WISHBONE DMA/Bridge IP Core

Author: Rudolf Usselmann
rudi@asics.ws

Rev. 0.2
January 28, 2001

Preliminary Draft

Revision History

Rev.	Date	Author	Description
0.1	23/1/01	Rudolf Usselmann	First Draft Internal release
0.2	28/1/01	RU	First public release

1 Introduction

This core provides DMA transfers between two WISHBONE interfaces. It can also act as a bridge, allowing masters on each WISHBONE interface to also directly access slaves on the other interface.

The DMA engine may have up to four channels. The actual number of channels is user selectable. See Appendix A "Core HW Configuration" on page 19 for more information.

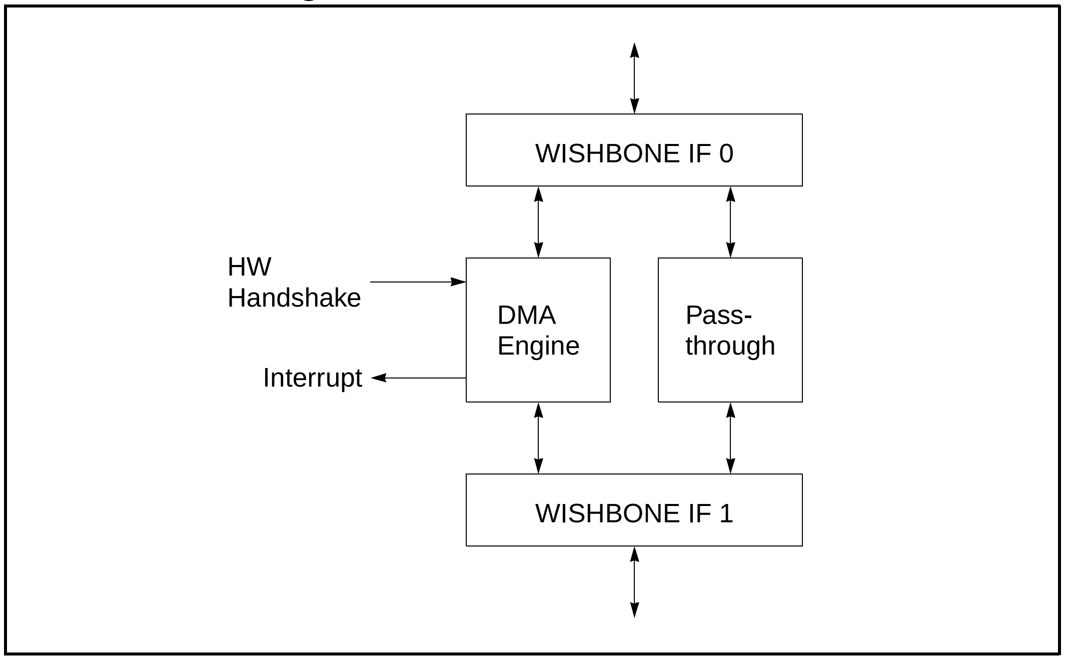
This implementation is designed to work between two WISHBONE interfaces running at the same clock.

(This page intentionally left blank)

2 Architecture

Below figure illustrates the overall architecture of the core.

Figure 1: Core Architecture Overview



It consists of 3 main building blocks: Two WISHBONE interfaces, a DMA engine and pass through logic.

2.1. WISHBONE Interface

The DMA/Bridge core has two master and slave capable WISHBONE interfaces. Both interfaces are WISHBONE SoC bus specification Rev. B compliant.

This implementation implements a 32 bit bus width and does not support other bus widths.

WISHBONE COMPATIBLE

2.2. DMA Engine

The DMA engine is a 4 channel DMA engine that support transfers between the two interfaces and transfers on the same interface (block copy). Each channel can be programmed to have a different priority. Channels with the same priority are serviced in a round robin fashion.

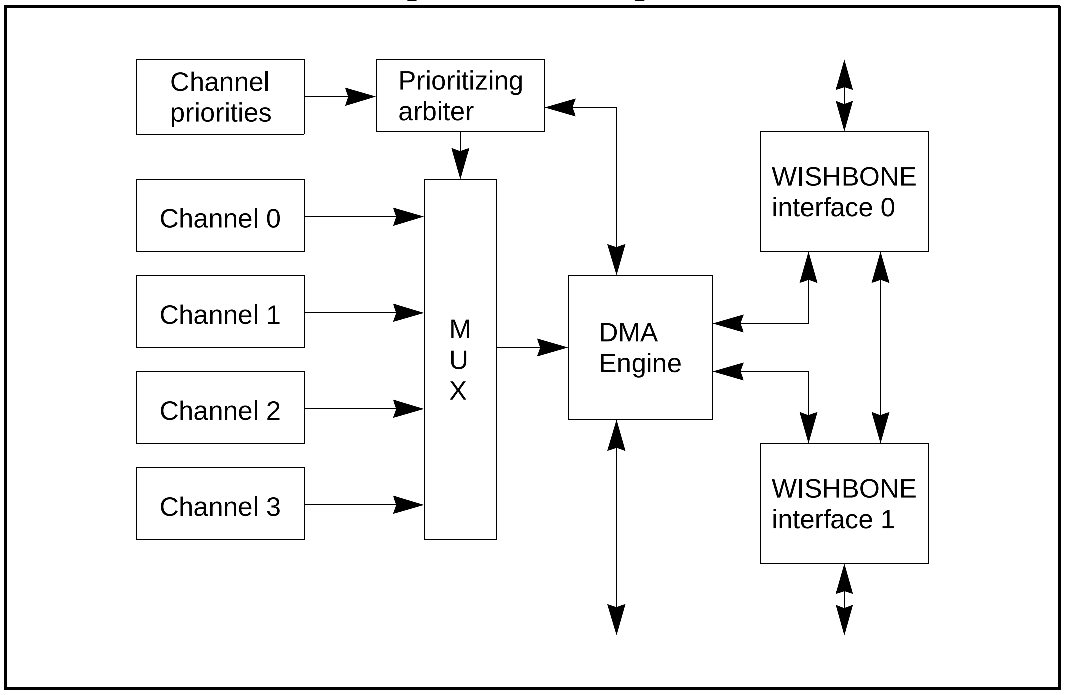
2.3. Pass Through

This block performs the pass through operation between the two WISHBONE interfaces. It includes a two entry deep write buffer in each direction. The write buffer can be disabled if desired.

3 Operation

The DMA engine consists of 4 DMA channels, the actual DMA engine and a channel prioritizing arbiter.

Figure 2: DMA Engine

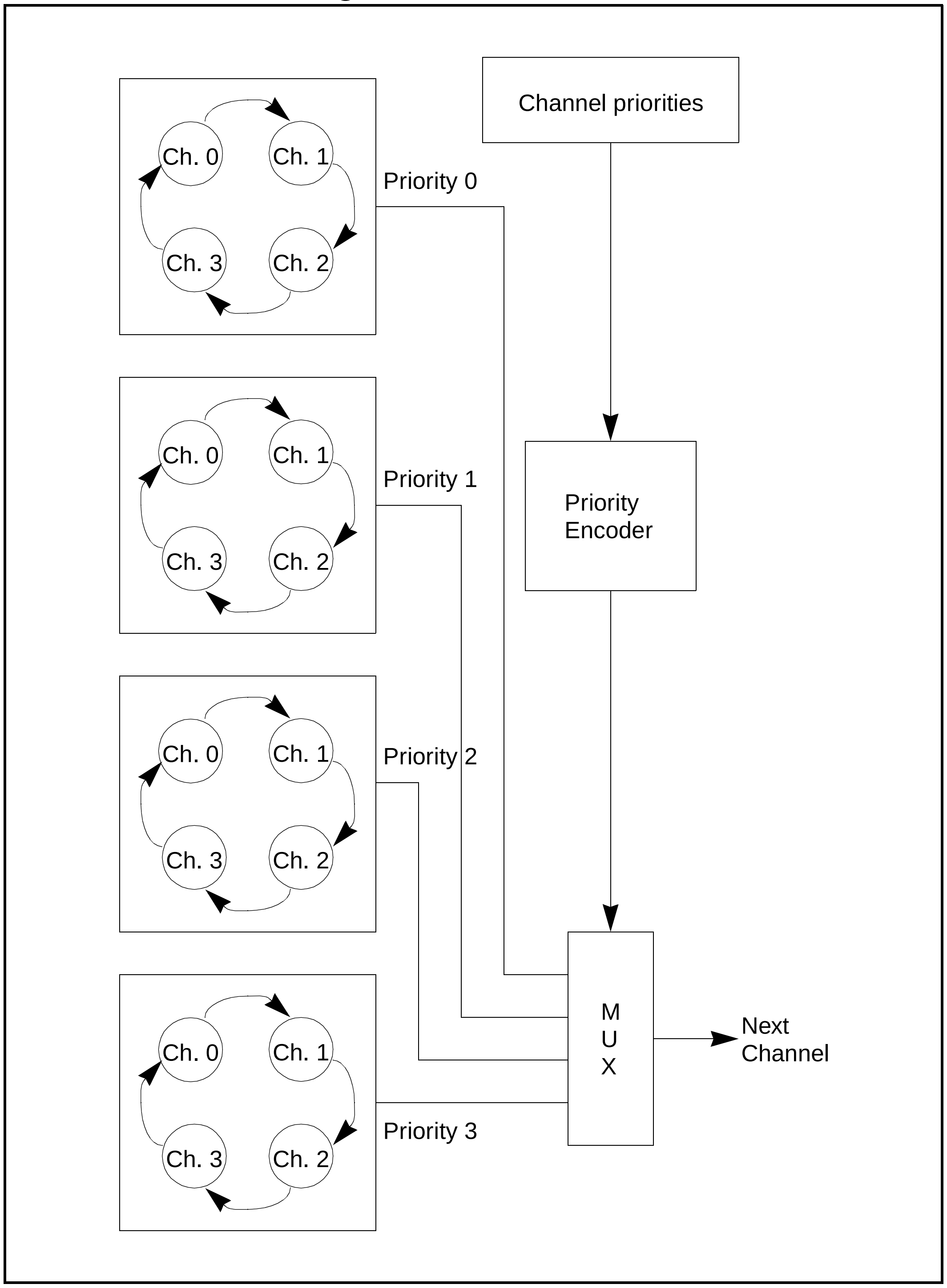


3.1. Prioritizing Arbiter

The prioritizing arbiter will select the next channel to process based first on priority, and secondarily, if all priorities are equal, in a round robin way. Each channel has a 2 bit priority value associated with it. A value of 0 identifies a channel with very low priority, a value of 3, identifies a channel with very high priority. Channels with the same priority are processed in a round robin way, as long as there are no channels with a higher priority.

Below figure illustrates the internal operation of the channel arbiter.

Figure 3: Channel Arbiter



Care should be taken when using priorities, as channels with lower priorities may be locked out and never serviced, if channels with higher priority are being continuously serviced.

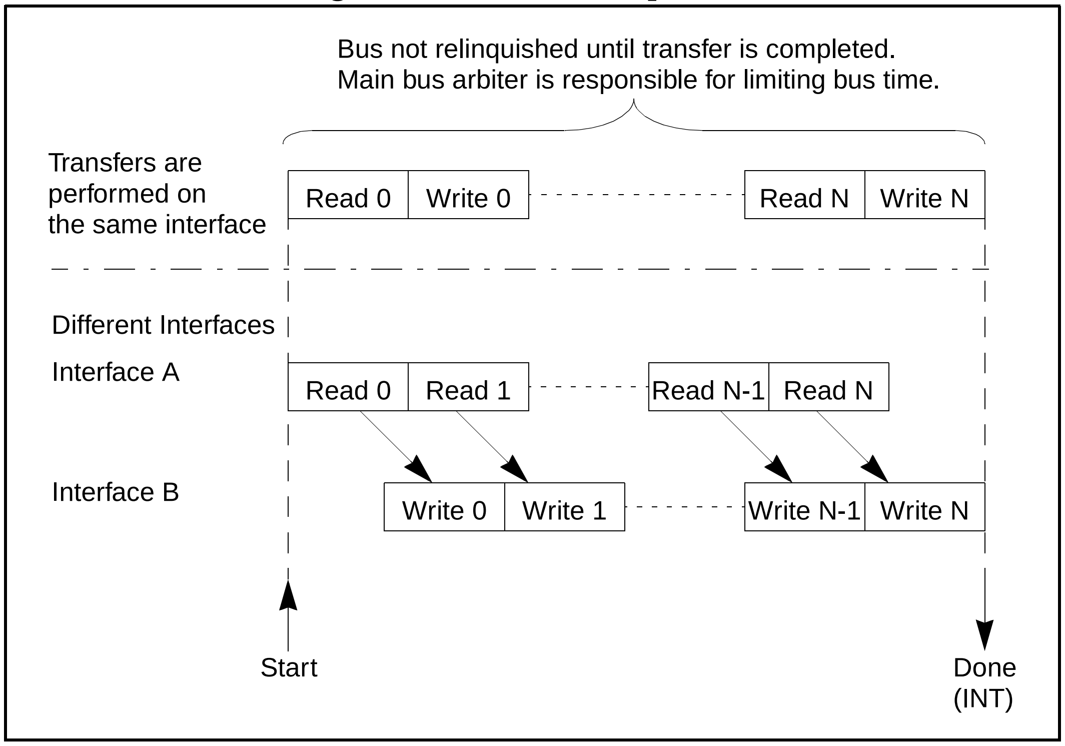
3.2. DMA Engine

The DMA engine can be programmed to perform various transfer operations. This section will illustrate several transfer options and their operation.

3.2.1. Normal DMA operation

This is a simple DMA operation performing a block copy. Below figure illustrates the operation.

Figure 4: Normal DMA operation


Bus not relinquished until transfer is completed.
Main bus arbiter is responsible for limiting bus time.
Transfers are performed on the same interface
Read 0
Write 0
Read N
Write N
Different Interfaces
Interface A
Read 0
Read 1
Read N-1
Read N
Interface B
Write 0
Write 1
Write N-1
Write N
Start
Done (INT)

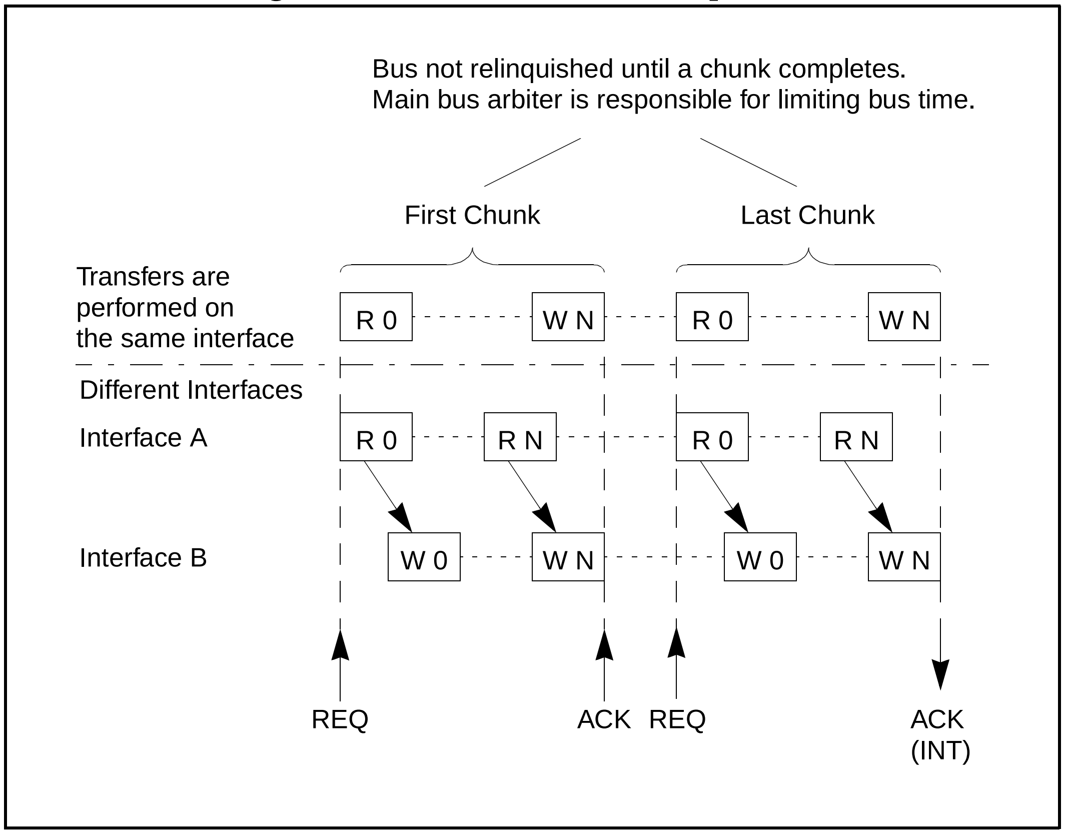
In this example the DMA engine performs a block copy from one location to another, either on the same interface or on a different interfaces. The DMA engine will leave the bus request signal asserted until it has completed the transfer. The transfer begins when either the local controller/CPU writes to the channel CSR register. When the transfer is completed, the DMA engine will assert an interrupt (if enabled) or go to the idle state. If the auto restart bit (ARS) is set, it immediately restart the operation. When the ARS bit is set, the DMA engine will continue restarting until the ARS bit is cleared in the channel CSR register. The software can also force the channel to stop, by writing a 1 to the STOP bit in the channel CSR register. In this case the DMA channel will immediately stop and indicate an error condition by setting the ERR bit in the channel CSR register and asserting an error interrupt (if enabled).

If CHK_SZ is not zero, the channel has to re-arbitrate for the interfaces after each CHK_SZ of word have been transferred. This is particularly useful when setting all channels up with the same priority and requiring “fair” bus usage distribution and low latency.

3.2.2. HW Handshake Mode

Below figure illustrates HW handshake DMA operations, where one full DMA transfer requires more than one external trigger.

Figure 5: HW Handshake DMA Operation


Bus not relinquished until a chunk completes.
Main bus arbiter is responsible for limiting bus time.
First Chunk
Last Chunk
Transfers are performed on the same interface
R 0
W N
R 0
W N
Different Interfaces
Interface A
R 0
R N
R 0
R N
Interface B
W 0
W N
W 0
W N
REQ
ACK
REQ
ACK (INT)

In this mode the DMA engine will wait for the external trigger (DMA_REQ) to be asserted before starting the DMA transfer. Each time the trigger is asserted it will transfer CHK_SZ number of words (one chunk). After each chunk transfer it will assert DMA_ACK to acknowledge the transfer. After TOT_SZ number of words have been transferred, a interrupt is asserted (if enabled).

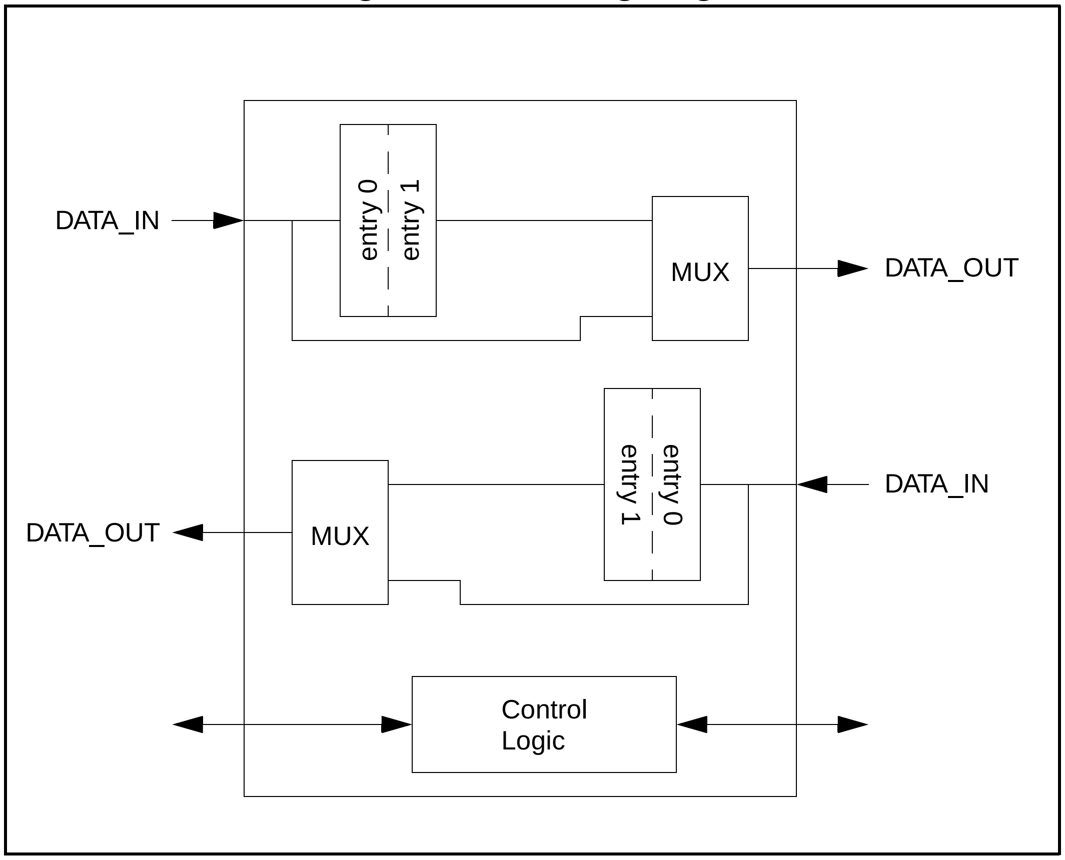
After each chunk transfer the DMA channel has to re-arbitrate internally for the usage of the WISHBONE interfaces.

If the ARS bit is set, the DMA channel will reload the values programmed in to the channel registers and restart the operation. This loop will continue until the ARS bit is cleared or the STOP bit is set. When the STOP bit is set, the DMA engine will immediately stop the transfer, set the ERR bit, and assert an error interrupt (if enabled).

3.3. Pass Through Operation

In pass through mode, this core acts as a bridge. It does not add any functionality to pass through traffic. The user may select if writes complete immediately or if they have to complete on the destination interface first. A two entry deep write buffer is provided in each direction. A delayed write that completes with an error, can be set up to assert an interrupt. Below figure illustrates the pass though logic.

Figure 6: Pass Through Logic



(This page intentionally left blank)

4 Core Registers

This section describes all control and status register inside the WISHBONE DMA/Bridge core. The *Address* field indicates a relative address in hexadecimal. *Width* specifies the number of bits in the register, and *Access* specifies the valid access types to that register. Where RW stands for read and write access, RO for read only access. A 'C' appended to RW or RO, indicates that some or all of the bits are cleared after a read.

Table 1: Control/Status Registers

Name	Addr.	Width	Access	Description
COR	0	32	RW	Configuration Register
INT_MSK	1	32	RW	Interrupt Mask
INT_SRCA	2	32	ROC	Interrupt Source for INTA_O output
INT_SRCB	3	32	ROC	Interrupt Source for INTB_O output
Channel 0 Registers				
CH0_CSR	8	32	RW	Control Status Register
CH0_SZ	9	32	RW	Transfer Size
CH0_A0	A	32	RW	Address 0
CH0_A1	B	32	RW	Address 1
Channel 1 Registers (Optional)				
CH1_CSR	C	32	RW	Control Status Register
CH1_SZ	D	32	RW	Transfer Size
CH1_A0	E	32	RW	Address 0
CH1_A1	F	32	RW	Address 1
Channel 2 Registers (Optional)				
CH2_CSR	10	32	RW	Control Status Register
CH2_SZ	11	32	RW	Transfer Size

Table 1: Control/Status Registers

Name	Addr.	Width	Access	Description
CH2_A0	12	32	RW	Address 0
CH2_A1	13	32	RW	Address 1
Channel 3 Registers (Optional)				
CH3_CSR	14	32	RW	Control Status Register
CH3_SZ	15	32	RW	Transfer Size
CH3_A0	16	32	RW	Address 0
CH3_A1	17	32	RW	Address 1

4.1. Configuration Register (COR)

This is the main configuration register of the DMA/Bridge core.

Table 2: COR Register

Bit #	Access	Description
31:10	RO	RESERVED
9	RW	WBE1 Enable write buffer from Interface 1 to interface 0 0: Write buffer disabled 1: Write buffer enabled
8	RW	WBE0 Enable write buffer from Interface 0 to interface 1 0: Write buffer disabled 1: Write buffer enabled
7:6	RW	PRI3 Priority of channel 3 (0: lowest priority; 3: highest priority)
5:4	RW	PRI2 Priority of channel 2 (0: lowest priority; 3: highest priority)
3:2	RW	PRI1 Priority of channel 1 (0: lowest priority; 3: highest priority)
1:0	RW	PRI0 Priority of channel 0 (0: lowest priority; 3: highest priority)

Reset Value:

COR: 00 h

4.2. Interrupt Mask Register (INT_MSK)

The interrupt mask register defines the functionality of INT_A and INT_B outputs.A bit set to a logical 1 enables the generation of the interrupt for that source, a zero disables the generation of an interrupt.

Table 3: Interrupt Mask Register

Bit #	**Access**	**Description**
31:26	RO	RESERVED
25	RW	Interrupt B Enable: Delayed write interface 1 to interface 0 error
24	RW	Interrupt B Enable: Delayed write interface 0 to interface 1 error
23	RW	Interrupt B Enable: Enable DMA Channel 3 Done Interrupt
22	RW	Interrupt B Enable: Enable DMA Channel 3 Error Interrupt
21	RW	Interrupt B Enable: Enable DMA Channel 2 Done Interrupt
20	RW	Interrupt B Enable: Enable DMA Channel 2 Error Interrupt
19	RW	Interrupt B Enable: Enable DMA Channel 1 Done Interrupt
18	RW	Interrupt B Enable: Enable DMA Channel 1 Error Interrupt
17	RW	Interrupt B Enable: Enable DMA Channel 0 Done Interrupt
16	RW	Interrupt B Enable: Enable DMA Channel 0 Error Interrupt
15:10	RO	RESERVED
9	RW	Interrupt A Enable: Delayed write interface 1 to interface 0 error
8	RW	Interrupt A Enable: Delayed write interface 0 to interface 1 error
7	RW	Interrupt A Enable: Enable DMA Channel 3 Done Interrupt
6	RW	Interrupt A Enable: Enable DMA Channel 3 Error Interrupt
5	RW	Interrupt A Enable: Enable DMA Channel 2 Done Interrupt
4	RW	Interrupt A Enable: Enable DMA Channel 2 Error Interrupt
3	RW	Interrupt A Enable: Enable DMA Channel 1 Done Interrupt
2	RW	Interrupt A Enable: Enable DMA Channel 1 Error Interrupt
1	RW	Interrupt A Enable: Enable DMA Channel 0 Done Interrupt
0	RW	Interrupt A Enable: Enable DMA Channel 0 Error Interrupt

Reset Value:

INT_MSK: 0000h

4.3. Interrupt Source Register (INT_SRCn)

This registers identifies the source of an interrupt. INT_SRCA register indicates the source for INTA_O output, INT_SRCB register indicates the source for INTB_O output. Whenever the function controller receives an interrupt, the interrupt handler must read this register to determine the source and cause of the interrupt. Some of the bits in this register will be cleared after a read. The software interrupt handler must make sure it keeps whatever information is required to handle the interrupt.

Table 4: Interrupt Source Register

Bit #	Access	Description
31:10	RO	RESERVED
9	ROC	Delayed write interface 1 to interface 0 error
8	ROC	Delayed write interface 0 to interface 1 error
7	ROC	DMA Channel 3 Done
6	ROC	DMA Channel 3 Error
5	ROC	DMA Channel 2 Done
4	ROC	DMA Channel 2 Error
3	ROC	DMA Channel 1 Done
2	ROC	DMA Channel 1 Error
1	ROC	DMA Channel 0 Done
0	ROC	DMA Channel 0 Error

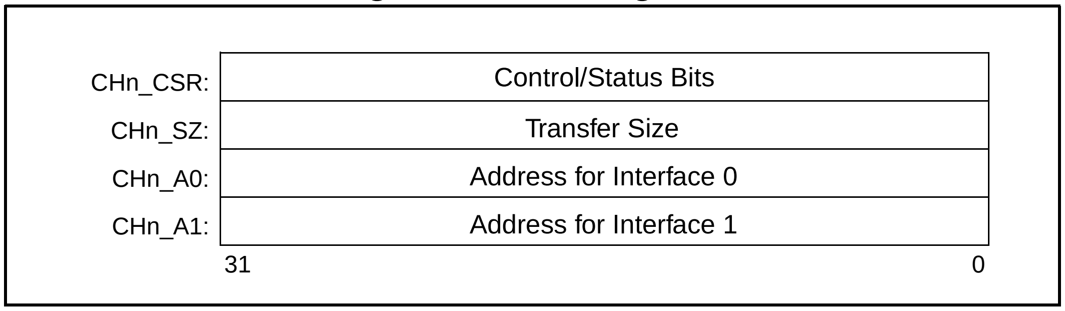
Reset Value:

INT_SRC: 0000h

4.4. Channel Registers

Each channel has 4 registers associated with it. These registers have exactly the same definition for each channel.

Figure 7: Channel Registers



4.4.1. Channel CSR Register (CHn_CSR)

The configuration and status bits specify the operation mode of the channel, as well as reporting any specific channel status.

Table 5: Channel CSR Register

Bit #	Access	Description
31:11	RO	RESERVED
10	RO	ERR DMA channel stopped due to error
9	RO	DONE DMA channel done (This bit will not be set unless the ARS bit is cleared)
8	RO	BUSY DMA channel busy
7	WO	STOP Writing a 1 to this register will cause it to stop it current transfer and set the ERR bit.
6	RO	RESERVED
5	RW	ARS Automatically restart the channel when transfer completes 0: Auto restart disabled 1: Automatically restart the DMA channel after TOT_SZ of bytes have been transferred. The original values programmed in to the channel registers are reloaded and the transfers starts over again.
4	RW	MODE 0: Normal Mode 1: HW Handshake Mode
3	RW	INC_SRC 0: Do not increment source address 1: Increment source address
2	RW	INC_DST 0: Do not increment destination address 1: Increment destination address
1	RW	SRC_SEL 0: Interface 0 is the source 1: Interface 1 is the source
0	RW	DST_SEL 0: Interface 0 is the destination 1: Interface 1 is the destination

Reset Value:

CHn_CSR: 0000h

4.4.2. Channel Size Register (CHn_SZ)

The transfer size register specifies the total and “chunk” transfer sizes for each channel.

Table 6: Channel Size Register

Bit #	Access	Description
31:27	RO	RESERVED
26:16	RW	CHK_SZ Chunk transfer size. Specifies the number of words (4 byte entities) to be transferred at one given time (Not implying they are buffered, but that they will be transferred for each start event in one bus request cycle). If chunk size is zero, the DMA engine will always perform TOT_SZ transfers. Maximum chunk size is 8K bytes.
15:12	RO	RESERVED
11:0	RW	TOT_SZ Total transfer Size. Specifies the number of words (4 byte entities) to be transferred. Maximum total transfer size is16K bytes.

Reset Value:

CHn_SZ: 0000h

4.4.3. Channel Address Registers (CHn_Am)

The address registers specify the source and destination address. Address register 0 is the source address, address register 1 is the destination address. Both registers are 32 bit wide.

Reset Value:

CHn_Am: 0000h

5 Core IOs

This chapter lists all IOs of the DMA/Bridge core.

5.1. Interface IOs

Both interfaces are WISHBONE Rev B compliant. The DMA/Bridge core, can be a slave or master on either interface.

Table 7: Host Interface (WISHBONE)

Name	Width	Direction	Description
CSP_I	1	I	Core pass-through select (From Address decoder)
CSR_I	1	I	Core registers select (From Address decoder), interface 0 only
ADDR_I	32	I	Address Input (for Slave)
ADDR_O	32	O	Address Output (from Master)
DATA_I	32	I	Data Input
DATA_O	32	O	Data Output
SEL_I	4	I	Input for Slave. Indicates which bytes are valid on the data bus. Whenever this signal is not 1111b during a valid access, the ERR_O is asserted.
SEL_O	4	O	Output from master. Indicates which bytes are valid on the data bus. Whenever this signal is not 1111b during a valid access, the ERR_O is asserted.
ACK_O	1	O	Output from slave. Acknowledgment Output. Indicates a normal Cycle termination.
ACK_I	1	I	Input for master. Acknowledgment Output. Indicates a normal Cycle termination.
ERR_O	1	O	Output from slave. Error acknowledgment output. Indicates an abnormal cycle termination.
ERR_I	1	I	Input for master. Error acknowledgment output. Indicates an abnormal cycle termination.

Table 7: Host Interface (WISHBONE)

Name	Width	Direction	Description
RTY_O	1	O	Output from slave. Retry Output. Indicates that the interface is not ready, and the master should retry this operation.
RTY_I	1	I	Input for master. Retry Output. Indicates that the interface is not ready, and the master should retry this operation.
WE_I	1	I	Input for slave. Indicates a Write Cycle when asserted high.
WE_O	1	O	Output from master. Indicates a Write Cycle when asserted high.
STB_I	1	I	Input for slave. Indicates the beginning of a valid transfer cycle.
STB_O	1	O	Output from master. Indicates the beginning of a valid transfer cycle.

5.2. Additional Control IOs

This section describes additional control signals. Except for the clock and reset signals all other signals are special extensions and directly a part of the WISHBONE specification.

Table 8: Additional IOs

Name	Width	Direction	Description
CLK_I	1	I	Clock input
RST_I	1	I	Reset Input
REQ_0	1	I	DMA Request to channel 0 (trigger input)
REQ_1	1	I	DMA Request to channel 1 (trigger input)
REQ_2	1	I	DMA Request to channel 2 (trigger input)
REQ_3	1	I	DMA Request to channel 3 (trigger input)
ACK_0	1	O	DMA Acknowledge channel 0 (Asserted when the DMA is done with the transfer)
ACK_1	1	O	DMA Acknowledge channel 1 (Asserted when the DMA is done with the transfer)
ACK_2	1	O	DMA Acknowledge channel 2 (Asserted when the DMA is done with the transfer)
ACK_3	1	O	DMA Acknowledge channel 3 (Asserted when the DMA is done with the transfer)
INTA_O	1	O	Interrupt Output A
INTB_O	1	O	Interrupt Output B

Appendix A

Core HW Configuration

This Appendix describes the configuration of the core. This step is performed before final Synthesis and tape-out of the core.