 3xf0 ….. 8xf0) shall be analysed during the harmonic summing process. Two harmonic summing analysis-runs on the same FOP with different configuration (via {a\_set}) shall be supported. | HSUM | **Intention is Yes.** |
| FDAS.HARMONIC\_SUM:055/A | A globally user configurable number of orbital acceleration ambiguity slopes “A {a\_set}[3:0]” up to a maximum of 11 slopes shall be used during the harmonic summing process. Two harmonic summing analysis-runs on the same FOP with different configuration (via {a\_set}) shall be supported. | HSUM | **Intention is Yes.** |
| FDAS.HARMONIC\_SUM:060/A | The FOP (SP0), Stretched Planes SP2 …. SP32 and Harmonic Planes H1… H32 may be seen as a way to visualise which terms need summing. | HSUM | **Intention is Yes.**  The actual method is to sum to appropriate locations the FOP. |
| FDAS.HARMONIC\_SUM:070/A | Harmonic summing function shall be required to select freq-bins from any of the 85 rows of the FOP, inferring that 85:1 muxing of 32-bit vectors shall be required. To support a default selection if the desired FOP row is greater than the FOP row range (+/- 42 rows) the mux shall actually be required to be 86:1. The value of the 86th 32-bit selected value shall be an MC configurable value “M [31:0]”. | HSUM | **Intention is Yes.**  The actual method shall use addressing to select the required data from a RAM, |
| FDAS.HARMONIC\_SUM:080/A | For correct operation the Harmonic Summing must start at the left side of the FOP (i.e. the low frequency end) and proceed for the different “seed\_f0” columns to the right side of the FOP (i.e. the high frequency end). | HSUM | **Intention is Yes.** |
| FDAS.THRESHOLD:010/A | Threshold values will be supplied by software via the MC interface and written to internal FPGA memory for the FDAS function to access.  The number of different thresholds is based on the following:-   * Each “seed\_f0” row in its FOP column representing a different orbital acceleration (i.e. FOP row = p) shall require a different threshold level for itself and its harmonics (total of 21 FOP rows). * Each fundamental and its harmonics shall require a different threshold level (total of 8 thresholds). * Pulsars of a lower frequency shall require different thresholds than those of a higher frequency and it is deemed that two sets of thresholds shall be required to support this.   Hence in total 21 x 8 x 2 = 336 threshold values shall be required.  A “seed\_f0” freq-bin value “THRESH\_SET [21:0]” shall indicate the pulsar frequency at which the thresholds shall change from one set to the other.  Each threshold value shall conform to the IEEE 754 32-bit single precision format. | HSUM | **Intention is Yes.** |
| FDAS.THRESHOLD:020/A | FDAS shall indicate in internal FPGA memory a list of locations which exceed the threshold in terms:-   * Harmonic Number (f0, 2xf0, 3xf0 ….. 8xf0) * Filter Convolution Number “p” (-42, -41 … 0…. 41, 42) * Freq-bin Number “b” (1 …… 4,194,304) * Power Level of the location   Each the four values shall be in IEEE 754 32-bit single precision format. | HSUM | **Intention is Yes.**  Some of the actual information reported shall need to be translated by software using the known configuration for FDAS. |
| FDAS.THRESHOLD:030/A | The list of locations that have crossed the threshold shall be of a finite size.  There shall be a defined limit to the number of threshold crossing locations that can be reported. The limit shall be 25 reports per harmonic. Since the same FOP can be analysed twice, results shall be provided for both analysis-runs.  i.e. total number of threshold reports is given by:-  No. of threshold reports = up to 2 analysis-runs  x (25 reports/harmonic)  x (8 harmonics)  = 400 reports  (each containing four IEEE 754 32-bit values)  If the reported list reaches its limit a counter for each harmonic (“T\_EXC{harmonic}[31:0]”) shall indicate the number of overflow reports. The “seed\_f0” location (“S\_EXC{harmonic}[28:0]”) and the filter number (p) (“P\_EXC{harmonic}[6:0]”) of the first overflowed threshold crossing shall be recorded against each harmonic. | HSUM | **Intention is Yes.** |
| FDAS.THRESHOLD:035/A | Ideally there shall be an MC configurable option (“T\_FILTER\_EN”) to store only the most significant threshold crossing for a particular “seed\_f0” postulate, since typically the threshold shall be crossed for a number of harmonics of “seed\_f0”, and it is sufficient to only store the threshold crossing for the highest harmonic. This shall require the 11-off acceleration ambiguity slopes for a “seed\_f0” to be completed before selection of the appropriate threshold crossings to be stored. | HSUM | **Intention is Yes.** |
| FDAS.THRESHOLD:040/A | The output memory containing the threshold results shall be paged to allow one page to be statically available to software via the MC interface over the PCIe, whilst to other page is being written to by FDAS. | HSUM | **Intention is Yes.** |
| FDAS.THRESHOLD:050/A | A free-running internal 32-bit unsigned count (“DM\_CNT[31:0]”) which increments for each DM shall be provided as a header to identify each set of threshold results to software via the MC interface over the PCIe. The software shall have the ability to reset this count to zero via “DM\_CNT\_RESET”. | HSUM | **Intention is Yes.** |
| 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 shown below for an IEEE 754 value as an example | CONV  CTRL  MCI\_TOP  PCIe Hard IP Macro  PCIF | **Yes.**  The actual values are shown in the Memory Map. |
| FDAS.MC:020/A | The MC Interface shall be supported by the PCIe interface to the FPGA  . | CTRL  MCI\_TOP  PCIe Hard IP Macro  PCIF | **Yes.** |
| FDAS.MC:030/A | The MC register map for the FDAS shall be supported, | CTRL  MCI\_TOP  PCIe Hard IP Macro  PCIF | **Yes.** |
| 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 possible so that a region of the DDR external memory can be accessed via the PCIe with a minimum number of read/write commands. | DDRIF2 | **Yes.** |
| FDAS.DIAGNOSTIC:020/A | It shall be possible to insert a count sequence into the FOP data written to the external DDR memory instead of the convolution results. | CONV | **No.**  This would involve muxing of the 512-bit data output to allow either the convolution result or a count sequence to be inserted. |
| FDAS.DIAGNOSTIC:030/A | It shall be possible to insert a count sequence into unused locations of the FOP. | CONV | **Yes.**  To reduce muxing and improve timing performance this feature is always enabled. |
| FDAS.DIAGNOSTIC:032/A | Each of the processing modules within FDAS shall have an MC configurable asynchronous reset. The individual resets are for the following modules:-   * CTRL * CLD * CONV * HSUM * DDRIF2#1 * DDRIF2#2 * DDR Controller #1 (Altera IP) * DDR Controller#2 (Altera IP)   This reset shall be ANDed with the main system reset from a pin on the FPGA. | MCI\_TOP | **Yes.** |
| FDAS.DIAGNOSTIC:035/A | The DDR Controllers shall be monitored to check the calibration of the interface to the external DDR SDRAM after a reset to the DDR Controllers. | MCI\_TOP | **Yes.** |
| 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. | CONV | **Yes.** |
| FDAS.DIAGNOSTIC:050/A | It shall be possible to trigger each main function within FDAS individually if desired via the MC interface. | CTRL  CONV | **Yes.** |
| FDAS.DIAGNOSTIC:060/A | It shall be possible to run each main function within FDAS for a configurable number of clock cycles, at which point processing will pause. Processing can be subsequently re-commenced for another configurable number of clock cycles. | CTRL  CONV | **Yes.** |
| FDAS.DIAGNOSTIC:070/A | Processing Timers shall be provided for each of the main functions (CLD, CONV and HSUM), measuring the time from when a function is triggered to when it finishes processing. The time shall be measured in terms of system clock cycles. Each timer shall be 32-bits. The timer shall not roll-over if it reaches its maximum value (0xFFFF,FFFF). | CTRL | **Yes.** |
| FDAS.MSI:010/A | FDAS shall support Message signalling Interrupts (MSI) via the PCIe Interface to indicate when a module has finished processing. | PCIF  PCIe Hard IP Macro | **Yes.** |
| FDAS.MSI:020/A | There shall be a separate MSI interrupt for each of the following indications:-   * CLD\_DONE Indicating the CLD module has finished processing a DM. * CONV\_DONE Indicating the CONV module has finished processing a DM. * HSUM\_DONE Indicating the HSUM module has finished processing a DM. | MSI  PCIe Hard IP Macro | **Yes.** |

Table 2‑6: Design Requirement Tag Compliance for this Implementation of FDAS

# FDAS Clocking Architecture

The Clocking Architecture for this implementation of the FDAS FPGA for the Intel Agilex F Development Board is shown in the figure below:-

100MHz

PCIe Clocks

**CLD**

**CTRL**

**CONV**

**HSUM**

**FDAS FPGA**

350MHz

**PCIE HARD IP MACRO**

**(INTEL IP)**

**MCI\_**

**TOP**

**PCIF**

**DDR\_CONTROLLER\_CALIBRATION\_HPS**

**(INTEL IP)**

**DDRIF2 #0**

CLK\_REF\_I\_0

(refclk)

CLK\_PCIE

CLK\_PCIE

**PLL**

**(INTEL IP)**

350MHz

CLK\_MC

PLL\_REF\_CLK\_0

CLK\_SYS

CLK\_MC

CLK\_SYS

CLK\_MC

CLK\_SYS

CLK\_SYS

CLK\_MC

CLK\_SYS

CLK\_MC

CLK\_DDR

EMIF\_USR\_CLK\_0

33.333MHz

PLL Reference clocks

MEM0\_CK/MEM0\_CK\_N

1200MHz Clocks to External SDRAM memories

MEM0\_CK Pin M37

MEM0\_CK\_N Pin P37

MEM1\_CK Pin M13

MEM1\_CK\_N Pin P13

CLK\_PCIE\_

REF\_0

Pin AJ48

PLL\_REF\_CLK\_0

Pin L40

CLK\_PCIE\_O

(coreclkout\_hip)

**Reset**

**Synchroniser**

CLK\_MC

**Reset Synchronisers**

CLK\_SYS

CLK\_PCIE

CLK\_DDR0 300MHz

C:LK\_DDR1 300MHz

CLK\_SYS & CLK\_MC

CLK\_PCIE\_

REF\_1

Pin AE48

CLK\_REF\_I\_1

(refclk)

**MSIX**

CLK\_SYS

CLK\_MC

CLK\_PCIE

PLL\_REF\_CLK\_1

Pin L10

EMIF\_USR\_CLK\_1

PLL\_REF\_CLK\_1

MEM1\_CK/MEM1\_CK\_N

**DDR\_CONTROLLER\_CALIBRATION\_HP**

**(INTEL IP)**

PLL\_REF\_CLK\_2

EMIF\_USR\_CLK\_2

MEM2\_CK/MEM2\_CK\_N

PLL\_REF\_CLK\_2

Pin CN38

CLK\_DDR2 300MHz

CLK\_DDR3 300MHz

PLL\_REF\_CLK\_3

Pin DC8

EMIF\_USR\_CLK\_3

PLL\_REF\_CLK\_3

MEM3\_CK/MEM3\_CK\_N

MEM2\_CK Pin CM35

MEM2\_CK\_N Pin CK35

MEM3\_CK Pin DB11

MEM3\_CK\_N Pin CY11

**DDRIF2 #2**

CLK\_PCIE

CLK\_SYS

CLK\_DDR

**DDRIF2 #3**

CLK\_PCIE

CLK\_SYS

CLK\_DDR

**Reset Synchronisers**

CLK\_PCIE

**Reset Synchronisers**

****

CLK\_DDR2

CLK\_DDR3

Figure ‑ : FDAS Clocking Architecture

The clocks required for internal processing in FDAS are the 100MHz PCIe clock on pins AJ48 and AE48 of the FPGA device and the 33.333MHz clock on pins L40, L10, CN38 and DC8 of the FPGA device.

The 100MHz clock is used by the Intel PCIe Hard IP Macro, which generates a 350MHz clock (CLK\_PCIE) for processing in the core of the FPGA. This 350MHz clock is used directly by the modules which have data interfaces connecting to the PCIe Hard IP Macro, namely the PCIF module, MSIX module and the three DDRIF2 modules.

The 350MHz clock is also used as the source of an Intel IP PLL (Phase Locked Loop) to generate the main 350MHz core processing clock “CLK\_SYS” of the FPGA. There is also a separate clock “CLK\_MC” for the Memory Map Micro Configuration (MC) interface registers, thus allowing the MC interface to run at a different rate (i.e. slower) than CLK\_SYS if desired. However in this implementation CLK\_MC and CLK\_SYS are both at 350MHz derived from the PLL.

The 33.333MHz clocks are used by DDR\_CONTROLLER\_CALIBRATION & DDR\_CONTROLLER\_CALIBRATION\_HPS IP blocks as the reference clock. This clock is multiplied by the DDR\_CONTROLLER to 1200.0MHz to provide the interface clock to the DDR4 External memory (MEM\*\_CK/MEM\*\_CK\_N). Both edges of this 1200.0MHz clock are used by the DDR4 External Memory, thus supporting 2400.0 Mtranfers per second.

All DDR Controllers also generate a 300.0MHz clock for their associated DDRIF2 modules.

# FDAS Reset Architecture

The Reset Architecture for this implementation of the FDAS FPGA for the Intel Agilex F Development Board is shown in the figure below:-

**CLD**

**CONV**

**HSUM**

**FDAS FPGA**

NINIT\_DONE\_I

**PCIE HARD IP MACRO**

**(INTEL IP)**

**DDR\_CONTROLLER\_CALIBRATION\_HPS**

**DDRIF2 #0**

RST\_PCIE\_N\_O

(app\_nreset\_status)

CLK\_PCIE

LOCAL\_RESET\_REQ\_0

LOCAL\_RESET\_REQ\_1

CLK\_SYS

RST\_SYS\_N

RST\_MC\_N

RST\_SYS\_N

CLK\_DDR0

CLK\_DDR0

Resets to External SDRAM memories

MEM0\_RESET\_N Pin P33

MEM1\_RESET\_N Pin P17

PIN\_PERST\_N

Pin BU58

PIN\_PERST\_I

(pin\_perst)

**CTRL**

**MCI\_**

**TOP**

**PCIF**

RST\_PCIE\_N

**PLL**

**(INTEL IP)**

RST\_SYS\_N

RST\_MC\_N

**Reset**

**Synchroniser**

**Reset**

**Synchronisers**

RST\_MC\_N

RESETN

RST\_PCIE\_N

global reset

(RESETN AND’ed with inverted MC configured individual resets in MCI\_TOP)

RST\_SYS\_N

RST\_MC\_N

CLK\_PCIE

CLK\_PCIE

CLK\_SYS &

CLK\_MC

RST\_MC\_N

CLK\_PCIE

CLK\_MC

CLK\_SYS

CLK\_MC

CLK\_MC

CLK\_MC

CLK\_MC

CLK\_MC

CLK\_SYS

CLK\_SYS

CLK\_SYS

CLK\_SYS

MEM0\_CK /

MEM0\_CK\_N

PLL\_REF\_CLK\_1

EMIF\_USR\_RESET\_N\_0

EMIF\_USR\_RESET\_N\_1

RST\_DDR\_N

RST\_PCIE\_N

RST\_SYS\_N

MEM0\_RESET\_N

MEM1\_RESET\_N

**Reset**

**Synchroniser**

CLK\_PCIE

**MSIX**

CLK\_PCIE

CLK\_MC

CLK\_SYS

**RESET\_**

**RELEASE**

**(INTEL IP)**

ninit\_done

DDR\_RST\_N\_0 to 3

RST\_MC\_N

RST\_SYS\_N

RST\_PCIE\_N

PLL\_REF\_CLK\_0

MEM1\_CK /

MEM1\_CK\_N

**DDRIF2 #2**

CLK\_PCIE

CLK\_SYS

CLK\_DDR2

RST\_DDR\_N

RST\_PCIE\_N

RST\_SYS\_N

**DDRIF2 #3**

CLK\_PCIE

CLK\_SYS

CLK\_DDR3

RST\_DDR\_N

RST\_PCIE\_N

RST\_SYS\_N

**DDR\_CONTROLLER\_CALIBRATION**

LOCAL\_RESET\_REQ\_2

LOCAL\_RESET\_REQ\_3

PLL\_REF\_CLK\_3

EMIF\_USR\_RESET\_N\_2

EMIF\_USR\_RESET\_N\_3

MEM2\_RESET\_N

MEM3\_RESET\_N

PLL\_REF\_CLK\_2

MEM2\_CK /

MEM2\_CK\_N

MEM3\_CK /

MEM3\_CK\_N

CLK\_DDR2

CLK\_DDR3

MEM2\_RESET\_N Pin CK31

MEM3\_RESET\_N Pin CY15

RST\_DDRIF\_0\_SYS

RST\_DDRIF\_3\_SYS

RST\_DDRIF\_2\_SYS

CLK\_DDR1

2

1

0

3

**Reset**

**Synchronisers**

CLK\_DDR2

CLK\_DDR3

CLK\_DDR0

RST\_DDRIF\_0\_DDR

RST\_DDRIF\_2\_DDR

RST\_DDRIF\_3\_DDR

Not used

Not used

Not used

Figure ‑ : FDAS Reset Architecture

There is one reset signal into the FDAS FPGA and “PIN\_PERST\_N” (pin BU58).

PIN\_PERST\_N is active low and resets the control registers on the PCIe Hard IP Macro and the data path registers.

The RESET\_RELEASE IP Block ensures that the PCIe Hard Macro is not operational until the FPGA image has fully configured the device. Once the active low “nint\_done” signal is asserted the PCIe becomes operational and the Host PC can pass a reset command to the FPGA if desired.

Due to the reset command from the Host PC the PCIe Hard IP Macro generates an active low reset “RST\_PCIE\_N\_O” (named “app\_rst\_n” in the Intel literature) which is used as the internal global reset for FDAS. This is treated as the “RST\_PCIE\_N” reset associated with the CLK\_PCIE clock domain.

RST\_PCIE\_N is used to reset the logic on the CLK\_PCIE clock domain in:-

* The PCIF module
* PCIe clocked circuits in DDRIF2#0, DDRIF2#2, DDRIF2#3 and MSIX modules in conjunction with a reset from the MCI\_TOP. The reset from the MCI\_TOP module is synchronised to the CLK\_PCIE clock domain and is logically ANDed with the RST\_PCIE\_N signal to provide a manual reset of the DDRIF2 and MSIX modules in addition to the automatic reset from the PCIe Hard IP Macro. This manual reset feature has been found to be necessary to ensure un-commanded data is not presented to the PCIe Hard IP Macro from the DDRIF2 modules when the other resets to the DDRIF2 modules (i.e. on the CLK\_DDR and CLK\_SYS domains) are asserted and then removed.
* The Intel IP PLL that is used to generate the CLK\_SYS/CLK\_MC clocks.

In addition RST\_PCIE\_N is also synchronised onto the CLK\_MC clock by a “reset synchroniser” to generate the “RST\_MC\_N” for the CLK\_MC domain. The “reset synchroniser” ensures that de-assertion of the reset is synchronous with the clock.

Finally RST\_PCIE\_N is also AND’ed with inverted MC configurable per-module resets (i.e. resets for CTRL, CLD, CONV, HSUM, DDRIF2#0, DDRIF2 #2, DDRIF2 #3 and MSIX) in the MCI\_TOP module. The resultant resets apart from the ones for the DDR\_CONTROLLER\_CALIBRATION modules are synchronised onto the required clock domain using “reset synchronisers”.

The reset to the DDR\_CONTROLLER\_CALIBRATION and DDR\_CONTROLLER\_CALIBRATION\_HPS modules from MCI\_TOP is asynchronous to the 300MHz clock into DDR\_CONTROLLER\_CALIBRATION and DDR\_CONTROLLER\_CALIBRATION\_HPS modules.

To apply a Reset perform the following steps:-

1. Apply all resets in the MCI\_TOP module.
2. Remove the DDR\_CONTROLLER\_CALIBRATION/ DDR\_CONTROLLER\_CALIBRATION\_HPS modules in the MCI\_TOP module.
3. Wait for all DDR Calibration signals to go high (this may take over one second)
4. Remove remaining resets in the MCI\_TOP module.

# FDAS FPGA Pinout

The pinout for this implementation of the FDAS FPGA is fixed by the Intel Agilex F Development Board.

The signals of the FDAS Design that are mapped to external FPGA pins are listed below:-

| **FDAS Signal** | **FPGA Pin** | **Dir** | **FPGA I/O Bank** | **Function** |
| --- | --- | --- | --- | --- |
| **PCIe Interface** |  |  |  |  |
| CLK\_PCIE\_REF\_0  CLK\_PCIE\_REF\_0\_N | AJ48 | IN | 10A | 100MHz PCIe Clock Inputs from the PC/Computer via the PCIe connector. |
| AH49 | IN | 10A |
| CLK\_PCIE\_REF\_1  CLK\_PCIE\_REF\_1\_N | AE48 | IN | 10A |
| AD49 | IN | 10A |
| PIN\_PERST\_N | BU58 | IN | 10A | Reset (Active Low). Resets the data path and control registers. |
| RX\_IN0  RX\_IN0\_N | BP61 | IN | 10A | PCIe Rx Data Lane 0 |
| BR62 | IN | 10A |
| RX\_IN1  RX\_IN1\_N | BN58 | IN | 10A | PCIe Rx Data Lane 1 |
| BM59 | IN | 10A |
| RX\_IN2  RX\_IN2\_N | BK61 | IN | 10A | PCIe Rx Data Lane 2 |
| BL62 | IN | 10A |
| RX\_IN3  RX\_IN3\_N | BJ58 | IN | 10A | PCIe Rx Data Lane 3 |
| BH59 | IN | 10A |
| RX\_IN4  RX\_IN4\_N | BF61 | IN | 10A | PCIe Rx Data Lane 4 |
| BG62 | IN | 10A |
| RX\_IN5  RX\_IN5\_N | BE58 | IN | 10A | PCIe Rx Data Lane 5 |
| BD59 | IN | 10A |
| RX\_IN6  RX\_IN6\_N | BB61 | IN | 10A | PCIe Rx Data Lane 6 |
| BC62 | IN | 10A |
| RX\_IN7  RX\_IN7\_N | BA58 | IN | 10A | PCIe Rx Data Lane 7 |
| AY59 | IN | 10A |
| RX\_IN8  RX\_IN8\_N | AV61 | IN | 10A | PCIe Rx Data Lane 8 |
| AW62 | IN | 10A |
| RX\_IN9  RX\_IN9\_N | AU58 | IN | 10A | PCIe Rx Data Lane 9 |
| AT59 | IN | 10A |
| RX\_IN10  RX\_IN10\_N | AP61 | IN | 10A | PCIe Rx Data Lane 10 |
| AR62 | IN | 10A |
| RX\_IN11  RX\_IN11\_N | AN58 | IN | 10A | PCIe Rx Data Lane 11 |
| AM59 | IN | 10A |
| RX\_IN12  RX\_IN12\_N | AK61 | IN | 10A | PCIe Rx Data Lane 12 |
| AL62 | IN | 10A |
| RX\_IN13  RX\_IN13\_N | AJ58 | IN | 10A | PCIe Rx Data Lane 13 |
| AH59 | IN | 10A |
| RX\_IN14  RX\_IN14\_N | AF61 | IN | 10A | PCIe Rx Data Lane 14 |
| AG62 | IN | 10A |
| RX\_IN15  RX\_IN15\_N | AE58 | IN | 10A | PCIe Rx Data Lane 15 |
| AD59 | IN | 10A |
| TX\_OUT0  TX\_OUT0\_N | BP55 | OUT | 10A | PCIe Tx Data Lane 0 |
| BR56 | OUT | 10A |
| TX\_OUT1  TX\_OUT1\_N | BN52 | OUT | 10A | PCIe Tx Data Lane 1 |
| BM53 | OUT | 10A |
| TX\_OUT2  TX\_OUT2\_N | BK55 | OUT | 10A | PCIe Tx Data Lane 2 |
| BL56 | OUT | 10A |
| TX\_OUT3  TX\_OUT3\_N | BJ52 | OUT | 10A | PCIe Tx Data Lane 3 |
| BH53 | OUT | 10A |
| TX\_OUT4  TX\_OUT4\_N | BF55 | OUT | 10A | PCIe Tx Data Lane 4 |
| BG56 | OUT | 10A |
| TX\_OUT5  TX\_OUT5\_N | BE52 | OUT | 10A | PCIe Tx Data Lane 5 |
| BD53 | OUT | 10A |
| TX\_OUT6  TX\_OUT6\_N | BB55 | OUT | 10A | PCIe Tx Data Lane 6 |
| BC56 | OUT | 10A |
| TX\_OUT7  TX\_OUT7\_N | BA52 | OUT | 10A | PCIe Tx Data Lane 7 |
| AY53 | OUT | 10A |
| TX\_OUT8  TX\_OUT8\_N | AV55 | OUT | 10A | PCIe Tx Data Lane 8 |
| AW56 | OUT | 10A |
| TX\_OUT9  TX\_OUT9\_N | AU52 | OUT | 10A | PCIe Tx Data Lane 9 |
| AT53 | OUT | 10A |
| TX\_OUT10  TX\_OUT10\_N | AP55 | OUT | 10A | PCIe Tx Data Lane 10 |
| AR56 | OUT | 10A |
| TX\_OUT11  TX\_OUT11\_N | AN52 | OUT | 10A | PCIe Tx Data Lane 11 |
| AM53 | OUT | 10A |
| TX\_OUT12  TX\_OUT12\_N | AK55 | OUT | 10A | PCIe Tx Data Lane 12 |
| AL56 | OUT | 10A |
| TX\_OUT13  TX\_OUT13\_N | AJ52 | OUT | 10A | PCIe Tx Data Lane 13 |
| AH53 | OUT | 10A |
| TX\_OUT14  TX\_OUT14\_N | AF55 | OUT | 10A | PCIe Tx Data Lane 14 |
| AG56 | OUT | 10A |
| TX\_OUT15  TX\_OUT15\_N | AE52 | OUT | 10A | PCIe Tx Data Lane 15 |
| AD53 | OUT | 10A |
|  |  |  |  |  |
| **DDR4 Interface #0** |  |  |  |  |
| PLL\_REF\_CLK\_0 | L40 | IN | 3B BOT | 33.3MHz ref clock for interface #0 DDR Controller |
| MEM0\_ALERT\_N[0] | U44 | IN | 3B BOT | DDR4 Interface #0  Alert Signal |
| MEM0\_ACT\_N[0] | N34 | OUT | 3B BOT | DDR4 Interface #0  ACTIVTE Command |
| MEM0\_PAR[0] | N38 | OUT | 3B BOT | Interface #0 Command/Address Parity |
| MEM0\_BA[0] | N44 | OUT | 3B BOT | DDR4 Interface #0  Bank Address |
| MEM0\_BA[1] | M45 | OUT | 3B BOT |
| MEM0\_BG[0] | P45 | OUT | 3B BOT | DDR4 Interface #0  Bank Group |
| MEM0\_BG[1] | M33 | OUT | 3B BOT |
| MEM0\_CK[0] | M37 | OUT | 3B BOT | DDR4 Interface #0  1200.0MHz Clock |
| MEM0\_CK\_N[0] | P37 | OUT | 3B BOT |
| MEM0\_CKE[0] | L36 | OUT | 3B BOT | DDR4 Interface #0  Clock Enable |
| MEM0\_CS\_N[0] | L34 | OUT | 3B BOT | DDR4 Interface #0  Chip Select |
| MEM0\_ODT[0] | M35 | OUT | 3B BOT | DDR4 Interface #0  On Die Termination |
| MEM0\_RESET\_N[0] | P33 | OUT | 3B BOT | DDR4 Interface #0  Reset |
| MEM0\_A[0] | T33 | OUT | 3B BOT | DDR4 Interface #0  Address |
| MEM0\_A[1] | V33 | OUT | 3B BOT |
| MEM0\_A[2] | U34 | OUT | 3B BOT |
| MEM0\_A[3] | W34 | OUT | 3B BOT |
| MEM0\_A[4] | T35 | OUT | 3B BOT |
| MEM0\_A[5] | V35 | OUT | 3B BOT |
| MEM0\_A[6] | U36 | OUT | 3B BOT |
| MEM0\_A[7] | W36 | OUT | 3B BOT |
| MEM0\_A[8] | T37 | OUT | 3B BOT |
| MEM0\_A[9] | V37 | OUT | 3B BOT |
| MEM0\_A[10] | U38 | OUT | 3B BOT |
| MEM0\_A[11] | W38 | OUT | 3B BOT |
| MEM0\_A[12] | P41 | OUT | 3B BOT |
| MEM0\_A[13] | L42 | OUT | 3B BOT |
| MEM0\_A[14] | N42 | OUT | 3B BOT |
| MEM0\_A[15] | M43 | OUT | 3B BOT |
| MEM0\_A[16] | P43 | OUT | 3B BOT |
| MEM0\_DBI\_N[0] | G36 | BIDI | 3B TOP | DDR4 Interface #0  Data Bus Inversion |
| MEM0\_DBI\_N[1] | A36 | BIDI | 3B TOP |
| MEM0\_DBI\_N[2] | B43 | BIDI | 3B TOP |
| MEM0\_DBI\_N[3] | F43 | BIDI | 3B TOP |
| MEM0\_DBI\_N[4] | G50 | BIDI | 3A TOP |
| MEM0\_DBI\_N[5] | F57 | BIDI | 3A TOP |
| MEM0\_DBI\_N[6] | B57 | BIDI | 3A TOP |
| MEM0\_DBI\_N[7] | T57 | BIDI | 3A BOT |
| MEM0\_DBI\_N[8] | A50 | BIDI | 3A TOP |
| MEM0\_DQ[0] | F33 | BIDI | 3B TOP | DDR4 Interface #0  Data Bus In/Out  DDR4 Interface #0  Data Bus In/Out |
| MEM0\_DQ[1] | H33 | BIDI | 3B TOP |
| MEM0\_DQ[2] | G34 | BIDI | 3B TOP |
| MEM0\_DQ[3] | J34 | BIDI | 3B TOP |
| MEM0\_DQ[4] | J38 | BIDI | 3B TOP |
| MEM0\_DQ[5] | G38 | BIDI | 3B TOP |
| MEM0\_DQ[6] | F37 | BIDI | 3B TOP |
| MEM0\_DQ[7] | H37 | BIDI | 3B TOP |
| MEM0\_DQ[8] | B33 | BIDI | 3B TOP |
| MEM0\_DQ[9] | D33 | BIDI | 3B TOP |
| MEM0\_DQ[10] | A34 | BIDI | 3B TOP |
| MEM0\_DQ[11] | C34 | BIDI | 3B TOP |
| MEM0\_DQ[12] | D37 | BIDI | 3B TOP |
| MEM0\_DQ[13] | A38 | BIDI | 3B TOP |
| MEM0\_DQ[14] | B37 | BIDI | 3B TOP |
| MEM0\_DQ[15] | C38 | BIDI | 3B TOP |
| MEM0\_DQ[16] | A40 | BIDI | 3B TOP |
| MEM0\_DQ[17] | C40 | BIDI | 3B TOP |
| MEM0\_DQ[18] | B41 | BIDI | 3B TOP |
| MEM0\_DQ[19] | D41 | BIDI | 3B TOP |
| MEM0\_DQ[20] | D45 | BIDI | 3B TOP |
| MEM0\_DQ[21] | B45 | BIDI | 3B TOP |
| MEM0\_DQ[22] | A44 | BIDI | 3B TOP |
| MEM0\_DQ[23] | C44 | BIDI | 3B TOP |
| MEM0\_DQ[24] | G40 | BIDI | 3B TOP |
| MEM0\_DQ[25] | J40 | BIDI | 3B TOP |
| MEM0\_DQ[26] | F41 | BIDI | 3B TOP |
| MEM0\_DQ[27] | H41 | BIDI | 3B TOP |
| MEM0\_DQ[28] | J44 | BIDI | 3B TOP |
| MEM0\_DQ[29] | H45 | BIDI | 3B TOP |
| MEM0\_DQ[30] | G44 | BIDI | 3B TOP |
| MEM0\_DQ[31] | F45 | BIDI | 3B TOP |
| MEM0\_DQ[32] | G48 | BIDI | 3A TOP |
| MEM0\_DQ[33] | F47 | BIDI | 3A TOP |
| MEM0\_DQ[34] | J48 | BIDI | 3A TOP |
| MEM0\_DQ[35] | H47 | BIDI | 3A TOP |
| MEM0\_DQ[36] | F51 | BIDI | 3A TOP |
| MEM0\_DQ[37] | H51 | BIDI | 3A TOP |
| MEM0\_DQ[38] | G52 | BIDI | 3A TOP |
| MEM0\_DQ[39] | J52 | BIDI | 3A TOP |
| MEM0\_DQ[40] | F55 | BIDI | 3A TOP |
| MEM0\_DQ[41] | G54 | BIDI | 3A TOP |
| MEM0\_DQ[42] | H55 | BIDI | 3A TOP |
| MEM0\_DQ[43] | J54 | BIDI | 3A TOP |
| MEM0\_DQ[44] | J58 | BIDI | 3A TOP |
| MEM0\_DQ[45] | F59 | BIDI | 3A TOP |
| MEM0\_DQ[46] | G58 | BIDI | 3A TOP |
| MEM0\_DQ[47] | H59 | BIDI | 3A TOP |
| MEM0\_DQ[48] | B55 | BIDI | 3A TOP |
| MEM0\_DQ[49] | A54 | BIDI | 3A TOP |
| MEM0\_DQ[50] | D55 | BIDI | 3A TOP |
| MEM0\_DQ[51] | C54 | BIDI | 3A TOP |
| MEM0\_DQ[52] | D59 | BIDI | 3A TOP |
| MEM0\_DQ[53] | C58 | BIDI | 3A TOP |
| MEM0\_DQ[54] | F61 | BIDI | 3A TOP |
| MEM0\_DQ[55] | H61 | BIDI | 3A TOP |
| MEM0\_DQ[56] | V55 | BIDI | 3A BOT |
| MEM0\_DQ[57] | T55 | BIDI | 3A BOT |
| MEM0\_DQ[58] | W54 | BIDI | 3A BOT |
| MEM0\_DQ[59] | U54 | BIDI | 3A BOT |
| MEM0\_DQ[60] | W58 | BIDI | 3A BOT |
| MEM0\_DQ[61] | T59 | BIDI | 3A BOT |
| MEM0\_DQ[62] | U58 | BIDI | 3A BOT |
| MEM0\_DQ[63] | V59 | BIDI | 3A BOT |
| MEM0\_DQ[64] | A48 | BIDI | 3A TOP |
| MEM0\_DQ[65] | B47 | BIDI | 3A TOP |
| MEM0\_DQ[66] | C48 | BIDI | 3A TOP |
| MEM0\_DQ[67] | D47 | BIDI | 3A TOP |
| MEM0\_DQ[68] | C52 | BIDI | 3A TOP |
| MEM0\_DQ[69] | D51 | BIDI | 3A TOP |
| MEM0\_DQ[70] | B51 | BIDI | 3A TOP |
| MEM0\_DQ[71] | A52 | BIDI | 3A TOP |
| MEM0\_DQS[0] | F35 | BIDI | 3B TOP | DDR4 Interface #0  Data Strobe |
| MEM0\_DQS\_N[0] | H35 | BIDI | 3B TOP |
| MEM0\_DQS[1] | B35 | BIDI | 3B TOP |
| MEM0\_DQS\_N[1] | D35 | BIDI | 3B TOP |
| MEM0\_DQS[2] | A42 | BIDI | 3B TOP |
| MEM0\_DQS\_N[2] | C42 | BIDI | 3B TOP |
| MEM0\_DQS[3] | G42 | BIDI | 3B TOP |
| MEM0\_DQS\_N[3] | J42 | BIDI | 3B TOP |
| MEM0\_DQS[4] | F49 | BIDI | 3A TOP |
| MEM0\_DQS\_N[4] | H49 | BIDI | 3A TOP |
| MEM0\_DQS[5] | G56 | BIDI | 3A TOP |
| MEM0\_DQS\_N[5] | J56 | BIDI | 3A TOP |
| MEM0\_DQS[6] | A56 | BIDI | 3A TOP |
| MEM0\_DQS\_N[6] | C56 | BIDI | 3A TOP |
| MEM0\_DQS[7] | U56 | BIDI | 3A BOT |
| MEM0\_DQS\_N[7] | W56 | BIDI | 3A BOT |
| MEM0\_DQS[8] | B49 | BIDI | 3A TOP |
| MEM0\_DQS\_N[8] | D49 | BIDI | 3A TOP |
| OCT\_RZQIN0 | M41 | IN | 3B BOT | DDR4 Interface #0  ZQ Reference |
|  |  |  |  |  |
| **DDR4 Interface #1 (unused)** |  |  |  |  |
| PLL\_REF\_CLK\_1 | L10 | IN | 3D BOT | 33.3MHz ref clock for interface #1 DDR Controller |
| MEM1\_ALERT\_N[0] | L6 | IN | 3D BOT | DDR4 Interface #1  Alert Signal |
| MEM1\_ACT\_N[0] | N16 | OUT | 3D BOT | DDR4 Interface #1  ACTIVTE Command |
| MEM1\_PAR[0] | N12 | OUT | 3D BOT | Interface #1 Command/Address Parity |
| MEM1\_BA[0] | N6 | OUT | 3D BOT | DDR4 Interface #1  Bank Address |
| MEM1\_BA[1] | M5 | OUT | 3D BOT |
| MEM1\_BG[0] | P5 | OUT | 3D BOT | DDR4 Interface #1  Bank Group |
| MEM1\_BG[1] | M17 | OUT | 3D BOT |
| MEM1\_CK[0] | M13 | OUT | 3D BOT | DDR4 Interface #1  1200.0MHz Clock |
| MEM1\_CK\_N[0] | P13 | OUT | 3D BOT |
| MEM1\_CKE[0] | L14 | OUT | 3D BOT | DDR4 Interface #1  Clock Enable |
| MEM1\_CS\_N[0] | L16 | OUT | 3D BOT | DDR4 Interface #1  Chip Select |
| MEM1\_ODT[0] | M15 | OUT | 3D BOT | DDR4 Interface #1  On Die Termination |
| MEM1\_RESET\_N[0] | P17 | OUT | 3D BOT | DDR4 Interface #1  Reset |
| MEM1\_A[0] | T17 | OUT | 3D BOT | DDR4 Interface #1  Address |
| MEM1\_A[1] | V17 | OUT | 3D BOT |
| MEM1\_A[2] | U16 | OUT | 3D BOT |
| MEM1\_A[3] | W16 | OUT | 3D BOT |
| MEM1\_A[4] | T15 | OUT | 3D BOT |
| MEM1\_A[5] | V15 | OUT | 3D BOT |
| MEM1\_A[6] | U14 | OUT | 3D BOT |
| MEM1\_A[7] | W14 | OUT | 3D BOT |
| MEM1\_A[8] | T13 | OUT | 3D BOT |
| MEM1\_A[9] | V13 | OUT | 3D BOT |
| MEM1\_A[10] | U12 | OUT | 3D BOT |
| MEM1\_A[11] | W12 | OUT | 3D BOT |
| MEM1\_A[12] | P9 | OUT | 3D BOT |
| MEM1\_A[13] | L8 | OUT | 3D BOT |
| MEM1\_A[14] | N8 | OUT | 3D BOT |
| MEM1\_A[15] | M7 | OUT | 3D BOT |
| MEM1\_A[16] | P7 | OUT | 3D BOT |
| MEM1\_DBI\_N[0] | B7 | BIDI | 3D TOP | DDR4 Interface #1  Data Bus Inversion |
| MEM1\_DBI\_N[1] | F7 | BIDI | 3D TOP |
| MEM1\_DBI\_N[2] | G14 | BIDI | 3D TOP |
| MEM1\_DBI\_N[3] | A14 | BIDI | 3D TOP |
| MEM1\_DBI\_N[4] | T21 | BIDI | 3C BOT |
| MEM1\_DBI\_N[5] | M21 | BIDI | 3C BOT |
| MEM1\_DBI\_N[6] | L28 | BIDI | 3C BOT |
| MEM1\_DBI\_N[7] | U28 | BIDI | 3C BOT |
| MEM1\_DBI\_N[8] | T7 | BIDI | 3D BOT |
| MEM1\_DQ[0] | A10 | BIDI | 3D TOP | DDR4 Interface #1  Data Bus In/Out  DDR4 Interface #1  Data Bus In/Out |
| MEM1\_DQ[1] | C10 | BIDI | 3D TOP |
| MEM1\_DQ[2] | B9 | BIDI | 3D TOP |
| MEM1\_DQ[3] | D9 | BIDI | 3D TOP |
| MEM1\_DQ[4] | A6 | BIDI | 3D TOP |
| MEM1\_DQ[5] | B5 | BIDI | 3D TOP |
| MEM1\_DQ[6] | C6 | BIDI | 3D TOP |
| MEM1\_DQ[7] | D5 | BIDI | 3D TOP |
| MEM1\_DQ[8] | G10 | BIDI | 3D TOP |
| MEM1\_DQ[9] | J10 | BIDI | 3D TOP |
| MEM1\_DQ[10] | F9 | BIDI | 3D TOP |
| MEM1\_DQ[11] | H9 | BIDI | 3D TOP |
| MEM1\_DQ[12] | G6 | BIDI | 3D TOP |
| MEM1\_DQ[13] | J6 | BIDI | 3D TOP |
| MEM1\_DQ[14] | F5 | BIDI | 3D TOP |
| MEM1\_DQ[15] | H5 | BIDI | 3D TOP |
| MEM1\_DQ[16] | F17 | BIDI | 3D TOP |
| MEM1\_DQ[17] | H17 | BIDI | 3D TOP |
| MEM1\_DQ[18] | G16 | BIDI | 3D TOP |
| MEM1\_DQ[19] | J16 | BIDI | 3D TOP |
| MEM1\_DQ[20] | F13 | BIDI | 3D TOP |
| MEM1\_DQ[21] | J12 | BIDI | 3D TOP |
| MEM1\_DQ[22] | H13 | BIDI | 3D TOP |
| MEM1\_DQ[23] | G12 | BIDI | 3D TOP |
| MEM1\_DQ[24] | B17 | BIDI | 3D TOP |
| MEM1\_DQ[25] | D17 | BIDI | 3D TOP |
| MEM1\_DQ[26] | A16 | BIDI | 3D TOP |
| MEM1\_DQ[27] | C16 | BIDI | 3D TOP |
| MEM1\_DQ[28] | B13 | BIDI | 3D TOP |
| MEM1\_DQ[29] | D13 | BIDI | 3D TOP |
| MEM1\_DQ[30] | A12 | BIDI | 3D TOP |
| MEM1\_DQ[31] | C12 | BIDI | 3D TOP |
| MEM1\_DQ[32] | U24 | BIDI | 3C BOT |
| MEM1\_DQ[33] | W24 | BIDI | 3C BOT |
| MEM1\_DQ[34] | T23 | BIDI | 3C BOT |
| MEM1\_DQ[35] | V23 | BIDI | 3C BOT |
| MEM1\_DQ[36] | U20 | BIDI | 3C BOT |
| MEM1\_DQ[37] | W20 | BIDI | 3C BOT |
| MEM1\_DQ[38] | T19 | BIDI | 3C BOT |
| MEM1\_DQ[39] | V19 | BIDI | 3C BOT |
| MEM1\_DQ[40] | L24 | BIDI | 3C BOT |
| MEM1\_DQ[41] | N24 | BIDI | 3C BOT |
| MEM1\_DQ[42] | M23 | BIDI | 3C BOT |
| MEM1\_DQ[43] | P23 | BIDI | 3C BOT |
| MEM1\_DQ[44] | L20 | BIDI | 3C BOT |
| MEM1\_DQ[45] | P19 | BIDI | 3C BOT |
| MEM1\_DQ[46] | N20 | BIDI | 3C BOT |
| MEM1\_DQ[47] | M19 | BIDI | 3C BOT |
| MEM1\_DQ[48] | M31 | BIDI | 3C BOT |
| MEM1\_DQ[49] | P31 | BIDI | 3C BOT |
| MEM1\_DQ[50] | L30 | BIDI | 3C BOT |
| MEM1\_DQ[51] | N30 | BIDI | 3C BOT |
| MEM1\_DQ[52] | M27 | BIDI | 3C BOT |
| MEM1\_DQ[53] | P27 | BIDI | 3C BOT |
| MEM1\_DQ[54] | L26 | BIDI | 3C BOT |
| MEM1\_DQ[55] | N26 | BIDI | 3C BOT |
| MEM1\_DQ[56] | T31 | BIDI | 3C BOT |
| MEM1\_DQ[57] | V31 | BIDI | 3C BOT |
| MEM1\_DQ[58] | U30 | BIDI | 3C BOT |
| MEM1\_DQ[59] | W30 | BIDI | 3C BOT |
| MEM1\_DQ[60] | T27 | BIDI | 3C BOT |
| MEM1\_DQ[61] | W26 | BIDI | 3C BOT |
| MEM1\_DQ[62] | V27 | BIDI | 3C BOT |
| MEM1\_DQ[63] | U26 | BIDI | 3C BOT |
| MEM1\_DQ[64] | U10 | BIDI | 3D BOT |
| MEM1\_DQ[65] | W10 | BIDI | 3D BOT |
| MEM1\_DQ[66] | T9 | BIDI | 3D BOT |
| MEM1\_DQ[67] | V9 | BIDI | 3D BOT |
| MEM1\_DQ[68] | U6 | BIDI | 3D BOT |
| MEM1\_DQ[69] | W6 | BIDI | 3D BOT |
| MEM1\_DQ[70] | T5 | BIDI | 3D BOT |
| MEM1\_DQ[71] | V5 | BIDI | 3D BOT |
| MEM1\_DQS[0] | A8 | BIDI | 3D TOP | DDR4 Interface #1  Data Strobe |
| MEM1\_DQS\_N[0] | C8 | BIDI | 3D TOP |
| MEM1\_DQS[1] | G8 | BIDI | 3D TOP |
| MEM1\_DQS\_N[1] | J8 | BIDI | 3D TOP |
| MEM1\_DQS[2] | F15 | BIDI | 3D TOP |
| MEM1\_DQS\_N[2] | H15 | BIDI | 3D TOP |
| MEM1\_DQS[3] | B15 | BIDI | 3D TOP |
| MEM1\_DQS\_N[3] | D15 | BIDI | 3D TOP |
| MEM1\_DQS[4] | U22 | BIDI | 3C BOT |
| MEM1\_DQS\_N[4] | W22 | BIDI | 3C BOT |
| MEM1\_DQS[5] | L22 | BIDI | 3C BOT |
| MEM1\_DQS\_N[5] | N22 | BIDI | 3C BOT |
| MEM1\_DQS[6] | M29 | BIDI | 3C BOT |
| MEM1\_DQS\_N[6] | P29 | BIDI | 3C BOT |
| MEM1\_DQS[7] | T29 | BIDI | 3C BOT |
| MEM1\_DQS\_N[7] | V29 | BIDI | 3C BOT |
| MEM1\_DQS[8] | U8 | BIDI | 3D BOT |
| MEM1\_DQS\_N[8] | W8 | BIDI | 3D BOT |
| OCT\_RZQIN1 | M9 | IN | 3D BOT | DDR4 Interface #1  ZQ Reference |
|  |  |  |  |  |
|  |  |  |  |  |
| **DDR4 Interface #2** |  |  |  |  |
| PLL\_REF\_CLK\_2 | CN38 | IN | 2B BOT | 33.3MHz ref clock for interface #2 DDR Controller |
| MEM2\_ALERT\_N[0] | CG42 | IN | 2B BOT | DDR4 Interface #2  Alert Signal |
| MEM2\_ACT\_N[0] | CL32 | OUT | 2B BOT | DDR4 Interface #2  ACTIVTE Command |
| MEM2\_PAR[0] | CL36 | OUT | 2B BOT | Interface #2 Command/Address Parity |
| MEM2\_BA[0] | CL42 | OUT | 2B BOT | DDR4 Interface #2  Bank Address |
| MEM2\_BA[1] | CM43 | OUT | 2B BOT |
| MEM2\_BG[0] | CK43 | OUT | 2B BOT | DDR4 Interface #2  Bank Group |
| MEM2\_BG[1] | CM31 | OUT | 2B BOT |
| MEM2\_CK[0] | CM35 | OUT | 2B BOT | DDR4 Interface #2  1200.0MHz Clock |
| MEM2\_CK\_N[0] | CK35 | OUT | 2B BOT |
| MEM2\_CKE[0] | CN34 | OUT | 2B BOT | DDR4 Interface #2  Clock Enable |
| MEM2\_CS\_N[0] | CN32 | OUT | 2B BOT | DDR4 Interface #2  Chip Select |
| MEM2\_ODT[0] | CM33 | OUT | 2B BOT | DDR4 Interface #2  On Die Termination |
| MEM2\_RESET\_N[0] | CK31 | OUT | 2B BOT | DDR4 Interface #2  Reset |
| MEM2\_A[0] | CH31 | OUT | 2B BOT | DDR4 Interface #2  Address |
| MEM2\_A[1] | CF31 | OUT | 2B BOT |
| MEM2\_A[2] | CG32 | OUT | 2B BOT |
| MEM2\_A[3] | CE32 | OUT | 2B BOT |
| MEM2\_A[4] | CH33 | OUT | 2B BOT |
| MEM2\_A[5] | CF33 | OUT | 2B BOT |
| MEM2\_A[6] | CG34 | OUT | 2B BOT |
| MEM2\_A[7] | CE34 | OUT | 2B BOT |
| MEM2\_A[8] | CH35 | OUT | 2B BOT |
| MEM2\_A[9] | CF35 | OUT | 2B BOT |
| MEM2\_A[10] | CG36 | OUT | 2B BOT |
| MEM2\_A[11] | CE36 | OUT | 2B BOT |
| MEM2\_A[12] | CK39 | OUT | 2B BOT |
| MEM2\_A[13] | CN40 | OUT | 2B BOT |
| MEM2\_A[14] | CL40 | OUT | 2B BOT |
| MEM2\_A[15] | CM41 | OUT | 2B BOT |
| MEM2\_A[16] | CK41 | OUT | 2B BOT |
| MEM2\_DBI\_N[0] | CH55 | BIDI | 2A BOT | DDR4 Interface #2  Data Bus Inversion |
| MEM2\_DBI\_N[1] | CV55 | BIDI | 2A TOP |
| MEM2\_DBI\_N[2] | DB55 | BIDI | 2A TOP |
| MEM2\_DBI\_N[3] | DC48 | BIDI | 2A TOP |
| MEM2\_DBI\_N[4] | CV41 | BIDI | 2B TOP |
| MEM2\_DBI\_N[5] | DB41 | BIDI | 2B TOP |
| MEM2\_DBI\_N[6] | DC34 | BIDI | 2B TOP |
| MEM2\_DBI\_N[7] | CU34 | BIDI | 2B TOP |
| MEM2\_DBI\_N[8] | CU48 | BIDI | 3A TOP |
| MEM2\_DQ[0] | CE52 | BIDI | 2A BOT | DDR4 Interface #2  Data Bus In/Out |
| MEM2\_DQ[1] | CF53 | BIDI | 2A BOT |
| MEM2\_DQ[2] | CG52 | BIDI | 2A BOT |
| MEM2\_DQ[3] | CH53 | BIDI | 2A BOT |
| MEM2\_DQ[4] | CE56 | BIDI | 2A BOT |
| MEM2\_DQ[5] | CG56 | BIDI | 2A BOT |
| MEM2\_DQ[6] | CF57 | BIDI | 2A BOT |
| MEM2\_DQ[7] | CH57 | BIDI | 2A BOT |
| MEM2\_DQ[8] | CR52 | BIDI | 2A TOP |
| MEM2\_DQ[9] | CT53 | BIDI | 2A TOP |
| MEM2\_DQ[10] | CU52 | BIDI | 2A TOP |
| MEM2\_DQ[11] | CV53 | BIDI | 2A TOP |
| MEM2\_DQ[12] | CR56 | BIDI | 2A TOP |
| MEM2\_DQ[13] | CU56 | BIDI | 2A TOP |
| MEM2\_DQ[14] | CT57 | BIDI | 2A TOP |
| MEM2\_DQ[15] | CV57 | BIDI | 2A TOP |
| MEM2\_DQ[16] | DA52 | BIDI | 2A TOP |
| MEM2\_DQ[17] | CY53 | BIDI | 2A TOP |
| MEM2\_DQ[18] | DB53 | BIDI | 2A TOP |
| MEM2\_DQ[19] | DC52 | BIDI | 2A TOP |
| MEM2\_DQ[20] | CY57 | BIDI | 2A TOP |
| MEM2\_DQ[21] | DB57 | BIDI | 2A TOP |
| MEM2\_DQ[22] | DA56 | BIDI | 2A TOP |
| MEM2\_DQ[23] | DC56 | BIDI | 2A TOP |
| MEM2\_DQ[24] | DA46 | BIDI | 2A TOP |
| MEM2\_DQ[25] | DC46 | BIDI | 2A TOP |
| MEM2\_DQ[26] | DB45 | BIDI | 2A TOP |
| MEM2\_DQ[27] | CY45 | BIDI | 2A TOP |
| MEM2\_DQ[28] | DC50 | BIDI | 2A TOP |
| MEM2\_DQ[29] | DA50 | BIDI | 2A TOP |
| MEM2\_DQ[30] | DB49 | BIDI | 2A TOP |
| MEM2\_DQ[31] | CY49 | BIDI | 2A TOP |
| MEM2\_DQ[32] | CT39 | BIDI | 2B TOP |
| MEM2\_DQ[33] | CV39 | BIDI | 2B TOP |
| MEM2\_DQ[34] | CU38 | BIDI | 2B TOP |
| MEM2\_DQ[35] | CR38 | BIDI | 2B TOP |
| MEM2\_DQ[36] | CU42 | BIDI | 2B TOP |
| MEM2\_DQ[37] | CV43 | BIDI | 2B TOP |
| MEM2\_DQ[38] | CR42 | BIDI | 2B TOP |
| MEM2\_DQ[39] | CT43 | BIDI | 2B TOP |
| MEM2\_DQ[40] | CY39 | BIDI | 2B TOP |
| MEM2\_DQ[41] | DB39 | BIDI | 2B TOP |
| MEM2\_DQ[42] | DC38 | BIDI | 2B TOP |
| MEM2\_DQ[43] | DA38 | BIDI | 2B TOP |
| MEM2\_DQ[44] | DC42 | BIDI | 2B TOP |
| MEM2\_DQ[45] | DB43 | BIDI | 2B TOP |
| MEM2\_DQ[46] | DA42 | BIDI | 2B TOP |
| MEM2\_DQ[47] | CY43 | BIDI | 2B TOP |
| MEM2\_DQ[48] | DA32 | BIDI | 2B TOP |
| MEM2\_DQ[49] | DC32 | BIDI | 2B TOP |
| MEM2\_DQ[50] | DB31 | BIDI | 2B TOP |
| MEM2\_DQ[51] | CY31 | BIDI | 2B TOP |
| MEM2\_DQ[52] | DA36 | BIDI | 2B TOP |
| MEM2\_DQ[53] | DC36 | BIDI | 2B TOP |
| MEM2\_DQ[54] | DB35 | BIDI | 2B TOP |
| MEM2\_DQ[55] | CY35 | BIDI | 2B TOP |
| MEM2\_DQ[56] | CR32 | BIDI | 2B TOP |
| MEM2\_DQ[57] | CU32 | BIDI | 2B TOP |
| MEM2\_DQ[58] | CV31 | BIDI | 2B TOP |
| MEM2\_DQ[59] | CT31 | BIDI | 2B TOP |
| MEM2\_DQ[60] | CV35 | BIDI | 2B TOP |
| MEM2\_DQ[61] | CU36 | BIDI | 2B TOP |
| MEM2\_DQ[62] | CT35 | BIDI | 2B TOP |
| MEM2\_DQ[63] | CR36 | BIDI | 2B TOP |
| MEM2\_DQ[64] | CU46 | BIDI | 2A TOP |
| MEM2\_DQ[65] | CT45 | BIDI | 2A TOP |
| MEM2\_DQ[66] | CR46 | BIDI | 2A TOP |
| MEM2\_DQ[67] | CV45 | BIDI | 2A TOP |
| MEM2\_DQ[68] | CT49 | BIDI | 2A TOP |
| MEM2\_DQ[69] | CU50 | BIDI | 2A TOP |
| MEM2\_DQ[70] | CV49 | BIDI | 2A TOP |
| MEM2\_DQ[71] | CR50 | BIDI | 2A TOP |
| MEM2\_DQS[0] | CG54 | BIDI | 2A BOT | DDR4 Interface #2  Data Strobe |
| MEM2\_DQS\_N[0] | CE54 | BIDI | 2A BOT |
| MEM2\_DQS[1] | CU54 | BIDI | 2A TOP |
| MEM2\_DQS\_N[1] | CR54 | BIDI | 2A TOP |
| MEM2\_DQS[2] | DC54 | BIDI | 2A TOP |
| MEM2\_DQS\_N[2] | DA54 | BIDI | 2A TOP |
| MEM2\_DQS[3] | DB47 | BIDI | 2A TOP |
| MEM2\_DQS\_N[3] | CY47 | BIDI | 2A TOP |
| MEM2\_DQS[4] | CU40 | BIDI | 2B TOP |
| MEM2\_DQS\_N[4] | CR40 | BIDI | 2B TOP |
| MEM2\_DQS[5] | DC40 | BIDI | 2B TOP |
| MEM2\_DQS\_N[5] | DA40 | BIDI | 2B TOP |
| MEM2\_DQS[6] | DB33 | BIDI | 2B TOP |
| MEM2\_DQS\_N[6] | CY33 | BIDI | 2B TOP |
| MEM2\_DQS[7] | CV33 | BIDI | 2B TOP |
| MEM2\_DQS\_N[7] | CT33 | BIDI | 2B TOP |
| MEM2\_DQS[8] | CV47 | BIDI | 2A TOP |
| MEM2\_DQS\_N[8] | CT47 | BIDI | 2A TOP |
| OCT\_RZQIN2 | CM39 | IN | 2B BOT | DDR4 Interface #2  ZQ Reference |
|  |  |  |  |  |
|  |  |  |  |  |
| **DDR4 Interface #3** |  |  |  |  |
| PLL\_REF\_CLK\_3 | DC8 | IN | 2D TOP | 33.3MHz ref clock for interface #3 DDR Controller |
| MEM3\_ALERT\_N[0] | CU4 | IN | 2D TOP | DDR4 Interface #3  Alert Signal |
| MEM3\_ACT\_N[0] | DA14 | OUT | 2D TOP | DDR4 Interface #3  ACTIVTE Command |
| MEM3\_PAR[0] | DA10 | OUT | 2D TOP | Interface #3 Command/Address Parity |
| MEM3\_BA[0] | CY3 | OUT | 2D TOP | DDR4 Interface #3  Bank Address |
| MEM3\_BA[1] | CV1 | OUT | 2D TOP |
| MEM3\_BG[0] | CT1 | OUT | 2D TOP | DDR4 Interface #3  Bank Group |
| MEM3\_BG[1] | DB15 | OUT | 2D TOP |
| MEM3\_CK[0] | DB11 | OUT | 2D TOP | DDR4 Interface #3  1066MHz Clock |
| MEM3\_CK\_N[0] | CY11 | OUT | 2D TOP |
| MEM3\_CKE[0] | DC12 | OUT | 2D TOP | DDR4 Interface #3  Clock Enable |
| MEM3\_CS\_N[0] | DC14 | OUT | 2D TOP | DDR4 Interface #3  Chip Select |
| MEM3\_ODT[0] | DB13 | OUT | 2D TOP | DDR4 Interface #3  On Die Termination |
| MEM3\_RESET\_N[0] | CY15 | OUT | 2D TOP | DDR4 Interface #3  Reset |
| MEM3\_A[0] | CV15 | OUT | 2D TOP | DDR4 Interface #3  Address |
| MEM3\_A[1] | CT15 | OUT | 2D TOP |
| MEM3\_A[2] | CU14 | OUT | 2D TOP |
| MEM3\_A[3] | CR14 | OUT | 2D TOP |
| MEM3\_A[4] | CV13 | OUT | 2D TOP |
| MEM3\_A[5] | CT13 | OUT | 2D TOP |
| MEM3\_A[6] | CU12 | OUT | 2D TOP |
| MEM3\_A[7] | CR12 | OUT | 2D TOP |
| MEM3\_A[8] | CV11 | OUT | 2D TOP |
| MEM3\_A[9] | CT11 | OUT | 2D TOP |
| MEM3\_A[10] | CU10 | OUT | 2D TOP |
| MEM3\_A[11] | CR10 | OUT | 2D TOP |
| MEM3\_A[12] | CY7 | OUT | 2D TOP |
| MEM3\_A[13] | DC6 | OUT | 2D TOP |
| MEM3\_A[14] | DA6 | OUT | 2D TOP |
| MEM3\_A[15] | DB5 | OUT | 2D TOP |
| MEM3\_A[16] | CY5 | OUT | 2D TOP |
| MEM3\_DBI\_N[0] | CN26 | BIDI | 2C BOT | DDR4 Interface #3  Data Bus Inversion |
| MEM3\_DBI\_N[1] | CG26 | BIDI | 2C BOT |
| MEM3\_DBI\_N[2] | CH19 | BIDI | 2C BOT |
| MEM3\_DBI\_N[3] | DC26 | BIDI | 2C TOP |
| MEM3\_DBI\_N[4] | CN12 | BIDI | 2D BOT |
| MEM3\_DBI\_N[5] | CG12 | BIDI | 2D BOT |
| MEM3\_DBI\_N[6] | CM5 | BIDI | 2D BOT |
| MEM3\_DBI\_N[7] | CH5 | BIDI | 2D BOT |
| MEM3\_DBI\_N[8] | CM19 | BIDI | 2C BOT |
| MEM3\_DQ[0] | CN28 | BIDI | 2C BOT | DDR4 Interface #3  Data Bus In/Out  DDR4 Interface #3  Data Bus In/Out |
| MEM3\_DQ[1] | CL28 | BIDI | 2C BOT |
| MEM3\_DQ[2] | CK29 | BIDI | 2C BOT |
| MEM3\_DQ[3] | CM29 | BIDI | 2C BOT |
| MEM3\_DQ[4] | CK25 | BIDI | 2C BOT |
| MEM3\_DQ[5] | CM25 | BIDI | 2C BOT |
| MEM3\_DQ[6] | CN24 | BIDI | 2C BOT |
| MEM3\_DQ[7] | CL24 | BIDI | 2C BOT |
| MEM3\_DQ[8] | CG28 | BIDI | 2C BOT |
| MEM3\_DQ[9] | CH29 | BIDI | 2C BOT |
| MEM3\_DQ[10] | CE28 | BIDI | 2C BOT |
| MEM3\_DQ[11] | CF29 | BIDI | 2C BOT |
| MEM3\_DQ[12] | CE24 | BIDI | 2C BOT |
| MEM3\_DQ[13] | CH25 | BIDI | 2C BOT |
| MEM3\_DQ[14] | CF25 | BIDI | 2C BOT |
| MEM3\_DQ[15] | CG24 | BIDI | 2C BOT |
| MEM3\_DQ[16] | CH21 | BIDI | 2C BOT |
| MEM3\_DQ[17] | CF21 | BIDI | 2C BOT |
| MEM3\_DQ[18] | CE22 | BIDI | 2C BOT |
| MEM3\_DQ[19] | CG22 | BIDI | 2C BOT |
| MEM3\_DQ[20] | CE18 | BIDI | 2C BOT |
| MEM3\_DQ[21] | CG18 | BIDI | 2C BOT |
| MEM3\_DQ[22] | CH17 | BIDI | 2C BOT |
| MEM3\_DQ[23] | CF17 | BIDI | 2C BOT |
| MEM3\_DQ[24] | DC28 | BIDI | 2C TOP |
| MEM3\_DQ[25] | DA28 | BIDI | 2C TOP |
| MEM3\_DQ[26] | CY29 | BIDI | 2C TOP |
| MEM3\_DQ[27] | DB29 | BIDI | 2C TOP |
| MEM3\_DQ[28] | CY25 | BIDI | 2C TOP |
| MEM3\_DQ[29] | DB25 | BIDI | 2C TOP |
| MEM3\_DQ[30] | DC24 | BIDI | 2C TOP |
| MEM3\_DQ[31] | DA24 | BIDI | 2C TOP |
| MEM3\_DQ[32] | CN14 | BIDI | 2D BOT |
| MEM3\_DQ[33] | CL14 | BIDI | 2D BOT |
| MEM3\_DQ[34] | CK15 | BIDI | 2D BOT |
| MEM3\_DQ[35] | CM15 | BIDI | 2D BOT |
| MEM3\_DQ[36] | CK11 | BIDI | 2D BOT |
| MEM3\_DQ[37] | CM11 | BIDI | 2D BOT |
| MEM3\_DQ[38] | CN10 | BIDI | 2D BOT |
| MEM3\_DQ[39] | CL10 | BIDI | 2D BOT |
| MEM3\_DQ[40] | CG14 | BIDI | 2D BOT |
| MEM3\_DQ[41] | CH15 | BIDI | 2D BOT |
| MEM3\_DQ[42] | CE14 | BIDI | 2D BOT |
| MEM3\_DQ[43] | CF15 | BIDI | 2D BOT |
| MEM3\_DQ[44] | CE10 | BIDI | 2D BOT |
| MEM3\_DQ[45] | CH11 | BIDI | 2D BOT |
| MEM3\_DQ[46] | CF11 | BIDI | 2D BOT |
| MEM3\_DQ[47] | CG10 | BIDI | 2D BOT |
| MEM3\_DQ[48] | CM7 | BIDI | 2D BOT |
| MEM3\_DQ[49] | CN8 | BIDI | 2D BOT |
| MEM3\_DQ[50] | CK7 | BIDI | 2D BOT |
| MEM3\_DQ[51] | CL8 | BIDI | 2D BOT |
| MEM3\_DQ[52] | CK3 | BIDI | 2D BOT |
| MEM3\_DQ[53] | CN4 | BIDI | 2D BOT |
| MEM3\_DQ[54] | CL4 | BIDI | 2D BOT |
| MEM3\_DQ[55] | CM3 | BIDI | 2D BOT |
| MEM3\_DQ[56] | CH7 | BIDI | 2D BOT |
| MEM3\_DQ[57] | CF7 | BIDI | 2D BOT |
| MEM3\_DQ[58] | CE8 | BIDI | 2D BOT |
| MEM3\_DQ[59] | CG8 | BIDI | 2D BOT |
| MEM3\_DQ[60] | CE4 | BIDI | 2D BOT |
| MEM3\_DQ[61] | CG4 | BIDI | 2D BOT |
| MEM3\_DQ[62] | CH3 | BIDI | 2D BOT |
| MEM3\_DQ[63] | CF3 | BIDI | 2D BOT |
| MEM3\_DQ[64] | CM21 | BIDI | 2C BOT |
| MEM3\_DQ[65] | CN22 | BIDI | 2C BOT |
| MEM3\_DQ[66] | CK21 | BIDI | 2C BOT |
| MEM3\_DQ[67] | CL22 | BIDI | 2C BOT |
| MEM3\_DQ[68] | CK17 | BIDI | 2C BOT |
| MEM3\_DQ[69] | CL18 | BIDI | 2C BOT |
| MEM3\_DQ[70] | CN18 | BIDI | 2C BOT |
| MEM3\_DQ[71] | CM17 | BIDI | 2C BOT |
| MEM3\_DQS[0] | CM27 | BIDI | 2C BOT | DDR4 Interface #3  Data Strobe |
| MEM3\_DQS\_N[0] | CK27 | BIDI | 2C BOT |
| MEM3\_DQS[1] | CH27 | BIDI | 2C BOT |
| MEM3\_DQS\_N[1] | CF27 | BIDI | 2C BOT |
| MEM3\_DQS[2] | CG20 | BIDI | 2C BOT |
| MEM3\_DQS\_N[2] | CE20 | BIDI | 2C BOT |
| MEM3\_DQS[3] | DB27 | BIDI | 2C TOP |
| MEM3\_DQS\_N[3] | CY27 | BIDI | 2C TOP |
| MEM3\_DQS[4] | CM13 | BIDI | 2D BOT |
| MEM3\_DQS\_N[4] | CK13 | BIDI | 2D BOT |
| MEM3\_DQS[5] | CH13 | BIDI | 2D BOT |
| MEM3\_DQS\_N[5] | CF13 | BIDI | 2D BOT |
| MEM3\_DQS[6] | CN6 | BIDI | 2D BOT |
| MEM3\_DQS\_N[6] | CL6 | BIDI | 2D BOT |
| MEM3\_DQS[7] | CG6 | BIDI | 2D BOT |
| MEM3\_DQS\_N[7] | CE6 | BIDI | 2D BOT |
| MEM3\_DQS[8] | CN20 | BIDI | 2C BOT |
| MEM3\_DQS\_N[8] | CL20 | BIDI | 2C BOT |
| OCT\_RZQIN3 | DB7 | IN | 2D TOP | DDR4 Interface #3  ZQ Reference |

Table 5‑1: FDAS FPGA pinout for Agilex F Development Board

# External DDR4 SDRAM Memory Organisation

The data in the External DDR4 SDRAM is accessible via the PCIe Interface and the CLD, CONV and HSUM modules within the FDAS FPGA.

The tables in the following sub-sections show how the DM Sample data and Convolution Results data is arranged at different interfaces in the FDAS design. Addressing for a particular interface is also related to the Byte Address (“BYTE\_ADDR”) so it is possible to compare the data for the various data bus widths.

The figure below is annotated with the different points within the FDAS design for the data arrangement described in the tables.

The Architecture for this implementation of FDAS is shown in the figure below:-

Figure 6‑1: FDAS Data Interfaces

PCIe

I/F

**CLD**

**CONV**

**HSUM**

**FDAS**

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

ADDR

DATA[511:0]

**DDRIF2 #0**

**DDRIF2 #2, #3**

ADDR

DATA[1023:0]

DATA[1023:0]

ADDR

ADDR

**DDR\_CONTROLLER\_CALIBRATION\_HPS (INTEL IP) (for DDRIF2 #0)**

**DDR\_CONTROLLER\_CALIBRATION (INTEL IP) (for DDRIF2 #2 and #3)**

**External 8 Gibi-Byte**

**DDR4 SDRAM Memory #0**

**External 8 Gibi-Byte**

**DDR4 SDRAM Memory #3**

DDR4 I/F #0

DDR4 I/F #3

DATA[511:0]

DATA[511:0]

ADDR

ADDR

DATA[511:0]

TLPs

**HOST PC/COMPUTER**

DATA11023:0]

ADDR

**A**

**B**

**D**

**C**

**F**

**F**

**J**

**K**

**F**

**C**

**L**

**External 8 Gibi-Byte**

**DDR4 SDRAM Memory #2**

DDR4 I/F #2

**H**

## Input Samples for a DM

In this implementation of FDAS for the Intel Agilex F card a single 8 Gibi-byte SDRAM supports the CLD module to store the incoming samples for a DM. However the CLD module is designed to also support either two or three 8 Gibi-byte SDRAMs for future implementations via the use of VHDL generics.

Each sample (e.g. “b[1]”) is 64-bit consisting of a 32-bit real and 32-bit imaginary value with the following organisation:-

|  |  |
| --- | --- |
| Bit 63 Bit 32  (MSB of Imaginary Part) (LSB of Imaginary Part) | Bit 31 Bit 0  (MSB of Real Part) (LSB of Real Part) |
| 32-bit Imaginary Part of the Sample | 32-bit Real Part of the Sample |

Table 6‑1: 64-bit Sample Data Organisation

### Point A: DM Samples: PC/Computer <> PCIe Hard IP Macro

Within the memory of the PC/Computer the data for a DM consisting of up to 222 (4,194,304) samples must be organised as shown in Table 6‑1 below so that it can be passed in the correct order to the FDAS FPGA via TLPs over the PCIe Interface:-

|  |  |
| --- | --- |
| BYTE\_ADDR[34:0]  / Hex | Observed Data Freq-bins “b”  64-bit per freq-bin: 32-bit real, 32-bit Imaginary  Bit 31 Bit 0  (MSB) (LSB) |
| 0x200000000 | b[1] Real Part |
| 0x200000004 | b[1] Imaginary Part |
| 0x200000008 | b[2] Real Part |
| 0x20000000C | b[2] Imaginary Part |
| : | : |
| 0x21FFFFF8 | b[4,194,304] Real Part |
| 0x21FFFFFC | b[4,194,304] Imaginary Part |

Table 6‑2: DM Sample Organisation in the PC/Computer to populate the TLPs over PCIe

Note that the byte address range for DDR4 SDRAM #0 as seen via the PCIe is 0x2,0000,0000 to 0x2,FFFF,FFFF.

### Point B: DM Samples: PCIe Hard IP Macro <> DDRIF2

Table 6‑3 below shows the organisation of the samples for a DM consisting of up to 222 (4,194,304) samples as they appear on the internal 512-bit Interface of the PCIe Hard IP Macro to the DDRIF2 module:-

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ADDR[25:0] =  BYTE\_ADDR[31:6]  /Hex | Observed Data Freq-bins “b”  64-bit per freq-bin: 32-bit real, 32-bit Imaginary  Bit 511 Bit 0 | | | | | | | |
| 0x0000000 | b[8] | b[7] | b[6] | b[5] | b[4] | b[3] | b[2] | b[1] |
| 0x0000001 | b[16] | b[15] | b[14] | b[13] | b[12] | b[11] | b[10] | b[9] |
| : | : | : | : | : | : | : | : | : |
| 0x007FFFE | b[4,194,296] | b[4,194,295] | b[4,194,294] | b[4,194,293] | b[4,194,292] | b[4,194,291] | b[4,194,290] | b[4,194,289] |
| 0x007FFFF | b[4,194,304] | b[4,194,303] | b[4,194,302] | b[4,194,301] | b[4,194,300] | b[4,194,299] | b[4,194,298] | b[4,194,297] |

Table 6‑3: DM Sample Organisation on the PCIe Hard IP Macro 512-bit bus to DDRIF2

Note that BYTE\_ADDR[5:0] is not shown in the above table since it is always 0 as each data word contains 64 bytes.

### Point C: DM Samples: DDRIF2 <> DDR CONTROLLER / CLD

Table 6‑4 below shows the organisation of the samples for a DM consisting of up to 222 (4,194,304) samples as they appear on the internal 512-bit Interface from the DDRIF2 module to the DDR CONTROLLER and the CLD module:-

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ADDR[25:0] =  BYTE\_ADDR[31:6]  /Hex | Observed Data Freq-bins “b”  64-bit per freq-bin: 32-bit real, 32-bit Imaginary  Bit 511 Bit 0 | | | | | | | |
| 0x0000000 | b[8] | b[7] | b[6] | b[5] | b[4] | b[3] | b[2] | b[1] |
| 0x0000001 | b[16] | b[15] | b[14] | b[13] | b[12] | b[11] | b[10] | b[9] |
| : | : | : | : | : | : | : | : | : |
| 0x007FFFE | b[4,194,296] | b[4,194,295] | b[4,194,294] | b[4,194,293] | b[4,194,292] | b[4,194,291] | b[4,194,290] | b[4,194,289] |
| 0x007FFFF | b[4,194,304] | b[4,194,303] | b[4,194,302] | b[4,194,301] | b[4,194,300] | b[4,194,299] | b[4,194,298] | b[4,194,297] |

Table 6‑4: DM Sample Organisation on the DDRIF2 512-bit bus to DDR CONTROLLER /CLD

Note that BYTE\_ADDR[5:0] is not shown in the above table since it is always 0 as each data word contains 64 bytes.

### Point D: DM Samples: DDR CONTROLLER <> SDRAM

Table 6‑5 below shows the organisation of the samples for a DM consisting of up to 222 (4,194,304) samples as they appear on external 64-bit Interface from the DDR Controller to the external DDR4 SDRAM:-

|  |  |
| --- | --- |
| ADDR[28:0] =  BG[1:0] (for Bank Group) &  BA[1:0] (for Bank within Group) &  A[14:0] (for Row) &  A[9:0] (for col)  = BYTE\_ADDR[31:3]  /Hex | Observed Data Freq-bins “b”  64-bit per freq-bin: 32-bit real, 32-bit Imaginary  Bit 63 Bit 0 |
| 0x00000000 | b[1] |
| 0x00000001 | b[2] |
| : | : |
| 0x003FFFFE | b[4,194,303] |
| 0x003FFFFF | b[4,194,304] |

Table 6‑5: DM Sample Organisation on the 64-bit Bus to External DDR4 SDRAM

Note that BYTE\_ADDR[2:0] is not shown in the above table since it is always 0 as each data word contains eight bytes.

## Convolution Results for a DM (FOP)

In this implementation of FDAS for the Intel Agilex F card two 8 Gibi-byte SDRAMs supports the CONV module to store the convolution results, known as the FOP (Filter Output Plane). However the CONV module is designed to also support either two or three 8 Gibi-byte SDRAMs for future implementations via the use of VHDL generics.

Each location of the FOP is a 32-bit value representing power with the following organisation:-

|  |
| --- |
| Bit 31 Bit 0  (MSB) (LSB) |
| 32-bit Power Value |

Table 6‑6: FOP Power Value

### Point F: FOP: DDRIF2 <> CONV/HSUM/DDR CONTROLLER

Table 6‑7 below shows the organisation of the FOP for a DM consisting of up to 85 x 222 (4,194,304) locations as they appear on the internal 1024-bit Interface between the DDRIF2 module, DDR CONTROLLER, CONV and HSUM modules:-

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| DDR\_ADDR  [31:6] | FOP  Col | FOP: PWR for filter p[\*] Mapped to the DDR \_DATA[1023:0] Interface  Each p[\*] value pertains to a row of the FOP  Bit 511 Bit 0 | | | | | | | | | | | | | | | |
| 0x0000000 | 1 | p[-7] | p[7] | p[-6] | p[6] | p[-5] | p[5] | p[-4] | p[4] | p[-3] | p[3] | p[-2] | p[2] | p[-1] | p[1] | p[0] | p[0] |
| 0x0000001 | p[-21] | p[21] | p[-20] | p[20] | p[-19] | p[19] | p[-18] | p[18] | p[-17] | p[17] | p[-16] | p[16] | p[-15] | p[15] |  |  |
| 0x0000002 | p[-35] | p[35] | p[-34] | p[34] | p[-33] | p[33] | p[-32] | p[32] | p[-31] | p[31] | p[-30] | p[30] | p[-29] | p[29] |  |  |
| 0x0000004 | 2 | p[-7] | p[7] | p[-6] | p[6] | p[-5] | p[5] | p[-4] | p[4] | p[-3] | p[3] | p[-2] | p[2] | p[-1] | p[1] | p[0] | p[0] |
| 0x0000005 | p[-21] | p[21] | p[-20] | p[20] | p[-19] | p[19] | p[-18] | p[18] | p[-17] | p[17] | p[-16] | p[16] | p[-15] | p[15] |  |  |
| 0x0000006 | p[-35] | p[35] | p[-34] | p[34] | p[-33] | p[33] | p[-32] | p[32] | p[-31] | p[31] | p[-30] | p[30] | p[-29] | p[29] |  |  |
| : |  | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : |
| 0x0FFFFFC | 222 | p[-7] | p[7] | p[-6] | p[6] | p[-5] | p[5] | p[-4] | p[4] | p[-3] | p[3] | p[-2] | p[2] | p[-1] | p[1] | p[0] | p[0] |
| 0x0FFFFFD | p[-21] | p[21] | p[-20] | p[20] | p[-19] | p[19] | p[-18] | p[18] | p[-17] | p[17] | p[-16] | p[16] | p[-15] | p[15] |  |  |
| 0x0FFFFFE | p[-35] | p[35] | p[-34] | p[34] | p[-33] | p[33] | p[-32] | p[32] | p[-31] | p[31] | p[-30] | p[30] | p[-29] | p[29] |  |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| DDR\_ADDR  [31:6] | FOP  Col | FOP: PWR for filter p[\*] Mapped to the DDR \_DATA[1023:0] Interface  Each p[\*] value pertains to a row of the FOP  Bit 1023 Bit 512 | | | | | | | | | | | | | | | |
| 0x0000000 | 1 | p[-14] | p[14] | p[-13] | p[13] | p[-12] | p[12] | p[-11] | p[11] | p[-10] | p[10] | p[-9] | p[9] | p[-8] | p[8] |  |  |
| 0x0000001 | p[-28] | p[28] | p[-27] | p[27] | p[-26] | p[26] | p[-25] | p[25] | p[-24] | p[24] | p[-23] | p[23] | p[-22] | p[22] |  |  |
| 0x0000002 | p[-42] | p[42] | p[-41] | p[41] | p[-40] | p[40] | p[-39] | p[39] | p[-38] | p[38] | p[-37] | p[37] | p[-36] | p[36] |  |  |
| 0x0000004 | 2 | p[-14] | p[14] | p[-13] | p[13] | p[-12] | p[12] | p[-11] | p[11] | p[-10] | p[10] | p[-9] | p[9] | p[-8] | p[8] |  |  |
| 0x0000005 | p[-28] | p[28] | p[-27] | p[27] | p[-26] | p[26] | p[-25] | p[25] | p[-24] | p[24] | p[-23] | p[23] | p[-22] | p[22] |  |  |
| 0x0000006 | p[-42] | p[42] | p[-41] | p[41] | p[-40] | p[40] | p[-39] | p[39] | p[-38] | p[38] | p[-37] | p[37] | p[-36] | p[36] |  |  |
| : |  | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : |
| 0x0FFFFFC | 222 | p[-14] | p[14] | p[-13] | p[13] | p[-12] | p[12] | p[-11] | p[11] | p[-10] | p[10] | p[-9] | p[9] | p[-8] | p[8] |  |  |
| 0x0FFFFFD | p[-28] | p[28] | p[-27] | p[27] | p[-26] | p[26] | p[-25] | p[25] | p[-24] | p[24] | p[-23] | p[23] | p[-22] | p[22] |  |  |
| 0x0FFFFFE | p[-42] | p[42] | p[-41] | p[41] | p[-40] | p[40] | p[-39] | p[39] | p[-38] | p[38] | p[-37] | p[37] | p[-36] | p[36] |  |  |

Table 6‑7: FOP Organisation on the CONV/HSUM 1024-bit bus

Note that DDR\_ADDR[5:0] is not shown in the above table since it is always 0 as each data word contains 64 bytes.

### Point H: FOP: DDR CONTROLLER <> SDRAM

Table 6‑8 below shows the organisation of the FOP at point H for a DM consisting of up to 85 x 222 (4,194,304) locations as they appear on external 64-bit Interface from the DDR Controller to the external DDR4 SDRAM:-

| ADDR[28:0] =  BG[1:0] (for Bank Group) &  BA[1:0] (for Bank within Group) &  A[14:0] (for Row) &  A[9:0] (for col)  = BYTE\_ADDR[31:3]  /Hex | FOP  Column | FOP: PWR for filter p[\*]  Each p[\*] value pertains to a row of the FOP  Bit 63 Bit 0 | |
| --- | --- | --- | --- |
| 0x00000000 | 1 | p[0] | p[0] |
| 0x00000001 | p[-1] | p[1] |
| 0x00000002 | p[-2] | p[2] |
| 0x00000003 | p[-3] | p[3] |
| 0x00000004 | p[-4] | p[4] |
| 0x00000005 | p[-5] | p[5] |
| 0x00000006 | p[-6] | p[6] |
| 0x00000007 | p[-7] | p[7] |
| 0x00000008 |  |  |
| 0x00000009 | p[-15] | p[15] |
| 0x0000000A | p[-16] | p[16] |
| 0x0000000B | p[-17] | p[17] |
| 0x0000000C | p[-18] | p[18] |
| 0x0000000D | p[-19] | p[19] |
| 0x0000000E | p[-20] | p[20] |
| 0x0000000F | p[-21] | p[21] |
| 0x00000010 |  |  |
| 0x00000011 | p[-29] | p[29] |
| 0x00000012 | p[-30] | p[30] |
| 0x00000013 | p[-31] | p[31] |
| 0x00000014 | p[-32] | p[32] |
| 0x00000015 | p[-33] | p[33] |
| 0x00000016 | p[-34] | p[34] |
| 0x00000017 | p[-35] | p[35] |
| 0x00000020 | 2 | p[0] | p[0] |
| 0x00000021 | p[-1] | p[1] |
| 0x00000022 | p[-2] | p[2] |
| 0x00000023 | p[-3] | p[3] |
| 0x00000024 | p[-4] | p[4] |
| 0x00000025 | p[-5] | p[5] |
| 0x00000026 | p[-6] | p[6] |
| 0x00000027 | p[-7] | p[7] |
| 0x00000028 |  |  |
| 0x00000029 | p[-15] | p[15] |
| 0x0000002A | p[-16] | p[16] |
| 0x0000002B | p[-17] | p[17] |
| 0x0000002C | p[-18] | p[18] |
| 0x0000002B | p[-19] | p[19] |
| 0x0000002E | p[-20] | p[20] |
| 0x0000002F | p[-21] | p[21] |
| 0x00000030 |  |  |
| 0x00000031 | p[-29] | p[29] |
| 0x00000032 | p[-30] | p[30] |
| 0x00000033 | p[-31] | p[31] |
| 0x00000034 | p[-32] | p[32] |
| 0x00000035 | p[-33] | p[33] |
| 0x00000036 | p[-34] | p[34] |
| 0x00000037 | p[-35] | p[35] |
| : |  | : | : |
| 0x07FFFFE0 | 222 | p[0] | p[0] |
| 0x07FFFFE1 | p[-1] | p[1] |
| 0x07FFFFE2 | p[-2] | p[2] |
| 0x07FFFFE3 | p[-3] | p[3] |
| 0x07FFFFE4 | p[-4] | p[4] |
| 0x07FFFFE5 | p[-5] | p[5] |
| 0x07FFFFE6 | p[-6] | p[6] |
| 0x07FFFFE7 | p[-7] | p[7] |
| 0x07FFFFE8 |  |  |
| 0x07FFFFE9 | 222 | p[-15] | p[15] |
| 0x07FFFFEA | p[-16] | p[16] |
| 0x07FFFFEB | p[-17] | p[17] |
| 0x07FFFFEC | p[-18] | p[18] |
| 0x07FFFFED | p[-19] | p[19] |
| 0x07FFFFEE | p[-20] | p[20] |
| 0x07FFFFEF | p[-21] | p[21] |
| 0x07FFFFF0 |  |  |
| 0x07FFFFF1 | p[-29] | p[29] |
| 0x07FFFFF2 | p[-30] | p[30] |
| 0x07FFFFF3 | p[-31] | p[31] |
| 0x07FFFFF4 | p[-32] | p[32] |
| 0x07FFFFF5 | p[-33] | p[33] |
| 0x07FFFFF6 | p[-34] | p[34] |
| 0x07FFFFF7 | p[-35] | p[35] |

Table 6‑8: FOP Organisation on the 64-bit Bus at point H to External DDR4 SDRAM

Note that BYTE\_ADDR[2:0] is not shown in the above table since it is always 0 as each data word contains eight bytes.

### Point J: FOP: DDR CONTROLLER <> SDRAM

Table 6‑9 below shows the organisation of the FOP at point J for a DM consisting of up to 85 x 222 (4,194,304) locations as they appear on external 64-bit Interface from the DDR Controller to the external DDR4 SDRAM:-

| ADDR[28:0] =  BG[1:0] (for Bank Group) &  BA[1:0] (for Bank within Group) &  A[14:0] (for Row) &  A[9:0] (for col)  = BYTE\_ADDR[31:3]  /Hex | FOP  Column | FOP: PWR for filter p[\*]  Each p[\*] value pertains to a row of the FOP  Bit 63 Bit 0 | |
| --- | --- | --- | --- |
| 0x00000000 | 1 |  |  |
| 0x00000001 | p[-8] | p[8] |
| 0x00000002 | p[-9] | p[9] |
| 0x00000003 | p[-10] | p[10] |
| 0x00000004 | p[-11] | p[11] |
| 0x00000005 | p[-12] | p[12] |
| 0x00000006 | p[-13] | p[13] |
| 0x00000007 | p[-14] | p[14] |
| 0x00000008 |  |  |
| 0x00000009 | p[-22] | p[22] |
| 0x0000000A | p[-23] | p[23] |
| 0x0000000B | p[-24] | p[24] |
| 0x0000000C | p[-25] | p[25] |
| 0x0000000D | p[-26] | p[26] |
| 0x0000000E | P[27] | p[27] |
| 0x0000000F | p[-28] | p[28] |
| 0x00000010 |  |  |
| 0x00000011 | p[-36] | p[36] |
| 0x00000012 | p[-37] | p[37] |
| 0x00000013 | p[-38] | p[38 |
| 0x00000014 | p[-39 | p[39] |
| 0x00000015 | p[-40 | p[40] |
| 0x00000016 | p[-41] | p[41 |
| 0x00000017 | p[-42] | p[42] |
| 0x00000020 | 2 |  |  |
| 0x00000021 | p[-8] | p[8] |
| 0x00000022 | p[-9] | p[9] |
| 0x00000023 | p[-10] | p[10] |
| 0x00000024 | p[-11] | p[11] |
| 0x00000025 | p[-12] | p[12] |
| 0x00000026 | p[-13] | p[13] |
| 0x00000027 | p[-14] | p[14] |
| 0x00000028 |  |  |
| 0x00000029 | p[-22] | p[22] |
| 0x0000002A | p[-23] | p[23] |
| 0x0000002B | p[-24] | p[24] |
| 0x0000002C | p[-25] | p[25] |
| 0x0000002B | p[-26] | p[26] |
| 0x0000002E | P[27] | p[27] |
| 0x0000002F | p[-28] | p[28] |
| 0x00000030 |  |  |
| 0x00000031 | p[-36] | p[36] |
| 0x00000032 | p[-37] | p[37] |
| 0x00000033 | p[-38] | p[38 |
| 0x00000034 | p[-39 | p[39] |
| 0x00000035 | p[-40 | p[40] |
| 0x00000036 | p[-41] | p[41 |
| 0x00000037 | p[-42] | p[42] |
| : |  | : | : |
| 0x07FFFFE0 | 222 |  |  |
| 0x07FFFFE1 | p[-8] | p[8] |
| 0x07FFFFE2 | p[-9] | p[9] |
| 0x07FFFFE3 | p[-10] | p[10] |
| 0x07FFFFE4 | p[-11] | p[11] |
| 0x07FFFFE5 | p[-12] | p[12] |
| 0x07FFFFE6 | p[-13] | p[13] |
| 0x07FFFFE7 | p[-14] | p[14] |
| 0x07FFFFE8 |  |  |
| 0x07FFFFE9 | 222 | p[-22] | p[22] |
| 0x07FFFFEA | p[-23] | p[23] |
| 0x07FFFFEB | p[-24] | p[24] |
| 0x07FFFFEC | p[-25] | p[25] |
| 0x07FFFFED | p[-26] | p[26] |
| 0x07FFFFEE | P[27] | p[27] |
| 0x07FFFFEF | p[-28] | p[28] |
| 0x07FFFFF0 |  |  |
| 0x07FFFFF1 | p[-36] | p[36] |
| 0x07FFFFF2 | p[-37] | p[37] |
| 0x07FFFFF3 | p[-38] | p[38 |
| 0x07FFFFF4 | p[-39 | p[39] |
| 0x07FFFFF5 | p[-40 | p[40] |
| 0x07FFFFF6 | p[-41] | p[41 |
| 0x07FFFFF7 | p[-42] | p[42] |

Table 6‑9: FOP Organisation on the 64-bit Bus at point J to External DDR4 SDRAM

Note that BYTE\_ADDR[2:0] is not shown in the above table since it is always 0 as each data word contains eight bytes.

### Point K: FOP: DDRIF2 <> PCIe Hard IP Macro

Table 6‑11 below shows the organisation of the FOP for a DM consisting of up to 85 x 222 (4,194,304) locations as they appear on the internal 512-bit Interfaces of the DDRIF2 #2 and #3 module to the PCIe Hard IP Macro:-

DDRIF2 #2

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ADDR[25:0] =  BYTE\_ADDR[31:6]  /Hex | FOP  Col | FOP: PWR for filter p[\*] Mapped to the DDR \_DATA[512:0] Interface  Each p[\*] value pertains to a row of the FOP  Bit 511 Bit 0 | | | | | | | | | | | | | | | |
| 0x0000000 | 1 | p[-7] | p[7] | p[-6] | p[6] | p[-5] | p[5] | p[-4] | p[4] | p[-3] | p[3] | p[-2] | p[2] | p[-1] | p[1] | p[0] | p[0] |
| 0x0000001 | p[-21] | p[21] | p[-20] | p[20] | p[-19] | p[19] | p[-18] | p[18] | p[-17] | p[17] | p[-16] | p[16] | p[-15] | p[15] |  |  |
| 0x0000002 | p[-35] | p[35] | p[-34] | p[34] | p[-33] | p[33] | p[-32] | p[32] | p[-31] | p[31] | p[-30] | p[30] | p[-29] | p[29] |  |  |
| 0x0000004 | 2 | p[-7] | p[7] | p[-6] | p[6] | p[-5] | p[5] | p[-4] | p[4] | p[-3] | p[3] | p[-2] | p[2] | p[-1] | p[1] | p[0] | p[0] |
| 0x0000005 | p[-21] | p[21] | p[-20] | p[20] | p[-19] | p[19] | p[-18] | p[18] | p[-17] | p[17] | p[-16] | p[16] | p[-15] | p[15] |  |  |
| 0x0000006 | p[-35] | p[35] | p[-34] | p[34] | p[-33] | p[33] | p[-32] | p[32] | p[-31] | p[31] | p[-30] | p[30] | p[-29] | p[29] |  |  |
| : |  | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : |
| 0x0FFFFFC | 222 | p[-7] | p[7] | p[-6] | p[6] | p[-5] | p[5] | p[-4] | p[4] | p[-3] | p[3] | p[-2] | p[2] | p[-1] | p[1] | p[0] | p[0] |
| 0x0FFFFFD | p[-21] | p[21] | p[-20] | p[20] | p[-19] | p[19] | p[-18] | p[18] | p[-17] | p[17] | p[-16] | p[16] | p[-15] | p[15] |  |  |
| 0x0FFFFFE | p[-35] | p[35] | p[-34] | p[34] | p[-33] | p[33] | p[-32] | p[32] | p[-31] | p[31] | p[-30] | p[30] | p[-29] | p[29] |  |  |

DDRIF2 #3

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ADDR[25:0] =  BYTE\_ADDR[31:6]  /Hex | FOP  Col | FOP: PWR for filter p[\*] Mapped to the DDR \_DATA[512:0] Interface  Each p[\*] value pertains to a row of the FOP  Bit 511 Bit 0 | | | | | | | | | | | | | | | |
| 0x0000000 | 1 | p[-14] | p[14] | p[-13] | p[13] | p[-12] | p[12] | p[-11] | p[11] | p[-10] | p[10] | p[-9] | p[9] | p[-8] | p[8] |  |  |
| 0x0000001 | p[-28] | p[28] | p[-27] | p[27] | p[-26] | p[26] | p[-25] | p[25] | p[-24] | p[24] | p[-23] | p[23] | p[-22] | p[22] |  |  |
| 0x0000002 | p[-42] | p[42] | p[-41] | p[41] | p[-40] | p[40] | p[-39] | p[39] | p[-38] | p[38] | p[-37] | p[37] | p[-36] | p[36] |  |  |
| 0x0000004 | 2 | p[-14] | p[14] | p[-13] | p[13] | p[-12] | p[12] | p[-11] | p[11] | p[-10] | p[10] | p[-9] | p[9] | p[-8] | p[8] |  |  |
| 0x0000005 | p[-28] | p[28] | p[-27] | p[27] | p[-26] | p[26] | p[-25] | p[25] | p[-24] | p[24] | p[-23] | p[23] | p[-22] | p[22] |  |  |
| 0x0000006 | p[-42] | p[42] | p[-41] | p[41] | p[-40] | p[40] | p[-39] | p[39] | p[-38] | p[38] | p[-37] | p[37] | p[-36] | p[36] |  |  |
| : |  | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : |
| 0x0FFFFFC | 222 | p[-14] | p[14] | p[-13] | p[13] | p[-12] | p[12] | p[-11] | p[11] | p[-10] | p[10] | p[-9] | p[9] | p[-8] | p[8] |  |  |
| 0x0FFFFFD | p[-28] | p[28] | p[-27] | p[27] | p[-26] | p[26] | p[-25] | p[25] | p[-24] | p[24] | p[-23] | p[23] | p[-22] | p[22] |  |  |
| 0x0FFFFFE | p[-42] | p[42] | p[-41] | p[41] | p[-40] | p[40] | p[-39] | p[39] | p[-38] | p[38] | p[-37] | p[37] | p[-36] | p[36] |  |  |

Table 6‑10: FOP Organisation on the DDRIF2 512-bit bus to PCIe Hard IP Macro

Note that BYTE\_ADDR[5:0] is not shown in the above table since it is always 0 as each data word contains 64 bytes.

### Point L: FOP: PCIe Hard IP Macro <> PC/Computer

Table 6‑12 below shows the organisation of the FOP for a DM consisting of up to 85 x 222 (4,194,304) locations as they appear in TLPs over the PCIe Interface.

For DDRIF2 #2 with base offset 0x600000000

| BYTE\_ADDR[34:0]  / Hex | FOP  Column | FOP: PWR for filter p[\*]  Each p[\*] value pertains to a row of the FOP  Bit 31 Bit 0  (MSB) (LSB) |
| --- | --- | --- |
| 0x600000000 | 1 | p[0] |
| 0x600000004 | p[0] |
| 0x600000008 | p[1] |
| 0x60000000C | p[-1] |
| 0x600000010 | p[2] |
| 0x600000014 | p[-2] |
| 0x600000018 | p[3] |
| 0x60000001C | p[-3] |
| 0x600000020 | p[4] |
| 0x600000024 | p[-4] |
| 0x600000028 | p[5] |
| 0x60000002C | p[-5] |
| 0x600000030 | p[6] |
| 0x600000034 | p[-6] |
| 0x600000038 | p[7] |
| 0x60000003C | p[-7] |
| 0x600000040 |  |
| 0x600000044 |  |
| 0x600000048 | p[15] |
| 0x60000004C | p[-15] |
| 0x600000050 | p[16] |
| 0x600000054 | p[-16] |
| 0x600000058 | p[17] |
| 0x60000005C | p[-17] |
| 0x600000060 | p[18] |
| 0x600000064 | p[-18] |
| 0x600000068 | p[19] |
| 0x60000006C | p[-19] |
| 0x600000070 | p[20] |
| 0x600000074 | p[-20] |
| 0x600000078 | p[21] |
| 0x60000007C | p[-21] |
| 0x600000080 |  |
| 0x600000084 |  |
| 0x600000088 | p[29] |
| 0x60000008C | p[-29] |
| 0x600000090 | p[30] |
| 0x600000094 | p[-30] |
| 0x600000098 | p[31] |
| 0x60000009C | p[-31] |
| 0x6000000A0 | p[32] |
| 0x6000000A4 | p[-32] |
| 0x6000000A8 | p[33] |
| 0x6000000AC | p[-33] |
| 0x6000000B0 | p[34] |
| 0x6000000B4 | p[-34] |
| 0x6000000B8 | p[35] |
| 0x6000000BC | p[-35] |
| 0x600000100 | 2 | p[0] |
| 0x600000104 | p[0] |
| 0x600000108 | p[1] |
| 0x60000010C | p[-1] |
| 0x600000110 | p[2] |
| 0x600000114 | p[-2] |
| 0x600000118 | p[3] |
| 0x60000011C | p[-3] |
| 0x600000120 | p[4] |
| 0x600000124 | p[-4] |
| 0x600000128 | p[5] |
| 0x60000012C | p[-5] |
| 0x600000130 | p[6] |
| 0x600000134 | p[-6] |
| 0x600000138 | p[7] |
| 0x60000013C | p[-7] |
| 0x600000140 |  |
| 0x600000144 |  |
| 0x600000148 | p[15] |
| 0x60000014C | p[-15] |
| 0x600000150 | p[16] |
| 0x600000154 | p[-16] |
| 0x600000158 | p[17] |
| 0x60000015C | p[-17] |
| 0x600000160 | p[18] |
| 0x600000164 | p[-18] |
| 0x600000168 | p[19] |
| 0x60000016C | p[-19] |
| 0x600000170 | p[20] |
| 0x600000174 | p[-20] |
| 0x600000178 | p[21] |
| 0x60000017C | p[-21] |
| 0x600000180 |  |
| 0x600000184 |  |
| 0x600000188 | p[29] |
| 0x60000018C | p[-29] |
| 0x600000190 | p[30] |
| 0x600000194 | p[-30] |
| 0x600000198 | p[31] |
| 0x60000019C | p[-31] |
| 0x6000001A0 | p[32] |
| 0x6000001A4 | p[-32] |
| 0x6000001A8 | p[33] |
| 0x6000001AC | p[-33] |
| 0x6000001B0 | p[34] |
| 0x6000001B4 | p[-34] |
| 0x6000001B8 | p[35] |
| 0x6000001BC | p[-35] |
|  | : | : |
| 0x63FFFFF00 | 222 | p[0] |
| 0x63FFFFF04 | p[0] |
| 0x63FFFFF08 | p[1] |
| 0x63FFFFF0C | p[-1] |
| 0x63FFFFF10 | p[2] |
| 0x63FFFFF14 | p[-2] |
| 0x63FFFFF18 | p[3] |
| 0x63FFFFF1C | p[-3] |
| 0x63FFFFF20 | p[4] |
| 0x63FFFFF24 | p[-4] |
| 0x63FFFFF28 | p[5] |
| 0x63FFFFF2C | p[-5] |
| 0x63FFFFF30 | p[6] |
| 0x63FFFFF34 | p[-6] |
| 0x63FFFFF38 | p[7] |
| 0x63FFFFF3C | p[-7] |
| 0x63FFFFF40 |  |
| 0x63FFFFF44 |  |
| 0x63FFFFF48 | p[15] |
| 0x63FFFFF4C | p[-15] |
| 0x63FFFFF50 | p[16] |
| 0x63FFFFF54 | p[-16] |
| 0x63FFFFF58 | p[17] |
| 0x63FFFFF5C | p[-17] |
| 0x63FFFFF60 | p[18] |
| 0x63FFFFF64 | p[-18] |
| 0x63FFFFF68 | p[19] |
| 0x63FFFFF6C | p[-19] |
| 0x63FFFFF70 | p[20] |
| 0x63FFFFF74 | p[-20] |
| 0x63FFFFF78 | p[21] |
| 0x63FFFFF7C | p[-21] |
| 0x63FFFFF80 |  |
| 0x63FFFFF84 |  |
| 0x63FFFFF88 | p[29] |
| 0x63FFFFF8C | p[-29] |
| 0x63FFFFF90 | p[30] |
| 0x63FFFFF94 | p[-30] |
| 0x63FFFFF98 | p[31] |
| 0x63FFFFF9C | p[-31] |
| 0x63FFFFFA0 | p[32] |
| 0x63FFFFFF4 | p[-32] |
| 0x63FFFFFA8 | p[33] |
| 0x63FFFFFAC | p[-33] |
| 0x63FFFFFB0 | p[34] |
| 0x63FFFFFB4 | p[-34] |
| 0x63FFFFFB8 | p[35] |
| 0x63FFFFFBC | p[-35] |

For DDRIF2 #3 with base offset 0x800000000

| BYTE\_ADDR[34:0]  / Hex | FOP  Column | FOP: PWR for filter p[\*]  Each p[\*] value pertains to a row of the FOP  Bit 31 Bit 0  (MSB) (LSB) |
| --- | --- | --- |
| 0x800000000 | 1 |  |
| 0x800000004 |  |
| 0x800000008 | p[8] |
| 0x80000000C | p[8] |
| 0x800000010 | p[9] |
| 0x800000014 | p[-9] |
| 0x800000018 | p[10] |
| 0x80000001C | p[-10] |
| 0x800000020 | p[11] |
| 0x800000024 | p[-11] |
| 0x800000028 | p[12] |
| 0x80000002C | p[-12] |
| 0x800000030 | p[13] |
| 0x800000034 | p[-13] |
| 0x800000038 | p[14] |
| 0x80000003C | p[-14] |
| 0x800000040 |  |
| 0x800000044 |  |
| 0x800000048 | p[22] |
| 0x80000004C | p[-22] |
| 0x800000050 | p[23] |
| 0x800000054 | p[-23] |
| 0x800000058 | p[24] |
| 0x80000005C | p[-24] |
| 0x800000060 | p[25] |
| 0x800000064 | p[-25] |
| 0x800000068 | p[26] |
| 0x80000006C | p[-26] |
| 0x800000070 | p[27] |
| 0x800000074 | p[-27] |
| 0x800000078 | p[28] |
| 0x80000007C | p[-28] |
| 0x800000080 |  |
| 0x800000084 |  |
| 0x800000088 | p[36] |
| 0x80000008C | p[-36] |
| 0x800000090 | p[37] |
| 0x800000094 | p[-37] |
| 0x800000098 | p[38] |
| 0x80000009C | p[-38] |
| 0x8000000A0 | p[39] |
| 0x8000000A4 | p[-39] |
| 0x8000000A8 | p[40] |
| 0x8000000AC | p[-40] |
| 0x8000000B0 | p[41] |
| 0x8000000B4 | p[-41] |
| 0x8000000B8 | p[42] |
| 0x8000000BC | p[-42] |
| 0x800000100 | 2 |  |
| 0x800000104 |  |
| 0x800000108 | p[8] |
| 0x80000010C | p[8] |
| 0x800000110 | p[9] |
| 0x800000114 | p[-9] |
| 0x800000118 | p[10] |
| 0x80000011C | p[-10] |
| 0x800000120 | p[11] |
| 0x800000124 | p[-11] |
| 0x800000128 | p[12] |
| 0x80000012C | p[-12] |
| 0x800000130 | p[13] |
| 0x800000134 | p[-13] |
| 0x800000138 | p[14] |
| 0x80000013C | p[-14] |
| 0x800000140 |  |
| 0x800000144 |  |
| 0x800000148 | p[22] |
| 0x80000014C | p[-22] |
| 0x800000150 | p[23] |
| 0x800000154 | p[-23] |
| 0x800000158 | p[24] |
| 0x80000015C | p[-24] |
| 0x800000160 | p[25] |
| 0x800000164 | p[-25] |
| 0x800000168 | p[26] |
| 0x80000016C | p[-26] |
| 0x800000170 | p[27] |
| 0x800000174 | p[-27] |
| 0x800000178 | p[28] |
| 0x80000017C | p[-28] |
| 0x800000180 |  |
| 0x800000184 |  |
| 0x800000188 | p[36] |
| 0x80000018C | p[-36] |
| 0x800000190 | p[37] |
| 0x800000194 | p[-37] |
| 0x800000198 | p[38] |
| 0x80000019C | p[-38] |
| 0x8000001A0 | p[39] |
| 0x8000001A4 | p[-39] |
| 0x8000001A8 | p[40] |
| 0x8000001AC | p[-40] |
| 0x8000001B0 | p[41] |
| 0x8000001B4 | p[-41] |
| 0x8000001B8 | p[42] |
| 0x8000001BC | p[-42] |
|  | : | : |
| 0x83FFFFF00 | 222 |  |
| 0x83FFFFF04 |  |
| 0x83FFFFF08 | p[8] |
| 0x83FFFFF0C | p[8] |
| 0x83FFFFF10 | p[9] |
| 0x83FFFFF14 | p[-9] |
| 0x83FFFFF18 | p[10] |
| 0x83FFFFF1C | p[-10] |
| 0x83FFFFF20 | p[11] |
| 0x83FFFFF24 | p[-11] |
| 0x83FFFFF28 | p[12] |
| 0x83FFFFF2C | p[-12] |
| 0x83FFFFF30 | p[13] |
| 0x83FFFFF34 | p[-13] |
| 0x83FFFFF38 | p[14] |
| 0x83FFFFF3C | p[-14] |
| 0x83FFFFF40 |  |
| 0x83FFFFF44 |  |
| 0x83FFFFF48 | p[22] |
| 0x83FFFFF4C | p[-22] |
| 0x83FFFFF50 | p[23] |
| 0x83FFFFF54 | p[-23] |
| 0x83FFFFF58 | p[24] |
| 0x83FFFFF5C | p[-24] |
| 0x83FFFFF60 | p[25] |
| 0x83FFFFF64 | p[-25] |
| 0x83FFFFF68 | p[26] |
| 0x83FFFFF6C | p[-26] |
| 0x83FFFFF70 | p[27] |
| 0x83FFFFF74 | p[-27] |
| 0x83FFFFF78 | p[28] |
| 0x83FFFFF7C | p[-28] |
| 0x83FFFFF80 |  |
| 0x83FFFFF84 |  |
| 0x83FFFFF88 | p[36] |
| 0x83FFFFF8C | p[-36] |
| 0x83FFFFF90 | p[37] |
| 0x83FFFFF94 | p[-37] |
| 0x83FFFFF98 | p[38] |
| 0x83FFFFF9C | p[-38] |
| 0x83FFFFFA0 | p[39] |
| 0x83FFFFFF4 | p[-39] |
| 0x83FFFFFA8 | p[40] |
| 0x83FFFFFAC | p[-40] |
| 0x83FFFFFB0 | p[41] |
| 0x83FFFFFB4 | p[-41] |
| 0x83FFFFFB8 | p[42] |
| 0x83FFFFFBC | p[-42] |

Table 6‑11: FOP Organisation in the TLPs over PCIe to the PC/Computer

Note that:-

The byte address range for DDR4 SDRAM #2 as seen via the PCIe is 0x6,0000,0000 to 0x6,FFFF,FFFF.

The byte address range for DDR4 SDRAM #3 as seen via the PCIe is 0x8,0000,0000 to 0x8,FFFF,FFFF.

# FDAS MCI Memory mapped Interface

## FDAS DDR SDRAM Memory Map

Note that the Base Address offset in the PCIe “Type 0” Configuration Space for BAR0 must be added to the addresses detailed in this section.

Note the addresses are byte based.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Usage | Intel Agilex DDR Interface | Base Address (Byte based) | End Address  (Byte based) | Storage Capacity supported |
| Store Incoming DM Spectrum | DDR #0 | 0x200000000 | 0x2FFFFFFFF | FDAS internal address bus is 32-bit (byte based). Hence 4 Mebi Bytes capacity (i.e. 232 bytes). |
| Store lower part of FOP date word. Data[511:0] of CONV and HSUM. | DDR #2 | 0x600000000 | 0x6FFFFFFFF | FDAS internal address bus is 32-bit (byte based). Hence 4 Mebi Bytes capacity (i.e. 232 bytes). |
| Store upper part of FOP data word. Data[1023:512] of CONV and HSUM. | DDR #3 | 0x800000000 | 0x8FFFFFFFF | FDAS internal address bus is 32-bit (byte based). Hence 4 Mebi Bytes capacity (i.e. 232 bytes). |

## FDAS Configuration Memory Map

Note that the Base Address offset in the PCIe “Type 0” Configuration Space for BAR2 must be added to the addresses detailed in this section.

Note the addresses in the Memory Map are byte based.

### Address Decode

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Instance | Module | Size | Base Address | End Address | Mapped To |
| TOPMCI\_1 | MCI\_TOP | 0x0800000 | 0x0000000 | 0x07FFFFF |  |
| CTRL\_1 | CTRL | 0x0400000 | 0x0800000 | 0x0BFFFFF |  |
| MSIX\_1 | MSIX | 0x0000080 | 0x0C00000 | 0x0C0007F |  |
| CONV\_1 | CONV | 0x0800000 | 0x1000000 | 0x17FFFFF |  |
| HSUM\_1 | HSUM | 0x0200000 | 0x1800000 | 0x19FFFFF |  |

### Module MCI\_TOP

#### Inventory

Register: PRODUCT\_ID

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| 0x000000 | RO | PRODUCT\_ID[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PRODUCT\_ID[7:0] | | | | | | | |

Register: CORE\_VERSION

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| 0x000008 | RO | CORE\_VERSION[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CORE\_VERSION[7:0] | | | | | | | |

Register: CORE\_REVISION

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| 0x000010 | RO | CORE\_REVISION[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CORE\_REVISION[7:0] | | | | | | | |

Register: TOP\_VERSION

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| 0x000018 | RO | TOP\_VERSION[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TOP\_VERSION[7:0] | | | | | | | |

Register: TOP\_REVISION

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| 0x000020 | RO | TOP\_REVISION[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| TOP\_REVISION[7:0] | | | | | | | |

PRODUCT\_ID[15:0]: FDAS FPGA Product Number. Hard Coded Value.

CORE\_VERSION[15:0]: FDAS FPGA Core level version number. Hard Coded Value.

CORE\_REVISION[15:0]: FDAS FPGA Core level revision number. Hard Coded Value.

TOP\_VERSION[15:0]: FDAS FPGA Top level version number. Hard Coded Value.

TOP\_REVISION[15:0]: FDAS FPGA Top level revision number. Hard Coded Value.

#### Monitor and Reset Control

Register: CTRL\_RESET (default: 0x00000000)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| 0x000080 | RW |  |  | MSIX\_RESET | DDR\_3\_RESET | DDR\_2\_RESET | DDR\_1\_RESET | DDR\_0\_RESET | DDRIF\_PCIE\_RESET |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DDRIF\_3\_RESET | DDRIF\_2\_RESET | DDRIF\_1\_RESET | DDRIF\_0\_RESET | HSUM\_RESET | CONV\_RESET | CLD\_RESET | CTRL\_RESET |

Register: DDR\_0\_CAL\_FAIL

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| 0x000088 | RO |  |  |  |  | DDR\_3\_RESET\_DONE | DDR\_2\_RESET\_DONE | DDR\_1\_RESET\_DONE | DDR\_0\_RESET\_DONE |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DDR\_3\_CAL\_PASS | DDR\_3\_CAL\_FAIL | DDR\_2\_CAL\_PASS | DDR\_2\_CAL\_FAIL | DDR\_1\_CAL\_PASS | DDR\_1\_CAL\_FAIL | DDR\_0\_CAL\_PASS | DDR\_0\_CAL\_FAIL |

CTRL\_RESET: Reset CTRL Module  
Logic '1' = Reset Default: 0.

CLD\_RESET: Reset CLD Module  
Logic '1' = Reset Default: 0.

CONV\_RESET: Reset CONV Module  
Logic '1' = Reset Default: 0.

HSUM\_RESET: Reset HSUM Module  
Logic '1' = Reset Default: 0.

DDRIF\_0\_RESET: Reset DDRIF2 Module #0 system clock domain and DDR clock domain  
Logic '1' = Reset Default: 0.

DDRIF\_1\_RESET: Reset DDRIF2 Module #1 system clock domain and DDR clock domain  
Logic '1' = Reset Default: 0.

DDRIF\_2\_RESET: Reset DDRIF2 Module #2 system clock domain and DDR clock domain  
Logic '1' = Reset Default: 0.

DDRIF\_3\_RESET: Reset DDRIF2 Module #3 system clock domain and DD clock domain  
Logic '1' = Reset Default: 0.

DDRIF\_PCIE\_RESET: Resets DDRIF2 Modules #0, #1, #2 & #3 PCIE clock domain, and also the MSIX module PCIE clock domain  
Logic '1' = Reset Default: 0.

DDR\_0\_RESET: Reset DDR Controller#0  
Logic '1' = Reset Default: 0.

DDR\_1\_RESET: Reset DDR Controller#1   
Logic '1' = Reset Default: 0.

DDR\_2\_RESET: Reset DDR Controller#2   
Logic '1' = Reset Default: 0.

DDR\_3\_RESET: Reset DDR Controller#3  
Logic '1' = Reset Default: 0.

MSIX\_RESET: Reset MSIX Module system clock domain  
Logic '1' = Reset Default: 0.

DDR\_0\_CAL\_FAIL: DDR Controller #0 has failed Calibration  
Logic '1' = Fail

DDR\_0\_CAL\_PASS: DDR Controller #0 has passed Calibration  
Logic '1' = Pass

DDR\_1\_CAL\_FAIL: DDR Controller #1 has failed Calibration   
Logic '1' = Fail

DDR\_1\_CAL\_PASS: DDR Controller #1 has passed Calibration   
Logic '1' = Pass

DDR\_2\_CAL\_FAIL: DDR Controller #2 has failed Calibration  
Logic '1' = Fail

DDR\_2\_CAL\_PASS: DDR Controller #2 has passed Calibration  
Logic '1' = Pass

DDR\_3\_CAL\_FAIL: DDR Controller #3 has failed Calibration  
Logic '1' = Fail

DDR\_3\_CAL\_PASS: DDR Controller #3 has passed Calibration  
Logic '1' = Pass

DDR\_0\_RESET\_DONE: DDR Controller #0 has reset  
Logic '1' = Reset Done

DDR\_1\_RESET\_DONE: DDR Controller #1 has reset  
Logic '1' = Reset Done

DDR\_2\_RESET\_DONE: DDR Controller #2 has reset  
Logic '1' = Reset Done

DDR\_3\_RESET\_DONE: DDR Controller #3 has reset  
Logic '1' = Reset Done

Reset procedure:  
1) Apply all resets.  
2) Remove DDR\_0\_RESETN, DDR\_1\_RESETN, DDR\_2\_RESETN and DDR\_3\_RESETN  
3) Wait for DDR\_0\_RESET\_DONE, DDR\_1\_RESET\_DONE, DDR\_2\_RESET\_DONE, DDR\_3\_RESET\_DONE, DDR\_0\_CAL\_PASS, DDR\_1\_CAL\_PASS, DDR\_2\_CAL\_PASS and   
DDR\_3\_CAL\_PASS to go high  
4) Remove remaining resets.

### Module CTRL

#### DM Trigger

Register: DM\_TRIG (default: 0x00000000)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0x000000 | RW |  |  |  |  |  |  |  | DM\_TRIG |

DM\_TRIG: A 0>1 transition of DM\_TRIG causes FDAS to process a DM. All FDAS modules are simultaneously triggered. Default: 0.

#### Page Selection

Register: PAGE (default: 0x00000000)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0x000040 | RW |  |  |  |  |  |  |  | PAGE |

PAGE: Indication of which page the processing modules should use as the source of data.  
CLD shall use PAGE  
CONV shall use PAGE  
HSUM shall use NOT(PAGE) Default: 0.

#### Convolution Overlap

Register: OVERLAP\_SIZE (default: 0x00000000)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| 0x000080 | RW |  |  |  |  |  |  | OVERLAP\_SIZE[9:8] | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OVERLAP\_SIZE[7:0] | | | | | | | |

OVERLAP\_SIZE[9:0]: Size of the Convolution Overlap  
"0000000000" = No overlap  
"0000000001" = Overlap of 1 sample  
etc Default: 0.

#### FOP Dimensions

Register: FOP\_SAMPLE\_NUM (default: 0x00000000)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| 0x0000C0 | RW |  | FOP\_SAMPLE\_NUM[22:16] | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| FOP\_SAMPLE\_NUM[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| FOP\_SAMPLE\_NUM[7:0] | | | | | | | |

Register: IFFT\_LOOP\_NUM (default: 0x00000000)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0x0000D0 | RW |  |  | IFFT\_LOOP\_NUM | | | | | |

FOP\_SAMPLE\_NUM[22:0]: The number of 64-bit samples of the DM to be processed to form the FOP.  
This determines the "frequency" dimension of the FOP (i.e. the x-axis).  
  
The number of samples processed is one more than the configuration value.  
  
FDAS processes an integral number of FFTs during the convolution process.   
  
To calculate the FOP\_SAMPLE\_NUMBER the size of the FFT and the OVERLAP\_SIZE configuration must be taken into account.  
  
For example if the FFT size is 1,024 points and the OVERLAP\_SIZE = 500, this means that 524 new samples are processed by each FFT. Hence if we wish to process 2,097,152 samples we will need to perform ROUNDUP(2097152/524) = 4003 FFTs, which equates to 4003 x 524 = 2,097,572 samples.  
  
Hence in this example the value entered into FOP\_SAMPLE\_NUMBER = 2,097,572 - 1 = 2,097,571 decimal = 0x2001A3 Default: 0.

IFFT\_LOOP\_NUM[5:0]: The number of times the CONV module loops through its Inverse Fourier Transforms.   
The number of Inverse Fourier Transforms (IFFTs) processed in each loop is set by the "ifft\_g" generic. Each IFFT generates two rows of the FOP (i.e. the y-axis), -one for positive acceleration "+[p]" and the compliment for negative acceleration -"[p]". The total number of rows of the FOP is given by:-  
  
Total FOP Rows = 1 + ifft\_g\* IFFT\_LOOP\_NUM  
  
Default: 0.

#### Manual Override

Register: MAN\_OVERRIDE (default: 0x00000000)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0x000100 | RW |  |  |  |  |  |  |  | MAN\_OVERRIDE |

MAN\_OVERRIDE: 1 = Enable individual manual control of the triggering of the FDAS modules. Default: 0.

#### Manual Trigger

Register: MAN\_CLD\_TRIG (default: 0x00000000)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0x000140 | RW |  |  |  |  |  | MAN\_HSUM\_TRIG | MAN\_CONV\_TRIG | MAN\_CLD\_TRIG |

MAN\_CLD\_TRIG: If MAN\_OVERRIDE = 1, then a 0>1 transition on MAN\_CLD\_TRIG shall trigger the CLD module to commence processing a DM. Default: 0.

MAN\_CONV\_TRIG: If MAN\_OVERRIDE = 1, then a 0>1 transition on MAN\_CONV\_TRIG shall trigger the CONV module to commence processing a DM. Default: 0.

MAN\_HSUM\_TRIG: If MAN\_OVERRIDE = 1, then a 0>1 transition on MAN\_HSUM\_TRIG shall trigger the HSUM module to commence processing a DM. Default: 0.

#### Manual Enable

Register: MAN\_CLD\_EN (default: 0x00000000)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0x000180 | RW |  |  |  |  |  | MAN\_HSUM\_EN | MAN\_CONV\_EN | MAN\_CLD\_EN |

MAN\_CLD\_EN: If MAN\_OVERRIDE = 1, then if MAN\_CLD\_EN =1 the CLD module operates, however if MAN\_CLD\_EN = 0 the CLD module halts at its current state. Default: 0.

MAN\_CONV\_EN: If MAN\_OVERRIDE = 1, then if MAN\_CONV\_EN =1 the CONV module operates, however if MAN\_CONV\_EN = 0 the CONV module halts at its current state. Default: 0.

MAN\_HSUM\_EN: If MAN\_OVERRIDE = 1, then if MAN\_HSUM\_EN =1 the HSUM module operates, however if MAN\_HSUM\_EN = 0 the HSUM module halts at its current state. Default: 0.

#### Manual Pause Enable

Register: MAN\_CLD\_PAUSE\_EN (default: 0x00000000)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0x0001C0 | RW |  |  |  |  |  | MAN\_HSUM\_PAUSE\_EN | MAN\_CONV\_PAUSE\_EN | MAN\_CLD\_PAUSE\_EN |

MAN\_CLD\_PAUSE\_EN: If MAN\_OVERRIDE = 1, then if MAN\_CLD\_PAUSE\_EN =1 the CLD module automatic pause feature is enabled which will pause the CLD module after MAN\_CLD\_PAUSE\_CNT clock cycles. Toggling MAN\_CLD\_PAUSE\_RST from 0>1 shall recommence CLD operation to the next pause. Default: 0.

MAN\_CONV\_PAUSE\_EN: If MAN\_OVERRIDE = 1, then if MAN\_CONV\_PAUSE\_EN =1 the CONV module automatic pause feature is enabled which will pause the CONV module after MAN\_CONV\_PAUSE\_CNT clock cycles. Toggling MAN\_CONV\_PAUSE\_RST from 0>1 shall recommence CONV operation to the next pause. Default: 0.

MAN\_HSUM\_PAUSE\_EN: If MAN\_OVERRIDE = 1, then if MAN\_HSUM\_PAUSE\_EN =1 the HSUM module automatic pause feature is enabled which will pause the HSUM module after MAN\_HSUM\_PAUSE\_CNT clock cycles. Toggling MAN\_HSUM\_PAUSE\_RST from 0>1 shall recommence HSUM operation to the next pause. Default: 0.

#### Manual Pause Reset

Register: MAN\_CLD\_PAUSE\_RST (default: 0x00000000)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0x000200 | RW |  |  |  |  |  | MAN\_HSUM\_PAUSE\_RST | MAN\_CONV\_PAUSE\_RST | MAN\_CLD\_PAUSE\_RST |

MAN\_CLD\_PAUSE\_RST: If MAN\_OVERRIDE = 1 a 0>1 transition of MAN\_CLD\_PAUSE\_RST shall recommence CLD operation if it is paused. Default: 0.

MAN\_CONV\_PAUSE\_RST: If MAN\_OVERRIDE = 1 a 0>1 transition of MAN\_CONV\_PAUSE\_RST shall recommence CONV operation if it is paused. Default: 0.

MAN\_HSUM\_PAUSE\_RST: If MAN\_OVERRIDE = 1 a 0>1 transition of MAN\_HSUM\_PAUSE\_RST shall recommence HSUM operation if it is paused. Default: 0.

#### Manual Pause Time

Register: MAN\_CLD\_PAUSE\_CNT (default: 0x00000000)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| 0x000240 | RW | MAN\_CLD\_PAUSE\_CNT[31:24] | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MAN\_CLD\_PAUSE\_CNT[23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MAN\_CLD\_PAUSE\_CNT[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MAN\_CLD\_PAUSE\_CNT[7:0] | | | | | | | |

Register: MAN\_CONV\_PAUSE\_CNT (default: 0x00000000)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| 0x000248 | RW | MAN\_CONV\_PAUSE\_CNT[31:24] | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MAN\_CONV\_PAUSE\_CNT[23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MAN\_CONV\_PAUSE\_CNT[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MAN\_CONV\_PAUSE\_CNT[7:0] | | | | | | | |

Register: MAN\_HSUM\_PAUSE\_CNT (default: 0x00000000)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| 0x000250 | RW | MAN\_HSUM\_PAUSE\_CNT[31:24] | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| MAN\_HSUM\_PAUSE\_CNT[23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| MAN\_HSUM\_PAUSE\_CNT[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| MAN\_HSUM\_PAUSE\_CNT[7:0] | | | | | | | |

MAN\_CLD\_PAUSE\_CNT[31:0]: If MAN\_OVERRIDE = 1 and MAN\_CLD\_PAUSE\_EN =1 then the CLD module will pause after MAN\_CLD\_PAUSE\_CNT clock cycles. Default: 0.

MAN\_CONV\_PAUSE\_CNT[31:0]: If MAN\_OVERRIDE = 1 and MAN\_CONV\_PAUSE\_EN =1 then the CONV module will pause after MAN\_CONV\_PAUSE\_CNT clock cycles. Default: 0.

MAN\_HSUM\_PAUSE\_CNT[31:0]: If MAN\_OVERRIDE = 1 and MAN\_HSUM\_PAUSE\_EN =1 then the HSUM module will pause after MAN\_HSUM\_PAUSE\_CNT clock cycles. Default: 0.

#### Done Indications

Register: LATCHED\_CLD\_DONE

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0x000280 | RO |  |  |  |  |  | LATCHED\_HSUM\_DONE | LATCHED\_CONV\_DONE | LATCHED\_CLD\_DONE |

LATCHED\_CLD\_DONE: Latched Indication that CLD has finished processing the DM. 1 = finished.

LATCHED\_CONV\_DONE: Latched Indication that CONV has finished processing the DM. 1 = finished.

LATCHED\_HSUM\_DONE: Latched Indication that HSUM has finished processing the DM. 1 = finished.

#### Pause Indications

Register: CLD\_PAUSED

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0x0002C0 | RO |  |  |  |  |  | HSUM\_PAUSED | CONV\_PAUSED | CLD\_PAUSED |

CLD\_PAUSED: '1' = Latched Indication CLD has paused. If MAN\_OVERRIDE = 1 a 0> 1 Transition of MAN\_CLD\_PAUSE\_RST shall clear CLD\_PAUSED.

CONV\_PAUSED: '1' = Latched Indication CONV has paused. If MAN\_OVERRIDE = 1 a 0> 1 Transition of MAN\_CONV\_PAUSE\_RST shall clear CONV\_PAUSED.

HSUM\_PAUSED: '1' = Latched Indication HSUM has paused. If MAN\_OVERRIDE = 1 a 0> 1 Transition of MAN\_HSUM\_PAUSE\_RST shall clear HSUM\_PAUSED.

#### CONV FFT Ready Indication

Register: CONV\_FFT\_READY

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0x000300 | RO |  |  |  |  |  |  |  | CONV\_FFT\_READY |

CONV\_FFT\_READY: '1' = Indication that the first FFT in the CONV module has completed its processing and is ready for the FFT output to be observed for diagnostic purposes. This signal is asserted when the enable to CONV has been de-asserted and the first FFT has completed its processing.

#### Processing Times

Register: CLD\_PROC\_TIME

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| 0x000310 | RO | CLD\_PROC\_TIME[31:24] | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| CLD\_PROC\_TIME[23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| CLD\_PROC\_TIME[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CLD\_PROC\_TIME[7:0] | | | | | | | |

Register: CONV\_PROC\_TIME

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| 0x000318 | RO | CONV\_PROC\_TIME[31:24] | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| CONV\_PROC\_TIME[23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| CONV\_PROC\_TIME[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CONV\_PROC\_TIME[7:0] | | | | | | | |

Register: HSUM\_PROC\_TIME

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| 0x000320 | RO | HSUM\_PROC\_TIME[31:24] | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| HSUM\_PROC\_TIME[23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| HSUM\_PROC\_TIME[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| HSUM\_PROC\_TIME[7:0] | | | | | | | |

CLD\_PROC\_TIME[31:0]: CLD Processing Time.  
The time from the triggering of CLD to CLD\_DONE measured in CLK\_SYS cycles.

CONV\_PROC\_TIME[31:0]: CONV Processing Time.  
The time from the triggering of CONV to CONV\_DONE measured in CLK\_SYS cycles.

HSUM\_PROC\_TIME[31:0]: HSUM Processing Time.  
The time from the triggering of HSUM to HSUM\_DONE measured in CLK\_SYS cycles.

#### CONV and HSUM DDR SDRAM Access Counts

Register: CONV\_REQ\_CNT

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| 0x000340 | RO | CONV\_REQ\_CNT[31:24] | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| CONV\_REQ\_CNT[23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| CONV\_REQ\_CNT[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CONV\_REQ\_CNT[7:0] | | | | | | | |

Register: HSUM\_REQ\_CNT

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| 0x000348 | RO | HSUM\_REQ\_CNT[31:24] | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| HSUM\_REQ\_CNT[23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| HSUM\_REQ\_CNT[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| HSUM\_REQ\_CNT[7:0] | | | | | | | |

Register: HSUM\_REC\_CNT

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| 0x000350 | RO | HSUM\_REC\_CNT[31:24] | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| HSUM\_REC\_CNT[23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| HSUM\_REC\_CNT[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| HSUM\_REC\_CNT[7:0] | | | | | | | |

CONV\_REQ\_CNT[31:0]: Number of CONV DDR SDRAM Requests

HSUM\_REQ\_CNT[31:0]: Number of HSUM DDR SDRAM Requests

HSUM\_REC\_CNT[31:0]: Number of HSUM DDR SDRAM Received Words

### Module MSIX

#### CLD MSI-X Interrupt Configuration

Register: CLD\_MSIX\_DATA (default: 0x00000000)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| 0x00 | RW |  |  |  |  |  |  |  | CLD\_MSIX\_EN |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| CLD\_MSIX\_DATA[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CLD\_MSIX\_DATA[7:0] | | | | | | | |

CLD\_MSIX\_DATA[15:0]: Data for the CLD MSI-X Interrupt Vector Table Access:-  
Bits[15:12] : rsvd[3:0]  
Bit[11] : msix\_queue\_dir (1 = Host to Device, 0 = Device to Host)  
Bits[10:0]: msix\_queue\_num[10:0] (DMA Channel number or "queue number) Default: 0.

CLD\_MSIX\_EN: 1 = CLD MSI-X Interrupts Enabled Default: 0.

#### CONV MSI-X Interrupt Configuration

Register: CONV\_MSIX\_DATA (default: 0x00000000)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| 0x20 | RW |  |  |  |  |  |  |  | CONV\_MSIX\_EN |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| CONV\_MSIX\_DATA[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| CONV\_MSIX\_DATA[7:0] | | | | | | | |

CONV\_MSIX\_DATA[15:0]: Data for the CONV MSI-X Interrupt Vector Table Access:-  
Bits[15:12] : rsvd[3:0]  
Bit[11] : msix\_queue\_dir (1 = Host to Device, 0 = Device to Host)  
Bits[10:0]: msix\_queue\_num[10:0] (DMA Channel number or "queue number) Default: 0.

CONV\_MSIX\_EN: 1 = CONV MSI-X Interrupts Enabled Default: 0.

#### HSUM MSI-X Interrupt Configuration

Register: HSUM\_MSIX\_DATA (default: 0x00000000)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| 0x40 | RW |  |  |  |  |  |  |  | HSUM\_MSIX\_EN |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| HSUM\_MSIX\_DATA[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| HSUM\_MSIX\_DATA[7:0] | | | | | | | |

HSUM\_MSIX\_DATA[15:0]: Data for the HSUM MSI-X Interrupt Vector Table Access:-  
Bits[15:12] : rsvd[3:0]  
Bit[11] : msix\_queue\_dir (1 = Host to Device, 0 = Device to Host)  
Bits[10:0]: msix\_queue\_num[10:0] (DMA Channel number or "queue number) Default: 0.

HSUM\_MSIX\_EN: 1 =HSUM MSI-X Interrupts Enabled Default: 0.

### Module CONV

#### RAM: FILTER\_COEFFICIENTS

Array, size 42x1024, indexed by 'filter' (0 to 41) and 'word' (0 to 1023). Address range 0x000000 to 0x0A7FFF.

Register: FILTER\_RE\_COEFFICIENTS (default: 0x00000000)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| 0x000000\* | RW | FILTER\_RE\_COEFFICIENTS[31:24] | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| FILTER\_RE\_COEFFICIENTS[23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| FILTER\_RE\_COEFFICIENTS[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| FILTER\_RE\_COEFFICIENTS[7:0] | | | | | | | |

\*Address = 0x000000 + filter \* 16384 + word \* 16.

Register: FILTER\_IM\_COEFFICIENTS (default: 0x00000000)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| 0x000008\* | RW | FILTER\_IM\_COEFFICIENTS[31:24] | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| FILTER\_IM\_COEFFICIENTS[23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| FILTER\_IM\_COEFFICIENTS[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| FILTER\_IM\_COEFFICIENTS[7:0] | | | | | | | |

\*Address = 0x000008 + filter \* 16384 + word \* 16.

FILTER\_RE\_COEFFICIENTS[31:0]: Real Coefficient. Default: 0.

FILTER\_IM\_COEFFICIENTS[31:0]: Imaginary Coefficient. Default: 0.

Filter #N Coefficients  
Base Address N\*2048  
Offset Address 0: 32-bit Real Part Element 0  
Offset Address 1: 32-bit Imaginary Part Element 0  
:  
Offset Address 2046: 32-bit Real Part Element 1023  
Offset Address 2047: 32-bit Imaginary Part Element 1023

#### RAM: FFT\_RESULTS

Array, size 1024, indexed by 'smpl' (0 to 1023). Address range 0x100000 to 0x103FFF.

Register: FFT\_RE

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| 0x100000\* | RO | FFT\_RE[31:24] | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| FFT\_RE[23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| FFT\_RE[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| FFT\_RE[7:0] | | | | | | | |

\*Address = 0x100000 + smpl \* 16.

Register: FFT\_IM

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| 0x100008\* | RO | FFT\_IM[31:24] | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| FFT\_IM[23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| FFT\_IM[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| FFT\_IM[7:0] | | | | | | | |

\*Address = 0x100008 + smpl \* 16.

FFT\_RE[31:0]: Real Sample Data.

FFT\_IM[31:0]: Imaginary Sample Data.

#### Row 0 Delay

Register: ROW0\_DELAY (default: 0x00000000)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| 0x140000 | RW |  |  |  |  |  |  | ROW0\_DELAY[9:8] | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| ROW0\_DELAY[7:0] | | | | | | | |

ROW0\_DELAY[9:0]: Delay applied to Row 0 results Default: 0.

#### ALARMS

Buffer Overflow Alarms

Register: OVERFLOW\_0

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| 0x180000 | RO | OVERFLOW[31:24] | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| OVERFLOW[23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| OVERFLOW[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OVERFLOW[7:0] | | | | | | | |

Register: OVERFLOW\_1

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| 0x180008 | RO | OVERFLOW[63:56] | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| OVERFLOW[55:48] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| OVERFLOW[47:40] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OVERFLOW[39:32] | | | | | | | |

Register: OVERFLOW\_2

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| 0x180010 | RO |  |  |  | OVERFLOW[84:80] | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| OVERFLOW[79:72] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| OVERFLOW[71:64] | | | | | | | |

OVERFLOW[84:0]: [No description].

'1' indicates overflow in the corresponding convolution result buffer  
bit 0: p(0)  
bit 1: p(1)  
bit 2: p(-1)  
:  
bit 83: p(42)  
bit 84: p(-42)  
  
Cleared when CONV module input signal CONV\_TRIGGER='1'

### Module HSUM

HARMONIC SUMMING MODULE

#### FOP SELECTION CONFIGURATION

##### FOP ROW (FILTER "P") SELECTION

Look up table for FOP rows to use in summing.

Array, size summer\_gx2x21x11x16 (max 3x2x21x11x16), indexed by 'SUMMER\_INSTANCE' (0 to summer\_g-1), 'ANALYSIS\_RUN' (0 to 1), 'SEED\_NUM' (0 to 20), 'ACC\_AMBIGUITY\_NUM' (0 to 10) and 'HARMONIC' (0 to 15). Address range 0x000000 to 0x000000+summer\_g\*131072-1 (max 0x000000 to 0x05FFFF).

Register: HPSEL (default: 0x00000000)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0x000000\* | RW |  | HPSEL | | | | | | |

\*Address = 0x000000 + SUMMER\_INSTANCE \* 131072 + ANALYSIS\_RUN \* 65536 + SEED\_NUM \* 2048 + ACC\_AMBIGUITY\_NUM \* 128 + HARMONIC \* 8.

HPSEL[6:0]: The value programmed into the HPSEL register is the FOP Row Number using the encoding shown below.  
The special value 0x60 selects the user configurable value M[31:0] instead.  
(Note: FOP Row 1 is at the bottom and FOP Row 85 is at the top.)  
Default: 0.

|  |  |  |
| --- | --- | --- |
| Value | Label | Description |
| 0x00 | P[0] | FOP row 43(central) |
| 0x01 | P[-0] | FOP row 43(central) |
| 0x02 | P[1] | FOP row 44 |
| 0x03 | P[-1] | FOP row 42 |
| 0x04 | P[2] | FOP row 45 |
| 0x05 | P[-2] | FOP row 41 |
| 0x06 | P[3] | FOP row 46 |
| 0x07 | P[-3] | FOP row 40 |
| 0x08 | P[4] | FOP row 47 |
| 0x09 | P[-4] | FOP row 39 |
| 0x0A | P[5] | FOP row 48 |
| 0x0B | P[-5] | FOP row 38 |
| 0x0C | P[6] | FOP row 49 |
| 0x0D | P[-6] | FOP row 37 |
| 0x0E | P[7] | FOP row 50 |
| 0x0F | P[-7] | FOP row 36 |
| 0x12 | P[8] | FOP row 51 |
| 0x13 | P[-8] | FOP row 35 |
| 0x14 | P[9] | FOP row 52 |
| 0x15 | P[-9] | FOP row 34 |
| 0x16 | P[10] | FOP row 53 |
| 0x17 | P[-10] | FOP row 33 |
| 0x18 | P[11] | FOP row 54 |
| 0x19 | P[-11] | FOP row 32 |
| 0x1A | P[12] | FOP row 55 |
| 0x1B | P[-12] | FOP row 31 |
| 0x1C | P[13] | FOP row 56 |
| 0x1D | P[-13] | FOP row 30 |
| 0x1E | P[14] | FOP row 57 |
| 0x1F | P[-14] | FOP row 29 |
| 0x22 | P[15] | FOP row 58 |
| 0x23 | P[-15] | FOP row 28 |
| 0x24 | P[16] | FOP row 59 |
| 0x25 | P[-16] | FOP row 27 |
| 0x26 | P[17] | FOP row 60 |
| 0x27 | P[-17] | FOP row 26 |
| 0x28 | P[18] | FOP row 61 |
| 0x29 | P[-18] | FOP row 25 |
| 0x2A | P[19] | FOP row 62 |
| 0x2B | P[-19] | FOP row 24 |
| 0x2C | P[20] | FOP row 63 |
| 0x2D | P[-20] | FOP row 23 |
| 0x2E | P[21] | FOP row 64 |
| 0x2F | P[-21] | FOP row 22 |
| 0x32 | P[22] | FOP row 65 |
| 0x33 | P[-22] | FOP row 21 |
| 0x34 | P[23] | FOP row 66 |
| 0x35 | P[-23] | FOP row 20 |
| 0x36 | P[24] | FOP row 67 |
| 0x37 | P[-24] | FOP row 19 |
| 0x38 | P[25] | FOP row 68 |
| 0x39 | P[-25] | FOP row 18 |
| 0x3A | P[26] | FOP row 69 |
| 0x3B | P[-26] | FOP row 17 |
| 0x3C | P[27] | FOP row 70 |
| 0x3D | P[-27] | FOP row 16 |
| 0x3E | P[28] | FOP row 71 |
| 0x3F | P[-28] | FOP row 15 |
| 0x42 | P[29] | FOP row 72 |
| 0x43 | P[-29] | FOP row 14 |
| 0x44 | P[30] | FOP row 73 |
| 0x45 | P[-30] | FOP row 13 |
| 0x46 | P[31] | FOP row 74 |
| 0x47 | P[-31] | FOP row 12 |
| 0x48 | P[32] | FOP row 75 |
| 0x49 | P[-32] | FOP row 11 |
| 0x4A | P[33] | FOP row 76 |
| 0x4B | P[-33] | FOP row 10 |
| 0x4C | P[34] | FOP row 77 |
| 0x4D | P[-34] | FOP row 9 |
| 0x4E | P[35] | FOP row 78 |
| 0x4F | P[-35] | FOP row 8 |
| 0x52 | P[36] | FOP row 79 |
| 0x53 | P[-36] | FOP row 7 |
| 0x54 | P[37] | FOP row 80 |
| 0x55 | P[-37] | FOP row 6 |
| 0x56 | P[38] | FOP row 81 |
| 0x57 | P[-38] | FOP row 5 |
| 0x58 | P[39] | FOP row 82 |
| 0x59 | P[-39] | FOP row 4 |
| 0x5A | P[40] | FOP row 83 |
| 0x5B | P[-40] | FOP row 3 |
| 0x5C | P[41] | FOP row 84 |
| 0x5D | P[-41] | FOP row 2 |
| 0x5E | P[42] | FOP row 85 |
| 0x5F | P[-42] | FOP row 1 |
| 0x60 | M | User configurable value M[31:0] instead of a FOP Row |

#### THRESHOLD CONFIGURATION

Power threshold values for each summer instance for each harmonic and for each of 21 different seed\_f0 row locations within a column of the FOP.  
There are two sets of Thresholds, one for low frequency Periodicity Candidates (T\_SET = 0) and one for high frequency Periodicity Candidates (T\_SET = 1).

Array, size summer\_gx2x21x16 (max 3x2x21x16), indexed by 'SUMMER\_INSTANCE' (0 to summer\_g-1), 'T\_SET' (0 to 1), 'SEED\_NUM' (0 to 20) and 'HARMONIC' (0 to 15). Address range 0x060000 to 0x060000+summer\_g\*8192-1 (max 0x060000 to 0x065FFF).

Register: THRESHOLD (default: 0x00000000)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| 0x060000\* | RW | THRESHOLD[31:24] | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| THRESHOLD[23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| THRESHOLD[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| THRESHOLD[7:0] | | | | | | | |

\*Address = 0x060000 + SUMMER\_INSTANCE \* 8192 + T\_SET \* 4096 + SEED\_NUM \* 128 + HARMONIC \* 8.

THRESHOLD[31:0]: Power Level Threshold at or above which the summed power at the associated location of the SUMMER "summing tree" indicates a potential existence of a Periodicity Candidate. The value is in IEEE 754 32-bit single precision format.  
Default: 0.

#### ANALYSIS RUN PARAMETERS

##### GLOBAL PARAMETERS

Parameters common to all SUMMERs.

Register: B\_START\_1 (default: 0x00000000)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| 0x068000 | RW |  |  | B\_START\_1[21:16] | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| B\_START\_1[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| B\_START\_1[7:0] | | | | | | | |

Register: B\_STOP\_1 (default: 0x00000000)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| 0x068008 | RW |  |  | B\_STOP\_1[21:16] | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| B\_STOP\_1[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| B\_STOP\_1[7:0] | | | | | | | |

Register: B\_START\_2 (default: 0x00000000)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| 0x068010 | RW |  |  | B\_START\_2[21:16] | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| B\_START\_2[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| B\_START\_2[7:0] | | | | | | | |

Register: B\_STOP\_2 (default: 0x00000000)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| 0x068018 | RW |  |  | B\_STOP\_2[21:16] | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| B\_STOP\_2[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| B\_STOP\_2[7:0] | | | | | | | |

Register: H\_1 (default: 0x00000000)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| 0x068020 | RW |  |  |  |  | H\_2 | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|  |  |  |  | H\_1 | | | |

Register: FOP\_ROW\_1 (default: 0x00000055)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| 0x068028 | RW |  | FOP\_ROW\_2 | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|  | FOP\_ROW\_1 | | | | | | |

Register: FOP\_COL\_OFFSET (default: 0x000000D2)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| 0x068030 | RW |  |  |  |  |  |  |  | FOP\_COL\_OFFSET[8] |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| FOP\_COL\_OFFSET[7:0] | | | | | | | |

Register: A\_SET (default: 0x00000000)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0x068040 | RW |  |  |  |  |  |  |  | A\_SET |

Register: THRESH\_SET (default: 0x00000000)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| 0x068048 | RW |  |  | THRESH\_SET[21:16] | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| THRESH\_SET[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| THRESH\_SET[7:0] | | | | | | | |

Register: M (default: 0x00000000)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| 0x068050 | RW | M[31:24] | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| M[23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| M[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| M[7:0] | | | | | | | |

Register: T\_FILTER\_EN (default: 0x00000000)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0x068060 | RW |  |  |  |  |  |  |  | T\_FILTER\_EN |

B\_START\_1[21:0]: The first FOP column containing a seed\_f0 analysed in the first analysis run of the DM. Default: 0.

B\_STOP\_1[21:0]: The last FOP column containing a seed\_f0 analysed in the first analysis run of the DM. Default: 0.

B\_START\_2[21:0]: The first FOP column containing a seed\_f0 analysed in the second analysis run of the DM. Default: 0.

B\_STOP\_2[21:0]: The last FOP column containing a seed\_f0 analysed in the second analysis run of the DM. Default: 0.

H\_1[3:0]: The number of harmonics analysed in the first analysis run of the DM. ("0000" = 1 harmonic analysed, "1111" = 16 harmonics analysed.) Default: 0.

H\_2[3:0]: The number of harmonics analysed in the second analysis run of the DM. Default: 0.

FOP\_ROW\_1[6:0]: The number of FOP rows to read from DDR for each column for the first analysis run. Valid values are 1 to 85. Default: 85.

FOP\_ROW\_2[6:0]: The number of FOP rows to read from DDR for each column for the second analysis run. Valid values are 1 to 85. Default: 0.

FOP\_COL\_OFFSET[8:0]: First valid column of FOP data. Default: 210.

A\_SET: The number of analysis runs performed on a DM. '0' = 1 analysis run performed, '1' = 2 analysis runs performed. Default: 0.

THRESH\_SET[21:0]: The FOP column number at which the power thresholds used in the harmonic summing change from the "Low Periodicity Candidate Frequency" set to the "High Periodicity Candidate Frequency Set". Default: 0.

M[31:0]: If the HPSEL value for a harmonic is set to 0x60 then instead of FOP location values being passed to the SUMMER sub-module, this value M[31:0] is passed instead. Default: 0.

T\_FILTER\_EN: If set to 1 then only the results from the highest harmonic are stored for a particular seed\_f0 run. Default: 0.

FOP Column Numbering:  
0x000000 = FOP Column 1  
0x000001 = FOP Column 2  
:  
0x3FFFFF = FOP Column 4,194,304

##### PER SUMMER RUN PARAMETERS

Array, size summer\_g (max 3), indexed by 'SUMMER\_INSTANCE' (0 to summer\_g-1). Address range 0x068080 to 0x068080+summer\_g\*16-1 (max 0x068080 to 0x0680AF).

Register: P\_EN\_1 (default: 0x00000000)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| 0x068080\* | RW |  |  |  | P\_EN\_2 | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|  |  |  | P\_EN\_1 | | | | |

\*Address = 0x068080 + SUMMER\_INSTANCE \* 16.

Register: A\_1 (default: 0x00000000)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| 0x068088\* | RW |  |  |  |  | A\_2 | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|  |  |  |  | A\_1 | | | |

\*Address = 0x068088 + SUMMER\_INSTANCE \* 16.

P\_EN\_1[4:0]:   
The number of orbital accelerations (i.e. seed\_f0s in a FOP column) analysed in the first analysis run of the DM.  
Default: 0.

P\_EN\_2[4:0]:   
The number of orbital accelerations (i.e. seed\_f0s in a FOP column) analysed in the second analysis run of the DM.  
"00000" = 1 orbital acceleration analysed  
"00001" = 2 orbital accelerations analysed  
:  
"10100" = 21 orbital accelerations analysed  
Default: 0.

A\_1[3:0]:   
The number of orbital acceleration ambiguity slopes analysed in the first analysis run of the DM.  
Default: 0.

A\_2[3:0]:   
The number of orbital acceleration ambiguity slopes analysed in the second analysis run of the DM.  
"0000" = 1 orbital acceleration ambiguity slope analysed  
"0001" = 2 orbital acceleration ambiguity slopes analysed  
:  
"1010" = 11 orbital acceleration ambiguity slopes analysed Default: 0.

##### DM COUNTER

Register: DM\_CNT

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| 0x068100 | RO | DM\_CNT[31:24] | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| DM\_CNT[23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| DM\_CNT[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| DM\_CNT[7:0] | | | | | | | |

Register: DM\_CNT\_RESET (default: 0x00000000)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0x068108 | RW |  |  |  |  |  |  |  | DM\_CNT\_RESET |

DM\_CNT[31:0]: Free Running count of the DMs that have been processed by HSUM.   
DM\_CNT can be reset to zero by writing to DM\_CNT\_RESET ('0' to '1' transition).

DM\_CNT\_RESET: A 0 to 1 transition resets the free running DM Counter which can be used to identify which DM the Threshold Crossing results belong to. Default: 0.

#### SUMMING RESULTS

##### HARMONIC THRESHOLD CROSSING RESULTS

Two pages of 25-off Results per harmonic per Analysis Run, each consisting of 4-off 32-bit words.  
The first page contains the static results from the previous run. The second page contains the results for the current run which will change as the run progresses, but may be accessed for debug purposes if the processing is paused.

Array, size 2x2x16x25, indexed by 'PAGE' (0 to 1), 'ANALYSIS\_RUN' (0 to 1), 'HARMONIC' (0 to 15) and 'HARMONIC\_REPORT\_NUM' (0 to 24). Address range 0x080000 to 0x08FFFF.

Register: HARMONIC

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| 0x080000\* | RO |  |  |  |  |  |  |  | VALID |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
|  |  |  |  | HARMONIC | | | |

\*Address = 0x080000 + PAGE \* 32768 + ANALYSIS\_RUN \* 16384 + HARMONIC \* 1024 + HARMONIC\_REPORT\_NUM \* 32.

Register: FOPCOL

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| 0x080008\* | RO |  |  | FOPCOL[21:16] | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| FOPCOL[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| FOPCOL[7:0] | | | | | | | |

\*Address = 0x080008 + PAGE \* 32768 + ANALYSIS\_RUN \* 16384 + HARMONIC \* 1024 + HARMONIC\_REPORT\_NUM \* 32.

Register: FOPROW

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0x080010\* | RO |  | FOPROW | | | | | | |

\*Address = 0x080010 + PAGE \* 32768 + ANALYSIS\_RUN \* 16384 + HARMONIC \* 1024 + HARMONIC\_REPORT\_NUM \* 32.

Register: PWR

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| 0x080018\* | RO | PWR[31:24] | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| PWR[23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| PWR[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| PWR[7:0] | | | | | | | |

\*Address = 0x080018 + PAGE \* 32768 + ANALYSIS\_RUN \* 16384 + HARMONIC \* 1024 + HARMONIC\_REPORT\_NUM \* 32.

VALID: Indicates this result record is valid.  
Valid records will be first followed by invalid records. They will not be mixed.

|  |  |  |
| --- | --- | --- |
| Value | Label | Description |
| 0 | INVALID |  |
| 1 | VALID |  |

HARMONIC[3:0]: Harmonic Number of the Threshold Crossing Location. (Note that the report address also inherently provides the harmonic).

FOPCOL[21:0]: FOP Column [b] of the Threshold Crossing Location.

FOPROW[6:0]: FOP Row [p] of the Threshold Crossing Point encoded with the same mapping as HPSEL.

PWR[31:0]: The summed power at the Threshold Crossing point in IEEE 754 Format.

##### STORAGE EXCEEDED REPORTS

Information regarding threshold crossing reports that could not be stored. This information is presented on a per harmonic per analysis basis.  
There are two pages. The first page contains the static results from the previous run. The second page contains the results for the current run which will change as the run progresses, but may be accessed for debug purposes if the processing is paused.

Array, size 2x2x16, indexed by 'PAGE' (0 to 1), 'ANALYSIS\_RUN' (0 to 1) and 'HARMONIC' (0 to 15). Address range 0x100000 to 0x1007FF.

Register: T\_EXC

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| 0x100000\* | RO | T\_EXC[31:24] | | | | | | | |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| T\_EXC[23:16] | | | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| T\_EXC[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| T\_EXC[7:0] | | | | | | | |

\*Address = 0x100000 + PAGE \* 1024 + ANALYSIS\_RUN \* 512 + HARMONIC \* 32.

Register: S\_EXC

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |
| 0x100008\* | RO |  |  | S\_EXC[21:16] | | | | | |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |
| S\_EXC[15:8] | | | | | | | |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| S\_EXC[7:0] | | | | | | | |

\*Address = 0x100008 + PAGE \* 1024 + ANALYSIS\_RUN \* 512 + HARMONIC \* 32.

Register: P\_EXC

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 0x100010\* | RO |  | P\_EXC | | | | | | |

\*Address = 0x100010 + PAGE \* 1024 + ANALYSIS\_RUN \* 512 + HARMONIC \* 32.

T\_EXC[31:0]: The number of threshold crossing results that could not be recorded.

S\_EXC[21:0]: FOP Column number of the seed\_f0 for the first threshold crossing that could not be recorded.

P\_EXC[6:0]: FOP Row [p] for the first threshold crossing that could not be recorded, encoded using the same mapping as HPSEL.

# FDAS Parameterisation

The FDAS design has the following generic parameters at the top level:

| Generic | Type | Value (decimal unless stated otherwise) | Description |
| --- | --- | --- | --- |
| product\_id\_g[15:0] | natural | 64933  (Hex value 0xFDA5) | Product ID value that is read back via the MC Interface. |
| Core\_version\_g[15:0] | natural | 2574  (Hex value 0x0A0E)  [12:11] = summer\_g =1  [10:6] = harmonic\_g = 8  [5:0] = ifft\_g = 14 | Core Version Number value that is read back via the MC Interface. |
| Core\_revision\_g[15:0] | natural | 1 | Core Revision Number value that is read back via the MC Interface. |
| Top\_version\_g[15:0] | natural | 290  (Hex value 0x0122)  [9:6] = FDAS version on SVN = 4  [5:3] = Total no of DDR I/Fs Required = 4  [2:0] = ddr\_g = 2 dec | Top Version Number value that is read back via the MC Interface.  Note that 4 DDR I/Fs are required (as there are 4 DDR Controllers, although only 3 are used) |
| top\_revision\_g[15:0] | natural | 1 | Top Revision Number value that is read back via the MC Interface. |
| Ddr\_g | natural | 2 | Number of external DDR of interfaces storing the FOP |
| fft\_g | natural | 1024 | Number of points in an FFT |
| fft\_abits\_g | natural | log2(fft\_g) = 10 | Number of bits for FFT point counter |
| ifft\_g | natural | 14 | The number of CONV\_IFFT sub-module Instantiations. Each instantiation generates two rows of the FOP per processing iteration loop. |
| Ifft\_loop\_g | natural | 3 | Number of times the CONV\_IFFT sub-module instantiations are used for a DM to produce the FOP.  FOP Rows = 1 + [ 2x(ifft\_g x ifft\_loop\_g)] |
| ifft\_loop\_bits\_g | natural | log2(ifft\_loop\_g) = 2 | Number of bits for iteration count |
| res\_pages\_g | natural | 3 | Number of pages to store convolution results in CONV |
| summer\_g | natural | 1 | The number of instances of the SUMMER sub-module within the HSUM module. |
| Harmonic\_g | natural | 8 | The maximum number of harmonics that the HSUM module is capable of supporting.  Currently for the Enhanced Summing Tree this generic is limited to 12. |

Table 8‑1: FDAS Top Level Generic Parameters

# FDAS FPGA Resource Utilisation

The FPGA resource utilisation for this implementation of the FDAS design for the generic settings detailed in sec 8 and with a .qsf SEED setting of 6 is shown in the table below:

Quartus Version: 22.4

Device: Agilex F, AGFB014R24B2E2V

|  |  |  |  |
| --- | --- | --- | --- |
| Resource | Used | Total available in Device | % Utilisation |
| Logic Utilisation ALMs | 178,048 | 487,200 | 37 |
| Register | 393,799 | 1,948,800 | 20 |
| External Pins | 599 | 924 | 65 |
| Memory block bits | 27,043,416 | 145,612,800 | 19 |
| RAM Blocks (M20K) | 2,396 | 7,110 | 34 |
| DSP Blocks | 1,321 | 4,510 | 29 |
| PLLs | 16 | 24 | 67 |

Table 9‑1: FPGA Resource Utilisation for this FDAS Implementation

# FDAS FPGA Timing Analysis

The maximum clock speeds for this implementation of the FDAS design for the generic settings detailed in sec 8 and with a .qsf SEED setting of 6 is shown in the table below:

Quartus Version: 22.4

Device: Agilex F, AGFB014R24B2E2V

|  |  |  |  |
| --- | --- | --- | --- |
| Clock  Factor | Description | Required Frequency  /MHz | Actual Frequency  (900mV 100°C Slow Silicon)  /MHz |
| CLK\_SYS | Main processing clock for the CLD, CONV and HSUM modules | 350 | 377.22 |
| CLK\_MC | Memory Map Micro Configuration clock. Connected to CLK\_SYS in this implementation | 350 | 377.22 |
| CLK\_PCIE | Clock generated by the PCIe Hard IP Macro to the DDRIF2 and PCIF modules and the PLL to generate CLK\_SYS. | 350 | 417.36 |
| CLK\_DDR0 | Clock from DDR\_CONTROLLER to the DDRIF2 module #0 | 300 | 329.16 |
| CLK\_DDR1 | Clock from DDR\_CONTROLLER to the DDRIF2 module #1 | 300 | 404.53 |
| CLK\_DDR2 | Clock from DDR\_CONTROLLER to the DDRIF2 module #2 | 300 | 335.01 |
| CLK\_DDR3 | Clock from DDR\_CONTROLLER to the DDRIF2 module #3 | 300 | 356.25 |

Table 10‑1: Timing Analysis for this Implementation of FDAS

# FDAS FPGA Power Estimate

The power estimation is from the build using the FDAS\_TOP.qptc file loaded into the Quartus Power and Thermal Calculator.

In Quartus on the top banner menu select “Tools” > “Power and Thermal Calculator” and the select “OK” for the window asking to “open file from Quartus Power Analyser”.

The estimated power dissipation is:-

Quartus Version: 22.4

Device: Agilex F, AGFB014R24B2E2V

**25°C Junction Temperature**:

Dynamic Power = 37.902W

Static Power = 2.963W

Smart VID power saving = -1.061W

Total Power =39.804W

**100°C Junction Temperature**:

Dynamic Power = 37.781W

Static Power = 13.357W

Smart VID power saving = -3.943W

Total Power =47.196W

Note the 100°C Junction Temperature is representative of the maximum temperature would be expected to operate at.

# FDAS Measured Processing Time

The Processing time for this implementation of the FDAS design for the generic settings detailed in sec 8

**Accelerated Pulsar Search with 8 Harmonics**

|  |  |  |  |
| --- | --- | --- | --- |
| Process | Processing Parameters | Time measured in CLK\_SYS clock cycles @ 350MHz | Time  /ms |
| DM observation transfer via PCIe to FDAS | 16 Lane PCIe Gen 4 connecting to 64-bit DD4 SDRAM operating at 1200MHz.  PCIe theoretical bandwidth is 32Gbyte/sec (2Gbyte/sec per lane).  DDR SDRAM theoretical bandwidth = 19.2Gbyte/sec |  | 3.34 Note 2  (for 32Mbyte transfer) |
| Convolution | * FOP\_SAMPLE\_NUM =4,194,175 dec (4,194,176 Samples) * OVERLAP\_SIZE = 420 dec * IFFT\_LOOP\_NUM = 3 dec (6 processing loops) * fft\_g generic = 1024 (1024 point FFTs) * ifft\_g generic = 14 (14 IFFT sub-modules generating 28 FOP rows per processing loop) * FOP = 85 row x 4,194,176 sample | 21,358,537 Note 1 | 61.03 |
| Harmonic Summing | * summer\_g generic = 1 * 262,144 seed\_f0 FOP columns analysed * 85 FOP rows analysed per seed\_f0 * 8 harmonics summed per seed\_f0 * 21 accelerations (rows) per seed\_f0 FOP Column * 11 orbital acceleration ambiguity slopes per seed\_f0 * Results filtered to only store the results for the highest harmonic (this has the greatest processing time). * One analysis run | 67,109,213 Note 1 | 191.74 |
| FDAS Accelerated Pulsar Search time/DM |  |  | 252.77 Note 3 |

Note 1: The processing times have been measured using the Processing Time counters in the CTRL module which measure the time from when each module in the table above is triggered to process a DM to when it finishes processing.

Note 2: The next DM Observation data is transferred to FDAS from the Host PC whilst HSUM is summing the previous DM.

Note 3: In this implementation of FDAS, the CLD and CONV modules operate simultaneously followed by the HSUM module when the CONV module has finished. Hence the overall DM processing time is the summation of the Convolution and Harmonic Summing processing times.

Note 4: For diagnostic purposes the FOP can be read by the Host PC via the PCIe. This takes 400ms to perform (to read 2Gbytes of data).

# FDAS Performance Comparisons

The resource useage and performance of the three DDR version of the FDAS FPGA can be compared with previous versions.

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Arria 10**  **10AX115N3F40E2SG** | **Agilex**  **AGFB014R24A2E2VR0** | |
| Geometry | 20nm | 10nm with “hyperflex” retime | |
| Max. Core clock speed | 200MHz | 350MHz | 350MHz |
| DM Samples Convoluted | 4,194,304 | 4,194,304 | 4,194,304 |
| FOP Rows supported | 85 | 85 | 85 |
| FOP DDR Interfaces | 1 | 1 | 2 |
| CONV Filters/loop | 7 +ve accn/  7 –ve accn | 7 +ve accn/  7 –ve accn | 1. +ve accn/   14 –ve accn |
| HSUM Harmonics | 8 processed/  12 supported | 8 processed/  12 supported | 8 processed/  8 supported |
| HSUM Seed\_f0 FOP Cols | 262,144 | 262,144 | 262,144 |
| HSUM Seed\_f0s per FOP Col | 21 | 21 | 21 |
| HSUM Accn Ambiguity Slopes | 11 | 11 | 11 |
| Logic Blocks used | 157,374/427,200 (37%) | 169,240 / 487,200 (35% ) | 178,114 / 487,200 (37% ) |
| Registers used | 268,509/1,708,800 (16%) | 294,692/1,948,800 (15%) | 393,703/1,948,800 (20%) |
| RAM Blocks used | 1,487/2,713 (55%) | 1,871 / 7,110 ( 26% ) | 2,396 / 7,110 ( 34% ) |
| DSP Blocks used | 1,379/1,518 (91%) | 959 / 4,510 ( 21% ) | 1,321 / 4,510 ( 29% ) |
| DDR Interfaces used | 2/2 (100%) | 2/4 (50%) | ¾ (75%) |
| Power Consumption | 30W @ 56°C  (measured) | 32W @ 100°C  (basic analysis) | 47W @ 100°C (accurated analysis) |
| Module Processing time/DM (measured) | CONV: 2x107ms  HSUM: 337ms | CONV: 122ms  HSUM: 254ms | CONV: 61ms  HSUM: 192ms |
| **Total Accelerated Search Processing time/ DM** | **551ms** | **376ms** | **253ms** |

Table 13‑1: FDAS Performance Comparisons

## Comparison Summary

The requirement is to process a DM in 357.9ms. The only FDAS design that meets this with the full 222 sample convolutuon is the three DDR build in Intel Agilex.

It should be noted that actual requirement is to process pulsars with a frequency of up to 500Hz.

Each frequency-bin of the FOP covers a range of:

1/(223 samples \* 64us sample period) = 1/536.87 Hz

Hence a 500Hz pulsar’s fundamental harmonic would be at frequency-bin

536.87\*500 = 268435.

Since 8 harmonics are analysed the highest FOP frequency-bin that is required is

8 \* 268435 = 2147480.

Taking these figures into account the estimated processing times would be:-

CONV Processing time =2147480/222 = 52% of the values stated in Table 13‑1

HSUM Processing time = 268435/262144 = 103% of the values stated in Table 13‑1

Hence the processing times would be:

|  |  |  |  |
| --- | --- | --- | --- |
| **Parameter** | **Arria 10**  **10AX115N3F40E2SG** | **Agilex**  **AGFB014R24A2E2VR0** | |
| Module Processing time/DM (estimate) | CONV: 112ms  HSUM: 348ms | CONV: 64ms  HSUM: 262ms | CONV: 32ms  HSUM: 198ms |
| **Total Accelerated Search Processing time/ DM**  **(estimate)** | **460ms** | **326ms** | **230ms** |

Table 13‑2: Esitimated FDAS Processing Times for Pulsars up to 500Hz

On this basis both the two DDR (one DDR storing the FOP) and three DDR (two DDRs storing the FOP) Intel Agilex FDAS versions would meet the required processing times.

# Intel Agilex F Card Measured Power Consumption

The Intel Agilex F card power monitor was observed with the FDAS Image loaded in the Agilex FPGA, with the generic settings detailed in sec 8.

A measurement was made with the FDAS image loaded but inactive and also a measurement with the FDAS CLD/CONV & HSUM modules operating continuously (i.e. as soon as processing of a DM has completed, CLD & CONV modules are re-triggered).

## Power Characteristic with FDAS Inactive

FDAS generics set as shown in Section 8.

|  |  |  |  |
| --- | --- | --- | --- |
| Measured Agilex F Card Parameter | Voltage | Current | Power |
| 0.9V Power Rail | 899.658mV | 7093mA | 6.381W |
| 3.3V Power Rail | 3297.485mV | 3226mA | 10.638W |
| 1.8V Power Rail | 1798.706mV | 2015mA | 3.624W |
| **TOTAL POWER** |  |  | **20.64 W** |
| FPGA Core Temperature |  |  | 78.5 °C |
| Board Temperature |  |  | 44.9 °C |

Table 14‑1: Power Characteristics with FDAS Inactive

## Power Characteristic with FDAS Operating Continuously

FDAS generics set as shown in Section 8.

CONV processing 4,194,176 samples.

HSUM Processing:-

* 262,144 seed\_f0 columns,
* Accelerated Pulsar Search:
  + 85 FOP rows
  + 8 Harmonics
  + 21 seed\_f0 rows per FOP column
  + 11 orbital acceleration ambiguity slopes

The next DM is triggered as soon as the current DM has completed processing.

FDAS generics set as shown in Section 8.

|  |  |  |  |
| --- | --- | --- | --- |
| Measured Agilex F Card Parameter | Voltage | Current | Power |
| 0.9V Power Rail | 899.658mV | 7000mA | 6.298W |
| 3.3V Power Rail | 3297.485mV | 4062mA | 13.394W |
| 1.8V Power Rail | 1798.706mV | 2042mA | 3.674W |
| **TOTAL POWER** |  |  | **23.36 W** |
| FPGA Core Temperature |  |  | 79.4 °C |
| Board Temperature |  |  | 45.0 °C |

Table 14‑2: Power Characteristics with FDAS CLD, CONV & HSUM Modules Operating Continuously (albeit with CLD/CONV and HSUM operating in anti-phase).

# Appendix A: Overview of PCIe in Agilex F

## PCIe Physical Protocol Layers

See the following website for a good basic overview of PCIe:-

<http://xillybus.com/tutorials/pci-express-tlp-pcie-primer-tutorial-guide-1>

PCIe is a point-to- point connection topology, with a “root” end, usually on the PC Mother board and an “endpoint” on a plug-in PCIe card.

PCIe uses a lane topology. Each lane consists of a pair of differential wires in each direction to allow full duplex communication. The number of lanes is indicated by:-

(x1) : 1 lane

(x2): 2 lanes

(x4): 4 lanes

(x8): 8 lanes

(x12): 12 lanes

(x16): 16 lanes

(x32): 32 lanes

There are various versions of PCIe with different coding schemes and transfer rates:-

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| PCIe  Version | Altera Gen | Line Code | Transfer rate  Per lane | Bandwidth per lane |
| 1.0 | Gen 1 | 8b/10b | 2.5 GT/s | 2 Gbit/s (250Mbyte/s) |
| 2.0 | Gen 2 | 8b/10b | 5 GT/s | 4 Gbit/s (500Mbyte/s) |
| 3.0 | Gen 3 | 128b/130b | 8 GT/s | 7.877 Gbit/s (984.6Mbyte/s) |
| 4.0 | Gen 4 | 128b/130b | 16 GT/s | 15.754 Gbit/s (1969.2Mbyte/s) |

Table 15‑1: PCIe Versions

The data protocol is a packetized architecture with a three layer stack, - the Transaction Layer, the Data Link Layer and the Physical Layer. The Transaction Layer Packets (TLPs) are shown as follows:-

TLP Write:-

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| D  W | 3  1 | 3  0 | 2  9 | 2  8 | 2  7 | 2  6 | 2  5 | 2  4 | 2  3 | 2  2 | 2  1 | 2  0 | 1  9 | 1  8 | 1  7 | 1  6 | 1  5 | 1  4 | 1  3 | 1  2 | 1  1 | 1  0 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | | 0 |
| D  W  0 | R  0 | Fmt  0x2 | | Type  0x00 | | | | | R  0 | TC | | | R  0 | | | | T  D | E  P | Attr | | R  0 | | Length  (number of 32-bit Data Words) | | | | | | | | | | |
| D  W  1 | Requester ID | | | | | | | | | | | | | | | | Tag  (unused) | | | | | | | | Last BE | | | | 1st BE | | | | |
| D  W  2 | Address[31:2] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | R  0 | |
| D  W  3 | Data[31:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

R = Reserved set to 0

Fmt/Type: The above value indicates this is a memory write request

Length: The number of 32-bit data words in the packet

Address: only [31:2] set so that a x4 puts address onto 32-bit boundaries. Note that for 64-bit addressing there is an extra DW after DW2 containing Address[63:32].

BE (1st/last) indicates the valid bytes within first and last words

TLP Read Request:-

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| D  W | 3  1 | 3  0 | 2  9 | 2  8 | 2  7 | 2  6 | 2  5 | 2  4 | 2  3 | 2  2 | 2  1 | 2  0 | 1  9 | 1  8 | 1  7 | 1  6 | 1  5 | 1  4 | 1  3 | 1  2 | 1  1 | 1  0 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| D  W  0 | R  0 | Fmt  0x0 | | Type  0x00 | | | | | R  0 | TC | | | R  0 | | | | T  D | E  P | Attr | | R  0 | | Length  (number of 32-bit Data Words | | | | | | | | | |
| D  W  1 | Requester ID | | | | | | | | | | | | | | | | Tag  (tracking number) | | | | | | | | Last BE | | | | 1st BE | | | |
| D  W  2 | Address[31:2] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | R  0 | |

R = Reserved set to 0

Fmt/Type: The above value indicates this is a read request

Length: The number of 32-bit data words in the packet

Address: only [31:2] set so that a x4 puts address onto 32-bit boundaries. Note that for 64-bit addressing there is an extra DW after DW2 containing Address[63:32].

Tag: This allows the completion packet to be related to the read request packet.

BE (1st/last) indicates the bytes to read within a the first and last words

TLP Completion:-

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| D  W | 3  1 | 3  0 | 2  9 | 2  8 | 2  7 | 2  6 | 2  5 | 2  4 | 2  3 | 2  2 | 2  1 | 2  0 | 1  9 | 1  8 | 1  7 | 1  6 | 1  5 | 1  4 | 1  3 | 1  2 | 1  1 | 1  0 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| D  W  0 | R  0 | Fmt  0x2 | | Type  0x0A | | | | | R  0 | TC | | | R  0 | | | | T  D | E  P | Attr | | R  0 | | Length  (number of 32-bit Data Words) | | | | | | | | | |
| D  W  1 | Completer ID | | | | | | | | | | | | | | | | Status | | | B  C  M | Byte Count | | | | | | | | | | | |
| D  W  2 | Requester ID | | | | | | | | | | | | | | | | Tag | | | | | | | | R  0 | Lower Address | | | | | | |
| D  W  3 | Data[31:0] | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |

R = Reserved set to 0

Fmt/Type: The above value indicates this is a memory write request

Length: The number of 32-bit data words in the packet

Status: 0x0 indicates the completion was successful

Byte Count: The number of bytes left for transmission – including the current packet (useful info when the completion needs to use several packets).

Lower Address: Least significant 7 bits of the address from which the first byte in the TLP was read.

BE (1st/last) indicates the valid bytes within a word

## PCIe Configuration Space

Some useful website links are:-

<https://en.wikipedia.org/wiki/PCI_Express>

<https://en.wikipedia.org/wiki/PCI_configuration_space>

<http://wiki.osdev.org/PCI>

<https://www.altera.com/en_US/pdfs/literature/ug/ug_a5_pcie_avmm.pdf>

As an Endpoint the PCIe function’s parameters are stored within “Type 0” “Configuration Space”. The read only fields of this Configuration Space are populated when the PCIe is created in Intel Quartus Prime Platform Designer.

Historically the PCI Configuration Space was 256 bytes. However the PCIe Configuration Space has been extended up to 4096 bytes, with the first 256 bytes having the same meaning as for PCI.

The specification allows for up to 256 busses (i.e. endpoints) in a system, with up to 32 devices on each bus, each supporting eight functions, with each function having its own Configuration Space.

Hence 256 busses x 32 devices x 8 functions x 4096 bytes of computer memory have to be allocated to store the Configuration Space values that need to be written to endpoint PCIe cards that can be plugged into the system.

Configuration Space Access Method #1 (PCI):

Originally only two I/O locations of the CPU were used to access the Configuration Space:-

0xCF8: CONFIG\_ADDRESS

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 31 | 30-34 | 23-16 | 15-11 | 10-8 | 7-2 | 1-0 |
| Enable | Reserved | Bus No | Device No | Function No | Register No | 00 |

(Each register is 32-bit, hence 64 required for 256 bytes).

0xCFC: CONFIG\_DATA

Configuration Space Access Method #2 (PCIe):

PCIe introduced a new way to access the Configuration Space, where it is simply memory mapped and no I/O ports are used. Hence part of the CPU’s memory is sacrificed for this purpose. The first part of the Configuration Space is shown below:-

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 31 24 | 23 16 | | | | 15 8 | 7 0 |
| 0x000 | Device ID | | | | | Vendor ID | |
| 0x004 | Status | | | | | Command | |
| 0x008 | Class Code | Sub class | | | | ProgIF | Revision ID |
| 0x00C | BIST | Header Type | | | | Latency Timer | Cache Line Size |
| 0x010 | Base Address Register# 0 ( BAR0) | | | | | | |
| 0x014 | Base Address Register# 1 ( BAR1) | | | | | | |
| 0x018 | Base Address Register# 2 ( BAR2) | | | | | | |
| 0x01C | Base Address Register# 3 ( BAR3) | | | | | | |
| 0x020 | Base Address Register# 4 ( BAR4) | | | | | | |
| 0x024 | Base Address Register# 5 ( BAR5) | | | | | | |
| 0x028 | Card Bus CIS Pointer | | | | | | |
| 0x02C | Subsystem Device ID | | | | | Subsystem Vendor ID | |
| 0x030 | Expansion ROM Base Address | | | | | | |
| 0x034 | Reserved | | | | | | Capabilities Pointer |
| 0x038 | Reserved | | | | | | |
| 0x03C | Max Latency | Min Grant | | | | Interrupt Pin | Interrupt Line |
| 0x040  0x04C | Reserved | | | | | | |
| 0x050 | Reserved | 6  4  B  i  t | Multiple MSI  Message  Enable  1,2,4,8,16,  32 | Multiple  MSI  Message  Capable  1,2,4,8,16,  32 | M  S  I  E  N | Pointer to Next ID | Capability ID = 0x05 |
| 0x054 | MSI Message Address | | | | | | |
| 0x058 | MSI Message Upper Address | | | | | | |
| 0x05C | Reserved | Reserved | | | | MSI Message Data | |

The following field descriptions are common to all Header Types:

* **Device ID:** Identifies the particular device. Where valid IDs are allocated by the vendor.
* **Vendor ID:** Identifies the manufacturer of the device. Where valid IDs are allocated by PCI-SIG to ensure uniqueness and 0xFFFF is an invalid value that will be returned on read accesses to Configuration Space registers of non-existent devices.
* **Status:** A register used to record status information for PCI bus related events.
* **Command:** Provides control over a device’s ability to generate and respond to PCI cycles. Where the only functionality guaranteed to be supported by all devices is when a 0 is written to this register, the device is disconnected from the PCI bus for all accesses except Configuration Space access.
* **Class Code:** A read-only register that specifies the type of function the device performs.
* **Subclass:** A read-only register that specifies the specific function the device performs.
* **Prog IF:** A read-only register that specifies a register-level programming interface the device has, if it has any at all.
* **Revision ID:** Specifies a revision identifier for a particular device. Where valid IDs are allocated by the vendor.
* **BIST:** Represents that status and allows control of a device’s BIST (built-in self-test).
* **Header Type:** Identifies the layout of the rest of the header beginning at byte 0x10 of the header and also specifies whether or not the device has multiple functions. Where a value of 0x00 specifies a general device, a value of 0x01 specifies a PCI-to-PCI bridge, and a value of 0x02 specifies a CardBus bridge. If bit 7 of this register is set, the device has multiple functions; otherwise, it is a single function device.
* **Latency Timer:** Specifies the latency timer in units of PCI bus clocks.
* **Cache Line Size:** Specifies the system cache line size in 32-bit units. A device can limit the number of cache line sizes it can support, if an unsupported value is written to this field, the device will behave as if a value of 0 was written.
* **CardBus CIS Pointer:** Points to the Card Information Structure and is used by devices that share silicon between CardBus and PCI.
* **Interrupt Line:** Specifies which input of the system interrupt controllers the device’s interrupt pin is connected to and is implemented by any device that makes use of an interrupt pin. For the x86 architecture this register corresponds to the PIC IRQ numbers 0-15 (and not I/O APIC IRQ numbers) and a value of 0xFF defines no connection.
* **Interrupt Pin:** Specifies which interrupt pin the device uses. Where a value of 0x01 is INTA#, 0x02 is INTB#, 0x03 is INTC#, 0x04 is INTD#, and 0x00 means the device does not use an interrupt pin.
* **Max Latency:** A read-only register that specifies how often the device needs access to the PCI bus (in ¼ microsecond units).
* **Min Grant:** A read-only register that specifies the burst period length, in ¼ microsecond units, that the device needs (assuming a 33 MHz clock rate).
* **Capabilities Pointer:** Points to a linked list of new capabilities implemented by the device. Used if bit 4 of the status register (Capabilities List bit) is set to 1. The bottom two bits are reserved and should be masked before the Pointer is used to access the Configuration Space.

Header Type Register:-

|  |  |
| --- | --- |
| Bit 7 | Bit 6- 0 |
| MF | Header Type (0x00 = Endpoint, 0x01 = PCI Bridge, 0x02 = CardBus Bridge) |

### Base Address Register

Base address Registers (or BARs) can be used to hold memory addresses used by the device, or offsets for port addresses. Typically, memory address BARs need to be located in physical RAM while I/O space BARs can reside at any memory address (even beyond physical memory). To distinguish between them, you can check the value of the lowest bit. The following tables describe the two types of BARs:

|  |  |  |  |
| --- | --- | --- | --- |
| Memory Space BAR Layout | | | |
| **31 – 4** | **3** | **2 – 1** | **0** |
| 16-Byte Aligned Base Address | Prefetchable | Type | Always 0 |

|  |  |  |
| --- | --- | --- |
| I/O Space BAR Layout (no longer used) | | |
| **31 – 2** | **1** | **0** |
| 4-Byte Aligned Base Address | Reserved | Always 1 |

The Type field of the Memory Space BAR Layout specifies the size of the base register and where in memory it can be mapped. If it has a value of 0x00 then the base register is 32-bits wide and can be mapped anywhere in the 32-bit Memory Space. A value of 0x02 means the base register is 64-bits wide and can be mapped anywhere in the 64-bit Memory Space (A 64-bit base address register consumes 2 of the base address registers available). A value of 0x01 is reserved as of revision 3.0 of the PCI Local Bus Specification. In earlier versions it was used to support memory space below 1MB (16-bit wide base register that can be mapped anywhere in the 16-bit Memory Space).

When you want to retrieve the actual base address of a BAR, be sure to mask the lower bits. For 16-Bit Memory Space BARs, you calculate (BAR[x] & 0xFFF0). For 32-Bit Memory Space BARs, you calculate (BAR[x] & 0xFFFFFFF0). For 64-Bit Memory Space BARs, you calculate ((BAR[x] & 0xFFFFFFF0) + ((BAR[x+1] & 0xFFFFFFFF) << 32)) For I/O Space BARs, you calculate (BAR[x] & 0xFFFFFFFC).

To determine the amount of address space needed by a PCI device, you must save the original value of the BAR, write a value of all 1’s to the register, then read it back. The amount of memory can then be determined by masking the information bits, performing a bitwise NOT (‘~’ in C), and incrementing the value by 1. The original value of the BAR should then be restored. The BAR register is naturally aligned and as such you can only modify the bits that are set. For example, if a device utilizes 16 MB it will have BAR0 filled with 0xFF000000 (0x01000000 after decoding) and you can only modify the upper 8-bits.

### PCIe Bus Enumeration

PCIe is intended to be “plug-and-play”.

When a CPU boots up it examines “function#0” for each Bus and Device, since every device must have function#0 as a minimum. If all 1’s is returned the Device is not fitted (all 1’s for the Device ID and Vendor ID is not a legal value).

For the functions that are identified as being fitted writing all 1’s to the BAR and then reading back the value can be used to determine how much address space is needed (via a special read-back code). The BAR address can then be set to the desired value by the CPU.

## Altera Agilex F Avalon-MM DMA

This transfer method is appropriate for high bandwidth PCIe communications, since a block of data requiring multiple TLPs can be passed over the PCIe Interface via a single instruction.

### Agilex PCIe with Avalon-MM Multi Channel DMA Engine

#### Overview of the DMA Operation

The CXFT module performs a complex Fourier Transform of the De-dispersed time series observation. A transform is performed for each Dispersion Measure (DM) of an observation to form a frequency spectrum for the FDAS FPGA to analyse.

The spectrum for a 536.87 second observation consists of 2^22 frequency “bins” with each bin containing a real and Imaginary value each in IEE754 32-bit single precision format. Hence each bin is 8 bytes.

Hence the frequency spectrum for each DM is 2^22 x 8 = 2^25 = 32 MBytes.

Section 5.3 in Appendix A of this document provides a detailed overview of how the Direct Memory Accesses over the PCIe interface operate in the Intel Agilex FPGA.

The figure below shows how in practice this is performed:-

**FDAS FPGA**

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

**External DDR4 SDRAM Memory #0**

Address 0x2,0000,0000

**Host PC**

**CPU Core**

**PCIe Root**

**CPU Core**

**CPU Core**

**DMA Engine**

|  |  |
| --- | --- |
| **Queue Control and Status Registers (QCSR)**  **For Host PC-to-Device Queue 0** | |
| **Address**  **(offset from BAR0)** | **Field** |
| 0x80000 | Q\_CTRL |
| 0x80008 | Q\_START\_ADDR\_L |
| 0x8000C | Q\_START\_ADDR\_H |
| 0x80010 | Q\_SIZE |
| 0x80014 | Q\_TAIL\_POINTER |
| 0x80018 | Q\_HEAD\_POINTER |
| 0x8001C | Q\_COMPLETED\_POINTER |
| 0x80020 | Q\_CONSUMED\_HEAD\_ADDR\_L |
| 0x80024 | Q\_CONSUMED\_HEAD\_ADDR\_H |
| 0x80028 | Q\_BATCH\_DELAY |
| 0x80038 | Q\_DEBUG\_STATUS\_1 |
| 0x8003C | Q\_DEBUG\_STATUS\_2 |
| 0x80040 | Q\_DATA\_DRP\_ERR\_COUNTER |
| 0x80044 | Q\_PYLD\_CNT |
| 0x80048 | Q\_RESET |

**Access**

**Arbitration**

PCIe

**RAM**



**Huge Pages**



DM Spectrum

32MByte

#1

#2

#3

#n

Index pointed to by Q\_HEAD\_POINTER

Index pointed to by Q\_TAIL\_POINTER

**Descriptor Ring Linked List**

located at address

Q\_START\_ADDR\_L/H

|  |  |
| --- | --- |
| Descriptor Index | |
| 32 bytes = 256 bits | |
| Bits | Field |
| [63:0] | SRC\_ADDR |
| [127:74] | DEST\_ADDR |
| [147:128] | PAYLD\_CNT |
| [175:160] | DESC\_IDX |
| [176] | MISX\_EN |
| [177] | WB\_EN |
| [211:192] | RX\_PYLD\_CNT |
| [222] | SOF |
| [223] | EOF |
| [254] | DESC\_INVALID |
| [255] | LINK |

**PCIe End**

**Point**

**Access Arbitration**

Figure ‑: DMA of DM Spectrum to FDAS

The reason that DMA (Direct Memory Access) is that it frees up the CPU from the task of reading the DM Spectrum from Host PC RAM to the FPGA.

The software only needs to provide a few instructions to allow the DMA Engine in the FPGA to carry out the data transfer.

To initially set up the DMA:-

* The software chooses the size and where to locate the Descriptor Ring Linked List in RAM and configures the Q\_START\_ADDR\_L/H and Q\_SIZE in the FPGA’s DMA Engine QCSR register space. This ensures the FPGA’s DMA Engine knows where the Descriptor Ring Linked List is located.
* The software configures the PAYLD\_CNT in each descriptor and the Q\_PAYLD\_CNT in the FPGA’s DMA Engine QCSR. It is advised to set this to the maximum size of 1Mbyte.
* The software configures the DESC\_IDX value in each descriptor. It is advised to set this to the actual index value for each descriptor, with the first descriptor (index = 1) residing at the Q\_START\_ADDR\_L/H.
* The software can configure MSIX\_EN in each descriptor as desired to cause an MSI-X interrupt to be sent from the FPGA to the software when the DM Spectrum transfer for the descriptor is complete
* The software can configure WB\_EN in each descriptor and Q\_CONSUMED\_HEAD\_ADDR\_L/H in the FPGA’s DMA Engine as desired to cause a writing of the completed descriptor information from the FPGA to location Q\_CONSUMED\_HEAD\_ADDR\_L/H in the Host PCs memory. The software can then interrogate this location in its memory to understand the status instead of reading the FPGA’s QCSR registers.

To perform a transfer of a DM Spectrum from the Huge Pages region in the Host PC RAM to the DDR SDRAM connected to the FDAS FPGA.

1. Assuming PAYLD\_CNT is set to 1Mbyte and the DM Spectrum is 32Mbytes the software configures the next 32 descriptors of the Descriptor Ring Linked List with the appropriate SRC\_ADDR of each 1Mbyte block of the DM Spectrum in Huge pages and the DEST\_ADDR of where the 1Mbyte block is to be stored in the DDR SDRAM. (Note that for the very first DMA after power up the Q\_TAIL\_POINTER and Q\_HEAD\_POINTER in the FPGA’s DMA Engine will be set to an index of #1).
2. The software then writes the Descriptor Ring Linked List index for the final 1Mbyte block to the Q\_TAIL\_POINTER in the FPGA’s DMA Engine. On detecting that the Q\_HEAD\_POINTER and Q\_TAIL\_POINTER no longer match the FPGA’s DMA reads descriptors from the Descriptor Ring Linked List in Host PC RAM and carries out the instruction to read data from SRC\_ADDR in the Host PCs Huge Pages and write it to DEST\_ADDR in the DDR SDRAM. processes the descriptors, advancing its Q\_HEAD\_POINTER as each descriptor is processed. The transfer is complete when Q\_TAIL\_POINTER and Q\_HEAD\_POINTER match.
3. The software can use one of five methods to know when the DM spectrum transfer is complete:-

* The Host PC software can poll the Q\_COMPLETED\_POINTER in the QCSR of the FPGA. When the Q\_COMPLETED\_POINTER, which is updated by the FPGA, contains the DESC\_IDX value from the last completed descriptor.
* MSI-X interrupts can be enabled in the descriptors and an interrupt will occur when the DMA transfer for the descriptor has completed. The Host PC software could either count the number of interrupts to determine when the expected number of DMA transfers have occurred, or on each interrupt inspect the Q\_COMPLETED\_POINTER which is updated by the FPGA in the QCSR.
* Write back can be enabled in the descriptors and when a descriptor has completed an indication will be sent to a Host PC address configured in the QCSR via the Q\_CONSUMED\_HEAD\_ADDR\_L/H register. Software can examine the Host PC address that contains the write back information and this can be used to determine the number of DMA transfers that have occurred.
* MSI-X interrupts can be enabled in the QCSR, and an interrupt will occur when all the DMA transfers have completed.
* Write back can be enabled in the QCSR, and when all the DMA transfers have completed an indication will be sent to a Host PC address configured in the QCSR via the Q\_CONSUMED\_HEAD\_ADDR\_L/H register.

It should be noted the software driver C code from Intel actually takes care of most of the low level operations described above.

#### DMA Descriptors

The Multi-Channel DMA Engine resides within the Intel Agilex PCIe Hard IP Macro. The PCIe Macro can support up to 2k channels, however the actual number is specified in Intel Quartus Platform Designer for the PCIe. For FDAS 4 channels have been specified. In reality, FDAS only needs one channel with a “Device to Host” (D2H) queue and a “Host to Device” (H2D) queue. However each channel can only support two user MSI-X interrupts and hence the extra channels are available in FDAS for this. The queues within the PCIe DMA Engine use descriptors which are effectively the commands for the DMA transfers.

For each DMA channel the Host PC needs to maintain a linked list of “Descriptor Pages” for H2D transfers and a separate linked list of “Descriptor Pages” for D2H transfers.

Each Descriptor Page has a capacity of 4 Kibi Bytes and contains up to 128 descriptors which are 32 bytes in length.

Each descriptor can instruct a DMA transfer over the PCIe interface of up to 1Mbyte.

The last descriptor in a 4KB Descriptor Page must be a “link” descriptor which points to the Host PC address start of the next descriptor. The “link” descriptor is identified by having the Link bit in the 32 byte descriptor set to 1.

The last entry in the linked list must be a “link” descriptor pointing to the base address programmed in the Queue Control and Status Registers (QCSR), which is the location of the first descriptor in Host PC memory. This allows a circular buffer to be created. The QCSR is described in the following section 15.3.1.3.1

The figure below shows the concept of the Decscriptor Ring Linked List in Host PC memory followed by a table which describes the function of the 32 bytes (= 256 bits) in a Descriptor.

|  |
| --- |
| Desc #1\* Link = 0 |
| :  : |
| Desc #128 Link = 1 |

|  |
| --- |
| Desc #129 Link = 0 |
| :  : |
| Desc #256 Link = 1 |

|  |
| --- |
| Desc #257 Link = 0 |
| :  : |
| Desc # 384 Link = 1 |

|  |
| --- |
| Desc #n-127 Link = 0 |
| :  : |
| Desc #n Link = 1 |

.

.

.

4kByte Descriptor Page

4kByte Descriptor Page

4kByte Descriptor Page

4kByte Descriptor Page

**\*= Descriptor Index always starts at #1**

**Q\_START\_ADDR\_L/H (location in Host PC memory of the first descriptor) from QCSR registers**

Figure 15‑2 Physical view of Descriptor Ring Linked List in Host PC memory

Each descriptor is 32 bytes and contains the following information:-

| Name | Width/ Bits | Description |
| --- | --- | --- |
| SRC\_ADDR  Descriptor Bits [63:0] | 64 | If the LINK bit is ‘0’ then this field contains the source address.  Starting address of allocated transmit buffer read by DMA.  If the queue is H2D then this field contains the address in host memory.  If the queue is D2H then this is the AVMM address in device memory.  If the LINK bit is set, then this contains the address of the next 4kByte page in host memory containing the descriptors. |
| DEST\_ADDR  Descriptor Bits [127:64] | 64 | Provided the LINK bit is ‘0’ this field means:-  Starting address of allocated receive buffer read by DMA.  If the queue is D2H then this field contains the address in host memory.  If the queue is H2D then this is the AVMM address in device memory. |
| PAYLD\_CNT  Descriptor Bits[147:128] | 20 | Provided the LINK Bit is ‘0’ this field means the DMA payload size in bytes, Max 1 Mebi Byte.  “00000000000000000001” binary indicating 1 Byte.  “00000000000000000010” binary indicating 2 Bytes.  :  “00000000000000000000” binary indicating 1 Mebi Byte. |
| RSRVD  Descriptor Bits [159:148] | 12 | Reserved |
| DESC\_IDX  Descriptor Bits[175:160] | 16 | Unique identifier for each descriptor, assigned by the software driver. This value is written to the Q\_COMPLETED\_POINTER DMA engine register when a descriptor transfer is complete.  First descriptor DESC\_IDX is ‘1’. |
| MISX\_EN  Descriptor Bit [176] | 1 | Enable MSI-X Interrupt per descriptor |
| WB\_EN  Descriptor Bit [177] | 1 | Enable Write Back per descriptor |
| RX\_PYLD\_CNT  Descriptor Bits[211:192] | 20 | Received actual payload for D2H data transfer |
| RSRVD  Descriptor Bits [221:212] | 10 | Reserved |
| SOF  Descriptor Bit [222] | 1 | SOF Indicator for Avalon Streaming |
| EOF  Descriptor Bit [223 | 1 | EOF Indicator for Avalon Streaming |
| RSRVD  Descriptor Bits [253:224] | 30 | Reserved |
| DESC\_INVALID  Descriptor Bit [254] | 1 | Indicates if the current descriptor content is valid or stale |
| LINK  Descriptor Bit [255] |  | LINK = 0  Descriptor contains the Source Address, Destination Address and Length.  LINK = 1  Descriptor contains the address of the next 4Kibi Byte page in Host memory containing the descriptors |

Table ‑ : Descriptor Definition

#### FPGA PCIe Multi Channel DMA Controller Registers

Registers on BAR0 allow the Host PC to configure the FPGA’s DMA engine within the PCIe macro with the descriptors and also configure the MSI-X Interrupts.

The Multi-Channel DMA Control and Status Registers (CSR) are located at the following address regions of BAR0

|  |  |  |  |
| --- | --- | --- | --- |
| Address Space Name | BAR0 Based  22 bit address (Byte based) | Size | Description |
| QCSR (D2H, H2D)  Queue Control and Status Registers | 0x00\_0000 to 0x0F\_FFFF | 1 Mebi Byte | Individual queue control registers. Up to 2048 D2H and 2048 H2D queues. |
| MSI-X (Table and Pending Bit Array (PBA) | 0x10\_0000 to 0x1F\_FFFF | 1 Mebi Byte | MSI-X Table and PBA space |
| GSCR  General/global Control and Status Registers | 0x20\_0000 to 0x2F\_FFFF | 1 Mebi Byte | General DMA control and status registers |
| Reserved | 0x30\_0000 to 0x3F\_FFFF | 1 Mebi Byte | Reserved |

Table 15‑3: PCIe Multi Channel DMA Address Space

##### Queue Control and Status Registers (QCSR)

The Queue Control and Status Register (QCSR) space is in the BAR0 based address range 0x00\_0000 to 0x0F\_FFFF (1 Mebi Byte space) is arranged as follows:-

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Address Space Name | Size | DMA channel | DMA Channel Size | Description |
| QCSR (D2H) | 512 Kibi Byte | DMA Channel 0 | 256 Bytes | QCSR for DMA Channel 0 |
| DMA Channel 1 | 256 Bytes | QCSR for DMA Channel 1 |
| : | : | : |
| DMA Channel N | 256 Bytes | QCSR for DMA Channel N |
| QCSR (H2D) | 512 Kibi Byte | DMA Channel 0 | 256 Bytes | QCSR for DMA Channel 0 |
| DMA Channel 1 | 256 Bytes | QCSR for DMA Channel 1 |
| : | : | : |
| DMA Channel N | 256 Bytes | QCSR for DMA Channel N |

Table 15‑4: QCSR Address Space

The QCSR space contains queue control and status information. The register space of 1 Mebi Byte can support 2048 H2D and 2048 D2H queues, where each queue is allocated 256 Bytes of register space.

The QCSR address space address bits are assigned as follows:-

Address Bits [7:0] : Registers for one DMA channel queue

Address Bits [18:8]: Queue number

Address Bit [19]: 0 = D2H, 1 = H2D

The QCSR for a DMA channel has the following assignments:-

| Register Name | BAR0 based Address Offset (Byte) | Access Type | Description |
| --- | --- | --- | --- |
| Q\_CTRL | 0x00 | R/W | Control Register  Bit [9]: If set upon completion generate an MSI-X interrupt  Bit [8] If set upon completion do a write back  Bit[0]: Enable. Once enabled DMA descriptors are fetched and executed. |
| RESERVED | 0x04 |  | Reserved |
| Q\_START\_ADDR\_L | 0x08 | R/W | Lower 32-bit of Queue base address in Host PC memory. This is the beginning of the Linked List of 4 Kibi Byte pages containing the descriptors |
| Q\_START\_ADDR\_H | 0x0C | R/W | Upper 32-bit of Queue base address in Host PC memory. This is the beginning of the Linked List of 4 Kibi Byte pages containing the descriptors |
| Q\_SIZE | 0x10 | R/W | Number of Max entries in a queue. Powers of 2 only. The unit is number of descriptors. A value of 1 means a queue size of 2^1 = 2 descriptors. A value of 16 (0x10) means a queue size of 2^16 = 64K descriptors. |
| Q\_TAIL\_POINTER | 0x14 | R/W | Current Pointer to the last valid descriptor queue entry in the Host PC memory. |
| Q\_HEAD\_POINTER | 0x18 | RO | Current Pointer to the last descriptor that was fetched. Updated by the Descriptor Fetch Engine in the PCIe DMA controller. |
| Q\_COMPLETED\_POINTER | 0x1C | RO | Last completed pointer after the DMA is done. Software can poll this for status if Writeback is disabled. |
| Q\_CONSUMED\_HEAD\_ADDR\_L | 0x20 | R/W | Lower 32-bit of the Host PC address where writeback targets after DMA of a descriptor is completed when writeback is enabled. |
| Q\_CONSUMED\_HEAD\_ADDR\_H | 0x24 | R/W | Upper 32-bit of the Host PC address where writeback targets after DMA of a descriptor is completed when writeback is enabled. |
| Q\_BATCH\_DELAY | 0x28 | R/W | Delay the descriptor fetch until the time elapsed from a prior fetch exceeds the delay value on this register to maximise fetching efficiency. Each unit is 2ns. |
| RESERVED | 0x2C |  | Reserved |
| RESERVED | 0x30 |  | Reserved |
| RESERVED | 0x34 |  | Reserved |
| Q\_DEBUG\_STATUS\_1 | 0x38 | RO | Reserved |
| Q\_DEBUG\_STATUS\_2 | 0x3C | RO | Reserved |
| Q\_DATA\_DRP\_ERR\_COUNTER | 0x40 | R/W | Data Drop Error Counter |
| Q\_PYLD\_CNT | 0x44 | R/W | 20-bit payload count. DMA payload size in bytes and must be 64 byte aligned. Max 1 Mebi Byte with a value of 0x0\_0000 indicating 1 Mebi Byte. The value set in this field must be the same as used by the Host PC software to populate the PYLD\_CNT field of the descriptors for the respective channel. Only applicable for D2H AVST 1 port mode. Unused in all other modes. |
| Q\_RESET | 0x48 | R/W | Request Reset for the queue by writing a 1 to this register and poll for a value of 0 when the reset has been completed by the FPGA. |

Table 15‑5: QCSR for a DMA Channel (D2H or H2D)

##### MSI-X Memory Space

The MSI-X Table and Pending Bit Array (PBA) space is in the BAR0 based address range 0x10\_0000 to 0x1F\_FFFF (1 Mebi Byte space) is arranged as follows:-

Allocated memory space can support up to 2048 MSI-X interrupts for a function (Virtual or Physical machine). Actual amount of memory space depends on the Multi-Channel DMA IP configuration on the Quartus Prime Platform Designer for the PCIe.

**MSI-X Table**

Each Entry (Vector) is 16 bytes (4 DWORDs) and is divided into Table Message Address, Data and Mask (Vector Control) fields. To support 2048 interrupts the MSI-X Table requires 32 Bibi Bytes of space per function, but is mapped to 512 Kibi Bytes of space.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| DWORD 3 | DWORD 2 | DWORD 1 | DWORD 0 | Entry No. | BAR0 Based  Byte Addr  (Base = 0x10\_0000 of BAR0) |
| Vector Control | Message Data | Table Message Upper Address | Table Message Address | Entry 0 | Base |
| Vector Control | Message Data | Table Message Upper Address | Table Message Address | Entry 1 | Base + 1\*16 |
| Vector Control | Message Data | Table Message Upper Address | Table Message Address | Entry 1 | Base + 2\*16 |
| : | : | : | : | : | : |
| Vector Control | Message Data | Table Message Upper Address | Table Message Address | Entry (N-1) | Base + (N-1)\*16 |

Table 15‑6: MSI-X Table

**MSI-X PBA (Pending Bit Array)**

MSI-X PBA memory space is mapped to a 512 Kibi Byte region. Actual amount of memory space depends on the Multi-Channel DMA IP configuration on the Quartus Prime Platform Designer for the PCIe. The Pending Bit Array contains the pending bits, one per MSI-X table entry in an array of QWORDs (64 bits).

|  |  |  |
| --- | --- | --- |
| PBA | QWORD | BAR0 Based Byte Addr  (Base = 0x18\_0000 of BAR0) |
| Pending Bits 0 to 63 | 0 | Base |
| Pending Bits 64 to 127 | 1 | Base + 1\*8 |
| : | : | : |
| Pending Bits ((N-1)div64)\*64 to N-1 | (N-1) div 64 | Base + ((N-1) div 64)\*8 |

Table 15‑7: MSI-X PBA Table

Each DMA channel is allocated four MSI-X vectors, with the bottom two Table Message Address bits defining the usage:-

* 2’b00: H2D DMA vector
* 2’b01: H2D Event Interrupt (user Interrupt)
* 2’b10: D2H DMA vector
* 2’b11: D2H Event Interrupt (user Interrupt)

##### General/Global Control and Status Register (GCSR)

The General/Global space is in the BAR0 based address range 0x20\_0000 to 0x2F\_FFFF (1 Mebi Byte space) is arranged as follows:-

|  |  |  |  |
| --- | --- | --- | --- |
| Register Name | BAR0 Based  Address Offset (Byte) | Access Type | Description |
| CTRL | 0x00 | R/W | Reserved |
| RESERVED | 0x04 |  | Reserved |
| WB\_INT\_DELAY | 0x08 | R/W | Delay the writeback and/or the MSI-X interrupt until the time elapsed from a prior Writeback/interrupt exceeds the delay value in this register. |
| RESERVED | 0x0c to 0x6F |  | Reserved |
| VER\_NUM | 0x70 | RO | Multi-Channel DMA IP for PCIe version number. |
| SW\_RESET | 0x120 | R/W | Write this register to issue Multi Channel DMA IP reset without disturbing the PCIe link. This resets all queues and erase all the context. Can only be issued by a physical machine. |

Table 15‑8: General/Global Control and Status Register

# APPENDIX B: Configuration Via PCIe (CvP)

CvP must be enabled in the PCIe Hard IP Macro when it is created in the Intel Quartus Prime Platform Designer environment.

Note the CvP is only available if the PCIe link is 16 lanes Gen 4, however the Host PC can down-train to a lower PCIe genetation if it does not support Gen 4.

The CvP configuration scheme creates separate images for the Periphery (.jic file) (i.e. PCIe Interface operation) and the Core logic (.rbf file).

Typically the Periphery image is stored in Flash memory on the board fitted with the Agilex FPGA and at power up is loaded into the FPGA (using the Active Serial x 4 “fast mode” configuration scheme) to configure and make operational the PCIe interface to the Host PC.

The Core Logic image is stored on the Host PC and can be downloaded vla the PCIe to configure the core circuits. This method allows for different core functions or updates/bug fixes to be downloaded to the FPGA with the user being remote from the Host PC.

A .jic file is a “JTAG Indirect Configuration file, so named since it is download to the Flash memory by connecting the Host PC via USB to an Intel “Byteblaster” chip which connects to the JTAG interface of the FPGA which itself has the connection to the Flash memory. The JTAG (Joint Test Action Group) is an interface that many larger chips have to allow it to be tested in the factory when it on the PCB, for example to perform a “boundary scan” to prove all the pins are correctly soldered or to feed test signals into the chip to prove it is not faulty. Intel also use this interface to either directly program the FPGA with a .sof image or store a .jic image in the associated flash memory by using the FPGA at a bridge to the flash.

A .rbf file is a “Raw Binary File”.

A “CvP” driver for Linux is needed to perform the .rbf file download to the FPGA.

There are two CvP modes:-

* CvP Initialisation mode
  + The PCIe is configured using the Periphery image (.jic file), and when the PCIe interface is operational the software on the Host PC downloads the Core Logic image (.rbf) file over the PCIe to complete the FPGA configuration
* CvP Update mode
  + With an already fully configured FPGA, in “user mode” a new Core Logic image (.rbf) file can be downloaded by the Host PC via the PCIe interface to effectively replace the existing image.

## Creating the CvP Image using Intel Quartus Prime

CvP is only supported on Intel Quartus Prime ver 20.4 onwards.

With CvP the .sof bitstream file generated by the build in Intel Quartus Prime has to be processed to split it into a Periphery image (.jic) for the PCIe Interface and a Core Logic image (.rbf).

The steps are as follows:-

1. To ensure the .sof file is CvP capable:-
   1. Ensure in Intel Quartus Prime Platform Designer for the PCIe Hard IP Macro that CVP is enabled.
   2. Ensure that the .qsf file contains the following assignments:-

set\_global\_assignment -name STRATIXV\_CONFIGURATION\_SCHEME “ACTIVE SERIAL X4”

set\_global\_assignment -name CONFIGURATION\_VCCIO\_LEVEL 1.8V

set\_global\_assignment -name CVP\_MODE “CORE INITIALIZATION AND UPDATE”

set\_global\_assignment -name ENABLE\_CVP\_CONFDONE ON

set\_global\_assignment -name DEVICE\_INITIALIZATION\_CLOCK OSC\_CLK\_1\_125MHZ

set\_global\_assignment -name PWRMGT\_SLAVE\_DEVICE0\_ADDRESS 47

set\_global\_assignment -name PWRMGT\_SLAVE\_DEVICE1\_ADDRESS 00

set\_global\_assignment -name PWRMGT\_SLAVE\_DEVICE2\_ADDRESS 00

set\_global\_assignment -name ACTIVE\_SERIAL\_CLOCK AS\_FREQ\_100MHZ

set\_global\_assignment -name USE\_PWRMGT\_SCL SDM\_IO14

set\_global\_assignment -name USE\_PWRMGT\_SDA SDM\_IO11

set\_global\_assignment -name USE\_CONF\_DONE SDM\_IO16

set\_global\_assignment -name USE\_CVP\_CONFDONE SDM\_IO10

set\_global\_assignment -name USE\_INIT\_DONE SDM\_IO0

set\_global\_assignment -name PWRMGT\_SLAVE\_DEVICE\_TYPE OTHER

set\_global\_assignment -name PWRMGT\_VOLTAGE\_OUTPUT\_FORMAT “LINEAR FORMAT”

set\_global\_assignment -name PWRMGT\_LINEAR\_FORMAT\_N “-13”

set\_global\_assignment -name PWRMGT\_PAGE\_COMMAND\_ENABLE ON

set\_global\_assignment -name AUTO\_RESTART\_CONFIGURATION OFF

set\_global\_assignment -name MINIMUM\_SEU\_INTERVAL 1075681689

set\_global\_assignment -name CVP\_CONFDONE\_OPEN\_DRAIN OFF

1. To convert the .sof file after successful compilation:-
   1. In Intel Quartus Prime under the **File** menu, select **Programming File Generator**. A new window appears.
   2. In the **Device family,** select Agliex
   3. For the **Configuration mode**, select **Active serial x 4**
   4. Under **Output files** tab, specify the following parameters:-
      1. Specify the **Output directory** and **Name** for the output file (this directory must already exist)
      2. Select **Raw Binary File for CvP Core Configuration (.rbf)**
      3. Select **JTAG Indirect Configuration File for Periphery Configuation (.jic)** if you want to use the Active serial configuration mode.
      4. Select **Memory Map File (.map)**

Graphical user interface, text, application, email

Description automatically generated

* 1. Under **Input files** tab, click add bitstream. Navigate your file system and select the .sof file

Graphical user interface, text, application, email

Description automatically generated

* 1. Under the **Configuration device** tab:
     1. Click **Add device**
     2. Under the **Configuration Device** tab, select your configuration device (MT25QU02G for Agilex Development Board) and click **OK**

Graphical user interface, application

Description automatically generated

* + 1. Click to select the configuration device in the list and click **Add partition**

Graphical user interface, text, application

Description automatically generated

* + 1. In the **Add Partition** window, select the file in the **Input file** box, select **<auto>**in the **Address Mode** box, and then click **OK**

Graphical user interface, application

Description automatically generated

* + 1. Click **Select**
    2. In the **Select Devices** window, click Agilex in the device family list, select your flash loader device in the **Device name** list, and then click **OK**

Graphical user interface, application

Description automatically generated

* + 1. Click **Generate**

The following files should now be in the desired directory for the CvP files:-

* FDAS\_CvP.core.rbf
* FDAS\_CvP.periph.jic
* FDAS\_CvP\_jic.map

## Downloading the FDAS Design Using CvP

### Downloading the Periphery Image

The FDAS\_CvP.periph.jic periphery image is downloaded to the flash on the Intel Agilex development board using the Intel Quartus Programmer in the normal way.

Following a re-boot of the card the presence of the PCIe can be checked using the “lspci” command in Linux. This should return the information from the Type 0” “Configuration Space” which should match the PCIe Hard IP Macro configuration parameters that were selected in the Intel Quartus Platform Designer:-

$ sudo lspci -d 1172:000 -v

01:00.0 Unassigned class [ff00]: Altera Corporation Device 0000 (rev 01)

    Flags: bus master, fast devsel, latency 0

    Memory at 6002000000 (64-bit, prefetchable) [size=4M]

    Memory at 6000000000 (64-bit, prefetchable) [size=32M]

    Capabilities: [40] Power Management version 3

    Capabilities: [70] Express Endpoint, MSI 00

    Capabilities: [b0] MSI-X: Enable- Count=16 Masked-

    Capabilities: [100] Advanced Error Reporting

    Capabilities: [148] Virtual Channel

    Capabilities: [178] Alternative Routing-ID Interpretation (ARI)

    Capabilities: [188] Secondary PCI Express

    Capabilities: [1b8] Physical Layer 16.0 GT/s <?>

    Capabilities: [1e8] Lane Margining at the Receiver <?>

    Capabilities: [470] Data Link Feature <?>

    Capabilities: [d00] Vendor Specific Information: ID=1172 Rev=0 Len=05c <?>

### Downloading the Core Image

The FDAS\_CvP.core.rbf core image requires a software driver on the Host PC to support the downloading over the PCIe interface to the FPGA.

From the Intel CvP document "Intel Agilex® 7 Device Configuration via Protocol (CvP) Implementation User Guide" :-

*5.1. CvP Driver Support  
You can develop your own custom CvP driver for Linux using the sample Linux driver source code provided by Intel® .*

*Note: The Linux driver provided by Intel is not a production driver. You must adapt this driver to your design's strategy.  
Note: CvP Downstream Driver is no longer supported from Quartus version 22.2. Intel recommends you to use Open Source Linux CvP Upstream Driver.  
Note: Open Source Linux CvP Upstream Driver: After you access the link, you have to select the kernel version to download. Refer to Upstream Open Source CvP Driver for steps on how to download and install the CvP upstream driver.*

The method described below uses the Intel CvP Downstream Driver :-

1. **Setting Up the Correct MSEL Switch State**    Select Active Serial x4 (Fast mode) for CvP operation.  
    Table 14.  MSEL Pin Settings for Active Serial x4 (Fast mode) Scheme of Intel Agilex® 7 Devices

    Configuration Scheme   MSEL[2:0]  
    AS (Fast mode - for CvP)6    001  
      
2. **Downstream driver:**    altera\_cvp\_nd – Linux driver for configuring Altera FPGAs via CvP  
    Specifically for the Stratix10 and later parts.  
    Description:  
    ------------  
    The altera\_cvp driver allows a Linux user to configure the core of an Altera  
    CvP-capable FPGA through the use of standard Unix utilities or custom-built  
    user-space applications.  
      
    NOTE: The altera\_cvp\_load script loads the kernel module and sets correct permissions on the device.  
          altera\_cvp\_unload removes the kernel modules and the device in the /dev/ directory.  
    

**3. Installation:**  
    -------------  
    In the directory where the release files are located, run the following commands:  
      
    1) make  
    2) make install (must be run as root)  
      
    VID/DID Notes:

   -------------  
    Default Vendor ID: 0x1172 (Altera vendor ID)  
    Default Device ID: 0x0000 (Altera PCIe high-performance reference design device ID)  
        If your VID/DID differ, please modify the code near the top of altera\_cvp.c accordingly  
      
    Uninstall:  
    ----------  
    To uninstall the driver, run the following commands (both require root privileges):  
      
    1) /sbin/rmmod altera\_cvp  
    2) rm -f /dev/altera\_cvp  
      
      
    Usage:  
    ------  
    The cvp driver is accessible through the device file located at /dev/altera\_cvp.  
      
    Reading from this file will return a binary representation of the VSEC registers  
    for CvP. Writing to this file will cause any data written between file open and  
    close to be sent to the FPGA as CvP core configuration data.  
      
    This allows the use of standard Unix utilities like dd to perform CvP, for example:

  The Unix dd utility is preferable to cp because it allows the user to specify the  
    block size using the bs option.  
      
    dd if=<filename>.core.rbf of=/dev/altera\_cvp bs=4K  
      
    The command above will cause the driver to attempt a CvP configuration of the  
    FPGA using the specified core RBF file.

# APPENDIX C: Agilex Development Board Switch Positions

The required Intel Agilex Developmemt Board Switch Positions are shown below:-

**SW1**

A close-up of a green circuit board

Description automatically generated

**SW1 Pin Connections**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SW1 Pin** | **Board Label** | **Function** | **Default Settings** | **FDAS**  **(as shown in above picture)** |
| SW1.1 | MSEL0 | Mode select 0 for configuration | OFF | OFF |
| SW1.2 | MSEL1 | Mode select 1 for configuration | ON | ON |
| SW1.3 | MSEL2 | Mode select 2 for configuration | ON | ON |

**SW6, SW4, SW3 and SW2**

A close-up of a circuit board

Description automatically generated

**SW2 Pin Connections**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SW2 Pin** | **Board Label** | **Function** | **Default Settings** | **FDAS**  **(as shown in above picture)** |
| SW2.1 | USB MAX JTAG SEL | ON = UBII MAX10 JTAG select External JTAG HEADER  OFF = UBII MAX10 JTAG select USB PHY. | OFF | OFF |
| SW2.2 | SI5341 Enable | ON = Disable SI5341’s outputs  OFF = Enable SI5341’s outputs | OFF | OFF |
| SW2.3 | SI52202 Power Down | ON = Power down SI52202  OFF = Power up SI52202 | OFF | OFF |
| SW2.4 | UART Enable | ON = Disable UART  OFF = Enable UART | OFF | OFF |

**SW3 Pin Connections**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SW3 Pin** | **Board Label** | **Function** | **Default Settings** | **FDAS**  **(as shown in above picture)** |
| SW3.1 | FPGA I2C Enable | ON = Isolate FPGA from main I2C chain.  OFF =connect FPGA to main I2C chain. | OFF | OFF |
| SW3.2 | HPS I2C Enable | ON = Isolate HPS from main I2C chain.  OFF = connect HPS to main I2C chain. | OFF | OFF |
| SW3.3 | Main PMBUS Enable | ON = Isolate power module of VCC\_core from main I2Cchain.  OFF= connect power module of VCC\_core to main I2C chain. | OFF | OFF |
| SW3.4 | FPGAPMBUS Enable | ON = Isolate power module of VCC\_core from SDMPMBUS.  OFF= connect power module of VCC\_core to SDM PMBUS. | OFF | OFF |

**SW4 Pin Connections**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SW4 Pin** | **Board Label** | **Function** | **Default Settings** | **FDAS**  **(as shown in above picture)** |
| SW4.1 | JTAG Input Source | ON = select PCIe edge as JTAG master when external JTAG is absent.  OFF = select On-Board Intel FPGA Download Cable as JTAG master when external JTAG is absent. | OFF | OFF |
| SW4.2 | Power Max10 Bypass | ON = bypass Power Max10 in the JTAG chain. | ON | ON |
| SW4.3 | Mictor Bypass | ON = bypass HPS in the JTAG chain.  OFF = enable HPS in the JTAG chain. | ON | ON |
| SW4.4 | FPGABypass | ON = bypass FPGA in the JTAG chain.  OFF = enable FPGA in the JTAG chain. | OFF | OFF |

**SW6 Pin Connections**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SW6 Pin** | **Board Label** | **Function** | **Default Settings** | **FDAS**  **(as shown in above picture)** |
| SW6.1 | PCIe EP Present x16 | ON = x16 select  OFF = x16 deselect | ON | ON |
| SW6.2 | PCIe EP Present x8 | ON = x8 select  OFF = x8 deselect | OFF | OFF |
| SW6.3 | PCIe EP Present x4 | ON = x4 select  OFF = x4 deselect | OFF | OFF |
| SW6.4 | PCIe EP Present x1 | ON = x1 select  OFF = x1 deselect | OFF | OFF |

**SW5**

A close up of a circuit board

Description automatically generated

Note, for this Switch, if the 12V auxillary power is provided via connector J16 by the PC and the card is in the PCs PCIe slot which also provides power then it does not matter which state this switch is in. However for FDAS when fitted in the PC it is advised to set to OFF as shown in the picture above.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Intended Use** | **12V Auxillary present on connector J16?** | **12V PCIe present on the PCIe connector J7?** | **SW5 Position** | **Result** |
| On the Bench | Yes | No | ON (to the right in above picture) | Card Powered up via 12V Auxillary connector J16 |
| On the Bench | Yes | No | OFF (to the left in above picture – as shown) | Card Off |
| In PCs PCIe Slot | Yes | Yes | Don’t Care | Card Powered up via 12V Auxillary connector J16 and PCIe connector J7 |
| In PCs PCIe Slot | No | Yes | ON (to the right in above picture) | Card Off |
| In PCs PCIe Slot | No | Yes | OFF (to the left in above picture – as shown) | Card Powered up via PCIe connector J7 |

SW7

A close up of a circuit board

Description automatically generated

In the state shown in the above picture the 100MHz PCIe reference clock is taken from the PC via the PCIe connector J7 (SW7 set to OFF). This is the state required for operation when the card is fitted in the PCs PCIe slot.

**SW7 Pin Connections**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **SW7 Pin** | **Board Label** | **Function** | **Default Settings** | **FDAS**  **(as shown in above picture)** |
| SW7.1 | SEL\_A\_B | ON = select SI52202 as clock source for PCIe Gen4  OFF = select PCIe edge as clock source for PCIe Gen4 | ON | OFF |

# APPENDIX D: Accelerating Pulsar Equations

## Accelerating Pulsar Equations

The frequency drift of an accelerating pulsar is given by the equation:-

Delta F = A\*T\*F/c

The bin drift of an accelerating pulsar is given by the equation:-

z = A\*F\*(T2)/c

Where F = Frequency in Hz

A = Acceleration in m/s/s

T = Observation Period in seconds

c = Speed of Light in m/s

## Derivation of Equations

Suppose a distant pulsar is initially stationary at time t = 0, and then accelerates uniformly towards an observer with acceleration “A”.

At time t = 0 the period of the pulsar seen by the distant observer is “P0” which is the “local” period. The frequency of the pulsar seen by the distant observer is “F0” (= 1/P0).

At the end of an observation time t = “T” the velocity of the pulsar in the direction towards the observer is:-

Velocity = acceleration x time

Velocity = A\* T …………………………………………………..eq 1

At the end of the observation the distance the pulsar travels towards the observer between successive pulses is:-

Distance = speed \* time

Distance = A\*T\*P0 …………………………………………… ..eq 2

Since the pulsar has moved this distance towards the observer in between successive pulses the second pulse has a shorter path to the observer and the observer sees this as a reduction in the period. Hence the period “PT” that the observer sees as time t = T is:-

PT = (local pulsar period) – (reduction in flight time to observer due to velocity)

PT = P0 - [A\*T\*P0/c] …………………………………………………..eq 3

PT = P0(1 - [A\*T/c])…………………………………………………….eq 4

Where c is the speed of light.

Using equation 3 the change in the period that the observer sees from t = 0 to t = T is:-

Delta P = (period at t = T) – (period at t = 0)

Delta P = P0 - [A\*T\*P0/c] – P0 ………………………………………eq 5

Delta P = - A\*T\*P0/c …………………………………………………..eq 6

(note that if the pulsar had been accelerating directly away from the observer the Delta P would have been positive).

We can express equation 4 in terms of pulsar frequency since F = 1/P

1/FT = (1/F0) \* (1- [A\*T/c]) ………………………………………eq 7

Re-arranging:-

FT = F0 / (1- [A\*T/c]) ………………………………………………eq 8

At this point there is an approximation we can make if the speed of the pulsar is very much less than the speed of light, that makes the Delta F equation simpler.

This approximation is based on the fact that for very small x we can say 1/(1-x) ≈ (1 + x), and also for the opposite sign case 1/(1+x) ≈ (1 – x),

Hence if A\*T << c then we can approximate equation 8 to:

FT ≈ F0 \* (1+ [A\*T/c]) ………………………………………………eq 9

FT ≈ F0 + [A\*T\*F0/c]) ………………………………………………eq 10

Using equation 10 the change in the frequency that the observer sees from t = 0 to t = T is:-

Delta F = (frequency at t = T) – (frequency at t = 0)

Delta F = F0 + [A\*T\*F0/c] – F0 ……………………………………..eq 11

Delta F = A\*T\*F0/c …………………………………………………eq 12

(note that if the pulsar had been accelerating directly away from the observer the Delta F would have been negative).

The change in the observed frequency Delta F from t = 0 to t = T, is seen as a “drift” of Delta F in magnitude of the fundamental tone of the pulsar in the spectrum.

Assuming a sampling technique is used by the observer to detect the pulsar over the observation time T, a Fourier Transform from the time domain to the frequency domain (i.e. the spectrum) will result in bin size range of 1/T Hz.

Hence, we can express the Delta F as a number of frequency bins “z” drift in the spectrum

z \* (1/T) = Delta F ………………………………………………eq 13

Hence:-

z \* (1/T) = A\*T\*F0/c ……………………………………………eq 14

z/T = A\* F0\*T/c …………………………………………………eq 15

z = A\*F0\*(T2)/c………………………………………………….eq 16

## Determining the Acceleration from the FOP

Each FOP Col from the HSUM results is deemed to be 1/536.87 Hz since the observation period (T) is 536.87 seconds.

FOP Row 43 (P[0]) is the centre of the FOP representing zero acceleration.

FOP Row 43 is effectively the result of a convolution of the DM spectrum with a 1-tap filter with a coefficient of 1 as this is an “all pass” filter

(We don’t actually bother use a filter to generate the central FOP row, but if we did it would be 1-tap with a coefficient of 1)

Each successive FOP Row away from the centre row has a filter which is 10 taps wider i.e.

FOP Row 44 (P[1]) & FOP Row 42 (P[-1]) are (1 x 10) + 1 = 11-taps wide filter

FOP Row 45 (P[2]) & FOP Row 41 (P[-2]) are (2 x 10) + 1 = 21-taps wide filter

etc

Filters P[1] to P[42] are the matched filters for pulsars accelerating away from the observer.

Filters P[-1] to P[-42] are the matched filters for pulsars accelerating towards the observer.

The filter coefficients are complex, which requires a complex conjugate part to form a proper complex filter and give phase symmetry

Hence the number of filter taps required is (twice the bin drift) + 1.

The “+1” is required in the above equation since, for zero drift a 1-tap filter (with a coefficient of 1) provides an “all pass” filter.

Since:

a) The number of filter taps required is (2 x bin drift) + 1 for proper matching

b) For a given observation period (T) the bin drift is proportional to the Acceleration (A) and Tone frequency (F)

c) The filters were designed for a 600 second observation with 85 FOP rows (including the centre row for zero accelaration) with the top and bottom rows correcting acceleration drift of 350m/s/s for a 500Hz tone. This resulted in an extra 2\*350/84 = 8.3m/s/s acceleration drift correction per successive FOP row from the centre row for the 500Hz tone over a 600 second observation. However the current observation period will be 536.87 seconds (223 samples with 64us sampling) and hence the acceleration corrected per successive FOP row will be 8.3 \*(6002)/(536.872) = 10.4m/s/s, noting that the bin drift is proportional to the square of the observaation period. Hence it is assumed that each successive FOP filter away from the centre row represents an extra acceleration of 10.4m/s/s for a 500Hz tone for a 536.87 second observation (from the template generator MATLAB script).

We can say:-

Filter taps = [(A x F)/(500 x 10.4/10)] + 1

i.e. for a zero acceleration pulsar we need [(0 x 500)/(500 x 10.4/10)] + 1 = 1-tap filter for proper matching

for 10.4m/s/s acceleration pulsar we need [(10.4 x 500)/(500 x 10.4/10)] + 1 = 11-tap filter for proper matching

for 20.8m/s/s acceleration pulsar we need [(20.8 x 500)/(500 x 10.4/10)] + 1 = 21-tap filter for proper matching

etc

Hence, if we want to calculate the acceleration from the filter taps we can re-arrange the equation:-

A = [(filter taps -1) x (500 x 10.4/10)]/F

A = [(filter taps -1) x 520]/F

The spreadsheet below allows the FOP Row to be transformed to an acceleration.



# APPENDIX E: CONV Filter Templates

The CONV filter templates are created using the following MATLAB scripts embedded in this document:-





## Template Generation

In section 18.3 it was stated that the number of filter taps required is (twice the bin drift) + 1.

The reasons for this can be seen by examining the code of the templateGenerator.m script:-

%% This module reads in Fourier series data and correlates them with given

% templates over a range of accelerations, producing a list of candidates

function template = templateGenerator(metadata)

numTimeSamps = metadata.numTimeSamps; % Total time samples

sampTime = metadata.sampTime; % Sampling time in seconds

obsLength = numTimeSamps \* sampTime; % Observation length in seconds

c = metadata.speedOfLight; % Speed of light in m s^-2

numTemplates = metadata.numTemplates;

% Create an array to hold the templates

template = cell(numTemplates + 1, 1);

% Calculate the acceleration step size based on the number of templates requested

accelStep = 2 \* 350/numTemplates;

for templateNum = 0:numTemplates

% Calcualte how many bins the signal would drift for a 500 Hz signal

binsSignalDrifts = (accelStep \* templateNum - 350) \* 500 \* (obsLength^2)/c;

% Calculate the width of the template. The shift by binsSignalDrifts/2 is to

% center the template so that the edges are not cut off during its calculation

templateWidth = (-abs(binsSignalDrifts):abs(binsSignalDrifts)) - binsSignalDrifts/2;

% Calculate limits for the Fresnel integrals below

Y = templateWidth \* sqrt(2/binsSignalDrifts);

Z = (templateWidth + binsSignalDrifts) \* sqrt(2/binsSignalDrifts);

% Calculate Fresnel integrals for use in generating the template

% sinY = mfun('FresnelS', Y);

sinY = fresnels(Y);

% sinZ = mfun('FresnelS', Z);

sinZ = fresnels(Z);

% cosY = mfun('FresnelC', Y);

cosY = fresnelc(Y);

% cosZ = mfun('FresnelC', Z);

cosZ = fresnelc(Z);

% Create a template for this acceleration

template{templateNum + 1} = 1/sqrt(2 \* binsSignalDrifts) \* exp(1i \* pi \* (templateWidth.^2)/binsSignalDrifts) .\* (sinZ - sinY - 1i \* (cosY - cosZ));

end

template{numTemplates/2 + 1} = 1;

end

Analysing the above MATLAB templateGenerator.m script:

**accelStep calculation**:

It is desired that the top and bottom rows of the FOP shall correct the drift of a tone that has been accelerated by +/-350m/s/s for a 600 sec observation (as defined in the metadata), and hence each row of the FOP, assuming 85 rows, with the central row for zero accelaration the accelStep = 2\*350/84 = 8.333m/s/s.

**binsSignalDrifts calculation:**

The bin drift correction for each filter away from the FOP central row is calculated based on the formula for the bin drift in sec 18.1. Remember this is for a 600 sec observation. Hence for FOP Row 85 (for which templateNum = 84) for a 500Hz tone:-

binsSignalDrifts = (8.333\*84 -350)\*500\*(600^2) /c = 210.145 bins

**templateWidth calculation:**

For this it is best to use templateNum 84 which will create the template for Row 85 (top row) of the FOP as an example. For this template the binsSignalDrifts is 210.145. If we created (-abs(binsSignalDrifts):abs(binsSignalDrifts)) we would get 421 values

-210.145, -209.145, -208.145……….. -1.145, 0.854, 1.854 ………208.854, 209.854.

Hence there are 421 elements, however the value that is almost 0 is at the centre of the range. As wil be seen later Fresnel Integral are used in creating the filter shape and around a value of 0 is when the Fresnel Integral rapidly changes which is used to create the filter edges. Hence it is necessary to shift the 0 value to where the filter edge is desired and this is done by subtracting binsSignalDrifts/2; to create the sequence

-315.218, -314.218, -313/218…………-1.218, -0.218,0.781……….103.781, 104.781.

So still 421 elements but -0.218 is now element 316 out of 421 and so is (316 – 211) = 105 elements from the centre of the range (which half the required binsSignalDrifts of 210.145).Hence using Fresnel Integrals the edges of the filter can be 105 bins from the centre of the filter.

**Y and Z calculations:**

The Y and Z calculations become the Fresnel Integral limits for each of the elements in templateWidth. Note that the calculation for Z has a shift of binsSignalDrifts to effectively move the transition to 0 to the left by drift value in the list of elements for Z compared to Y.

For the example of templateNum 84 which will create the template for Row 85, the 421 Y and Z values become:-

Y = -30.751,-30.653………… -0.021, 0.076…………………… 8.563, 8.66

Z = -10.250,-10.152………… -0.007, 0.090…………………… 30.235, 30.332

Y has the transition to 0 located 105 elements to the right of centre of the 421 elements

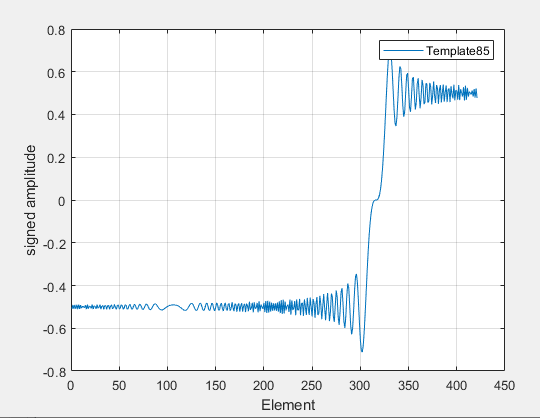
Z has the transition to 0 located 105 elements to the left of centre of the 421 elements

The transitions to 0 are the point where the Fresnel integrals rapidly change.

**SinY, SinZ, CosY and CosZ calculations:**

These are the results of Fresnel Integrals using the associated element of Y and Z as the limits. For the example of templateNum 84 which will create the template for Row 85, the 421 Y and Z values the SinY, SinZ, CosY and CosZ are:-

SinY

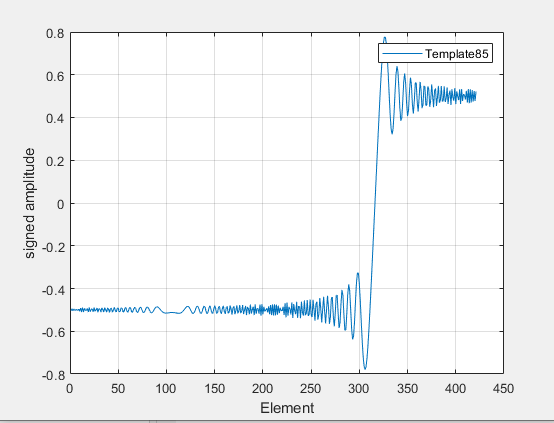


SinZ

A graph with a line graph

Description automatically generated

CosY



CosZ

A graph with a blue line

Description automatically generated

For the plots above for example of templateNum 84 which will create the template for Row 85, each plot is 421 elements, with each element being the Fresnel Integral result for the associated integral element of the 421 elements in Y and Z. The centre of the rising edges occur when the Y or Z element value is 0.

For the plots above for example of templateNum 84 which will create the template for Row 85, if we now plot (SinZ -SinY) and (CosY – CosZ) we see:-

SinZ - SinY

A graph with blue lines

Description automatically generated

CosY – CosZ

A graph of a diagram

Description automatically generated

For example of templateNum 84 which will create the template for Row 85 if we now plot the absolute values we see:

abs(SinZ -SinY)

A graph with blue lines

Description automatically generated

abs(CosY – CosZ)

A screen shot of a graph

Description automatically generated

For example of templateNum 84 which will create the template for Row 85 if we now plot the absolute and phase values of:

(sinZ - sinY - 1i \* (cosY - cosZ))

we see:

abs(sinZ - sinY - 1i \* (cosY - cosZ))

A graph with a line graph

Description automatically generated

Phase (sinZ - sinY - 1i \* (cosY - cosZ))

A graph with blue lines

Description automatically generated

Note that the phase is flat in the filter central region, ane hence a phase term needs to be added to the equaation for the template to match the phase variation of the pulsar tone as the frequency varies.

**exp(1i \* pi \* (templateWidth.^2)/binsSignalDrifts) calculation**

This is a phase term that creates a phase progession in the central region of the filter.

For example of templateNum 84 which will create the template for Row 85 the templateWidth^2/binsSignalsDrifts is calculated in the following spreadsheet for the central region of the filter:-



The green shaded parts of columns S and U of this spreadsheet show that there is a symmetry in the Delta phase between successive elements.

For example of templateNum 84 which will create the template for Row 85 the phase plot of exp(1i \* pi \* (templateWidth.^2)/binsSignalDrifts) is

A graph with blue lines

Description automatically generated

It can be seen that in the central region of the filter there is symmetry in the phase response around the centre of the filter.

**template calculation:**

The template calculation uses the Fresnel Integral results to create the overall filter shape combined with a phase shift shown previously and also a scaling factor of

1/sqrt(2 \* binsSignalDrifts)

For example of templateNum 84 which will create the template for Row 85, since the edges of these responses are centered at the point where the Y and Z elements are 0 which are 105 elements to the right and left of the 421 elements the width of the response at the centre of the rising/falling edges is (105\*2) = 210 elements which will become the bin drift correction for the template, and the total number of elements in the plot is 421 which will become the number of FIR filter taps for the template. This why the bin drift correction is half the number of taps in the template.

The template is the FIR filter tap coefficients.

For example of templateNum 84 which will create the template for Row 85 the template is the FIR filter complex tap coefficients. The figures below shows the absolute values of these tap coefficients and also the phase:-

abs(1/sqrt(2 \* binsSignalDrifts) \* exp(1i \* pi \* (templateWidth.^2)/binsSignalDrifts) .\* (sinZ - sinY - 1i \* (cosY - cosZ)))

A graph of a diagram

Description automatically generated

phase(1/sqrt(2 \* binsSignalDrifts) \* exp(1i \* pi \* (templateWidth.^2)/binsSignalDrifts) .\* (sinZ - sinY - 1i \* (cosY - cosZ)))

A graph of a diagram

Description automatically generated

It should be noted that for a 536.87 second observation the acceleration for each successive FOP row will be 10.4m/s/s (i.e. 8.3 X (600^2)/(536.97^2) as bin drift is proportional to the square of the observation period)

## Template Alignment, Padding and Fourier Transform

For correct alignment in the FOP each filter template has to have its centre tap aligned and this is performed in the conv\_filters.m script which symmetrically pads the filters with (0 + 0i)s to make them all 421 taps and the appends (0 + 0i)s to bring up to 1024 elements which is the CONV FFT size. The result is the Fourier Transformed to create the template.dat file used to configure FDAS. The template.dat represents the filter coefficient configuration in little endian IEEE754 as this is the endian needed for the PC, however the values when loaded into the CONV module are big endian.

It should be noted that the conv\_filters.m script can be configured to either use the templateGenerator.m script or read pre-created values from the templates\_c.mat file.

# Appendix F: FDAS Configuration

The CONV filter templates are described in Appendix E

The HPSEL Configuration is created using the embedded spreadsheet:-



This spreadsheet allows the user to choose what filter tap width to use for the ideal tap width for the actual drift. This is due to the fact that the FOP Rows are quantised with regard to tap length.

An overall FDAS configuration (except the CONV filters) that uses the above spread sheet which may be a useful starting point is:-



# Appendix G: PowerPoint Presentation of FDAS

A useful PowerPoint presentation of FDAS is;



# References

|  |  |  |
| --- | --- | --- |
| Bookmark | Reference | Description |
| [FDAS\_RA] | FDAS\_REQUIREMENTS\_AND\_  ARCHITECTURE\_130.doc | FDAS Requirements and Architecture Specification. |
| [Ref: Agilex Development Board] | Dev Kit: DK-DEV-AGF014EA  Intel Agilex Part: AGFB014R24B2E2V  DDR4 SDRAM Modules: RDIMM DDR4 8GB  K41197-001  MTA9ASF1G72PZ-2G9E1UI  Board Schematic:  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 AGFB014R24B2E2V FPGA. The board supports a PCIe interface and four DDR4 SDRAM interfaces. |

# Abbreviations and Acronyms

|  |  |
| --- | --- |
| CLD | Convolution Load Module |
| CONV | Convolution Module |
| CTRL | Control Module |
| DM | Dispersion Measurement |
| DMA | Direct Memory Access |
| DDR | Double Data Rate (Interface) |
| DDR4 | Double Data Rate Interface Version 4 |
| DDRIF2 | DDR Interface Module |
| DSP | Digital Signal Processing |
| ECC | Error Correcting Code |
| FDAS | Fourier Domain Acceleration Function |
| FFT | Fast Fourier Transform |
| FIR | Finite Impulse Response filter |
| FOP | Filter Output Plane |
| FPGA | Field Programmable Gate Array |
| Gibi | 230 = 1,073,741,824 |
| IEEE | Institute of Electrical Engineers |
| HSUM | Harmonic Summing Module |
| IFFT | Inverse Fast Fourier Transform |
| IP | Intellectual Property |
| IRQ | Interrupt ReQuest |
| JTAG | Joint Test Action Group |
| MC | Monitor and Control Memory Map Interface (to microprocessor) |
| MCI\_TOP | MC Interface Top Level Module |
| MHz | Mega Hertz (106 Hertz) |
| ms | Millisecond (10-3 seconds) |
| MSI | Message Signalled IRQ |
| PCIe | Peripheral Component Interconnect Express |
| PCIF | PCIe Interface Module |
| PLL | Phase Locked Loop |
| RAM | Random Access Memory |
| Rx | Receive |
| SDRAM | Synchronous Dynamic Random Access Memory |
| SW | Software |
| Tx | Transmit |
| VHDL | VHSIC Hardware Description Language |
| VHSIC | Very High Speed Integrated Circuit |