

**FDAS CONV Module Design Specification for Intel Agilex F Implementation**

FDAS\_CONV\_DS Revision 2 Draft A

Classification: UNRESTRICTED

Document type: DTE

Date: 2022-07-29

Status:

© Copyright 2022 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 | Jonathan Taylor | FPGA Design Engineer | SKAO |  | 29/07/2022 |
| 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 1 Draft A | 01 August 2017 | Initial Draft |
| Issue 1 Draft B | 05 December 2017 | Updated with additional information. Incorporating changes based on comments from internal review and from UoM. |
| Issue 1 Draft C | 21 February 2019 | Added delay for Row 0 results to match filter delays. MCI updated with latest cmgedit. |
| Issue 1 Draft D | 9 September 2021 | Updated the “Place in the System” diagram |
| Issue 2 Daft A | 29 July 2022 | Updated design to use the Intel Agilex FPGA |

**Table of Contents**

1 Introduction 5

2 Place in the System 6

3 Functional Specification 7

3.1 CONV\_FFT 7

3.1.1 FFT1024 IP 7

3.1.2 Control Signal Generation 8

3.2 CONV\_FFT\_STR 8

3.3 CONV\_COEF\_STR 9

3.4 CONV\_MULT 10

3.5 CONV\_IFFT 11

3.5.1 IFFT1024 IP 11

3.5.2 Control Signal Generation 11

3.6 CONV\_PWR 12

3.7 CONV\_RESULT\_STR 12

3.8 FOP\_STR 13

3.9 CONVMCI: Control Module Micro Controller Interface 17

3.9.1 Write to CONVMCI 17

3.9.2 Read from CONVMCI 18

3.10 CONV Flow Control 20

3.10.1 Input to FFT Store 20

3.10.2 FFT Store to Result Store 21

3.10.3 Result store to output DDR interface 22

4 Design Tags 23

5 Interface Specification 27

6 MCI Memory Mapped Interface 29

6.1 RAM: FILTER\_COEFFICIENTS 29

6.2 RAM: FFT\_RESULTS 30

6.3 Row 0 Delay 30

6.4 ALARMS 31

7 Design Parameterisation 32

8 References 32

9 Abbreviations and Acronyms 32

Appendix A : IEEE-754 single precision 33

Appendix B : FFT Overlap-Save Method of Convolution 34

Appendix C : Intel Agilex Variable Precision DSP Blocks 35

Appendix D : Intel Agilex IP Generation 37

FFT IP 37

IFFT IP 37

Floating Point Multiplier IP 38

Floating Point Multiplier with Adder IP 39

Floating Point Multiplier with Subtraction IP 41

**Figures**

[Figure 1 – CONV Location in FDAS 6](#_Toc109825818)

[Figure 2 – CONV Module Architecture 7](#_Toc109825819)

[Figure 4 - CONVMCI Write Access Timing Diagram showing the minimum access time 18](#_Toc109825820)

[Figure 5 - CONVMCI Read Access Timing Diagram showing the minimum access time 19](#_Toc109825821)

[Figure 3 – CONV Flow Control 20](#_Toc109825822)

**Tables**

[Table 1 - DDR SDRAM #1 data bus arrangement out of CONV when ddr\_g = 1 14](#_Toc109825840)

[Table 2 - DDR SDRAM #1 data bus arrangement out of CONV when ddr\_g = 2 15](#_Toc109825841)

[Table 3 - DDR SDRAM #2 data bus arrangement out of CONV when ddr\_g = 2 15](#_Toc109825842)

[Table 4 - DDR SDRAM #1 data bus arrangement out of CONV when ddr\_g = 3 16](#_Toc109825843)

[Table 5 - DDR SDRAM #2 data bus arrangement out of CONV when ddr\_g = 3 16](#_Toc109825844)

[Table 6 - DDR SDRAM #3 data bus arrangement out of CONV when ddr\_g = 3 17](#_Toc109825845)

[Table 7 - Design Tags 25](#_Toc109825846)

[Table 8 - Interface Specifications 28](#_Toc109825847)

[Table 9 - Generic definitions 32](#_Toc109825848)

# Introduction

This document describes the design implementation of the CONV Module used in the FDAS FPGA.

The Convolution (CONV) module performs a convolution in the frequency domain between the observed data and a set of filter coefficients to allow determination of any possible pulsar indications by reversing the effects of any orbital motion of the pulsar.

Dependent on which filter convolution is being processed, different data processing shall be performed:-

* Filter convolution 0 (for the p = 0 row of the FOP) shall only require conversion of each freq-bin from a complex amplitude to power.
* Filter convolutions +/-1 to +/-42 (for the p = +/-1 to +/-42 rows of the FOP) shall require the “FFT Overlap with Save” convolution method with the padded transformed filter coefficients and subsequent conversion of the convolution result from a complex amplitude to power.

The processing in the CONV module maintains counters so that the freq-bin numbers of the convolution result are known. These counters support frequency bin numbers up to (223 -1). These ensure that convolution results are stored in the correct external DDR memory location of the FOP.

Flow control is implemented throughout CONV to prevent corruption when it cannot accept any more data due to bandwidth limitations of the external DDR memory interface.

The CONV module shall operate at the core system clock and micro clock, CLK\_SYS and CLK\_MC, frequency of 350MHz.

# Place in the System

The CONV Module’s place in the system is shown highlighted in the figure below:-

PCIe

**CLD**

**CTRL**

FDAS Control

CLD\_DONE

**CONV**

data[63:0]

ready

**HSUM**

**FDAS**

CONV\_DONE CONV\_FFT\_REDAY

valid

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

**MCI\_TOP**

CLD\_DM\_TRIGGER

CLD\_ENABLE

CLD\_PAGE[31:0]

CONV\_DM\_TRIGGER

CONV\_ENABLE

CONV\_PAGE[31:0]

IFFT\_LOOP\_NUM[5:0]

HSUM\_DM\_TRIGGER

HSUM\_ENABLE

HSUM\_PAGE[31:0]

HSUM\_DONE

**PCIF**

ADDR

DATA[511:0]

**DDRIF2 #1**

**DDRIF2 #2**

ADDR

DATA[511:0]

DATA[511:0]

ADDR

ADDR

DATA[511:0]

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

*Note: CLD, CONV and HSUM are designed with generically sized data width interfaces to DDRIF2 for a future implementation, but in this implementation the DDRIF2 data width is fixed. CLD, CONV and HSUM are also designed with paging of the DDR memory for a future implementation.*

**FDAS\_DDR\_CONTROLLER\_**

**CALIBRATION (INTEL IP) #1**

**FDAS\_DDR\_CONTROLLER\_**

**CALIBRATION (INTEL IP) #2**

DMA Transfer

**External 4 Gibi-Byte**

**DDR4 SDRAM Memory #1**

**External 4 Gibi-Byte**

**DDR4 SDRAM Memory #2**

MC

Interface

Via

PCIe

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

MC Bus

OVERLAP\_SIZE[9:0]

FOP\_SAMPLE\_NUM[22:0]

**MSIX**

MSI-X

Figure 1 – CONV Location in FDAS

# Functional Specification

The CONV Module architecture is shown in the figure below:-

FFT

FFT STORE

FOP STORE

Result store

Mult

Mult

Result store

Result store

IFFT

IFFT

x IFFT\_LOOP\_G

x IFFT\_G

Power

Power

Power

MCI

IDATA[31:0]

RDATA[31:0]

READYOUT

SOF

EOF

VALID

MCDATA[31:0]

MCADDR[19:0]

MCRWN

MCMS

MCDATAOUT[31:0]

DDR\_DATAOUT[255:0]

DDR\_ADDROUT[31:0]

DDR\_VALIDOUT

DDR\_WAITREQ

FOP\_NUM

CONV\_TRIGGER

CONV\_DONE

CONV\_ENABLE

IFFT\_LOOP\_NUM

OVERLAP\_SIZE

coef

PAGE\_START

MCREADYOUT

CONV\_FFT\_STR

CONV\_FFT

CONV\_MULT

CONV\_MULT

CONV\_IFFT

CONV\_IFFT

CONV\_PWR

CONV\_PWR

CONV\_PWR

CONV\_RESULT\_STR

FOP\_STR

CONVMCI

CONV\_COEF\_STR

CONJ

Filter 0

Filter 1 to 42

Filter -1 to -42

CONV\_RESULT\_STR

CONV\_RESULT\_STR

Figure 2 – CONV Module Architecture

## CONV\_FFT

VHDL entity: conv\_fft.vhd

VHDL architecture: conv\_fft\_synth.vhd

### FFT1024 IP

CONV\_FFT performs a floating point fast fourier transform (FFT). The number of points can be up to 1024 and is determined by the generic fft\_g.

Input: AXI streaming, natural order, complex number, IEEE-754 single precision floating point format.

Output: AXI streaming, bit reversed order, complex number, IEEE-754 single precision floating point format.

The FFT is implemented in Intel Agilex with the Unified FFT Intel FPGA IP. Generated using Quartus Platform Designer tool (see Appendix D).

Parameter setting:

1024 points

Forward transform

Natural order input

Bit-reversed order output

Hard macros enabled

IP design:

fft1024\_lib/fft1024

Resource usage:

|  |  |
| --- | --- |
| DSP Blocks | 36 |
| M20Ks | 18 |
| Registers | 6632 |
| ALUTs | 2101 |

### Control Signal Generation

The Unified FFT Intel FPGA IP requires that the ValidIn signal must not deassert during an FFT. It must be asserted from the first input to last input of an FFT.

The READYOUT signal passed back to the preceeding CLD module controls the input data. READYOUT is generated from the READY signal received from the following FFT store submodule, however it is only de-asserted when the VALID signal is inactive between 1024 word frames of data.

The Unified FFT Intel FPGA IP does not include SOF or EOF. The SOF and EOF signals required by the following FFT store submodule are regenerated from the FFT’s output valid signal. The SOF from the rising edge of the valid signal and the EOF from the falling edge. Retiming ensures that the SOF coincides with first word of the frame and the EOF with the last word of the frame.

## CONV\_FFT\_STR

VHDL entity: conv\_fft\_str.vhd

VHDL architecture: conv\_fft\_str\_synth.vhd

CONV\_FFT\_STR stores the results of the CONV\_FFT until they are required by the CONV\_MULT submodules. Data is read out along with a read address and read enables to obtain coefficients from CONV\_COEF\_STR submodules. The coefficients are supplied together with the data to the CONV\_MULT submodule.

Input: AXI streaming, bit-reversed order, complex number, IEEE-754 single precision floating point format.

Output: AXI streaming, bit-reversed order, complex number, IEEE-754 single precision floating point format.

For diagnostic purposes, MCI read access is provided to allow FFT results to be read back.

The generic, pages\_g determines the number of pages provided by CONV\_FFT\_STR.

It implements “pages\_g” pages of 1024 frequency bin storage, each bin contains a real and imaginary value, both in IEEE-754 single precision floating point format.

Writes occur in bit-reverse order and therefore the write address is bit-reversed. When a page is complete as indicated by receipt of EOF high, the write page number is incremented.

A difference in the write and read page addresses indicates whether data is available for the read process. The read address is bit-reversed and data is output with the VALIDOUT set high. SOFOUT is set on the first data word and EOFOUT is set on the last data word.

The read process is repeated a number of times as determined by the loop\_g generic and the LOOP\_NUM input configuration signal.

To ensure that only complete frames of data are sent out through the IFFT submodules. The decision to start sending data is made before each sequence of iterations begins. Once started, the sequence will continue to completion.

Data flow into the FFT store is controlled by outputting the READYOUT signal back to the CONV\_FFT. This is set high to indicate ready when the read and write page addresses indicate space is available.

An address is generated and output from this block to allow coefficients to be read from the coefficient store. This address is also in bit-reverse order to match the output data.

If a diagnostic micro read is required, “CONV\_ENABLE” input is set low. When a segment has been written into the store, the block indicates micro reads can commence by setting the DONEOUT output high. This signal becomes MCREADYOUT on the CONV module. The data read by the micro shall automatically be from the last page written. Normal internal reads may continue whilst MCI data is being accessed. Writes can continue until either the current MCI read page or the internal read page is reached. Further writes are stopped by setting the READYOUT output low.

## CONV\_COEF\_STR

VHDL entity: conv\_coef\_str.vhd

VHDL architecture: conv\_coef\_str\_synth.vhd

Each instance of CONV\_COEF\_STR contains the coefficients for all loop iterations of one CONV\_IFFT pair as determined by the generic, loop\_g. The coefficients have a real and imaginary component, both in IEEE-754 single precision floating point format.

Within CONV\_COEF\_STR, one coefficient store is provided for each loop iteration. Each coefficient store is configured via the CONVMCI. The inputs MCRDEN and MCWREN contain a separate bit for each coefficient store.

A read address, RDADDR, and read enable RDEN are provided by the CONV\_FFT\_STR. A second address is generated which is sequence reversed. The two addresses are applied to all coefficient stores and data is read from the store which is valid for the current iteration as indicated by the bit set in RDEN.

The data read with the second address is converted to its conjugate by inverting the sign of the imaginary component. See Appendix A for details of the IEEE-754 single precision floating point format.

## CONV\_MULT

VHDL entity: conv\_mult.vhd

VHDL architecture: conv\_mult\_scm.vhd

CONV\_MULT multiplies complex valued samples with complex coefficients and their conjugates.

All arithmetic is floating point in IEEE-754 single precision format.

Input and output data is basically AXI streaming, but no flow control is provided.

CONV contains ifft\_g instances of CONV\_MULT.

Complex multiplication can be expanded:-

It can be seen from the equation that the real part of the result is obtained from a multiply and subtract function. While the imaginary part of the result is obtained from a multiply and add function.

Each multiply and add/subtract function makes use of 2 of the hard floating point (HFP) DSP blocks provided by the FPGA. See Appendix C. Details of the multiply and add is shown in Figure C-2. For the multiply and subtract the adder is configured to be a subtract operation.

The HFP blocks are designed using Intel Quartus Platform Designer tool (see Appendix D).

IP designs:

dsp\_prim\_lib/cmplxmult\_fp

dsp\_prim\_lib/mult\_fp\_co

dsp\_prim\_lib/multadd\_fp\_ci

dsp\_prim\_lib/multsub\_fp\_ci

Latency through cmplxmult\_fp is 5 clock cycles.

CONV\_MULT also provides a delay in the data path so that the delay in reading coefficients is matched by the input data.

## CONV\_IFFT

VHDL entity: conv\_ifft.vhd

VHDL architecture: conv\_ifft\_synth.vhd

### IFFT1024 IP

Performs a floating point inverse fast fourier transform (IFFT). Number of points can be up to 1024 and is determined by the generic fft\_g.

Input and output data is basically AXI streaming, including flow control.

FDAS contains 2\*ifft\_g instances of CONV\_IFFT.

The IFFT is implemented in Intel Agilex with the Unified FFT Intel FPGA IP. Generated using Quartus Platform Designer tool (see Appendix D). The IFFT function is completed by scaling the output values. This is accomplished by dividing by N, where N is the number of points of the IFFT determined by fft\_g.

Parameter setting:

1024 points

Reverse transform

Bit-reversed order input

Natural order output

Hard macros enabled

IP design:

ifft1024\_lib/ifft1024

Resource usage:

|  |  |
| --- | --- |
| DSP Blocks | 36 |
| M20Ks | 18 |
| Registers | 5686 |
| ALUTs | 2032 |

### Control Signal Generation

The Unified FFT Intel FPGA IP requires that the ValidIn signal must not deassert during an FFT. It must be asserted from the first input to last input of an FFT. The FFT store submodule ensures this requirement is met. The data it supplies will only stop at the end of a frame.

The Unified FFT Intel FPGA IP does not include SOF or EOF. The SOF and EOF signals required by the following FFT store submodule are regenerated from the IFFT’s output valid signal. The SOF from the rising edge of the valid signal and the EOF from the falling edge. Retiming ensures that the SOF coincides with first word of the frame and the EOF with the last word of the frame.

## CONV\_PWR

VHDL entity: conv\_pwr.vhd

VHDL architecture: conv\_pwr\_scm.vhd

CONV\_PWR converts a complex amplitude value to a power value. Real and imaginary floating point values are squared and then added together.

Input and output data is basically AXI streaming, but no flow control is provided.

FDAS contains 2\*ifft\_g+1 instances of CONV\_PWR

The submodule contains two multipliers and an adder that operate on IEEE-754 single precision floating point values. The power calculator makes use of 2 of the hard floating point (HFP) DSP blocks provided by the FPGA. See Appendix C. Details of the multiply-add is shown in Figure C-2.

The HFP blocks are designed using Intel Platform Designer tool.

IP designs:

dsp\_prim\_lib/mult\_fp\_co

dsp\_prim\_lib/multadd\_fp\_ci

## CONV\_RESULT\_STR

VHDL entity: conv\_result\_str.vhd

VHDL architecture: conv\_result\_str\_synth.vhd

CONV\_RESULT\_STR stores the output of the convolution process after it has been converted to a power value.

The number of pages is determined by setting the generic, pages\_g.

The input is basically AXI streaming with no flow control. The output is read on demand from a specified address.

FDAS contains 2 \* ifft\_g \* ifft\_loop\_g + 1 instances of CONV\_RESULT\_STR.

Each page stores 1024 x 32 bit single precision floating point values. After each write the write address is incremented. Following a write with the WREOF input active, the next write address is set to zero and the write page is incremented.

The read address and read enable are input to the submodule. The read data is output after two cycle delay. The input RDEOF is active on last read request. After the last read, the read page is incremented.

If a write is attempted to the current read page, the OVERFLOW output is asserted.

## FOP\_STR

VHDL entity: conv\_fop\_str.vhd

VHDL architecture: conv\_fop\_str\_synth.vhd

Reads samples from CONV\_RESULT\_STR instances and transfers them out to the DDR write port. The sub-module supports up to 3 DDR interfaces as set by the generic, ddr\_g.

Data is output as a ddr\_g\*256-bit word, DDR\_DATAOUT. For each DDR interface the data output consists of up to 8x32-bit single precision floating point values. Unused bits are set to zero. The common DDR address is output on DDR\_ADDROUT.

Full AXI-4 flow control is provided on the output and is common across all the DDR interfaces. Valid data is indicated by setting DDR\_VALIDOUT to ‘1’. The input DDR\_WAITREQ indicates that data was taken from the output when it is set low.

Read requests are generated in the form of RDENOUT bits assigned to each result store being set to ‘1’ and the common RDADDROUT being set to required address. A separate address, RDADDR0OUT, is supplied to the result store of Row 0. RDADDR0OUT is calculated by subtracting the configuration value ROW0\_DELAY from the value assigned to RDADDROUT. The following algorithm is used to generate the required sequence of RDENOUT bits.

RDENOUT <= (others => '0');

if wordcnt=0 then

RDENOUT(0) <= '1';

end if;

for j in 1 to ddr\_g\*14 loop

if wordcnt\*ddr\_g\*14+j<=2\*ifft\_g\*ifft\_loop\_g then

RDENOUT(wordcnt\*ddr\_g\*14+j) <= '1';

end if;

end loop;

After 3 clock cycles read data is expected at input RDDATA. Data is saved until requested by DDR\_WAITREQ being set to zero. Four output words can be saved at any one time so that as demand stops and starts, the delay fetching new data results in no delay in data at the output.

In order to complete the FFT Overlap-Save Method of Convolution as shown in Appendix B, data reads from the CONV\_RESULT\_STR instances begins at the address specified by the OVERLAP\_SIZE input.

Data from each CONV\_RESULT\_STR instance is mapped to its correct position in the output data words. The following algorithm is used to map read data to output data.

if wordcnt=0 then

outdata(31:0) <= readdata(31:0);

outdata(63:32) <= readdata(31:0);

else

outdata(63:0) <= (others => '0');

end if;

for i in 0 to ddr\_g-1 loop

for j in 1 to 14 loop

if wordcnt\*ddr\_g\*14+i\*14+j<=2\*ifft\_g\*ifft\_loop\_g then

outdata((i\*16+j+2)\*32-1:(i\*16+j+1)\*32) <= readdata(wordcnt\*ddr\_g\*14+i\*14+j+1)\*32-1: (wordcnt\*ddr\_g\*14+i\*14+j)\*32);

end if;

end loop;

end loop;

The full set of output data may need several cycles to output as is shown in the following examples.

After a rising edge on CONV\_TRIGGER, DDR\_ADDROUT begins at the value specified by PAGE\_START. When the number of words written reaches the value specified by the input FOP\_NUM minus one, the DONEOUT signal is asserted for one clock cycle

For diagnostic purposes the address offset is inserted into the unused 64 bits. Only DDR locations containing data are written.

Example data output word mapping:-

* ddr\_g = 1

DDR SDRAM #1

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CONV\_  DDR\_  ADDR  Page &  cnt\_c &  cnt\_q &  cnt\_p  /Hex | Freq-Bin  Page | Convolution Results: PWR p[\*] Mapped to the CONV DDR \_DATA[512:0] Interface  Bit 511 Bit 0 | | | | | | | | | | | | | | | |
| 0x0000000 | freq-bin1  Page0 | 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[-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] |  |  |
| 0x0000002 | 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] |  |  |
| 0x0000003 | 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] |  |  |
| 0x0000004 | 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] |  |  |
| 0x0000005 | 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] |  |  |
| 0x0000008 | freq-bin2  Page0 | 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] |
| 0x0000009 | 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] |  |  |
| 0x000000A | 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] |  |  |
| 0x000000B | 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] |  |  |
| 0x000000C | 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] |  |  |
| 0x000000D | 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] |  |  |
| : |  | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : |
| 0x1FFFFF8 | freq-bin222  Page0 | 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] |
| 0x1FFFFF9 | 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] |  |  |
| 0x1FFFFFA | 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] |  |  |
| 0x1FFFFFB | 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] |  |  |
| 0x1FFFFFC | 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] |  |  |
| 0x1FFFFFD | 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] |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0x2000000 | freq-bin1  Page1 | 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] |
| 0x2000001 | 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] |  |  |
| 0x2000002 | 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] |  |  |
| 0x2000003 | 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] |  |  |
| 0x2000004 | 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] |  |  |
| 0x2000005 | 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] |  |  |
| : |  | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : |
| 0x3FFFFF8 | freq-bin222  Page1 | 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] |
| 0x3FFFFF9 | 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] |  |  |
| 0x3FFFFFA | 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] |  |  |
| 0x3FFFFFB | 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] |  |  |
| 0x3FFFFFC | 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] |  |  |
| 0x3FFFFFD | 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 1 - DDR SDRAM #1 data bus arrangement out of CONV when ddr\_g = 1

* ddr\_g = 2

DDR SDRAM #1

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CONV\_  DDR\_  ADDR  Page &  cnt\_c &  cnt\_q &  cnt\_p  /Hex | Freq-Bin  Page | Convolution Results: PWR p[\*] Mapped to the CONV DDR \_DATA[512:0] Interface  Bit 511 Bit 0 | | | | | | | | | | | | | | | |
| 0x0000000 | freq-bin1  Page0 | 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 | freq-bin2  Page0 | 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 | freq-bin222  Page0 | 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] |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0x1000000 | freq-bin1  Page1 | 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] |
| 0x1000001 | 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] |  |  |
| 0x1000002 | 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] |  |  |
| : |  | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : |
| 0x1FFFFFC | freq-bin222  Page1 | 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] |
| 0x1FFFFFD | 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] |  |  |
| 0x1FFFFFE | 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] |  |  |

Table 2 - DDR SDRAM #1 data bus arrangement out of CONV when ddr\_g = 2

DDR SDRAM #2

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CONV\_  DDR\_  ADDR  Page &  cnt\_c &  cnt\_q &  cnt\_p  /Hex | Freq-Bin  Page | Convolution Results: PWR p[\*] Mapped to the CONV DDR \_DATA[512:0] Interface  Bit 511 Bit 0 | | | | | | | | | | | | | | | |
| 0x0000000 | freq-bin1  Page0 | 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 | freq-bin2  Page0 | 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 | freq-bin222  Page0 | 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] |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0x1000000 | freq-bin1  Page1 | 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] |  |  |
| 0x1000000 | 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] |  |  |
| 0x1000000 | 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] |  |  |
| : |  | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : |
| 0x1FFFFFC |  | 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] |  |  |
| 0x1FFFFFD | 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] |  |  |
| 0x3FFFFFE | 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 3 - DDR SDRAM #2 data bus arrangement out of CONV when ddr\_g = 2

* ddr\_g = 3

DDR SDRAM #1

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CONV\_  DDR\_  ADDR  Page &  cnt\_c &  cnt\_q &  cnt\_p  /Hex | Freq-Bin  Page | Convolution Results: PWR p[\*] Mapped to the CONV DDR \_DATA[512:0] Interface  Bit 511 Bit 0 | | | | | | | | | | | | | | | |
| 0x0000000 | freq-bin1  Page0 | 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[-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 | freq-bin2  Page0 | 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] |
| 0x0000003 | 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] |  |  |
| : |  | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : |
| 0x07FFFFE | freq-bin222  Page0 | 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] |
| 0x07FFFFF | 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] |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0x0800000 | freq-bin1  Page1 | 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] |
| 0x0800001 | 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 | freq-bin222  Page1 | 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] |
| 0x0FFFFFF | 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] |  |  |

Table 4 - DDR SDRAM #1 data bus arrangement out of CONV when ddr\_g = 3

DDR SDRAM #2

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CONV\_  DDR\_  ADDR  Page &  cnt\_c &  cnt\_q &  cnt\_p  /Hex | Freq-Bin  Page | Convolution Results: PWR p[\*] Mapped to the CONV DDR \_DATA[512:0] Interface  Bit 511 Bit 0 | | | | | | | | | | | | | | | |
| 0x0000000 | freq-bin1  Page0 | 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[-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] |  |  |
| 0x0000002 | freq-bin2  Page0 | 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] |  |  |
| 0x0000003 | 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] |  |  |
| : |  | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : |
| 0x07FFFFE | freq-bin222  Page0 | 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] |  |  |
| 0x07FFFFF | 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] |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0x0800000 | freq-bin1  Page1 | 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] |  |  |
| 0x0800001 | 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] |  |  |
| : |  | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : |
| 0x0FFFFFE | freq-bin222  Page1 | 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] |  |  |
| 0x0FFFFFF | 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] |  |  |

Table 5 - DDR SDRAM #2 data bus arrangement out of CONV when ddr\_g = 3

DDR SDRAM #3

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| CONV\_  DDR\_  ADDR  Page &  cnt\_c &  cnt\_q &  cnt\_p  /Hex | Freq-Bin  Page | Convolution Results: PWR p[\*] Mapped to the CONV DDR \_DATA[512:0] Interface  Bit 511 Bit 0 | | | | | | | | | | | | | | | |
| 0x0000000 | freq-bin1  Page0 | 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] |  |  |
| 0x0000001 | 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] |  |  |
| 0x0000002 | freq-bin2  Page0 | 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] |  |  |
| 0x0000003 | 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] |  |  |
| : |  | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : |
| 0x07FFFFE | freq-bin222  Page0 | 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] |  |  |
| 0x07FFFFF | 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] |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0x0800000 | freq-bin1  Page1 | 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] |  |  |
| 0x0800001 | 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] |  |  |
| : |  | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : | : |
| 0x0FFFFFE | freq-bin222  Page1 | 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] |  |  |
| 0x0FFFFFF | 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 - DDR SDRAM #3 data bus arrangement out of CONV when ddr\_g = 3

## CONVMCI: Control Module Micro Controller Interface

VHDL entity: convmci.vhd

VHDL architecture: convmci\_synth.vhd

The CONVMCI module provides a microcontroller interface to allow configuration data to be written into CONV and reading back of configuration data and status from CONV.

Data is written in via MCDATAIN and read out via MCDATAOUT.

To minimise the address decoding within CONVMCI, the CONV module is supplied with a MCMS signal (active high) to indicate that the CONVMCI module has been selected.

Back-to-back accesses without an intermediate “idle” cycle are supported.

The CONVMCI module operates at the CLK\_MC frequency of 350MHz.

### Write to CONVMCI

When MCMS = 1 and MCRWN = 0, if MCADDR is a valid address of the memory map the value of MCDATAIN shall be written. This operation shall be synchronous to CLK\_MC, with the D-type storage identified by MCADDR being loaded with MCDATAIN each clock cycle at which MCMS = 1 and MCRWN = 0.

All internal stored values in D-types shall be set to 0 if the asynchronous reset “RST\_MC\_N” is asserted (active low). Filter coefficients stored in RAM are not initialised by “RST\_MC\_N”.

The figure below shows a write access with the minimum access time that is required to store the data.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  | A |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  | D |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  | D |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

MCDATAIN

MCADDR

MCMS

Internal Storage at Location “A”

CLK\_MC

**DATAIN “D” stored at MCADDR “A” here, hence the access can complete on this cycle**

MCRWN

Figure 4 - CONVMCI Write Access Timing Diagram showing the minimum access time

### Read from CONVMCI

When MCMS = 1, if MCADDR is a valid address of the memory map the value of MCDATAOUT shall be set to the value of the targeted location. This operation shall be synchronous to CLK\_MC, with the targeted location identified by MCADDR driving MCDATAOUT each clock cycle at which MCMS = 1 and MCRWN = 0.

If MCMS = 0 or MCADDR is not a valid address of the memory map MCDATAOUT shall be set to 0x0000,0000

The figure below shows a Read access with the minimum access time that is required to read the data.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  | |  |  | |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  | |  |  | |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  | |  |  | |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  | |  |  | |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  | Don’t care for Read Access | | | | | | | | |  |  |  |  |  |  |  |
|  |  |  |  | |  |  | |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  | A | |  |  | |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  | |  |  | |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  | D |  |  |  |  |  |  |  |  |
|  |  |  |  | |  |  | |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  | |  |  | |  |  |  |  |  |  |  |  |  |  |  |

MCDATAOUT

MCADDR

MCMS

CLK\_MC

**MCDATAOUT “D” from MCADDR “A” available here to be sampled.**

MCRWN

Figure 5 - CONVMCI Read Access Timing Diagram showing the minimum access time

## CONV Flow Control

The flow of data through CONV is controlled by passing back ready signals.

The flow control for CONV can be split into 3 sections:

Input to FFT store

FFT store to result store

Result store to output DDR interface.

FFT

FFT STR

MULT/ IFFT/ PWR

RESULT STR

FOP STR

READY

READY

READ

DDR

READY

VALID SOF, EOF

DATA

READY

1118 cycles

1109 cycles

Figure 3 – CONV Flow Control

### Input to FFT Store

The ready output from the FFT store is only asserted when there is sufficient room in the store to take all blocks of data currently in the pipeline from the input through the FFT submodule.

The ready output from the FFT submodule is output from the CONV module as READYOUT to control data input from the CLD module. The FFT submodule receives the ready signal from the FFT store, but will only de-assert its ready output signal between blocks of input data. This satisfies the requirement that once started the FFT must continue without breaks to completion.

Latency through FFT/IFFT has now been determined by simulation to be 1109 cycles. Therefore it has been determined that it is possible to receive a further 3 blocks into the store even after de-asserting the FFT store Ready output. Therefore FFT store is set to 4 and the Ready output is only asserted if 3 pages are free. This situation is as follows:

Starting from empty:

1. First block starts into FFT

2. 1025 cycles, second block starts into FFT

3. 1109 cycles, first block starts into FFT store (3 pages remain)

4. 2050 cycles third block starts into FFT

5. 2134 cycles, second block starts into FFT store. FFT store write page increments (2 pages remain). Ready output de-asserted.

6. 3075 cycles, ready prevents fourth block starts into FFT

7. 3161 cycles, third block starts into FFT store. FFT store write page increments (1 pages remain).

### FFT Store to Result Store

When the last result store is approaching being full the ready signal READYOUT is deasserted. This stops any further sequences of repeat reads and therefore data from the FFT store. This ensures that data stops before it can overflow the results store.

In the FFT store, a sequence of repeat reads only begins when the ready from the last result store is asserted. Once a sequence has started it is unaffected by the ready until the sequence is complete. This satisfies the requirement that once started the IFFT must continue without breaks to completion.

1106 cycles though FFT/IFFT, 1109 including retimes and sof/eof generation

5 cycles through Mult

4 cycles through Pwr

A single READY is taken from the last result store of the last iteration. This is the last to be read to the DDR and so the last to re-assert if de-asserted.

FFT store decides whether to start a new repeat sequence depending on the ready. Once started, the sequence will continue until completion. As de-assertion of the READY from the last iteration may not come in time to stop the next sequence starting, this READY signal is de-asserted when the last page is reached.

Taking 2 iterations as worse case.

With 2 pages:

1. block of data output from FFT store for first iteration.

2. 1025 cycles, block of data output from FFT store for second iteration.

3. 1118 cycles, first iteration result store starts write.

4. 2050 cycles, block of data output from FFT store for first iteration.

5. 2143 cycles, first iteration result store ready de-asserted as last page (ready ignored).

6. 2143 cycles, second iteration result store starts write.

7. 3075 cycles, block of data output from FFT store for second iteration.

8. 3168 cycles, second iteration result store ready de asserted as last page.

9. 3168 cycles, first iteration full

10. 4100 cycles, ready prevents block of data output from FFT store for first iteration

11. 4193 cycles, second iteration full

12. Ready only re-asserted when result store is empty

Therefore the store cannot overflow. However, it has to wait for the store to be completely empty before resuming fill.

To improve performance, increase pages to 3. Hence ready de-asserted when 2 pages filled (last page) and re-asserted when 1 page filled.

### Result store to output DDR interface

DDR data output from the CONV can be paused by setting DDR\_WAITREQ high. This signal is inverted and connected to the READY input of the FOP\_STR module.

CONV\_RESULT modules indicate when a page is available by asserting the AVAILOUT output. AVAILOUT from the last result store is used by FOP\_STR to decide whether to begin transmitting a new page.

# Design Tags

The relevant design tags from the FDAS Design Specification Document for the CONV module are listed below.

| **Design Tag** | **Description** | **Comment** |
| --- | --- | --- |
| 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. |  |
| 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. |  |
| 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. | Current design supports up to 222 freq-bins. Number is set by FOP\_NUM input from CTRL module. Therefore, no generic required. |
| 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. |  |
| FDAS.CONVOLUTION:040/A | Each convolution of the 222 freq-bin input sequence can be performed by either:-   * Direct time domain convolution by passing the input sequence through a FIR filter (see sec 2.8.1.1 of “FDAS REQUIREMENTS AND ARCHITECTURE SPECIFICATION”).   OR   * The FFT overlap-add method, with Fourier (FTT) transformation of the input sequence to the time domain (with sufficient padding for the overlap), multiplication element by element with Fourier transformed filter coefficients and inverse Fourier (IFFT) transformation to the frequency domain (see sec 2.8.1.2 of “FDAS REQUIREMENTS AND ARCHITECTURE SPECIFICATION”).   OR   * The FFT overlap-save method, with Fourier (FTT) transformation of the input sequence to the time domain, multiplication element by element with Fourier transformed filter coefficients which have been padded to the FFT length and inverse Fourier (IFFT) transformation to the frequency domain (see sec 2.8.1.3 of “FDAS REQUIREMENTS AND ARCHITECTURE SPECIFICATION”).   The current intention is to use the overlap-save method. | Overlap-save method implemented. |
| 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. | Only Fourier Transform convolutions implemented. |
| 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. |  |
| 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. |  |
| 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. | In addition to being complex conjugates of the +ve filter coefficients, the –ve filter also requires coefficients in reverse sequence order. |
| 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”). |  |
| 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”. |  |
| 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. | Arithmetic implemented in Altera Hard Floating Point macros. |
| 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. | Arithmetic implemented in Altera Hard Floating Point macros. |
|  |  |  |
| 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 | 32-bit data Bus |
|  |  |  |
| FDAS.DIAGNOSTIC:050/A | It shall be possible to trigger each main function within FDAS individually if desired via the MC interface. | This is a diagnostic aid to allow each function to be triggered individually to study the effect of the function. |
| 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. | This is a diagnostic aid to allow a function to run for a set period, thus allowing the function’s output to be observed before the end of a DM.  This is achieved by control of the CONV\_ENABLE input. |

Table 7 - Design Tags

# Interface Specification

| Signal | Direction | Clock Domain | Description |
| --- | --- | --- | --- |
| **Interface to MCI\_TOP** |  |  |  |
| MCMS | IN | CLK\_MC | Module Select. When MCMS =1 the CTRL module is selected by the MCI\_TOP module. |
| MCADDR[19:0] | IN | CLK\_MC | Micro Configuration Address. |
| MCRWN | IN | CLK\_MC | Micro Configuration Read Not Write.  ‘1’ = Read from CONV  ‘0’ = Write to CONV. |
| MCDATAIN[31:0] | IN | CLK\_MC | Micro Configuration Data In from MCI\_TOP. This data will be written to address location MCADDR if MCCSN =0 and MCRWN = 0 . |
| MCDATAOUT[31:0] | OUT | CLK\_MC | Micro Configuration Data Out to MCI\_TOP. This data will be valid from a valid MCADDR if MCMS =1  If MCMS = 0 then MCDATAOUT will be 0x00000000 |
| **CONV Data from CLD Module** |  |  |  |
| RDATA[31:0] | IN | CLK\_SYS | Real Sample Data input, IEEE-754 single precision |
| IDATA[31:0] | IN | CLK\_SYS | Imaginary Sample Data input, IEEE-754 single precision |
| VALID | IN | CLK\_SYS | Input data valid.indicator  ‘1’ = valid |
| SOF | IN | CLK\_SYS | Start of Frame indicator |
| EOF | IN | CLK\_SYS | End of Frame indicator |
| READYOUT | OUT | CLK\_SYS | Ready indicator |
| **CONV Data to DDRIF** |  |  |  |
| DDR\_DATAOUT[ddr\_g\*512-1:0] | OUT | CLK\_SYS | DDR Data output |
| DDR\_ADDROUT[25:0] | OUT | CLK\_SYS | DDR Address output |
| DDR\_VALIDOUT | OUT | CLK\_SYS | Output DDR data valid  ‘1’=valid |
| DDR\_WAITREQ | IN | CLK\_SYS | DDR Wait Request  ‘1’=wait |
| **CONV Control and Done Return** |  |  |  |
| CONV\_TRIGGER | IN | CLK\_SYS | A 0 > 1 transition resets all counters in the CONV module so it is ready to commence processing a DM. |
| CONV\_ENABLE | IN | CLK\_SYS | Enable the CONV processing. The CONV module only processes when CONV\_ENABLE = 1. If CONV\_ENABLE is set to 0 the processing in CONV halts and only re-commences from its current position when CONV\_ENABLE is set to 1. |
| CONV\_DONEOUT | OUT | CLK\_SYS | Indication that the CONV module has completed processing the DM. CONV\_DONE shall pulse for a single cycle when the processing has completed. |
| MCREADYOUT | OUT | CLKSYS | Indication that diagnostic FFT data is ready to be read via the MCI. After CONV\_ENABLE is set to 0 and a page from FFT has been stored, then MCREADYOUT is set to 1. MCREADYOUT remains cleared when CONV\_ENABLE is set to 1. After a 0 > 1 transition on CONV\_TRIGGER, a new page from FFT must be stored before MCREADYOUT is raised. |
| OVERLAP\_SIZE[abits\_g-1:0] | IN | CLK\_SYS | The size of the overlap in the “Overlap-Save” convolution. |
| FOP\_NUM(fop\_num\_bits\_g-1:0) | IN | CLK\_SYS | The number of frequency bins in a fop. |
| IFFT\_LOOP\_NUM[ifft\_loop\_bits\_g:0] | IN | CLK\_SYS | The number of times the CONV module loops round its instantiated IFFTs.  This defines the “y-axis” (frequency derivative) of the FOP.  Number of filters in FOP y-axis =  (IFFT\_LOOP\_NUM x ifft\_g x 2) + 1  Where ifft\_g is the generic applied to the CONV module to define the number if instantiated IFFTs. |
| PAGE\_START[25:0] | IN | CLK\_SYS | DDR memory start address. |
| **Global Clock/Resets** |  |  |  |
| CLK\_SYS | IN | - | Core System Clock |
| RST\_SYS\_N | IN | CLK\_SYS | Asynchronous Logic Reset  ‘0’=Reset |
| CLK\_MC | IN | - | Micro Clock |
| RST\_MC\_N | IN | CLK\_MC | Asynchronous Logic Reset  ‘0’=Reset |

Table 8 - Interface Specifications

# MCI Memory Mapped Interface

## RAM: FILTER\_COEFFICIENTS

Array, size 42x1024, indexed by 'filter' (0 to 41) and 'word' (0 to 1023). Address range 0x000000 to 0x053FFF.

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 \* 8192 + word \* 8.

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

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| 0x000004\* | 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 = 0x000004 + filter \* 8192 + word \* 8.

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 0x080000 to 0x081FFF.

Register: FFT\_RE

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| 0x080000\* | 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 = 0x080000 + smpl \* 8.

Register: FFT\_IM

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Address | Mode | 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |
| 0x080004\* | 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 = 0x080004 + smpl \* 8.

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 |
| 0x0A0000 | 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 |
| 0x0C0000 | 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 |
| 0x0C0004 | 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 |
| 0x0C0008 | 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'

# Design Parameterisation

The CONV module has the following generic parameters:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Generic | Type | Value | Nominal | Description |
| ddr\_g | natural | 1, 2, 3 | 1 | Number of external DDR of interfaces |
| fop\_num\_bits\_g | natural | >=1 | 23 | Number of bits for fop count |
| ifft\_g | natural | >=1 | 7 | Number of filter instance pairs |
| ifft\_loop\_g | natural | >=1 | 6 | Number of filter iterations |
| ifft\_loop\_bits\_g | natural | log2(ifft\_loop\_g) | 3 | Number of bits for iteration count |
| fft\_g | natural | 8, 16, 32, 64, 128, 256, 512, 1024 | 1024 | FFT points |
| abits\_g | natural | log2(fft\_g) | 10 | Number of bits for FFT points |

Table 9 - Generic definitions

# References

|  |  |
| --- | --- |
| [Agilex Handbook] | Intel Arria 10 Core Fabric and General Purpose  I/Os Handbook |

# Abbreviations and Acronyms

|  |  |
| --- | --- |
| AXI |  |
| DM | Dispersion Measure |
| DMA | Direct Memory Access |
| DDR | Double Data Rate |
| EOF | End Of Frame |
| FDAS | Fourier Domain Acceleration Search |
| FIR | Finite Impulse Response (filter) |
| FFT | Fast Fourier Transform |
| FOP | Filter Output Plane |
| FPGA | Field Programmable Gate Array |
| IEEE | Institute of Electrical Engineers |
| IFFT | Inverse Fast Fourier Transform |
| IP | Intellectual Property |
| MCI | Micro Controller Interface |
| MSI-X | Extended Message Signalled Interrupt |
| PCIe | Peripheral Component Interconnect Express |
| SDRAM | Synchronous Dynamic RAM |
| SOF | Start Of Frame |

# : IEEE-754 single precision

Most significant bit is sign bit (S), 0 for positive, 1 for negative.

Following 8-bits represent exponent (E).

Remaining 23-bits represent fraction (F).

|  |  |  |
| --- | --- | --- |
| 31 | 30 23 | 22 0 |
| S | Exponent (E) | Fraction (F) |

Normalised form (E /= 0)

Implicit leading 1.

Exponent has bias of 127, so that actual exponent is E-127

Gives: 1.F x 2(E-127)

DeNormalised form (E=0)

No implicit 1.

Can represent zero.

Actual exponent value is -126

Gives: 0.F x 2-126

# : FFT Overlap-Save Method of Convolution

CONV module is supplied with data preformatted with overlapping data.

During the first run, the overlap portion is filled with zeros.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |
| Input | zeros |  |  |  |  |  |  |  |  |  |  |
| data | └───────┬─────┘ | | |  |  |  |  |  |  |  |  |
|  |  |  | └───────┬─────┘ | | |  |  |  |  |  |  |
| Run 1 | zeros |  | new |  |  |  |  |  |  |  | |
|  |  |  |  |  |  |
| Run 2 |  |  | overlap |  | new |
|  |  |  |  |  |  |
| Run 3 |  |  | overlap |  | new |
|  |  |  |  |  |  |
| Run 4 |  |  | overlap |  | new |
|  |  |  |  |  | |
| Run 5 |  |  | overlap |  | new |
|  |  |  |  |  |  |  |  |  |  |  |  |
| Output data | |  | 1 |  | 2 |  | 3 |  | 4 |  | 5 |

Figure B-1 Input-output data format

The output data shows the portion of processed data output by CONV module for each run.

# : Intel Agilex Variable Precision DSP Blocks

As detailed in [Agilex Handbook].



Figure C-1 Intel Agilex Variable Precision DSP Blocks





Figure C-2 Complex Multiplier

# : Intel Agilex IP Generation

## FFT IP

IP Variation Name: fft1024

The FFT is generated using the Quartus Prime IP Catalog.

The “IP Catalog” shall be used with the following selected:-

DSP > Transforms > Unified FFT > FFT Intel FPGA IP

This launches IP Platform Designer” which then asks for a name for the IP Variation which shall be entered as “fft1024”

The settings are as follows:-

**Core Parameters**

**Inverse FFT:** OFF

**Bit-reversed Input:** OFF

**Log2(Size):** 10

**Datatypes:** Floating Point

**Floating Point Type:** Single

**Reset polarity:** Active Low

**Signal Widths**

**Output width:** 32 (greyed out)

**Generation Parameters**

**Generate Global Enable Signal:** OFF

**Generate Software Model:** OFF

**Frequency Target:** Set this to the core clock frequency

**Device family:** Agilex (greyed out)

**Speed Grade:** 2 (greyed out)

This populates all the necessary FFT settings. This can then be saved and the HDL can be generated (click “Generate HDL” button) ensuring that the selected language is VHDL.

## IFFT IP

IP Variation Name: ifft1024

The IFFT is generated using the Quartus Prime IP Catalog.

The “IP Catalog” shall be used with the following selected:-

DSP > Transforms > Unified FFT > FFT Intel FPGA IP

This launches IP Platform Designer” which then asks for a name for the IP Variation which shall be entered as “ifft1024”

The settings are as follows:-

**Core Parameters**

**Inverse FFT:** ON

**Bit-reversed Input:** ON

**Log2(Size):** 10

**Datatypes:** Floating Point

**Floating Point Type:** Single

**Reset polarity:** Active Low

**Signal Widths**

**Output width:** 32 (greyed out)

**Generation Parameters**

**Generate Global Enable Signal:** OFF

**Generate Software Model:** OFF

**Frequency Target:** Set this to the core clock frequency

**Device family:** Agilex (greyed out)

**Speed Grade:** 2 (greyed out)

This populates all the necessary IFFT settings. This can then be saved and the HDL can be generated (click “Generate HDL” button) ensuring that the selected language is VHDL.

## Floating Point Multiplier IP

IP Variation Name: mult\_fp\_co

The Floating Point Multiplier is generated using the Quartus Prime IP Catalog.

The “IP Catalog” shall be used with the following selected:-

DSP > Primitive DSP > Native Floating Point DSP Intel Agilex FPGA IP

This launches IP Platform Designer” which then asks for a name for the IP Variation which shall be entered as “mult\_fp\_co” (floating point multiplier with chain out).

The settings are as follows:-

**General Tab**

**Operation Mode**

**Choose the operation mode:** fp32\_mult

**Enable fp32\_chainin:** No

**Enable fp32\_chainout:** Yes

**FP32 Operation**

**Perform subtraction in fp32\_adder:** No

**FP16 Representation/Operation**

**Select the mode for fp16:** FLUSHED

**Select the width size for fp16 (omly for bfloat16 mode):** 16

**Perform subtraction in fp16\_adder:** No

**Exception Flag**

**Enable Exception Flag:** No

**Registers Tab**

**Clear Signal Setting**

**Type of clear signal:** sclr

**Enable clr0 signal for input registers:** Yes

**Enable clr1 for output and pipeline registers:** Yes

**Input Registers**

**Enable for input “accumulate”:** no\_reg

**Enable for input “fp32\_adder\_a”:** no\_reg

**Enable for input “fp32\_adder\_b”:** no\_reg

**Enable for input “fp32\_mult\_a”:** ena0

**Enable for input “fp32\_mult\_b”:** ena0

**Enable for input “fp16\_mult\_input”:** no\_reg

**Output Registers**

**Enable output register:** ena0

**Pipeline Registers**

**Enable “accum\_adder”register:** no\_reg

**Enable “adder\_input”register:** no\_reg

**Enable “adder\_pl”register:** no\_reg

**Enable “fp32\_adder\_a\_chainin\_pl”register:** no\_reg

**Enable “accum\_pipeline”register:** no\_reg

**Enable “mult\_pipeline”register:** no\_reg

**Enable “fp32\_adder\_a\_chainin\_2nd\_pl”register:** no\_reg

**Enable “accum\_2nd\_pipeline”register:** no\_reg

**Enable “mult\_2nd\_pipeline”register:** no\_reg

This populates all the necessary Floating Point Multiplier settings. This can then be saved and the HDL can be generated (click “Generate HDL” button) ensuring that the selected language is VHDL.

## Floating Point Multiplier with Adder IP

IP Variation Name: multadd\_fp\_ci

The Floating Point Multiplier with Adder is generated using the Quartus Prime IP Catalog.

The “IP Catalog” shall be used with the following selected:-

DSP > Primitive DSP > Native Floating Point DSP Intel Agilex FPGA IP

This launches IP Platform Designer” which then asks for a name for the IP Variation which shall be entered as “multadd\_fp\_ci” (floating point multiplier with adder with chain in).

The settings are as follows:-

**General Tab**

**Operation Mode**

**Choose the operation mode:** fp32\_mult\_add

**Enable fp32\_chainin:** Yes

**Enable fp32\_chainout:** No

**FP32 Operation**

**Perform subtraction in fp32\_adder:** No

**FP16 Representation/Operation**

**Select the mode for fp16:** FLUSHED

**Select the width size for fp16 (omly for bfloat16 mode):** 16

**Perform subtraction in fp16\_adder:** No

**Exception Flag**

**Enable Exception Flag:** No

**Registers Tab**

**Clear Signal Setting**

**Type of clear signal:** sclr

**Enable clr0 signal for input registers:** Yes

**Enable clr1 for output and pipeline registers:** Yes

**Input Registers**

**Enable for input “accumulate”:** no\_reg

**Enable for input “fp32\_adder\_a”:** no\_reg

**Enable for input “fp32\_adder\_b”:** no\_reg

**Enable for input “fp32\_mult\_a”:** ena0

**Enable for input “fp32\_mult\_b”:** ena0

**Enable for input “fp16\_mult\_input”:** no\_reg

**Output Registers**

**Enable output register:** ena0

**Pipeline Registers**

**Enable “accum\_adder”register:** no\_reg

**Enable “adder\_input”register:** ena0

**Enable “adder\_pl”register:** no\_reg

**Enable “fp32\_adder\_a\_chainin\_pl”register:** no\_reg

**Enable “accum\_pipeline”register:** no\_reg

**Enable “mult\_pipeline”register:** no\_reg

**Enable “fp32\_adder\_a\_chainin\_2nd\_pl”register:** no\_reg

**Enable “accum\_2nd\_pipeline”register:** no\_reg

**Enable “mult\_2nd\_pipeline”register:** ena0

This populates all the necessary Floating Point Multiplier with Adder settings. This can then be saved and the HDL can be generated (click “Generate HDL” button) ensuring that the selected language is VHDL.

## Floating Point Multiplier with Subtraction IP

IP Variation Name: multsub\_fp\_ci

The Floating Point Multiplier with Subtraction is generated using the Quartus Prime IP Catalog.

The “IP Catalog” shall be used with the following selected:-

DSP > Primitive DSP > Native Floating Point DSP Intel Agilex FPGA IP

This launches IP Platform Designer” which then asks for a name for the IP Variation which shall be entered as “multsub\_fp\_ci” (floating point multiplier with subtraction with chain in).

The settings are as follows:-

**General Tab**

**Operation Mode**

**Choose the operation mode:** fp32\_mult\_add

**Enable fp32\_chainin:** Yes

**Enable fp32\_chainout:** No

**FP32 Operation**

**Perform subtraction in fp32\_adder:** Yes

**FP16 Representation/Operation**

**Select the mode for fp16:** FLUSHED

**Select the width size for fp16 (only for bfloat16 mode):** 16

**Perform subtraction in fp16\_adder:** No

**Exception Flag**

**Enable Exception Flag:** No

**Registers Tab**

**Clear Signal Setting**

**Type of clear signal:** sclr

**Enable clr0 signal for input registers:** Yes

**Enable clr1 for output and pipeline registers:** Yes

**Input Registers**

**Enable for input “accumulate”:** no\_reg

**Enable for input “fp32\_adder\_a”:** no\_reg

**Enable for input “fp32\_adder\_b”:** no\_reg

**Enable for input “fp32\_mult\_a”:** ena0

**Enable for input “fp32\_mult\_b”:** ena0

**Enable for input “fp16\_mult\_input”:** no\_reg

**Output Registers**

**Enable output register:** ena0

**Pipeline Registers**

**Enable “accum\_adder”register:** no\_reg

**Enable “adder\_input”register:** ena0

**Enable “adder\_pl”register:** no\_reg

**Enable “fp32\_adder\_a\_chainin\_pl”register:** no\_reg

**Enable “accum\_pipeline”register:** no\_reg

**Enable “mult\_pipeline”register:** ena0

**Enable “fp32\_adder\_a\_chainin\_2nd\_pl”register:** no\_reg

**Enable “accum\_2nd\_pipeline”register:** no\_reg

**Enable “mult\_2nd\_pipeline”register:** ena0

This populates all the necessary Floating Point Multiplier with Subtraction settings. This can then be saved and the HDL can be generated (click “Generate HDL” button) ensuring that the selected language is VHDL.

END OF DOCUMENT