IP CORE MANUAL

AXI I2C Bus Interface IP

px_axil_i2c_mstr

PENTEK

Pentek, Inc.
One Park Way
Upper Saddle River, NJ 07458
(201) 818-5900
http://www.pentek.com/

Manual Part Number: 807.48303 Rev: 1.0 - December 09, 2016

Manual Revision History

Date	Version	Comments
12/09/16	1.0	Initial Release

Legal Notices

Copyright

Trademarks

Printed in the United States of America.

Table of Contents

Page

Table of Contents

Page

Chapter 4: Register Space

Chapter 5: Designing with the Core

Table of Contents

Page

Chapter 5: Designing with the Core (continued)

Chapter 6: Design Flow Steps

Table of Contents

Page

This page is intentionally blank

IP Facts

Description

Pentek's Navigator™ AXI I2C Bus Interface Core provides communication links between a large number of devices through a bidirectional two-wire serial bus interface.

This core complies with the ARM® AMBA® AXI4 Specification and also provides a control/status register interface. This user manual defines the hardware interface, software interface, and parameterization options for the AXI I2C Bus Interface Core.

Features

- Compliant to industry standard I^2C Protocol
- Register access through AXI4-Lite interface
- Software programmable I2C clock rate, supporting Standard mode 100kHz and Fast mode 400kHz operation
- START and STOP signal generation
- Acknowledge bit to show the status of a transfer
- 7-bit addressing on the I2C Bus
- Read and Write data FIFOs 8-bits wide by 16 bytes deep
- Supports clock stretching by the slave
- Supports Repeated Start Mode

Table 1-1: IP Facts Table

Core Specifics	
Supported Design Family[a]	Kintex® Ultrascale
Supported User Interfaces	AXI4-Lite
Resources	See Table 2-1
Provided with the Core	
Design Files	VHDL
Example Design	Not Provided
Test Bench	VHDL
Constraints File	Not Provided[b]
Simulation Model	VHDL
Supported S/W Driver	HAL Software Support
Tested Design Flows	
Design Entry	Vivado® Design Suite 2016.3 or later
Simulation	Vivado VSim
Synthesis	Vivado Synthesis
Support	
Provided by Pentek fpgasupport@pentek.com	

a.For a complete list of supported devices, see the *Vivado Design Suite Release Notes*.
b.Clock constraints can be applied at the top level module of the user design.

This page is intentionally blank

Chapter 1: Overview

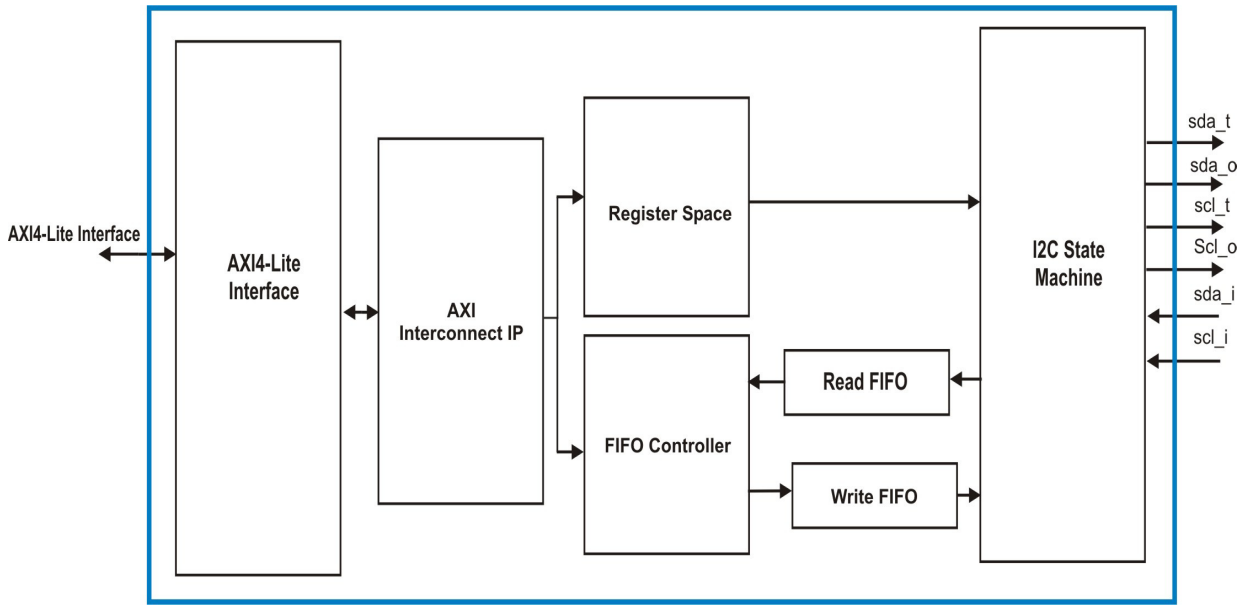
1.1 Functional Description

The AXI I2C Bus Interface Core provides a transaction interface to the AXI4-Lite Interface. The AXI4-Lite interface acts as a slave and is connected to AXI Interconnect IP as shown in Figure 1-1.

The I2C Bus Core (AXI I2C Bus Interface Core) does not provide explicit electrical connectivity to the I2C Bus. The design is expected to be externally connected to tri-state buffers that implement open collector drivers for **scl** and **sda** signals. It must also include external pull-up devices to hold the bus at logic '1' state when the driver is released.

Figure 1-1 is a top-level block diagram of the Pentek AXI I2C Bus Interface Core. The modules within the block diagram are explained in the later sections of this manual.

Figure 1-1: AXI I2C Bus Interface Core Block Diagram



1.1 Functional Description (continued)

- **AXI4-Lite Interface:** This module implements a 32-bit AXI4-Lite Slave Interface to access the Register Space through an AXI Interconnect IP. For additional details about the AXI4-Lite Interface, refer to Section 3.1 AXI4-Lite Core Interfaces.

- **AXI Interconnect IP:** This IP connects the AXI4-Lite Slave Interface to the AXI memory mapped Register Space, and the FIFO Controller.

- **Register Space:** This module contains control and status registers including Interrupt Enable, Interrupt Status and Interrupt Flag registers. Registers are accessed through the AXI4-Lite Interface.

- **Read and Write FIFOs:** These two FIFOs are used as input and output buffers by the AXI I2C Bus Interface Core. The Read FIFO accepts data from the I2C Bus, and stores it until the core is ready to process it. The Write FIFO stores the data that is ready to be output to the I2C Bus, until the bus is ready to accept the data.

- **I2C State Machine:** This state machine is used to control the I2C interface of the core.

1.2 Applications

The AXI I2C Bus Interface Core can be used for interfacing any Kintex Ultrascale FPGA to one or more I2C compliant devices.

1.3 System Requirements

For a list of system requirements, see the Vivado Design Suite Release Notes.

1.4 Licensing and Ordering Information

This core is included with all Pentek Navigator FPGA Design Kits for Pentek Jade series board products. Contact Pentek for Licensing and Ordering Information (www.pentek.com).

1.5 Contacting Technical Support

Technical Support for Pentek's Navigator FPGA Design Kits is available via e-mail (fpgasupport@pentek.com) or by phone (201-818-5900 ext. 238, 9 am to 5 pm EST).

1.6 Documentation

This user manual is the main document for this IP core. The following documents provide supplemental material:

1) *Vivado Design Suite User Guide: Designing with IP*

2) *Vivado Design Suite User Guide: Programming and Debugging*

3) *ARM AMBA AXI4 Protocol Version 2.0 Specification*
http://www.arm.com/products/system-ip/amba-specifications.php

4) NXP Semiconductor I^2C Bus Specification, Revision 6.0, April 2014

This page is intentionally blank

Chapter 2: General Product Specifications

2.1 Standards

The AXI I2C Bus Interface Core has bus a interface that complies with the *ARM AMBA AXI4-Lite Protocol Specification*.

2.2 Performance

The performance of the AXI I2C Bus Interface Core is limited only by the FPGA logic speed. The values presented in this section should be used as an estimation guideline. Actual performance can vary.

2.2.1 Maximum Frequencies

The I2C Bus Core is designed to meet a target frequency of 250 MHz on a Kintex Ultrascale -2 speed grade FPGA. 250 MHz is typically the PCI Express (PCIe®) AXI bus clock frequency.

2.3 Resource Utilization

The resource utilization of the AXI I2C Bus Interface Core is shown in Table 2-1. Resources have been estimated for the Kintex Ultrascale XCKU060 -2 speed grade device. These values were generated using the Vivado Design Suite.

Table 2-1: Resource Usage and Availability

Resource	# Used
LUTs	483
Flip-Flops	704
Memory LUTs	16

NOTE: Actual utilization may vary based on the user design in which the I2C Bus Core is incorporated.

2.4 Limitations and Unsupported Features

- The I2C Bus Core does not support multi-master operation.
- Fast-mode Plus and High Speed mode of 1 MHz and 3.4 MHz operation respectively are not supported by this IP core.
- Arbitration and busy-bus detection are not supported by the core.
- 10-bit addressing is also not supported by this core.
- This IP core does not support a transition from write to read or read to write during repeated start mode of operation.

2.5 Generic Parameters

The generic parameters of the I2C Bus Core are described in Table 2-2. These parameters can be set as required by the user application while customizing the core.

Table 2-2: Generic Parameters

Port/Signal Name	Type	Description
port_has_mga	Boolean	**Port has Geographical Address In:** This can be set to True if the user design has a geographical address input to the I2C Bus Core. This value is stored in the status register bits 22 downto 20.
in_clk_rate_mhz	Integer	**Input Clock Rate in MegaHertz:** This is the value of the input clock rate in MegaHertz. It can range from 1 to 500 MHz.
port_init_rate_khz	Integer	**Output I2C Clock Rate:** This is the initial output I2C bus clock rate setting at reset. It can range from 61 to 400 kHz.

Chapter 3: Port Descriptions

This chapter provides details about the port descriptions for the following interface types:

- AXI4-Lite Core Interfaces

3.1 AXI4-Lite Core Interfaces

The AXI I2C Bus Interface Core uses the Control/Status Register (CSR) interface to control, and receive status from, the user design.

3.1.1 Control/Status Register (CSR) Interface

The CSR interface is an AXI4-Lite Slave Interface that can be used to access the control and status registers in the I2C Bus Core. Table 3-1 defines the ports in the CSR interface. See Chapter 4 for a Control/Status Register memory map and bit definitions. See the *AMBA AXI4-Lite Specification* for more details on operation of the AXI4-Lite interfaces.

Table 3-1: Control/Status Register (CSR) Interface Port Descriptions

Port	Direction	Width	Description
s_axi_csr_aclk	Input	1	**Clock**
s_axi_csr_aresetn	Input	1	**Reset:** Active low. This signal will reset all control registers to their initial states.
s_axi_csr_awaddr	Input	6	**Write Address:** Address used for write operations. It must be valid when **s_axi_csr_awvalid** is asserted and must be held until **s_axi_csr_awready** is asserted by the I2C Bus Core.
s_axi_csr_awprot	Input	3	**Protection:** The I2C Bus Core ignores these bits.
s_axi_csr_awvalid	Input	1	**Write Address Valid:** This input must be asserted to indicate that a valid write address is available on **s_axi_csr_awaddr**. The I2C Bus Core asserts **s_axi_csr_awready** when it is ready to accept the address. The **s_axi_csr_awvalid** must remain asserted until the rising clock edge after the assertion of **s_axi_csr_awready**.
s_axi_csr_awready	Output	1	**Write Address Ready:** This output is asserted by the I2C Bus Core when it is ready to accept the write address.The address is latched when **s_axi_csr_awvalid** and **s_axi_csr_awready** are high on the same cycle.

Table 3-1: Control/Status Register (CSR) Interface Port Descriptions (Continued)

Port	Direction	Width	Description
s_axi_csr_wdata	Input	32	**Write Data:** This data will be written to the address specified by **s_axi_csr_awaddr** when **s_axi_csr_wvalid** and **s_axi_csr _wready** are both asserted. The value must be valid when **s_axi_csr_wvalid** is asserted and held until **s_axi_csr_wready** is also asserted.
s_axi_csr_wstrb	Input	4	**Write Strobes:** This signal, when asserted, indicates the number of bytes of valid data on the **s_axi_csr_wdata** signal. Each of these bits, when asserted, indicate that the corresponding byte of **s_axi_csr_wdata** contains valid data. Bit 0 corresponds to the least significant byte, and bit 3 to the most significant.
s_axi_csr_wvalid	Input	1	**Write Valid:** This signal must be asserted to indicate that the write data is valid for a write operation. The value on **s_axi_csr_wdata** is written into the register at address **s_axi_csr_awaddr** when **s_axi_csr_wready** and **s_axi_csr_wvalid** are high on the same cycle.
s_axi_csr_wready	Output	1	**Write Ready:** This signal is asserted by the I2C Bus Core when it is ready to accept data. The value on **s_axi_csr _wdata** is written into the register at address **s_axi_csr_awaddr** when **s_axi_csr_wready** and **s_axi_csr_wvalid** are high on the same cycle, assuming that the address has already or simultaneously been submitted.
s_axi_csr_bresp	Output	2	**Write Response:** The I2C Bus Core indicates success or failure of a write transaction through this signal, which is valid when **s_axi_csr_bvalid** is asserted; 00 = Success of normal access 01 = Success of exclusive access 10 = Slave Error 11 = Decode Error Note: For more details about this signal refer to the *AMBA AXI Specification*.
s_axi_csr_bready	Input	1	**Write Response Ready:** This signal must be asserted by the user logic when it is ready to accept the Write Response.
s_axi_csr_bvalid	Output	1	**Write Response Valid:** This signal is asserted by the I2C Bus Core when the write operation is complete and the Write Response is valid. It is held until **s_axi_csr_bready** is asserted by the user logic.
s_axi_csr_araddr	Input	6	**Read Address:** Address used for read operations. It must be valid when **s_axi_csr_arvalid** is asserted and must be held until **s_axi_csr_arready** is asserted by the I2C Bus Core.
s_axi_csr_arprot	Input	3	**Protection:** These bits are ignored by the I2C Bus Core

Table 3-1: Control/Status Register (CSR) Interface Port Descriptions (Continued)

Port	Direction	Width	Description
s_axi_csr_arvalid	Input	1	**Read Address Valid:** This input must be asserted to indicate that a valid read address is available on the **s_axi_csr_araddr**. The I2C Bus Core asserts **s_axi_csr_arready** when it ready to accept the Read Address. This input must remain asserted until the rising clock edge after the assertion of **s_axi_csr_arready**.
s_axi_csr_arready	Output	1	**Read Address Ready:** This output is asserted by the I2C Bus Core when it is ready to accept the read address. The address is latched when **s_axi_csr_arvalid** and **s_axi_csr_arready** are high on the same cycle.
s_axi_csr_rdata	Output	32	**Read Data:** This value is the data read from the address specified by the **s_axi_csr_araddr** when **s_axi_csr_arvalid** and **s_axi_csr_arready** are high on the same cycle.
s_axi_csr_rresp	Output	2	**Read Response:** The I2C Bus Core indicates success or failure of a read transaction through this signal, which is valid when **s_axi_csr_rvalid** is asserted; 00 = Success of normal access 01 = Success of exclusive access 10 = Slave Error 11 = Decode Error Note: For more details about this signal refer to the *AMBA AXI Specification*.
s_axi_csr_rvalid	Output	1	**Read Data Valid:** This signal is asserted by the I2C Bus Core when the read is complete and the read data is available on **s_axi_csr_rdata**. It is held until **s_axi_csr_rready** is asserted by the user logic.
s_axi_csr_rready	Input	1	**Read Data Ready:** This signal is asserted by the user logic when it is ready to accept the Read Data.
irq	Output	1	**Interrupt:** This is an active high, edge-type interrupt output.

3.1 I/O Signals

The I/O port/signal descriptions of the top level module of the AXI I2C Bus Interface core are discussed in Table 3-2.

Table 3-2: I/O Signals

Port	Direction	Width	Description
port_scl_i	std_logic	Input	**Serial Clock Input:** This is the serial clock input signal from a tri-state buffer.
port_scl_o	std_logic	Output	**Serial Clock Output:** This is the serial clock output signal to a tri-state buffer.
port_scl_t	std_logic	Output	**Serial Clock Output Enable:** This is the serial clock output enable signal to a tri-state buffer.
port_sda_i	std_logic	Input	**Serial Data Input:** This is the serial data input signal from a tri-state buffer.
port_sda_o	std_logic	Output	**Serial Data Output:** This is the serial data output signal to a tri-state buffer.
port_sda_t	std_logic	Output	**Serial Data Output Enable:** This is the serial data output enable signal to a tri-state buffer.
port_mga[2:0]	std_logic	Input	**Geographical Address Input:** This is the geographical address input when the generic parameter **port_has_mga** is True. This value stored in the status register bits 22 downto 20.

Chapter 4: Register Space

This chapter provides the memory map and register descriptions for the register space of the AXI I2C Bus Interface Core. The memory map is provided in Table 4-1.

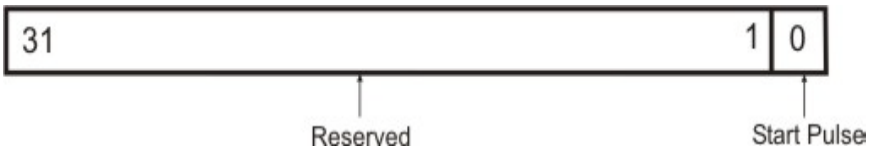
Table 4-1: Register Space Memory Map

Register Name	Address (Base Address +)	Access	Description
Start Pulse Control	0x00	R/W	Control the start pulse for transferring data on the I2C bus.
Clock Division Control	0x04	R/W	Controls input clock rate division.
I2C Address Control	0x08	R/W	Controls the address on the I2C bus.
Control Register	0x0C	R/W	Controls I2C enable, data direction, repeated start mode, FIFO flush and number of bytes to transfer.
Status Register	0x10	R	Indicate the status of the read and write FIFOs, and the I2C bus.
Interrupt Enable Register	0x14	R/W	Interrupt enable bits
Interrupt Status Register	0x18	R	Interrupt source status bits
Interrupt Flag Register	0x1C	R/Clr	Interrupt flag bits
Data Read or Write FIFO	0x20	R/W	Indicates the data stored in the read or write data FIFOs.
FIFO Status	0x24	R/W	Indicates the status of write FIFO when full, and read FIFO when empty.

4.1 Start Pulse Control Register

This register controls the start pulse for transferring data through the I2C port. The Start Pulse Control Register illustrated in Figure 4-1, and described in Table 4-2, has a single defined bit, in the least significant bit position, which is used to control the start pulse to enable (or disable) a write or read operation on the I2C port.

Figure 4-1: Start Pulse Control Register



.

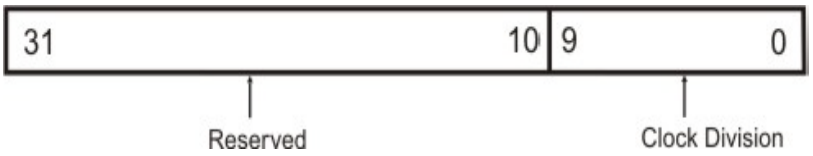
Table 4-2: Start Pulse Control Register (Base Address + 0x00)

Bits	Field Name	Default Value	Access Type	Description
31:1	Reserved	N/A	N/A	**Reserved**
0	strt_pls	0	R/W	**Start Pulse**: This bit enables/ disables a transfer on the I2C bus. When this bit toggled from '0' to '1', a new transfer is initiated. This bit does not self clear to '0'. It must be toggled.

4.2 Clock Division Control Register

This register controls the division of the input clock rate to achieve the required output I2C clock rate. The Clock Division Control Register is illustrated in Figure 4-2 and described in Table 4-3.

Figure 4-2: Clock Division Control Register



.

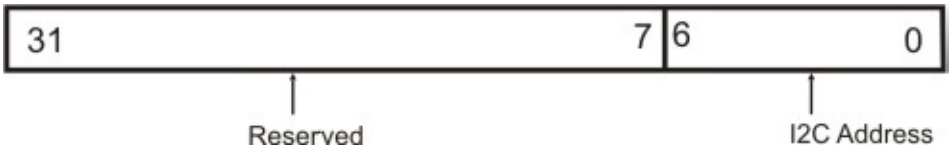
Table 4-3: Clock Division Control Register (Base Address + 0x04)

Bits	Field Name	Default Value	Access Type	Description
31:10	Reserved	N/A	N/A	**Reserved**
9:0	clk_division	Defaults to value set by generic frequency parameters at reset	R/W	**Clock Division:** The input clock rate division required to achieve the I2C output clock rate. Clock Division = (clock frequency in MHz * 1000) / (output rate in kHz * 4)

4.3 I2C Address Control Register

This register controls the address on the I2C Bus at which a slave acknowledges an address transfer operation from the bus master. This core uses 7-bit addressing on the I2C bus. The I2C Address Control Register is illustrated in Figure 4-3 and described in Table 4-4.

Figure 4-3: I2C Address Control Register



.

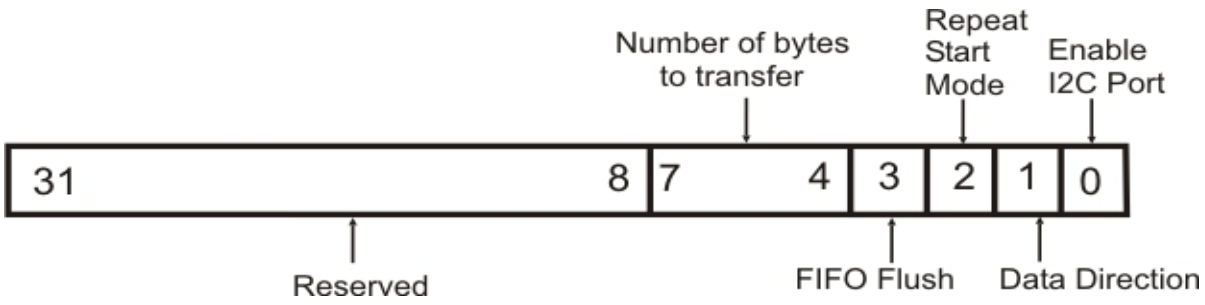
Table 4-4: I2C Address Control Register (Base Address + 0x08)

Bits	Field Name	Default Value	Access Type	Description
31:7	Reserved	N/A	N/A	**Reserved**
6:0	addr	000000	R/W	**I2C Address:** This is the 7-bit I2C address of the slave on the I2C Bus.

4.4 Control Register

This register controls the number of bytes to be transferred over the I2C bus, data direction on the I2C bus, and the enable signal required to enable the I2C port of this core. The Control Register is illustrated in Figure 4-4 and described in Table 4-5.

Figure 4-4: Control Register



.

Table 4-5: Control Register (Base Address + 0x0C)

Bits	Field Name	Default Value	Access Type	Description
31:8	Reserved	N/A	N/A	**Reserved**
7:4	num_bytes	0000	R/W	**Number of bytes to transfer over the I2C bus**
3	fifo_flush	0	R/W	**FIFO Flush:** This bit is used to reset the read and write FIFOs within the core. 0 = Run 1 = Reset
2	repeat_start_mode	0	R/W	**Repeat Start Mode:** This bit is used to enable/ disable repeated start mode of operation of the core. When repeated start mode is enabled, the current data transfer on the I2C bus does not end in a stop condition, allowing a repeat start condition with the next transfer. When the repeated start mode is disabled, the transaction ends with a normal stop condition. 0 = Disable 1 = Enable

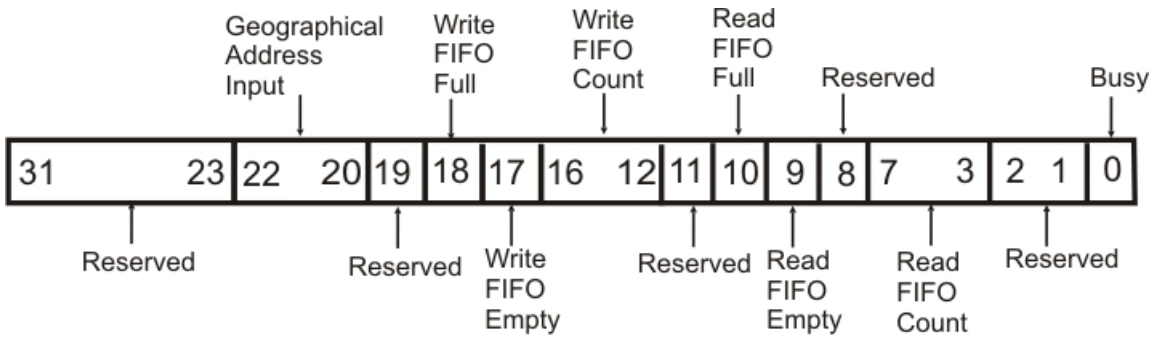
Table 4-5: Control Register (Base Address + 0x0C) (Continued)

Bits	Field Name	Default Value	Access Type	Description
1	r_w_n	0	R/W	**Data Direction:** This bit defines the direction of the data at the I2C port. 0 = Write 1 = Read
0	enable	0	R/W	**Enable the I2C Port:** This bit is used to enable/ disable the I2C port. When disabled, the I2C Bus Core is held in reset. When enabled, the IP core is activated for data transfer on the I2C bus. This bit must be set to ‘0’ to reset the I2C State Machine. 0 = Disable 1 = Enable

4.5 Status Register

The Status Register indicates the status of the AXI I2C Bus Interface core. This register is illustrated in Figure 4-5 and described in Table 4-6.

Figure 4-5: Status Register



.

Table 4-6: Status Register (Base Address + 0x10)

Bits	Field Name	Default Value	Access Type	Description
31:23	Reserved	N/A	N/A	**Reserved**
22:20	port_mga	000	R	**Geographical Address Input:** These bits indicate the geographical address input to the core.
19	Reserved	N/A	N/A	**Reserved**
18	w_fifo_full	0	R	**Write FIFO Full:** This bit indicates FIFO full status of the write FIFO within the I2C Bus Core. 0 = FIFO not full 1 = FIFO full
17	w_fifo_emp	0	R	**Write FIFO Empty:** This bit indicates FIFO empty status of the write FIFO within the I2C Bus Core. 0 = FIFO not empty 1 = FIFO empty
16:12	w_fifo_cnt[4:0]	00000	R	**Write FIFO Count:** These bits indicates the number of data bytes in the write FIFO that can be transmitted over the I2C bus.
11	Reserved	N/A	N/A	**Reserved**

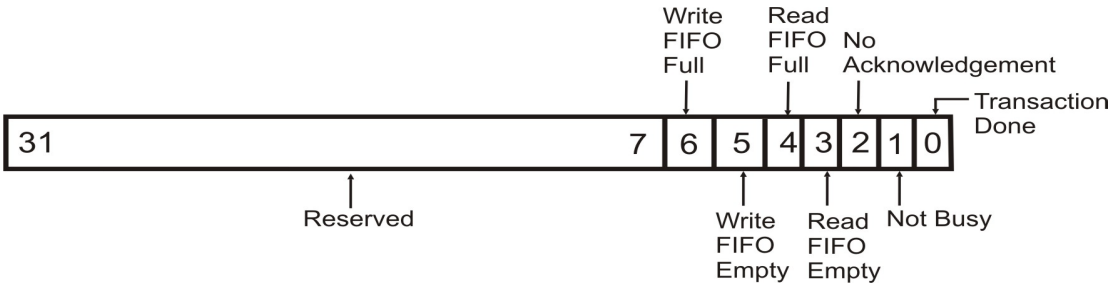
Table 4-6: Status Register (Base Address + 0x10) (Continued)

Bits	Field Name	Default Value	Access Type	Description
10	r_fifo_full	0	R	**Read FIFO Full:** This bit indicates FIFO full status of the read FIFO within the I2C Bus Core. 0 = FIFO not full 1 = FIFO full
9	r_fifo_emp	0	R	**Read FIFO Empty:** This bit indicates FIFO empty status of the read FIFO within the I2C Bus Core. 0 = FIFO not empty 1 = FIFO empty
8	Reserved	N/A	N/A	**Reserved**
7:3	r_fifo_cnt[4:0]	00000	R	**Read FIFO Count:** These bits indicates the number of data bytes available to read from the read FIFO.
2:1	Reserved	N/A	N/A	**Reserved**
0	busy	0	R	**Busy:** This bit indicates if a transaction is in progress at the I2C port. 0 = Not active 1 = Transaction in progress Note: Ensure that this bit is ‘0’ before starting a new transaction or reading or writing the FIFOs.

4.6 Interrupt Enable Register

The bits in the interrupt enable register are used to enable (or disable) the generation of interrupts based on the condition of certain circuit elements, known as interrupt sources. When a bit in this register associated with a given interrupt source is High, an interrupt will be generated by the rising edge of that source's Interrupt Status Register bit (see Section 4.7). This register is illustrated in Figure 4-6 and described in Table 4-7.

Figure 4-6: Interrupt Enable Register



.

Table 4-7: Interrupt Enable Register (Base Address + 0x14)

Bits	Field Name	Default Value	Access Type	Description
31:7	Reserved	N/A	N/A	**Reserved**
6	w_fifo_full	0	R/W	**Write FIFO Full:** This bit enables/ disables the write FIFO full interrupt source. 0 = Disable interrupt 1 = Enable interrupt
5	w_fifo_emp	0	R/W	**Write FIFO Empty:** This bit enables/ disables the write FIFO empty interrupt source. 0 = Disable interrupt 1 = Enable interrupt
4	r_fifo_full	0	R/W	**Read FIFO Full:** This bit enables/ disables the read FIFO full interrupt source. 0 = Disable interrupt 1 = Enable interrupt
3	r_fifo_emp	0	R/W	**Read FIFO Empty:** This bit enables/ disables the read FIFO empty interrupt source. 0 = Disable interrupt 1 = Enable interrupt

Table 4-7: Interrupt Enable Register (Base Address + 0x14) (Continued)

Bits	Field Name	Default Value	Access Type	Description
2	no_ack	0	R/W	**No Acknowledgement:** This bit enables/ disables the no acknowledgement interrupt source. The no acknowledgement interrupt source is set high when there is no acknowledgement after an address request on the I2C bus. 0 = Disable interrupt 1 = Enable interrupt
1	not busy	0	R/W	**Not Busy:** This bit enables/ disables the I2C bus not busy interrupt source. The I2C bus not busy interrupt source is the negation of the busy bit from the status register of the I2C Bus Core. 0 = Disable interrupt 1 = Enable interrupt
0	trns_done	0	R/W	**Transaction Done:** This bit enables/ disables the transaction done interrupt source. The transaction done interrupt source indicates that the current transaction on the I2C bus is complete. 0 = Disable interrupt 1 = Enable interrupt

4.7 Interrupt Status Register

The Interrupt Status Register has read-only access associated with each interrupt condition. A status bit changes to '1' when the source interrupt occurs. When a status bit in this register changes to '1' the corresponding flag bit in the Interrupt Flag Register is set to '1'. A status bit in this register clears to '0' when that interrupt condition clears, whereas the associated flag bit in the Interrupt Flag Register remains at logic '1' until it is explicitly cleared by the user.

Some of the interrupt sources are transient and so may not appear in the Interrupt Status Register at the time it is read. In such cases use the Interrupt Flag Register to see the interrupt conditions that have occurred. The Interrupt Status Register is illustrated in Figure 4-7 and described in Table 4-8.

Figure 4-7: Interrupt Status Register

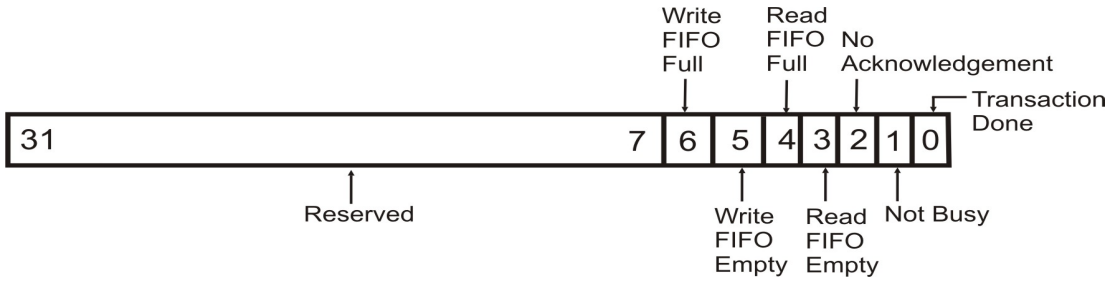


Table 4-8: Interrupt Status Register (Base Address + 0x18)

Bits	Field Name	Default Value	Access Type	Description
31:7	Reserved	N/A	N/A	**Reserved**
6	w_fifo_full	0	R	**Write FIFO Full:** This bit indicates the status of the write FIFO full interrupt source. 0 = No interrupt 1 = Interrupt condition asserted
5	w_fifo_emp	0	R	**Write FIFO Empty:** This bit indicates the status of the write FIFO empty interrupt source. 0 = No interrupt 1 = Interrupt condition asserted
4	r_fifo_full	0	R	**Read FIFO Full:** This bit indicates the status of the read FIFO full interrupt source. 0 = No interrupt 1 = Interrupt condition asserted

Table 4-8: Interrupt Status Register (Base Address + 0x18) (Continued)

Bits	Field Name	Default Value	Access Type	Description
3	r_fifo_emp	0	R	**Read FIFO Empty:** This bit indicates the status of the read FIFO empty interrupt source. 0 = No interrupt 1 = Interrupt condition asserted
2	no_ack	0	R	**No Acknowledgement:** This bit indicates the status of the no acknowledgement interrupt source. The no acknowledgement interrupt source is set high when there is no acknowledgement after an address request on the I2C bus. 0 = No interrupt 1 = Interrupt condition asserted
1	not busy	0	R	**Not Busy:** This bit indicates the status of the I2C bus not busy interrupt source. The I2C bus not busy interrupt source is the negation of the busy bit from the status register of the I2C Bus Core. 0 = No interrupt 1 = Interrupt condition asserted
0	trns_done	0	R	**Transaction Done:** This bit indicates the status of the transaction done interrupt source. The transaction done interrupt source indicates that the current transaction on the I2C bus is complete. 0 = No interrupt 1 = Interrupt condition asserted

4.8 Interrupt Flag Register

The Interrupt Flag Register has a read/clear access associated with each interrupt condition. When reset, this register has all bits set to '0' (cleared). Each flag bit in this register latches an interrupt occurrence. A '1' in any flag bit in this register indicates that an interrupt has occurred.

Note that when any status bit in the Interrupt Status Register, changes to '1' the corresponding flag bit in this register will also be set to '1'. However, when a status bit in the Interrupt Status Register clears from '1' to '0', the corresponding latched flag bit in this register does not clear, but remains at '1'. To clear the flag bits, write '1's to the desired bits. The flags are not affected by the Interrupt Enable Register. The Interrupt Flag Register is illustrated in Figure 4-8 and described in Table 4-9.

Figure 4-8: Interrupt Flag Register

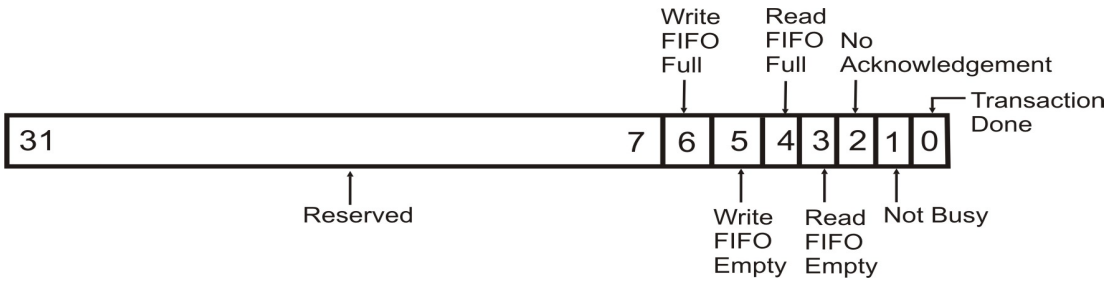


Table 4-9: Interrupt Flag Register (Base Address + 0x1C)

Bits	Field Name	Default Value	Access Type	Description
31:7	Reserved	N/A	N/A	**Reserved**
6	w_fifo_full	0	R/Clr	**Write FIFO Full:** This bit indicates the write FIFO full interrupt flag. **Read:** 0 = No interrupt 1 = Interrupt latched **Clear:** 1 = Clear latch
5	w_fifo_emp	0		**Write FIFO Empty:** This bit indicates the write FIFO empty interrupt flag. **Read:** 0 = No interrupt 1 = Interrupt latched **Clear:** 1 = Clear latch

Table 4-9: Interrupt Flag Register (Base Address + 0x1C) (Continued)

Bits	Field Name	Default Value	Access Type	Description
4	r_fifo_full	0	R/Clr	**Read FIFO Full:** This bit indicates the read FIFO full interrupt flag. **Read:** 0 = No interrupt 1 = Interrupt latched **Clear:** 1 = Clear latch
3	r_fifo_emp	0		**Read FIFO Empty:** This bit indicates the read FIFO empty interrupt flag. **Read:** 0 = No interrupt 1 = Interrupt latched **Clear:** 1 = Clear latch
2	no_ack	0		**No Acknowledgement:** This bit indicates the no acknowledgement interrupt flag. The no acknowledgement interrupt source is set high when there is no acknowledgement after an address request on the I2C bus. **Read:** 0 = No interrupt 1 = Interrupt latched **Clear:** 1 = Clear latch
1	not busy	0		**Not Busy:** This bit indicates the I2C bus not busy interrupt flag. The I2C bus not busy interrupt source is the negation of the busy bit from the status register of the I2C Bus Core. **Read:** 0 = No interrupt 1 = Interrupt latched **Clear:** 1 = Clear latch
0	trns_done	0		**Transaction Done:** This bit indicates the transaction done interrupt flag. The transaction done interrupt source indicates that the current transaction on the I2C bus is complete. **Read:** 0 = No interrupt 1 = Interrupt latched **Clear:** 1 = Clear latch

4.9 Data Read or Write FIFO Register

During a read operation from the user design on the I2C bus, this register indicates the data received from the I2C bus and stored in the read FIFO. During a write operation to the user design on the I2C bus, the data to be written over the I2C bus can be written into this register which is then stored in the write FIFO. The Data Read/ Write FIFO Register is illustrated in Figure 4-9 and described in Table 4-10.

Figure 4-9: Data Read/ Write FIFO Register

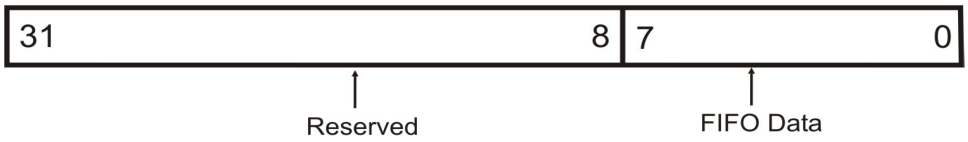


Table 4-10: Data Read/ Write FIFO Register (Base Address + 0x20)

Bits	Field Name	Default Value	Access Type	Description
31:8	Reserved	N/A	N/A	**Reserved**
7:0	data	0x00	R/W	**FIFO Data**

4.10 FIFO Status Register

The FIFO status register is a read-only register that indicates the status of the read and write FIFOs. It shows the status of the read FIFO when empty, and the write FIFO when full. Note that the write FIFO, 16 bytes deep, when full, the **w_fifo_full** is set High. This Register is illustrated in Figure 4-10 and described in Table 4-11.

Figure 4-10: FIFO Status Register

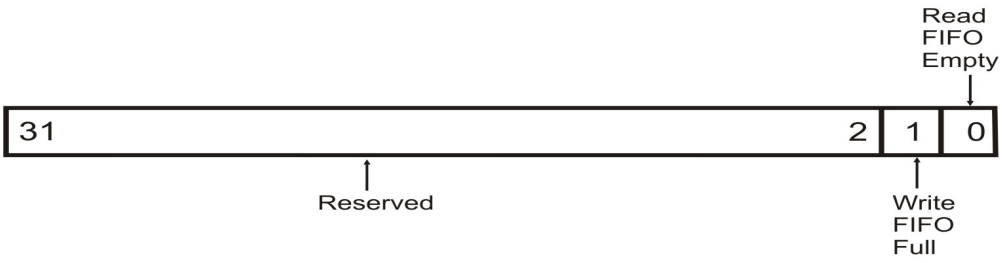


Table 4-11: FIFO Status Register (Base Address + 0x24)

Bits	Field Name	Default Value	Access Type	Description
31:2	Reserved	N/A	N/A	**Reserved**
1	w_fifo_full	0	R	**Write FIFO Full:** This bit indicates the FIFO full status of the write FIFO. 0 = FIFO not full 1 = FIFO full
0	r_fifo_emp	0	R	**Read FIFO Empty:** This bit indicates the FIFO empty status of the read FIFO. 0 = FIFO not empty 1 = FIFO empty

Chapter 5: Designing with the Core

This chapter includes guidelines and additional information to facilitate designing with the AXI I2C Bus Interface Core.

5.1 I2C Protocol and Electrical Characteristics

To understand and use the AXI I2C Bus Interface Core, it is helpful to have a basic understanding of the I2C Protocol, and electrical characteristics of the bus. For more details see the *I^2C Bus Specification*.

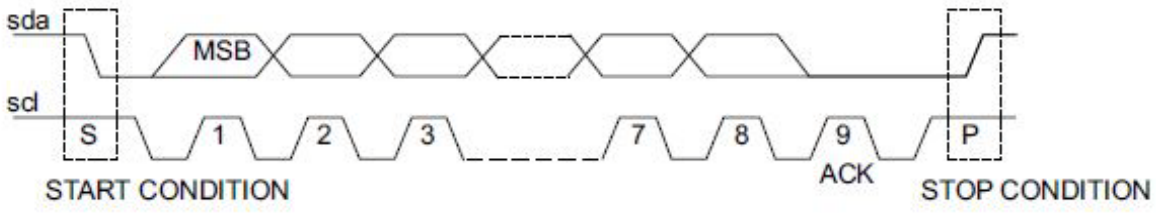
5.1.1 Protocol for Address and Data Transfer

Each device on the I2C bus has a unique 7-bit address, that operates both as transmitter and receiver, and additionally acts as a master or a slave. A master device initiates data transfer on the bus and generates the clock signal for that transfer. The slaves respond to the address clocked into them by the master and either accepts (write) data from or provides (read) data to the master.

Data transfers on the I2C bus are initiated with a START condition and are terminated with a STOP condition. After reaching the bus free state, a master signals a START defined by a High-to-Low transition on **sda** while **scl** is High. Between the START and STOP conditions of the bus, data on the **sda** signal must be stable during the High period of the **scl** signal and must meet any required setup and hold times during the Low period of the **scl** signal.

Figure 5-1 illustrates the START and STOP signals.

Figure 5-1: Data Transfer on I2C Bus



5.1.1 Protocol for Address and Data Transfer (continued)

Each transfer on the I2C bus consists of 9 clock cycles on **scl** to move eight bits of data and one acknowledge bit. Master and Slave transmitters send data with the most significant bit (MSB) first.

After providing data for eight clock cycles, the master or slave transmitter releases the **sda** line during the acknowledgement clock period to permit the receiver to transfer a 1-bit acknowledgement.

If a slave-receiver issues not-acknowledge (by releasing the **sda** signal during the acknowledgement period) this indicates that the slave receiver was unable to accept the prior eight bits transferred (consisting of address or data bits). After a byte of data is transferred and acknowledged, the slave (receiver/transmitter) has the unique capability to hold the transfer by keeping the **scl** line in its Low state by actively pulling the **scl** line Low for an arbitrary period of time (Clock Stretching), forcing the master into a wait state until the slave is ready for the next byte transfer.

Standard communication on the bus between a master and slave is composed of four parts:

- START
- Slave Address
- Data transfer
- STOP

The I2C protocol defines a transfer for 7-bit addressing, with the slave address sent after the START condition. This address is seven bits long followed by an eighth bit which defines the read/write operation. A High indicates a request for data read and a Low indicates a data write. The slave whose address matches the one transmitted by the master (and no other slave) should respond by sending back an acknowledge bit by pulling the **sda** line Low on the ninth clock cycle.

5.1.2 Electrical Issues

An I2C bus consists of two wires (hence the alternate name TWSI, or Two-Wire Serial Interface) named serial data (**sda**) and serial clock (**scl**), which carry information between the devices connected on the bus. The 400pF maximum signal load capacitance limits the maximum number of devices connectable to the same bus.

Both **sda** and **scl** transport data bidirectionally between connected devices using wired-AND electrical connectivity. To implement the wired-AND, each device connected to the bus utilizes an open collector (or open-drain) output, which can only sink current, with the emitter (or source) grounded, to pull the signal to logic ‘0’. Electrically that means it must not drive a logic ‘1’ on to either bus signal but can only release or float the output. When no device asserts a logic ‘0’ onto the bus, external pull-up devices (typically resistors) bring the signal state High. This may be a source of confusion, because no device can actually set the state of **scl** or **sda** to its High state.

When all devices on the bus release their drivers and both **sda** and **scl** are High for a specified period of time the bus is considered to be in the bus free state.

5.2 Clocking

Main Clock: **s_axi_csr_aclk**

This clock is used to clock all ports of the core.

5.3 Resets

Main reset: **s_axi_csr_aresetn**

This is an active low synchronous reset associated with **s_axi_csr_aclk**. When asserted, all state machines in the core are reset, all FIFOs are flushed and all the control registers are cleared back to their initial default states.

5.4 Interrupts

This core has an edge type (rising edge-triggered) interrupt output. It is synchronous with the **s_axi_csr_aclk**. On the rising edge of any interrupt signal, a one clock cycle wide pulse is output from the core on it’s **irq** output. Each interrupt event is stored in two registers, accessible on the **s_axi_csr** bus.

5.4 Interrupts (continued)

The Interrupt Status Register always reflects the current state of the interrupt condition, which may have changed since the generation of the interrupt. The Interrupt Flag Register latches the occurrence of each interrupt, in a bit that retains its state until explicitly cleared. The Interrupt flags can be cleared by writing '1' to the associated bit's location. All interrupt sources that are enabled (via the Interrupt Enable Register) are "OR ed" onto the **irq** output.

NOTE: All interrupt sources are latched in the interrupt flag register, even when an interrupt source is not enabled to create an interrupt.

NOTE: Because this core uses edge-triggered interrupts, the fact that an interrupt condition may remain active after servicing will not cause the generation of a new interrupt. A new interrupt will only be generated by another rising edge on an interrupt source.

5.5 Clock Stretching

The AXI I2C Bus Interface Core supports clock stretching. Clock stretching pauses a transaction by holding the **scl** line Low. A slave can hold the **scl** line Low after reception and acknowledgement of a byte to force the master into a wait state until the slave is ready for the next byte transfer, in a type of handshake procedure.

5.6 Interface Operation

CSR Interface: This is the Control/Status Register Interface and is associated with the **s_axi_csr_aclk**. It is a standard AXI4-Lite Interface. This interface connects through an AXI4-Lite Interconnect Core to the Register Space. See Chapter 4 for the control/ status register memory map, for more details on the registers that can be accessed through this interface.

5.7 Programming Sequence

This section briefly describes the programming sequence of registers in the I2C Bus Core.

❑ I2C Master Transmitter

- Ensure that the Interrupt Flag Register is cleared.
- Enable the interrupt enable bit for transaction done interrupt. This indicates whether the bus is busy in another transaction or free for a new transaction.
- Make sure the start bit is Low.
- Enable the I2C interface and set the slave address.

5.7 Programming Sequence (continued)

❑ I2C Master Transmitter (continued)

- Set the control register for writing data to the slave address, and set the data length value (number of bytes to transmit). Set repeat start mode based on the user application requirement.
- Write the data into the Read/Write FIFO Register.
- Toggle the start pulse bit High then Low.
- After transmission of data, wait for the acknowledgement and the transaction done interrupt.
- When the transaction done interrupt is received, check the Interrupt Flag Register and clear the interrupts.

❑ I2C Master Receiver

- Ensure that the Interrupt Flag Register is cleared.
- Enable the interrupt enable bit for transaction done interrupt.
- Enable the I2C interface and set the slave address.
- Set the control register for reading data from slave address and set the data length value (number of bytes to receive). Set repeat start mode based on the user application requirement.
- Toggle the start pulse bit High then Low.
- Wait for the transaction done interrupt.
- Read data from the Read/Write FIFO Register.
- Clear the interrupts.

NOTE: If interrupts are not used, you can poll the busy status bit low for transaction completion instead.

5.8 Timing Diagrams

The timing diagrams for the AXI I2C Bus Interface Core are shown in Figures 5-2 and 5-3. These timing diagrams were obtained by running the test bench for the core with simulation performed in the Vivado VSim environment. These timing diagrams depict the functionality of the core for Master transmitter and receiver operations.

Figure 5-2: AXI I2C Bus Interface Core - Master Transmitting Data

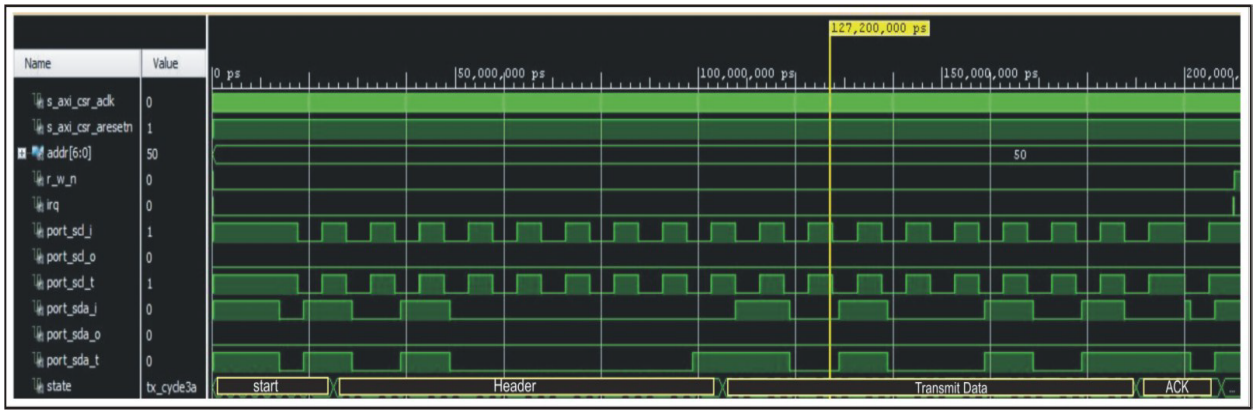
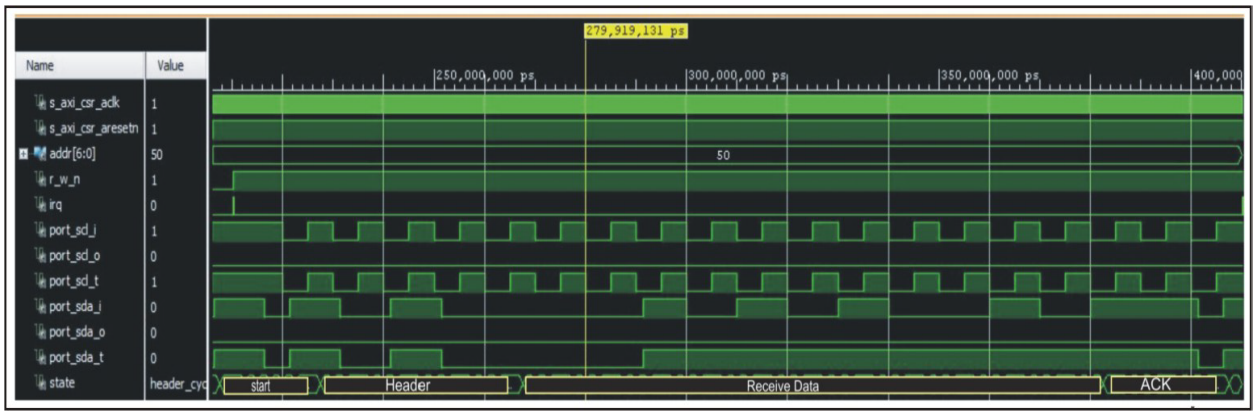


Figure 5-3: AXI I2C Bus Interface Core - Master Receiving Data

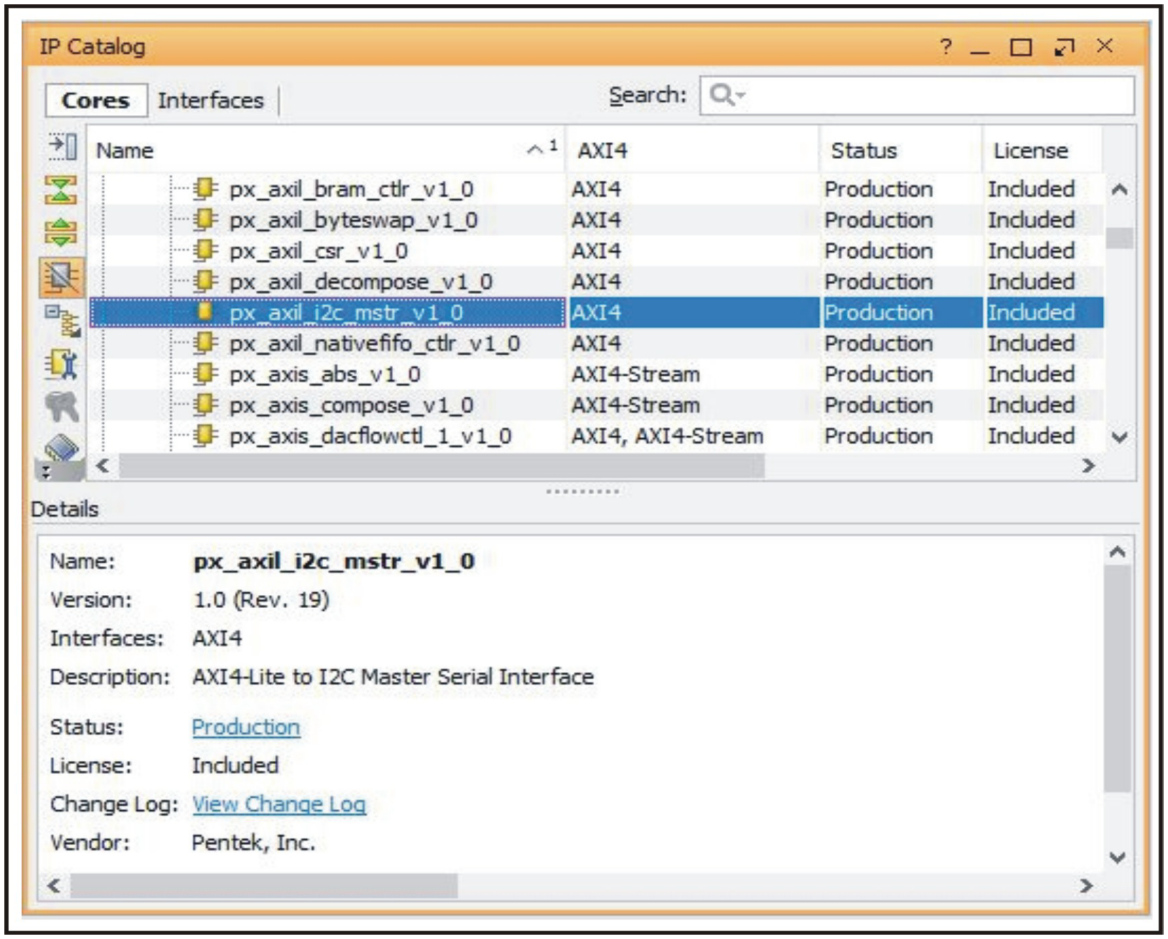


Chapter 6: Design Flow Steps

6.1 Pentek IP Catalog

This chapter describes customization and generation of the Pentek AXI I2C Bus Interface Core. It also includes simulation, synthesis, and implementation steps that are specific to this IP core. This core can be generated from the Vivado IP Catalog when the Pentek IP Repository has been installed. It will appear in the IP Catalog list as **px_axil_i2c_mstr_v1_0** as shown in Figure 6-1.

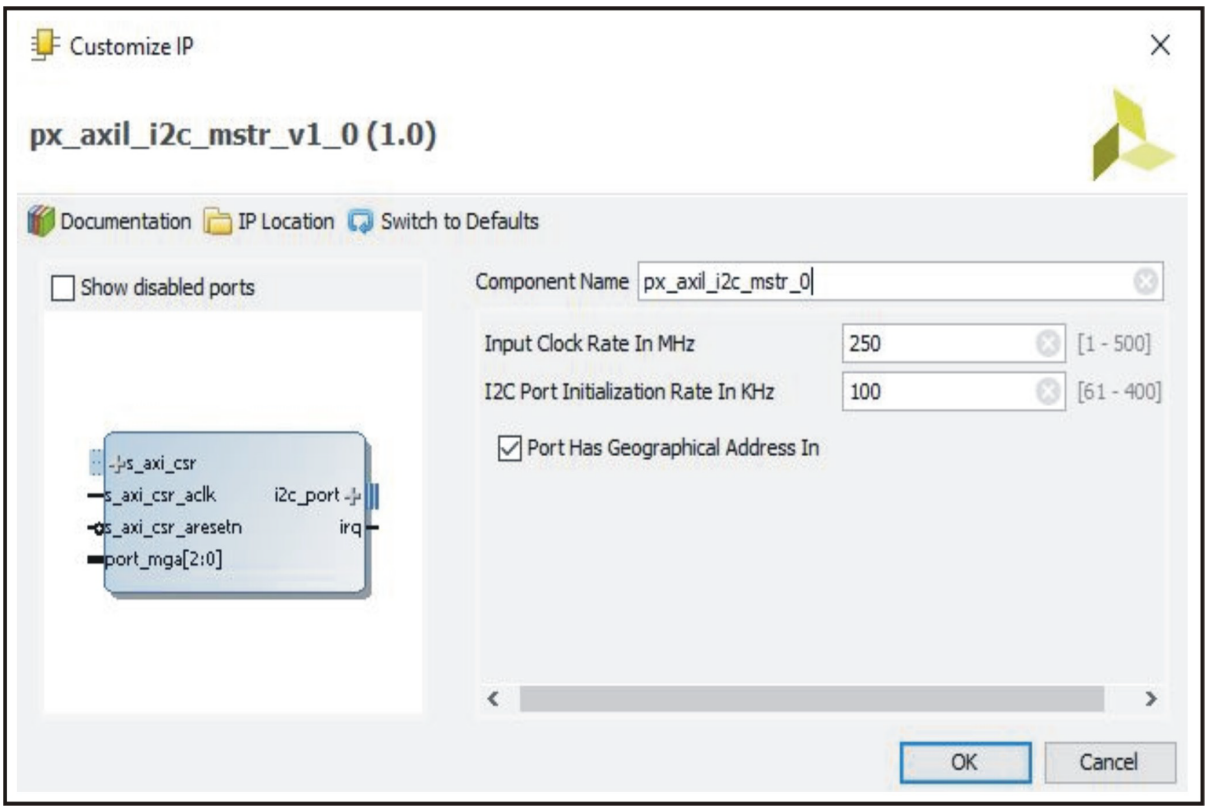
Figure 6-1: AXI I2C Bus Interface Core in Pentek IP Catalog



6.1 Pentek IP Catalog (continued)

When you select the **px_axil_i2c_mstr_v1_0** core, a screen appears that shows the core's symbol and the core's parameters (see Figure 6-2). The core's symbol is the box on the left side.

Figure 6-2: AXI I2C Bus Interface Core IP Symbol



6.2 User Parameters

The user parameters of this core are described in Section 2.5 of this user manual.

6.3 Generating Output

For more details about generating and using IP in the Vivado Design Suite, refer to the *Vivado Design Suite User Guide - Designing with IP*.

6.4 Constraining the Core

This section contains information about constraining the I2C Bus Core in Vivado Design Suite.

Required Constraints

The XDC constraints are not provided with the I2C Bus Core. Clock constraints can be applied in the top-level module of the user design.

Device, Package, and Speed Grade Selections

This IP works for the Kintex Ultrascale FPGAs.

Clock Frequencies

The main clock (**s_axi_csr_aclk**) of this IP core can take frequencies up to 250 MHz.

Clock Management

This section is not applicable for this IP core.

Clock Placement

This section is not applicable for this IP core.

Banking and Placement

This section is not applicable for this IP core.

Transceiver Placement

This section is not applicable for this IP core.

I/O Standard and Placement

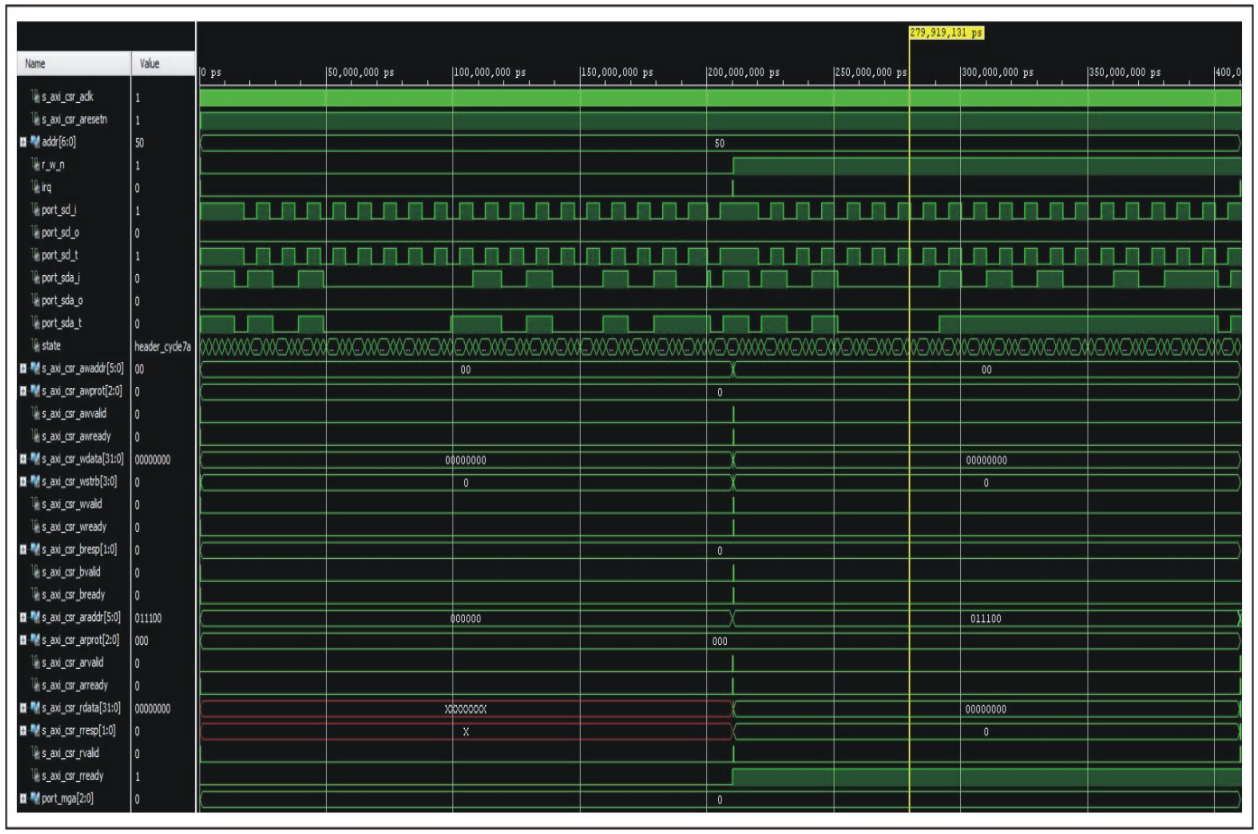
This section is not applicable for this IP core.

6.5 Simulation

The AXI I2C Bus Interface Core has a test bench which generates the output waveforms using the Vivado VSim environment.

The test bench is designed to run at 250 MHz input clock frequency and port initialization rate of 100 kHz (Standard Mode). The test bench also assigns an I2C address 0b1010000 to the module. The module writes and reads 1 byte of data. The data is first written to the given I2C address and then it is read from the same address. The programming procedure is the same as described in Section 5.7. When run, the simulation produces the results shown in Figure 6-3.

Figure 6-3: AXI I2C Bus Interface Core Test Bench Simulation Output



6.6 Synthesis and Implementation

For details about synthesis and implementation see the *Vivado Design Suite User Guide - Designing with IP*.