**OpenCPI HDL Application Workers Interface Specification**

Authors:

Shepard Siegel, Atomic Rules LLC ([Shepard.Siegel@atomicrules.com](mailto:Shepard.Siegel@atomicrules.com))

Jim Kulp, Mercury Federal Systems, Inc. ([jkulp@mercfed.com)](mailto:jkulp@mercfed.com)

*Revision History*

|  |  |  |  |
| --- | --- | --- | --- |
| **Revision** | **Description of Change** | **By** | **Date** |
| 1.01 | Creation from Atomic Rules, LLC, OpenCPI-FRMAM v29 | jkulp | 2010-07-01 |
| 1.02 | Added DDCWorker | Ssiegel | 2010-08-08 |
| 1.03 | DDCWorker prop change | Ssiegel | 2010-08-09 |

Table of Contents

1 References 4

2 Overview 5

3 GCD Worker 6

3.1 Function performed 6

3.2 Configuration Properties 6

3.3 Using the GCD Worker 6

3.4 Other Features 6

4 Stream-Message Adapter (SMAdapter) Worker 7

4.1 Function performed 7

4.2 Configuration Properties 8

4.3 Using the SMAdapter Worker 8

5 Delay Worker 10

5.1 Function performed 10

5.2 Configuration Properties 10

5.3 Using the Delay Worker 11

6 FrameGate Worker 12

6.1 Function Performed 12

6.2 Configuration Properties 12

6.3 Using the FrameGate Worker 12

7 Bias Worker 13

7.1 Function performed 13

7.2 Configuration Properties 13

7.3 Using the Bias Worker 13

8 PSD Worker 14

8.1 Function performed 14

8.2 Configuration Properties 14

8.3 Using the PSD Worker 14

9 Splitter2x2 Worker 16

9.1 Function performed 16

9.2 Configuration Properties 16

9.3 Using the Splitter2x2 Worker 17

# References

This document depends on several others. Primarily, it depends on the “OpenCPI Generic Authoring Model Reference Manual”, which describes concepts and definitions common to all OpenCPI authoring models, and the “OpenCPI HDL Authoring Model Reference” which describes how HDL workers must be written.

|  |  |  |
| --- | --- | --- |
| **Title** | **Published By** | **Link** |
| OpenCPI Generic Authoring Model Reference | [OpenCPI](http://opencpi.org) | Public URL: <http://www.opencpi.org/doc> |
| OpenCPI HDL Authoring Model Reference | [OpenCPI](http://opencpi.org) | Public URL: <http://www.opencpi.org/doc> |

# Overview

This document describes the HDL application workers in the OpenCPI component library. A future version of this document will be completely heterogeneous, meaning that it will describe all components, and then all the workers (implementations) in any authoring model for each component.

Configuration properties shaded in yellow are meant only for debugging and can be “compiled out” when not desired to save resources.

As with all HDL workers, all application workers must follow the control pattern defined in the HDL Authoring Model Reference [AMR]. First, workers are “taken out of reset”, by deasserting the OCP MReset\_n signal on its WCI OCP Slave interface. Next, if supported by the worker, the ***Initialize*** control operation is performed. Finally, the ***Start*** control operation is performed to cause the worker to begin operating (enter the operational state).

# GCD Worker

## Function performed

This worker has no data plane. It computes the GCD of the arguments (in configuration properties) provided as configuration properties. It is primarily used for testing.

## Configuration Properties

This section describes the configuration properties of the GCD worker.

|  |  |  |
| --- | --- | --- |
| **Property Offset** | **Access** | **Description** |
| +0x0000 | RW | GCD Argument “r0” |
| +0x0004 | RW | GCD Argument “r4” |
| +0x0008 | RO | GCD Result |
| +0x000C | RO | GCD Result Ready (bit0) |
| +0x0010 | RO | Worker Ordinal Number |
| +0x0014 | RO | Counter |
| +0x0018 | RW | B18 |
| +0x0018 | RW | B19 |
| +0x0018 | RW | B1A |
| +0x0018 | RW | B1B |
| +0x001C | RW | <No Response Generated> |
| +0x0020 | RW | Bit[31:1] - Reserved  Bit[0] - sFlagState |

## Using the GCD Worker

Write GCD arguments to r0 and r4 registers. A write to either property r0 or r4 will initiate GCD calculation when in the Operating state. The GCD Result Ready register will indicate with b0 True when the result is ready. When result is ready, read the GCD Result register. The GCD Result register will return the code 0xBADBADBA if there is not a valid GCD result ready.

## Other Features

The counter at location 0x14 will count when the control state is “Operating” and will reset when the control state is “Initialized”.

Four discrete byte-sized registers with the power-on reset value equal to their address are packed at the DWORD 0x0018. Individual byte or word writes may be tested by writing these locations.

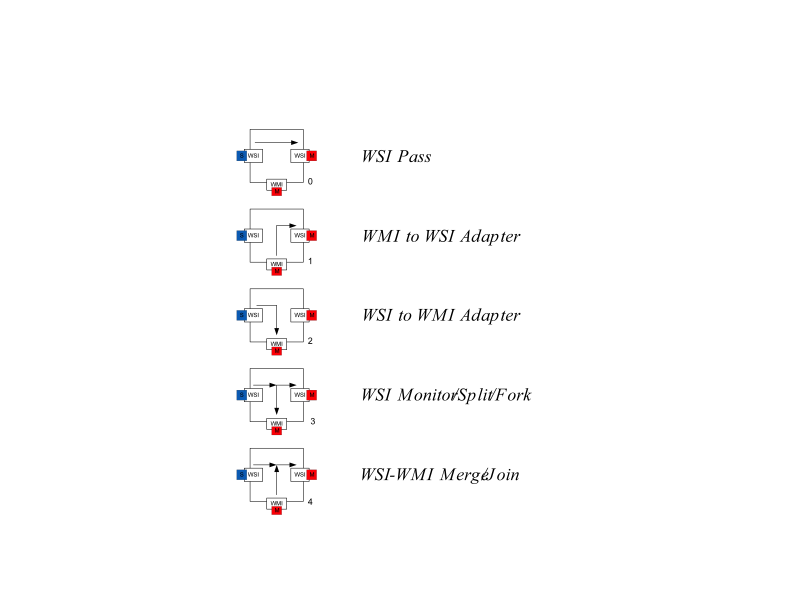
The location 0x001C will (on purpose) never generate a response and may be used to test the timeout handling.

The location 0x0020 allows WCI SFlag to be set or cleared at Bit-0

# Stream-Message Adapter (SMAdapter) Worker

## Function performed

This worker has one WSI input, one WSI output, and one WMI port that can be input or output. Depending on a configuration property, it routes data between these ports in several ways described below. The most basic function of this worker is as it is named, namely, to adapt between WSI and WMI, in either direction (modes 1 and 2). But it also can perform some splitting and joining functions too. All 5 modes are shown below, with the “mode number” to the lower right of the picture.



## Configuration Properties

This section describes the configuration properties of the SMAdapter

|  |  |  |  |
| --- | --- | --- | --- |
| **Property Offset** | **Property Name** | **Access** | **Description** |
| +0x0000 | smaCtrl | RW | Control Register (see text) |
| +0x0004 | mesgCount | RO | WMI Rolling 32b message count |
| +0x0008 | abortCount | RO | WMI Rolling 32b abort count (imprecise) |
| +0x000C |  | RO | Reserved |
| +0x0010 | thisMesg | RO | WSI-S {opcode[7:0], mesgLength[23:0]} |
| +0x0014 | lastMesg | RO | WSI-S {opcode[7:0], mesgLength[23:0]} |
| +0x0018 | portStatus | RO | Port Status: WSI-S[15:8], WSI-M[7:0] |
| +0x001C | 0 | RO |  |
| +0x0020 | pMesgCount | RO | WSI-S |
| +0x0024 | iMesgCount | RO | WSI-S |
| +0x0028 | tBusyCount | RO | WSI-S |
| +0x002C | pMesgCount | RO | WSI-M |
| +0x0030 | iMesgCount | RO | WSI-M |
| +0x0034 | tBusyCount | RO | WSI-M |

The control register at offset 0x0 can take these values:

|  |  |  |
| --- | --- | --- |
| **Bits** | **Name** | **Description** |
| 31:6 |  | Reserved |
| 5 | impWsiM | 0=Normal; 1=Force Imprecise on WSI-M egress |
| 4 | nixWsiM | 0=Normal; 1=No Put to WSI-M port (/dev/null) |
| 3:0 | mode | 0 = Pass WSI-S to WSI-M  1 = WMI to WSI-M  2 = WSI-S to WMI  3 = Fork WSI-S to both WSI-M and WMI  4 = Merge WSI-S and WMI to WSI-M  5:15 = Reserved |

## Using the SMAdapter Worker

The Stream-Message Adapter (SMAdapter) implements one of the five adapter topologies shown above. A control register set prior to START determines the module’s function. Six RO diagnostic registers measure the flow of messages past each port.

The following read-only (instrumentation) properties are initialized to zero at reset and on an INITIALIZE control-op. They are available for each of the three ports WMI, WSI-S, and WMI-M individually

The \_MesgCnt registers maintain a 32b unsigned rolling message count. This count is updated at message completion and indicate “how many messages have moved through this port”.

The lastMesg and thisMesg registers sample the opcode and mesgLength of the last message processed. This data is updated as soon as new information is available; but cannot be relied on to determine if a particular message has completed ingress or egress from any port. The mesgLength is given in Bytes.

# Delay Worker

## Function performed

This worker has a WSI input and a WSI output, and passes input messages to the output after a certain amount of delay, counted either in bytes or clock periods.

## Configuration Properties

This section describes the configuration properties of the Delay worker

|  |  |  |  |
| --- | --- | --- | --- |
| **Property Offset** | **Property Name** | **Access** | **Description** |
| +0x0000 | dlyCtrl | RW | Bits[3:0] 4’h0=Bypass (no delay); 4’h7=Delay |
| +0x0004 | dlyHoldoffBytes | RW | Number of bytesWritten to holdoff before read |
| +0x0008 | dlyHoldoffCycles | RW | Number of cyclesPassed to holdoff before read |
| +0x000C | mesgWtCount | RO | Incremented by Message Finalize (WSI-S) |
| +0x0010 | mesgRdCount | RO | Incremented by Message Body Response (WSI-M) |
| +0x0014 | bytesWritten | RO | Saturating WSI-S Monitor |
| +0x0018 | portStatus | RO | 0[31:24], WMEMI-M[23:16], WSI-S[15:8], WSI-M[7:0] |
| +0x001C | 0 | RO |  |
| +0x0020 | WSI-S: pMesgCount | RO |  |
| +0x0024 | WSI-S: iMesgCount | RO |  |
| +0x0028 | WSI-S: tBusyCount | RO |  |
| +0x002C | WSI-M: pMesgCount | RO |  |
| +0x0030 | WSI-M: iMesgCount | RO |  |
| +0x0034 | WSI-M: tBusyCount | RO |  |
| +0x0038 | wmemiWrReq | RO | WMemI Write Requests Issued (LSB=16B) |
| +0x003C | wmemiRdReq | RO | WMemI Read Requests Issued (LSB=16B) |
| +0x0040 | wmemiRdResp | RO | WMemI Read Responses Received (LSB=16B) |
| +0x0044 | dlyWordsStored | RO | Accumulator Tracking Words Stored (LSB=16B) |
| +0x0048 | dlyReadCredit | RO | Number of outstanding Read Credits (LSB=16B) |
| +0x004C | dlyWAG | RO | Write Address Generator (LSB = 16B) |
| +0x0050 | dlyRAG | RO | Read Address Generator (LSB= 16B) |
| +0x0054 | dlyMaxReadCredit | RW | 4=Default (design debug use only) |
| +0x0058 | dlyRdOpZero | RO | Rolling 32b count of Read Side Meta opcode==0 |
| +0x005C | dlyRdOpOther | RO | Rolling 32b count of Read Side Meta opcode!=0 |

## Using the Delay Worker

The Delay worker delays messages by storing them in a memory-based circular buffer. Incoming messages are stored as they arrive. Outgoing messages do not begin until both the dlyHoldoffBytes and dlyHoldoffCycles constraints have been satisfied. The cycle counter, (nominally counting 125 MHz / 8nS cycles), begins counting when the first message arrives. The dlyHoldoffBytes includes the bytes needed to store (in a delay buffer), the opcode and length of messages.

For example, to delay a message stream by 1 second, set the delayHoldoffCycles to 125E6; and leave dlyHoldoffBytes at its default of 0.

To use the Delay worker, de-assert reset, and then issue the INITIALIZE and START control operations in sequence.

The Delay worker delay function ***requires*** a WMemI interface with memory, typically provided by a DRAM Device Worker. It is essential to the message delay function that the WMemI service provider be successfully initialized and started ***before*** using the Delay worker for delay.

The yellow-lined, read-only instrumentation properties provide details of the internal state. These include: Multiplexing the metadata and message streams into a sequence of 16B words and write these words to DRAM memory. When the read criterion has been satisfied, begin reading these 16B words from DRAM memory and splitting them back into metadata and message streams.

# FrameGate Worker

## Function Performed

This worker has an input WSI and an output WSI and passes messages from input to output, gating (dropping) some of the messages according to the control settings. It also rebuffers input messages (frames) into output frames of a specified size.

## Configuration Properties

This section describes the configuration properties of the FrameGate

|  |  |  |  |
| --- | --- | --- | --- |
| **Property Offset** | **Property Name** | **Access** | **Description** |
| +0x0000 | frameGateStatus | RO |  |
| +0x0004 | frameGateCtrl | RW |  |
| +0x0008 | frameSize | RW |  |
| +0x000C | gateSize | RW |  |
| +0x0010 |  | RO | WSI-S[15:8], WSI-M[7:0] |
| +0x0014 | WSI-S: pMesgCount | RO |  |
| +0x0018 | WSI-S: iMesgCount | RO |  |
| +0x001C | WSI-S: tBusyCount | RO |  |
| +0x0020 | WSI-M: pMesgCount | RO |  |
| +0x0024 | WSI-M: iMesgCount | RO |  |
| +0x0028 | WSI-M: tBusyCount | RO |  |
| +0x002C | Op0MesgCnt | RO |  |
| +0x0030 | otherMesgCnt | RO |  |

## Using the FrameGate Worker

The FrameGate worker *frames* and *gates* an incoming stream of imprecise messages into precise sized messages of frameSize. It first *frames* the data with minimal buffering and latency by counting frameSize words of opcode-0 data and moves them from WSI-S to WSI-M. Then it *gates* off the input by discarding gateSize opcode-0 words on the WSI-S input. Messages that are not opcode 0 are passed directly from input to output.

The yellow-lined, read-only instrumentation properties provide details of the internal state. These include the usual monitor and extended-monitor functions for WSI ports; as well as rolling counts of the number of opcode 0 and other opcodes counted.

# Bias Worker

## Function performed

The BiasWorker accepts a message from its WSI input and passes it to its WSI output. It transforms all message data by adding the 32b UInt32 “biasValue” to each 4B element received. Both the message data and the biasValue are handled as 32b unsigned integers (unsigned 32.0 format). A biasValue of zero (the default) lets the message pass unaltered.

## Configuration Properties

This section describes the configuration properties of the BiasWorker

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Property Offset** | **Property Name** | **Access** | | **Description** |
| +0x0000 | biasValue | RW | | Bias (offset) to add to each 4B message element. 32b UInt |
| +0x0004 | controlReg | RW | |  |
| +0x0008 | messagePush | | RO | Rolling 32b count of doMessagePush firing |
| +0x000C | lastOpcode0 | | RO | Data associated with last Opcode 0 |
| +0x0018 | portStatus | | RO | Unused[31:16]; Port Status: WSI-S[15:8], WSI-M[7:0] |
| +0x0020 | pMesgCount | | RO | WSI-S |
| +0x0024 | iMesgCount | | RO | WSI-S |
| +0x0020 | pMesgCount | | RO | WSI-M |
| +0x0024 | iMesgCount | | RO | WSI-M |

## Using the Bias Worker

To start the Bias Worker, de-assert reset, and then issue the INITIALIZE and START control operations in sequence.

# PSD Worker

## Function performed

The PSD worker calculates an approximation of the Power Spectral Density function of 4K input sample frames. Only the PsdPass, PsdPrecise, and PsdFFT modes are supported.

## Configuration Properties

This section describes the configuration properties of the PSD

|  |  |  |  |
| --- | --- | --- | --- |
| **Property Offset** | **Property Name** | **Access** | **Description** |
| +0x0000 | psdStatus | RO | Bit0=hasDebugLogic |
| +0x0004 | psdCtrl | RW | Bits[1:0]={0=PsdPass;1=PsdPrecise;2=PsdFFT;3=PSDSpare} |
| +0x0008 |  |  |  |
| +0x000C |  |  |  |
| +0x0010 |  | RO | WSI-S[15:8], WSI-M[7:0] |
| +0x0014 | WSI-S: pMesgCount | RO |  |
| +0x0018 | WSI-S: iMesgCount | RO |  |
| +0x001C | WSI-S: tBusyCount | RO |  |
| +0x0020 | WSI-M: pMesgCount | RO |  |
| +0x0024 | WSI-M: iMesgCount | RO |  |
| +0x0028 | WSI-M: tBusyCount | RO |  |
| +0x002C | fftFrameCounts | RO | Bits[31:16]=framesLoaded, Bits[15:0]=framesUnloaded |

## Using the PSD Worker

Setting PsdFFT yields the desired PSD function by approximately the steps described here:

1. *Precise Frame Formatting, WsiToPrecise*  
   Format an imprecise-burst of time-domain samples from a prior stage into a precisely-sized message buffer.
2. *Windowing of Time-Domain Data*  
   Prior to the forward FFT, the real input data is windowed with a raised-cosine (Hamming) function to balance spectral leakage against sensitivity in the finite length transform to follow.
3. *Gear Shift 125MHz to 250 MHz*  
   Two samples per cycle at 125 MHz are converted to One sample per cycle at 250 MHz
4. *8K Streaming, Pipelined, Forward FFT, Natural Ordered Output*  
   Implemented by wrapping a FPGA Vendor Core (eg fft-v5-4k-stream-natural) Core operates at 250 MHz nominal. The output of the FFT core is a naturally ordered (F-bin 0, 1, 2, ..., 4095) vector of fixed-point complex numbers.
5. *Magnitude Approximation*  
   The magnitude of each complex number is approximated by folding all data into the first-quadrant |i|+|q| and then using a technique described by Lyons to estimate magnitude to within 1 dB. The input to this stage are 250 MSPS complex numbers; the output are 250 MSPS unsigned magnitudes. Each vector of 4096 magnitudes is termed a "Power Vector Frame".
6. *Power Vector Frame Averaging*  
   Each Power Vector Frame is added into a power vector frame accumulator until the four frames have been accumulated. Then the result is shifted down by two to yield a N=4 Power Vector Frame Average. This is implemented at 250 MHz.
7. *Output Formatting*  
   The PSD Output is then a 8KB message containing the 4096 16b unsigned entries of the N=4 averaged power vector frame. This step includes the Gear Shift from 16b 250 MHz (250 MSPS) to 32b 125 MHz (250 MSPS)

The yellow-lined, read-only instrumentation properties provide details of the internal state. These include the usual monitor and extended-monitor functions for WSI ports. The fftFrameCounts expose two 16b rolling counts the reconcile the loading and unloading of complete frames to the FFT

# Splitter2x2 Worker

## Function performed

The Splitter2x2 is a 2x2 crossbar. There are two WSI Slave inputs S0 and S1; and two WSI Master outputs. Prior to starting the splitter, it must be programmed to specify which input’s messages are to be produced at each output.

## Configuration Properties

This section describes the configuration properties of the Splitter2x2

|  |  |  |  |
| --- | --- | --- | --- |
| **Property**  **Offset** | **Property Name** | **Access** | **Description** |
| +0x0000 |  |  |  |
| +0x0004 | splitCtrl | RW |  |
| +0x0008 |  |  |  |
| +0x000C |  |  |  |
| +0x0010 |  |  |  |
| +0x0014 |  |  |  |
| +0x0018 |  |  |  |
| +0x001C |  | RO | WSI-S0[31:24], WSI-S1[23:16],WSI-M0[15:8],WSI-M1[7:0] |
| +0x0020 | WSI-S0: pMesgCount | RO |  |
| +0x0024 | WSI-S0: iMesgCount | RO |  |
| +0x0028 | WSI-S1: pMesgCount | RO |  |
| +0x002C | WSI-S1: iMesgCount | RO |  |
| +0x0030 | WSI-M0: pMesgCount | RO |  |
| +0x0034 | WSI-M0: iMesgCount | RO |  |
| +0x0038 | WSI-M1: pMesgCount | RO |  |
| +0x003C | WSI-M1: iMesgCount | RO |  |

The splitCtrl property has the following bit fields:

|  |  |
| --- | --- |
| **Bit** | **Function** |
| 15 | Disable M1 output |
| 8 | Select M1 Source 0=S0, 1=S1 |
| 7 | Disable M0 output |
| 0 | Select M0 Source 0=S0, 1=S1 |

## Using the Splitter2x2 Worker

Prior to starting the splitter, it must be programmed by setting bits in the splitCtrl register. Note that the powerup default is a “split” function where messages arriving on S0 are sent out both outputs M0 and M1.

The yellow-lined, read-only instrumentation properties provide details of the internal state. These include the usual monitor and extended-monitor functions for WSI ports.

# DDC Worker

## Function performed

This worker is a four-channel Digital Down Converter (also called a “tuner”). There is one WSI Slave input; and one WSI Master output. The WSI input accepts a stream of real data, as from a single antenna. The WSI output, when the DDC is bypassed, is a replica of the input. The WSI output, when the DDC is enabled, contains the down-converted complex (I/Q) baseband of the four tuned channels.

## Configuration Properties

This section describes the configuration properties of the DDCWorker

|  |  |  |  |
| --- | --- | --- | --- |
| **Property**  **Offset** | **Property Name** | **Access** | **Description** |
| +0x0000 | ddcStatus | RO |  |
| +0x0004 | ddcCtrl | RW |  |
| +0x0008 |  |  |  |
| +0x000C |  |  |  |
| +0x0010 |  | RO | WSI-S[15:8],WSI-M[7:0] |
| +0x0014 | WSI-S: pMesgCount | RO |  |
| +0x0018 | WSI-S: iMesgCount | RO |  |
| +0x001C | WSI-S: tBusyCount | RO |  |
| +0x0020 | WSI-M: pMesgCount | RO |  |
| +0x0024 | WSI-M: iMesgCount | RO |  |
| +0x0028 | WSI-M: tBusyCount | RO |  |
| +0x002C | ambaWriteReqCnt | RO | Number of Write requests by AMBA3 to DDC core |
| +0x0030 | ambaReadReqCnt | RO | Number of Read requests by AMBA3 to DDC core |
| +0x0034 | ambaRespCount | RO | Number of Responses by AMBA3 from DDC core |
| +0x0038 | outFrameCount | RO | Number of 8KB output frames |
| +0x1000 –  0x2FFC | DDC  Control  Space | RW | See 4KB register map starting on page 19 of Xilinx DS766 (July 23, 2010) LogiCore IP DUC/DDC Compiler v1.0 |

The ddcStatus property has the following bit fields:

|  |  |
| --- | --- |
| **Bit** | **Function** |
| 7 | INT\_MISSINPUT |
| 6 | INT\_ERRPACKET |
| 5 | INT\_LOSTOUTPUT |
| 4 | INT\_DUCDDC |
| [3:1] | 0 |
| 0 | hasHardwareDebug |

The ddcCtrl property has the following bit fields:

|  |  |
| --- | --- |
| **Bit** | **Function** |
| 0 | Enable DDC |

## Using the DDC Worker

Prior to starting the DDC, it must be programmed by setting bits in the ddcCtrl register. Note that the powerup default has the DDC disabled (bypass).

The yellow-lined, read-only instrumentation properties provide details of the internal state. These include the usual monitor and extended-monitor functions for WSI ports.