|  |  |  |  |
| --- | --- | --- | --- |
| Revision history | | | |
| **Revision** | **Date** | **Description** | **Author** |
| v0.1 | 2019-03-07 | Initial version after first review of requirements | Soheil Bahrami |
| v0.2 | 2019-12-01 | Minor updates | Soheil Bahrami |
| v0.3 | 2020-06-17 | Interrupt structure updated as per Quantum global approach | Soheil Bahrami |
| v1.0 | 2020-06-19 | External start triggers added – PDA review version | Soheil Bahrami |
| v1.1 | 2020-06-25 | Updates after PDA review | Soheil Bahrami |
| v1.2 | 2020-08-12 | Updates after RS update (v0.9) | Soheil Bahrami |
| v1.3 | 2020-12-16 | Matching document to implementation | Soheil Bahrami |
|  |  |  |  |

Contents

[1 Introduction 4](#_Toc43389510)

[1.1 Document Purpose 4](#_Toc43389511)

[1.2 Architectural Considerations 4](#_Toc43389512)

[1.2.1 Summary 4](#_Toc43389513)

[1.2.2 Use cases 4](#_Toc43389514)

[1.2.3 Place in system 4](#_Toc43389515)

[2 Implementation 6](#_Toc43389516)

[2.1 Block Diagram 6](#_Toc43389517)

[2.2 Functional description 6](#_Toc43389518)

[2.2.1 Start/Stop mechanism 6](#_Toc43389519)

[2.2.1.1 Buffer size 7](#_Toc43389520)

[2.2.2 Arbitration 7](#_Toc43389521)

[2.2.3 Interrupts/Flags 7](#_Toc43389522)

[2.2.3.1 Address-triggered interrupt 8](#_Toc43389523)

[2.2.3.2 Pending Interrupt 8](#_Toc43389524)

[2.2.3.3 FIFO overflow interrupt 8](#_Toc43389525)

[2.2.3.4 Bus error interrupt 8](#_Toc43389526)

[2.2.3.5 AGC trigger 8](#_Toc43389527)

[2.2.4 Debugging features 8](#_Toc43389528)

[2.3 Register File 9](#_Toc43389529)

[2.3.1 APB Block integration 9](#_Toc43389530)

[2.4 Software Driver APIs 10](#_Toc43389531)

[3 Integration 11](#_Toc43389532)

[3.1 Clock and Reset 11](#_Toc43389533)

[3.2 IP-XACT 11](#_Toc43389534)

[3.3 PINNIG INTERFACE 12](#_Toc43389535)

[3.4 Database management? 14](#_Toc43389536)

[4 Memory Map 15](#_Toc43389537)

[4.1 Base region map 15](#_Toc43389538)

[4.2 Stream region map 16](#_Toc43389539)

[5 Abbreviations and terminology 18](#_Toc43389540)

[5.1 Referenced documents 19](#_Toc43389541)

[Figure 1 – Possible topologies in a system 5](#_Toc44064110)

[Figure 2 – Block diagram of WDMA 6](https://nxp1-my.sharepoint.com/personal/soheil_bahrami_nxp_com/Documents/Projects/WDMA/Documents/ida_wdma_ads_1v1.docx#_Toc44064111)

[Figure 3 – structure of generated interrupt register logic 8](#_Toc44064112)

[Table 1 – clock, control and sample signals 13](#_Toc44064296)

[Table 2 – AXI4 Interface signals 14](#_Toc44064297)

[Table 3 - APB3 interface signals 15](#_Toc44064298)

# Introduction

## Document Purpose

This document contains the architecture design specification of the WDMA IP which has been developed for use in BL-RFP products. The IP can be used to deliver data streams to the bus system capable of AXI4 protocol. The purpose of this document is to cover the architecture, design and integration aspects of the IP.

This implementation fulfills the requirements specification found in the reference section of this document.

## Architectural Considerations

### Summary

* Accepts streaming data from ‘*N\_CHANNEL’* independent sources with parameterizable data width
* Native AXI4 write interface with 32-bit address and parameterizable width data equal to the width of incoming data
* Internal design-time configurable FIFO for multiple clock operation and to compensate for data irregularities
* Dynamically configurable AXI burst size
* 32-bit APB interface for programming
* Round robin arbitration between all sources. Shared control block results in area efficient design
* Local generation of debug data
* Programmable interrupt generation based on the progress of the write pointer
* Interrupt generation at FIFO overflow/bus error

### Use cases

* Single/Multiple shot data capture for debugging and non-streaming standards
* Continuous data capture into circular buffer

### Place in system

Depending on the SoC requirements, one or more instances of WDMA can be placed in a subsystem. If the arbitration of parallel streams can be left to the bus interconnect, the WDMA can be instantiated multiple times with the *N\_CHANNEL* set to 1.

When one instance is used, the arbitration can be done internally where the round-robin scheme takes care of sending samples onto the bus one burst per stream after another. Care needs to be taken to specify the clock speeds in a way that all streams will have a fair chance of being served.

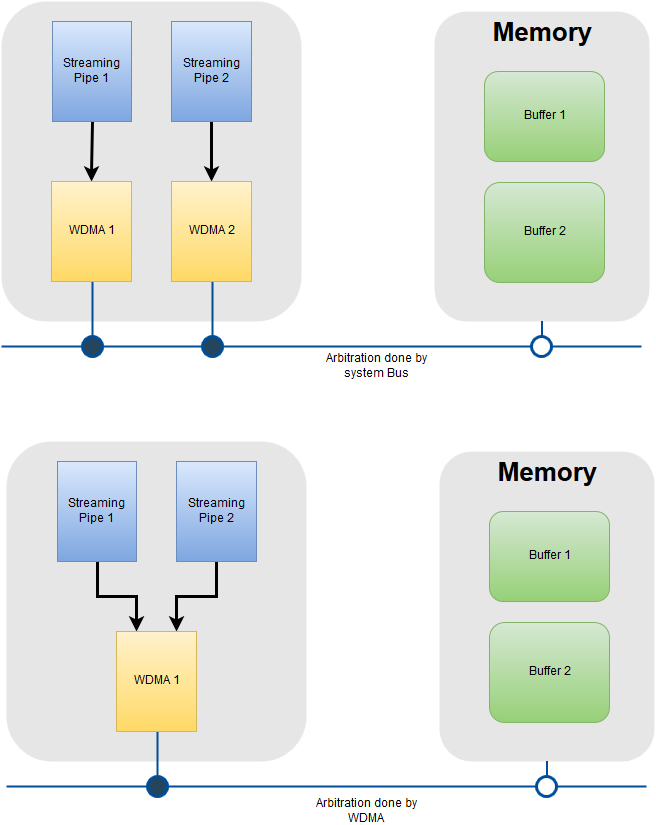


Figure 1 – Possible topologies in a system

# Implementation

## Block Diagram

Figure 2 – Block diagram of WDMA

## Functional description

### Start/Stop mechanism

Each stream can be controlled independently of the others. A buffer needs to be defined in the available SoC memory map by its start address and size. Upon setting the Enable bit, the corresponding buffer settings are read into the IP and delivery of incoming samples will commence. The buffers can be filled in circular mode or could be filled a defined number of times in “multi-shot” mode. This mode can be used to deliver a determined number of rounds of buffers without intervention of a processor.

Overwriting the buffer settings will not have an effect until the stream has been disabled and re-enabled. During the operation of the WDMA, the status of each stream can be monitored in terms of the address of the current AXI burst being written and the number of times the buffer has been filled since its enabling. Incrementing the current pointer happens when the burst starts sending samples onto the bus.

The bursts on the AXI interface are initiated only when enough samples have been gathered in the corresponding FIFO, therefore, the WDMA will not stall the bus once a burst has been initiated. However, if a slave pauses a burst, the transaction will take longer pending readiness of the slave.

To stop a stream, the enable bit must be set to zero. This will ensure that the FIFO of the corresponding stream will not receive any more samples. In case the FIFO has enough samples for a burst, even after disabling, the burst will be queued and sent.

After the last outstanding transaction has been finalized by receiving its corresponding BRESP signal from the interconnect, termination of a stream can be confirmed by checking the status bit of the stream in the status register. Note that status register will show enabled status if the stream is enabled but receives no samples.

It is not possible to pause the activity of WDMA to momentarily skip samples. The only way to stop samples to enter the FIFO is to de-assert the enable bit. After setting the enable bit, the FIFO will be reset.

#### External trigger

Each DMA channel can have a coordinated start with other blocks in the system. Typically, the preceding block that needs a synchronized start along with the WDMA. *start\_trigger\_out*:generates a pulse (in sample clock domain) that can be used by other blocks for a synchronized start.

#### Buffer size

Buffers can be defined with resolution of a single AXI word. If the length of a buffer is not multiple of the programmed burst size, the remaining samples will generate a smaller burst. Example:

*Burst length = 4 words, Buffer size = 17words 🡪 4 bursts with 4 beats + 1 single-word transfer*

### Arbitration

A Round-robin scheme takes care of arbitrating the access to the AXI bus from the multiple incoming streams. The round-robin counter is always enabled and checks for a signal from the FIFOs associated with each stream, one per clock cycle. As soon as enough samples for a burst have been gathered a flag is set by the FIFO which leads the round-robin to lock onto the corresponding stream and stop circulating. It stays locked until the bus is ready and the transfer of the burst has been completed. As soon as the writing of the beats are done, the round-robin lock gets lifted and the count will continue in the search of the next flag from the other FIFOs. A few points need to be considered in understanding this scheme to be able to define the right design parameters:

* The counter will not wait for a bus response, this will be tracked in an independent process to avoid delaying of the other streams.
* There is no queuing mechanism in place to reserve a slot for the next FIFO which may become ready while the round-robin is locked. Whichever FIFO is next on the rotary scheme, will be considered. For a *N\_CHANNEL* = N, this delay could potentially amount to *(N-2)\*(duration of a burst)*.
* Round-robin will ensure that once a burst has been sent by a certain stream, the turn will be given to the next potential stream even if the first FIFO becomes full in the meantime. This will ensure that all the streams will get a fair chance at the cost of possible overflow of the fast streams.
* For a configuration of N streams, round-robin will always count through 0 to N-1 to check for ready flags from FIFOs even if some of the streams have been disabled. This will mean an additional clock cycle delay per disabled stream in serving the active ones.

Care needs to be taken in guarantying sufficient bandwidth on the bus interface of the WDMA to ensure adequate throughput.

### FIFO

FIFOs are used at the input of each stream, independent of the others. They are used to synchronize the incoming samples for the Sample clock domain into the core (AXI) domain. They also provide buffering for burst generation and delay compensation.

FIFO depth for all streams is determined by the parameter *FIFO\_LEN\_LOG2* at design time. The chosen value needs to be at least as big as the maximum desired AXI burst-length plus a few more to act as buffer for delays. The fill state of the FIFO is available to the control logic and can be used to decide by the arbiter if enough samples are available for a burst.

Burst size is programmable through an APB register and can be set upon enabling a stream. The fill state of the FIFO will be compared to the programmed value to determine whether a burst is ready.

### Interrupts/Flags

Interrupts can be utilized by the processor to follow the state of each of the WDMA streams. The structure of interrupts is depicted in the diagram below. Software can artificially set the interrupt for testing purposes.

Interrupts must be enabled using the corresponding xxx\_enable\_set register. To test the interrupt, it can be manually triggered by setting the corresponding xxx\_sw\_set register. This is a self-clearing register, therefore, no clearing is required.

The status of the interrupt is available in xxx\_status before the enable stage and in xxx\_intstat after enable stage. Interrupt can be cleared be writing a 1 in the xxx\_status register.

All the interrupt-related registers are multi-bit fields with each bit index corresponding to the same stream index.

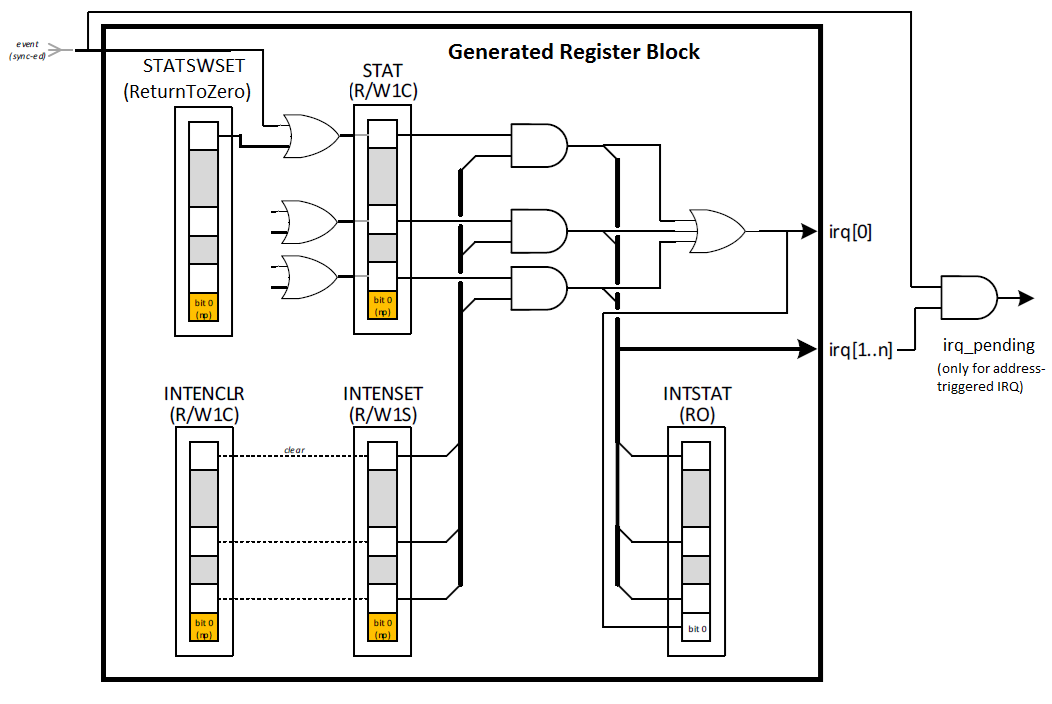


Figure 3 – structure of generated interrupt register logic

#### Address-triggered interrupt

WDMA can generate an interrupt when a programmed address in the buffer has been reached. This interrupt is triggered when a burst begins to be written onto the programmed address.

#### Pending Interrupt

An additional interrupt can be issued if an address-triggered IRQ has been issued and not cleared in time when it gets re-triggered. This interrupt does not apply to the other interrupt sources.

#### FIFO overflow interrupt

Once an overflow flag is asserted by a FIFO, this can trigger an interrupt.

#### Bus error interrupt

Every burst sent to the bus is acknowledged by a response. In case of an error (BRESP ≠ 0) an interrupt is triggered. This is also the case when the response hasn’t arrived after a programmable timeout register, in which case the interrupt is also triggered even if BRESP = 0.

#### AGC trigger

Similar to Address-triggered interrupt, there is a possibility to generate a signal based on a programmed address which will generate a pulse. This output signal can be wired directly to other IPs to act as trigger for signal processing operations.

### Debugging features

A test pattern generator is also included. The generator must be activated independently and any of the streams can be activated in debug mode. In this mode the incoming samples are ignored, and debug data are fed into the FIFO. The generated pattern is a rotating walking one.

The streams can be enabled in debug mode prior to enabling of the pattern generator, in order to have synchronized data for all the streams.

Enabling and disabling the streams in debug mode – as well as in functional mode - will restart the writing of the buffers from start. Likewise, the buffer and multi-shot settings also apply in debug mode.

The generator can be programmed to generate data at various rates. The value of this register represents the number of clock cycles between samples. For example, 0 will write data into the FIFO continuously while 7 will write into the FIFO every 8 cycles. Enabling the generator on multiple streams with a high rate will produce more data than the AXI bus can sink. This can be used to test overflow of the internal FIFOs. Using the generator, it is possible to program the WDMA to generate an arbitrary amount of data on AXI, including an amount that exceeds typical use cases of the system. This feature can be used to stress the system beyond its intended limits (or to determine exactly where this limit lies).

The debug generator uses the *sample\_clock*. This should be considered in the data rate calculations.

## Register File

Register space of WDMA is divided into a base register block and one block per stream. Each block covers a space of 0x100. For example, if the WDMA is configured to support two streams (N\_CHANNEL = 2), it will need a memory map space of 0x300. Two Verilog modules are generated for each and a wrapper includes the modules (1 x base + N\_CHANNELS x stream) and the decode logic for APB connections.

The first region will accommodate the base registers followed by the streams. Twelve bits are accepted by *paddr* which corresponds to 0x1000 bits. If less than this space is used, the unallocated addresses will result in *slverr*, so that no hang-up occurs on the APB bus. Currently a memory map region of 0x1000 is required for WDMA. This also dictates 15 as maximum number of supported channels.

The base region includes global status, debug and interrupt registers. This allows for monitoring the status of all the streams by reading only one register.

## Software Driver APIs

TO BE UPDATED

# Integration

The WDMA is a parameterizable design and the same code can be instantiated with appropriate parameters to fulfill different system requirements. Following parameters are configurable at the time of instantiation:

* ***N\_CHANNELS***: between 1 and 15; restricted by memory-map limit (0x1000 for the IP: 0x100 for base registers + 0x100 \* 15 for stream registers)

Number of incoming streams, potentially from different clock domains which get arbitrated. This will decide the number of FIFOs in the design. When N\_CHANNELS = 1, arbitration logic is reduced to direct connection to the AXI state machine. This parameter is limited to 15, the reason being the reserved APB address space of 0x1000 for the IP.

* ***AXI\_WIDTH***: any supported AXI4 bit-width can be used.

This will determine the width of the incoming samples as well.

* ***FIFO\_LEN\_LOG2:*** ‘0’,’1’and ‘2’ will result in 4-deep FIFOs, no upper limit;

Determines the depth of FIFOs for all streams. Maximum programmable AXI burst length is affected by this.

The RTL has been developed using SystemVerilog and takes advantage of two-dimensional arrays to parameterize many of the interfacing signals. In case the block is instantiated in Verilog, a wrapper will be required.

## Clock and Reset

The IP uses three clock signals.

* ***sample\_clk*** is typically the clock that is used by the IPs that provide the incoming samples to the WDMA. It is also used in WDMA to generate debug patterns. In functional mode this clock can be gated if the streams are not active
* ***axi\_clk*** is the system-bus facing clock which matches the frequency with which the AXI interface runs. This is also the core clock of the internal logic of WDMA
* ***pclk*** is the APB clock used by the APB bus of the subsystem where WDMA is used.

There are no structural dependencies between the clock speeds. However AXI and Sample clocks must be chosen having throughput requirements in mind.

Resets are asynchronous and are assumed to be released synchronously to their respective clocks outside of WDMA.

RESET SYNCHRONISER DFT

### Clock domain crossing

Dynamic signals between all clock domains are synchronized. Sample and AXI domains are synchronized using the data FIFOs and APB domain signals go through a sync block that uses suitable synchronization for signals crossing between APB and AXI domains.

Interrupt pulses coming from core logic go through pulse synchronizers and live status signals like pointer update and buffer wrap count use FIFOs. Single controls use 2-stage flipflops.

## IP-XACT

The memory map information is represented in two excel spreadsheets, one corresponding the base region and one for the streams. These spreadsheets follow the format which is compatible with Magillem register generation flow. The necessary scripts for generation of APB blocks and IP-XACT xml files are included under *magillem* directory of the IP library in the design environment.

Packaging the IP (generation of an overall XML for the IP which include complete memory map and interface) needs to be done for each configuration separately depending on N\_CHANNELS.

## PINNING INTERFACE

Table 1 – clock, control and sample signals

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Pin Name | Direction | Dimensions [bit width x no. of ports] | Clock Domin | Description |
| axi\_clk | In | 1 | NA | AXI clock used as the functional clock of the IP |
| sample\_clk | In | 1 | NA | Core clock with which the samples are delivered |
| axi\_rst\_an | In | 1 | axi\_clk | Asynchronous reset for AXI domain. It is assumed that the release of this reset is synchronized to axi\_clk outside of WDMA |
| sample\_rst\_an | In | 1 | sample\_clk | Asynchronous reset for core domain. It is assumed that the release of this reset is synchronized to sample\_clk clock outside of WDMA |
| p\_data\_in | In | AXI\_WIDHT × N\_CHANNELS | sample\_clk | Incoming samples |
| p\_data\_valid | In | 1 × N\_CHANNELS | sample\_clk | Valid for incoming samples |
| irq\_bus\_error | out | N\_CHANNELS × 1 | apb\_clk | Interrupt vector for bus response/timeout for all streams |
| irq\_fifo | out | N\_CHANNELS × 1 | apb\_clk | Interrupt vector for FIFO overflow for all streams |
| irq\_trigger | out | N\_CHANNELS × 1 | apb\_clk | Interrupt vector for programmable trigger for all streams |
| irq\_trigger\_pending | out | N\_CHANNELS × 1 | apb\_clk | Interrupt vector for pending status of programmable IRQ for all streams |
| agc\_trigger | out | 1 × N\_CHANNELS | axi\_clk | Programmable trigger pulse for direct AGC control. Pulse-width is one AXI clock-cycle |
| start\_trigger\_out | out | 1 × N\_CHANNELS | sample\_clk | DMA start trigger pulse initiated by SW, to be used by preceding blocks to start delivery of samples |

Table 2 – AXI4 Interface signals

|  |  |  |  |
| --- | --- | --- | --- |
| Pin Name | Direction | Width | Description |
| axi\_m\_awvalid | out | 1 | AXI write command valid |
| axi\_m\_awready | in | 1 | AXI write command accept |
| axi\_m\_awaddr | out | 32 | AXI write address |
| axi\_m\_awlen | out | 8 | AXI write burst length (AXI4) |
| axi\_m\_awid | out | 4 | AXI write address id. It is used to differentiate the AXI transactions related to different streams. Also facilitates pipelining of the write and response channels. |
| axi\_m\_awburst | out | 2 | AXI write burst type. Only supported type is INCR |
| axi\_m\_awsize | out | 3 | AXI write burst size. Indicates the size of each transfer in a burst. This is statically set for all of the bytes in the AXI word. |
| axi\_m\_awlock | out | 2 | AXI write lock type. Supported type is NORMAL. EXCLUSIVE and LOCKED transfers are not supported |
| axi\_m\_awcache | out | 4 | AXI write cache type. Statically set to non-bufferable |
| axi\_m\_awprot | out | 3 | AXI write protection. Statically set to Unprivileged, Secure, Data access |
| axi\_m\_wvalid | out | 1 | AXI write command valid. Indicates that valid write address and control information are available |
| axi\_m\_wready | in | 1 | AXI write data accept |
| axi\_m\_wdata | out | AXI\_WIDTH | AXI write data. Width is configured through parameter AXI\_WIDTH |
| axi\_m\_wstrb | out | AXI\_WIDTH/8 | AXI write byte enable. Statically set to all of the bytes |
| axi\_m\_wlast | out | 1 | AXI write last. Indicates the last transfer in the write burst |
| axi\_m\_bvalid | in | 1 | AXI write response valid |
| axi\_m\_bid | in | 4 | AXI write response ID |
| axi\_m\_bready | out | 1 | AXI write response accept. Statically set to 1 Flow control of the response is not supported |
| axi\_m\_bresp | in | 2 | AXI write response. Used as IRQ trigger for bus errors/timeout |

Table 3 - APB3 interface signals

|  |  |  |  |
| --- | --- | --- | --- |
| Pin Name | Direction | Width | Description |
| pclk | in | 1 | APB clock. Must be the same clock as axi\_clk |
| presetn | in | 1 | APB reset. Must be the same reset as axi\_rst\_an |
| pwdata[31:0] | in | 32 | APB write data |
| penable | in | 1 | APB enable |
| pwrite | in | 1 | APB write enable |
| psel | in | 1 | APB slave select |
| prdata[31:0] | out | 32 | APB read data |
| pready | out | 1 | APB ready |
| pslverr | out | 1 | APB slave error. This be asserted for any undefined address in the 0x1000 range from the base address of WDMA |
| paddr[11:0] | in | 12 | APB address. 0x1000 region is defined and decoded even if smaller space is required |

# Memory Map

## Base region map

Base address: 0x0

Interrupt registers are multibit registers with each bit index corresponding to the stream index associated with it.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Register Name** | **Field Name** | **Offset** | **Width** | **Description** | **Reset Value** | **Access** |
| enable\_status |  | 0x00 | N\_CHANNELS | Status of all the streams in the WDMA. The enable status bit is set when the corresponding stream is enabled and de-asserted when it has been disabled and the remining AXI transaction has been finalized by receiving a BRESP. Bit index corresponds to streams index |  | r |
| debug\_generator |  | 0x04 | 32 |  |  | rw |
|  | rate | 16 | 16 | Data rate for generation of Debug pattern:  one sample per (debug\_rate+1) clock cycle Pattern generator uses sample clock (e.g. sample\_clk=166.4MHz, 63 will result in 2.6Msps) | 0x100 | rw |
|  | enable | 0 | 1 |  | 0x0 | rw |
| irq\_trigger\_status |  | 0x10 | N\_CHANNELS | Status/Clear register. Collection of Status flags including Interrupt Status before the enable stage. Writing one will clear the interrupt | 0x0 | rw |
| irq\_trigger\_enable\_set |  | 0x14 | N\_CHANNELS | Interrupt Enable Read and Set register | 0x0 | rw |
| irq\_trigger\_enable\_clr |  | 0x18 |  | Interrupt Enable Clear register. Collection of interrupt enable clear commands affecting the irq\_trigger\_enable\_set register state. | 0x0 | rw |
| irq\_trigger\_intstat |  | 0x1C | N\_CHANNELS | Interrupt Status register. Collection of interrupt status flags after the enable stage. | 0x0 | r |
| irq\_trigger\_sw\_set |  | 0x20 | N\_CHANNELS | Interrupt Software Set register. Collection of software interrupt set status flags before the enable stage. | 0x0 | rw |
| irq\_bus\_status |  | 0x30 | N\_CHANNELS | Status/Clear register. Collection of Status flags including Interrupt Status before the enable stage. Writing one will clear the interrupt | 0x0 | rw |
| irq\_bus\_enable\_set |  | 0x34 | N\_CHANNELS | Interrupt Enable Read and Set register. | 0x0 | rw |
| irq\_bus\_enable\_clr |  | 0x38 | N\_CHANNELS | Interrupt Enable Clear register. Collection of interrupt enable clear commands affecting the irq\_bus\_enable\_set register state. | 0x0 | rw |
| irq\_bus\_intstat |  | 0x3C | N\_CHANNELS | Interrupt Status register. Collection of interrupt status flags after the enable stage. | 0x0 | r |
| irq\_bus\_sw\_set |  | 0x40 | N\_CHANNELS | Interrupt Software Set register. Collection of software interrupt set status flags before the enable stage. | 0x0 | rw |
| irq\_fifo\_status |  | 0x50 | N\_CHANNELS | Status/Clear register. Collection of Status flags including Interrupt Status before the enable stage. Writing one will clear the interrupt | 0x0 | rw |
| irq\_fifo\_enable\_set |  | 0x54 | N\_CHANNELS | Interrupt Enable Read and Set register. | 0x0 | rw |
| irq\_fifo\_enable\_clr |  | 0x58 | N\_CHANNELS | Interrupt Enable Clear register. Collection of interrupt enable clear commands affecting the irq\_fifo\_enable\_set register state. | 0x0 | rw |
| irq\_fifo\_intstat |  | 0x5C | N\_CHANNELS | Interrupt Status register. Collection of interrupt status flags after the enable stage. | 0x0 | r |
| irq\_fifo\_sw\_set |  | 0x60 | N\_CHANNELS | Interrupt Software Set register. Collection of software interrupt set status flags before the enable stage. | 0x0 | rw |
| Module ID comprising config params |  |  |  |  |  |  |

## 

## Stream region map

Base address: (stream\_index + 1) \* 0x100 ; 0 stream\_index N\_CHANNELS

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Register Name** | **Field Name** | **Offset** | **Width** | **Description** | **Reset Value** | **Access** |
| Stream\_activate |  | 0x0 |  |  |  |  |
|  | Enable | 0 | 2 | Enable the stream: 0: set to disabled (check status bit for live status) 1: activate 3: Debug mode (walking-one) | 0x0 | rw |
| stream\_control |  | 0x4 | 32 |  | 0x0 | rw |
|  | fixed\_addr | 24 | 1 | When set, AXI bursts are generated in FIXED mode |  | rw |
|  | burst\_length | 20 | 8 | length of AXI bursts. Values more than 2^FIFO\_LEN\_LOG2-1 will be ignored(max.15) burst\_length is one less that the actual number of beats in a burst |  | rw |
|  | multishot | 0 | 16 | Number of times the buffer needs to be written. 0: indefinitely (Circular mode) ONLY EFFECTIVE UPON ENABLING | 0x0 | rw |
| stream\_status |  | 0x8 | 32 |  |  | r |
|  | wrap\_count | 16 | 16 | shows the number of time the buffer has been filled since last enable | 0x0 | r |
|  | write\_pointer | 0 | 16 | offset in relation to buffer.start\_address where currently being written to (updated when transfer is completed) | 0x0 | r |
| buffer\_settings1 |  | 0xC | 32 |  |  | rw |
|  | address | 0 | 32 | 32bit address where the buffer will be written. Must be aligned with bit-width of AXI bus APPLIED UPON ENABLING THE STREAM | 0x0 | rw |
| buffer\_settings2 |  | 0x10 | 32 |  |  | rw |
|  | size | 0 | 16 | size of buffer in terms of AXI words. Must adhere to AXI boundary crossing limit (in combination with start address) APPLIED UPON ENABLING THE STREAM | 0x0 | rw |
| irq\_trigger |  | 0x14 | 32 |  |  | rw |
|  | address | 0 | 32 | when writing onto this address, an interrupt will be triggered. Only following addresses will cause a trigger: buffer\_settings1.address + (i \*stream\_control.burst\_length \* AXI\_WIDTH) (0=<i<buffer\_settings2.size) APPLIED UPON ENABLING THE STREAM | 0xFFFFFFFF | rw |
| agc\_trigger |  | 0x18 | 32 |  |  | rw |
|  | address | 0 | 32 | when writing onto this address, a gating pulse for AGC will be generated. Only following addresses will cause a trigger: buffer\_settings1.address + (i \* stream\_control.burst\_length \* AXI\_WIDTH) (0=<i<buffer\_settings2.size) APPLIED UPON ENABLING THE STREAM | 0xFFFFFFFF | rw |
| bus\_error\_timeout |  | 0x1C | 32 |  |  | rw |
|  | value | 0 | 32 | bus\_error interrupt will be triggered if BRESP signals arrives later that this value in AXI clock cycles APPLIED UPON ENABLING THE STREAM | 0xFFFFFFFF | rw |

# Abbreviations and terminology

Abbreviations and terminology

| **Abbreviation / terminology** | **Description** |
| --- | --- |
| WDMA | Write Direct Memory Access |
| AXI | Advanced eXtensIble bus |
| FIFO | First In First Out |
| APB | Advanced Peripheral Bus |
| AGC | Automatic Gain Control |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# Acceptance Reviews and Approvals

Reviewers

| Name | Role | Location | Date |
| --- | --- | --- | --- |
| Manfred Zupke | Subsystem Architect | Hamburg |  |
| Frank Siebler | Design Engineer | Dresden |  |
| Jan Hahlbeck | Verification Engineer | Hamburg |  |
| Steffen Loebel | Verification Architect | Hamburg |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# Referenced documents

Referenced documents

| **Doc ID** | **Doc Title** | **Version** | **Author** | **Issue Date** |
| --- | --- | --- | --- | --- |
| doc435007 | |  |  | | --- | --- | |  | [150\_205\_090\_IDA\_WDMA\_RS](https://www.collabnet.nxp.com/sf/go/doc435007?nav=1&pagenum=1&pagesize=15) | | 2 | Soheil Bahrami | 2020-06-19 |
|  |  |  |  |  |
|  |  |  |  |  |