AXI Interrupt Controller (INTC) v4.1

LogiCORE IP Product Guide

Vivado Design Suite

PG099 November 14, 2018

XILINX

Table of Contents

Appendix C: Additional Resources and Legal Notices

Introduction

The LogiCORE™ IP AXI Interrupt Controller (INTC) core receives multiple interrupt inputs from peripheral devices and merges them into an interrupt output to the system processor. The registers used for storing interrupt vector addresses, checking, enabling and acknowledging interrupts are accessed through the AXI4-Lite interface.

Features

- Register access through the AXI4-Lite interface.
- Fast Interrupt mode.
- Supports up to 32 interrupts. Cascadable to provide additional interrupt inputs.
- Bus or single interrupt output.
- Priority between interrupt requests is determined by vector position. The least significant bit (LSB, in this case bit 0) has the highest priority.
- Interrupt Enable Register for selectively enabling individual interrupt inputs.
- Master Enable Register for enabling interrupts request output.
- Each input is configurable for edge or level sensitivity.
- Output interrupt request pin is configurable for edge or level generation.
- Configurable Software Interrupt capability.
- Support for nested interrupts.

LogiCORE IP Facts Table	
Core Specifics	
Supported Device Family[1]	UltraScale+™ UltraScale™ Zynq®-7000 SoC 7 Series
Supported User Interfaces	AXI4-Lite
Resources	Performance and Resource Utilization web page
Provided with Core	
Design Files	RTL
Example Design	Not Provided
Test Bench	Not Provided
Constraints File	Not Applicable
Simulation Model	Not Applicable
Supported S/W Driver[2]	Standalone
Tested Design Flows[3]	
Design Entry	Vivado® Design Suite
Simulation	For supported simulators, see the Xilinx Design Tools: Release Notes Guide.
Synthesis	Vivado Synthesis
Support	
Provided by Xilinx at the Xilinx Support web page	

Notes:

1. For a complete list of supported devices, see the Vivado IP catalog.
2. Standalone driver details can be found in the SDK directory (<install_directory>/SDK/<release>/data/embeddedsw/doc/xilinx_drivers.htm). Linux OS and driver support information is available from the Xilinx Wiki page.
3. For the supported versions of the tools, see the Xilinx Design Tools: Release Notes Guide.

Overview

The LogiCORE™ IP INTC core concentrates multiple interrupt inputs from peripheral devices to a single interrupt output to the system processor. The registers used for checking, enabling, and acknowledging interrupts are accessed through the AXI4-Lite interface.

Figure 1-1 illustrates the top-level block diagram for the AXI INTC core. The three main blocks in the AXI INTC core are described in this section.

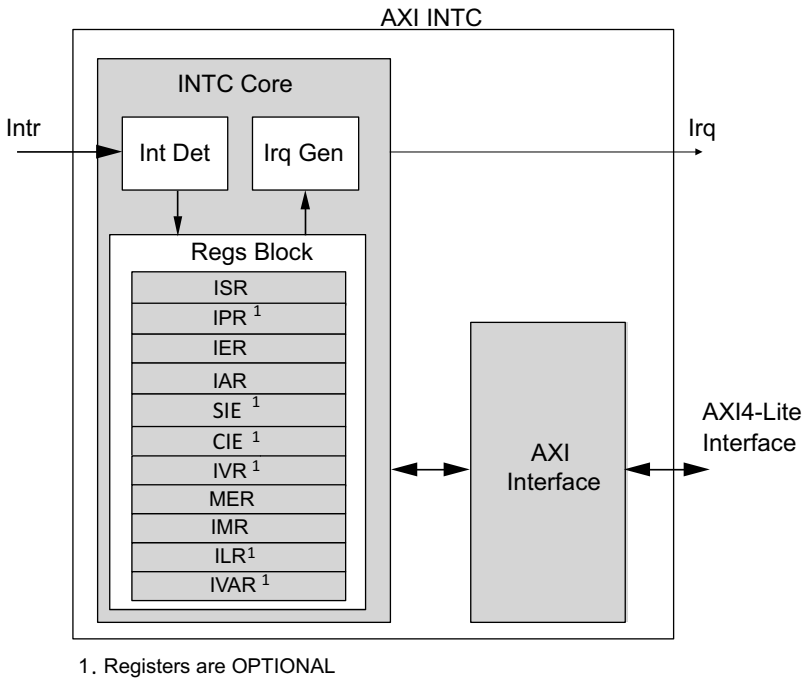

Figure 1-1: **AXI INTC Core Block Diagram**

- **Registers Block**: This block contains control and status registers. They are accessed through the AXI4-lite slave interface. For a detailed description of the AXI INTC core registers, see Register Space.
- **Interrupt Detection**: This block detects the interrupts input. It can be configured for either level or edge detection for each interrupt input.
- **Interrupt Generation**: This block performs the following functions:
  - Generates the final output interrupt from the interrupt controller core.
  - Interrupt sensitivity is determined by the configuration parameters.

- Checks for enable conditions in control registers (MER and IER) for interrupt generation.
- Resets the interrupt after acknowledge.
- Writes the vector address of the active interrupt in IVR register and enables the IPR register for pending interrupts.

Feature Summary

Interrupt conditions are captured by the AXI INTC core and retained until explicitly acknowledged. Interrupts can be enabled/disabled either globally or individually. The processor is signaled with an interrupt condition when all interrupts are globally enabled, and at least one captured interrupt is individually enabled.

Edge-Sensitive and Level-Sensitive Modes

Two modes of interrupts are supported, as shown in Figure 1-2.

- **Edge-sensitive**: Records a new interrupt condition when an active edge occurs on the interrupt input, and an interrupt condition does not already exist. (The polarity of the active edge, rising or falling, is a per-input option.) The interrupt is recorded irrespective of whether it is enabled or not, and is retained until acknowledged. Any active edges during this time have no effect.
- **Level-sensitive**: Records an interrupt condition any time the input is at the active level and the interrupt condition does not already exist. (The polarity of the active level, High or Low, is a per-input option.) The interrupt is recorded irrespective of whether it is enabled or not, and is retained until acknowledged even if the input level becomes inactive during this time.

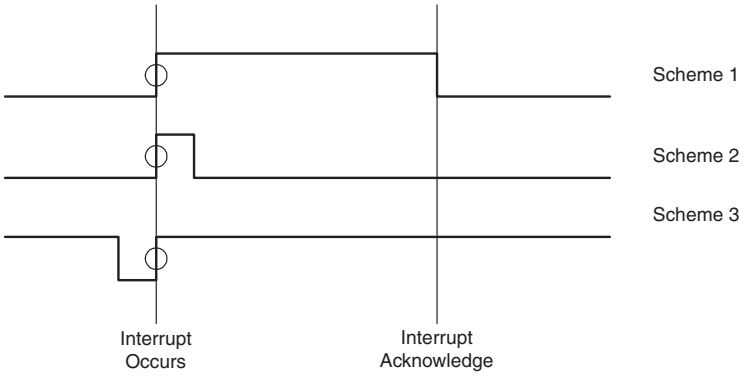

Figure 1-2: **Schemes for Generating Edges**

Fast Interrupt Mode

Each device connected to the AXI INTC core can use either normal or fast interrupt mode, based on the latency requirement. Fast interrupt mode can be chosen for designs requiring lower latency. Fast interrupt mode is enabled by setting the corresponding bit in the Interrupt Mode register (IMR).

The interrupt vector address is taken from the corresponding IVAR or IVEAR register, and sent to the processor via the `interrupt_address` port. This allows the processor to jump directly to the interrupt service routine.

The interrupt is acknowledged through `processor_ack` ports driven by the processor for interrupts configured in fast interrupt mode. The IRQ generated is cleared based on the `processor_ack` signal, and the corresponding IAR bit is updated after acknowledgment is received by `processor_ack`.

Cascade Mode

When the system requires more than 32 interrupts, it is necessary to expand the AXI INTC core capability to handle more interrupt. This can be achieved by instantiating one or more additional cores, and setting the Cascade Mode parameters accordingly. For additional details, see Cascade Mode Interrupt in Chapter 3.

Software Interrupts

The core also supports a configurable number of software interrupts, which are primarily intended for inter-processor interrupts in multi-processor systems. These interrupts are triggered by software writing to the Interrupt Status Register.

Nested Interrupts

The core provides support for nested interrupts, by implementing an Interrupt Level Register. This can be used by software to prevent lower priority interrupts from occurring when handling an interrupt, thus allowing interrupts to be enabled during interrupt handling to immediately take a higher priority interrupt. Software must save and restore the Interrupt Level Register and return address.

Because the processor jumps directly to the unique Interrupt vector address to service a particular interrupt when using fast interrupt mode, the user interrupt service routine code itself must save and restore the Interrupt Level Register and Return Address in this case. In normal interrupt mode, this is handled by the software driver.

Licensing and Ordering Information

This Xilinx® LogiCORE IP module is provided at no additional cost with the Xilinx Vivado® Design Suite under the terms of the Xilinx End User License. Information about this and other Xilinx LogiCORE IP modules is available at the Xilinx Intellectual Property page. For information about pricing and availability of other Xilinx LogiCORE IP modules and tools, contact your local Xilinx sales representative.

Product Specification

The AXI INTC core receives multiple interrupt inputs from peripheral devices and merges them to a single interrupt output to the system processor. The registers used for storing interrupt vector addresses, checking, enabling and acknowledging interrupts are accessed through the AXI4-Lite interface.

Performance

Maximum Frequencies

For details about performance, visit Performance and Resource Utilization.

Resource Utilization

For details about resource utilization, visit Performance and Resource Utilization.

Port Descriptions

This section describes both the input and output ports and the design parameters that are used to tailor the AXI INTC core for your design.

I/O Signals

The AXI INTC core I/O signals are listed and described in Table 2-1.

Table 2-1: **I/O Signal Description**

Signal Name	Interface	I/O	Initial State	Description
AXI Global System Signals				
s_axi_aclk	S_AXI	I	-	AXI Clock
s_axi_aresetn	S_AXI	I	-	AXI Reset, active-Low
AXI Interface Signals				
s_axi_*	S_AXI	I	-	For a description of AXI4, AXI4-Lite and AXI Stream signals, see the *Vivado Design Suite AXI Reference Guide* (UG1037) [Ref 2].
INTC Interface Signals				
intr[No. of interrupts - 1:0](1)(3)	INTC	I	-	Interrupt inputs
irq	INTC	O	0x1	Interrupt request output
interrupt_address[w - 1:0](2)	INTC	O	0x0	Interrupt address output (w = C_ADDR_WIDTH, 32 to 64 bits)
processor_ack[1:0]	INTC	I	-	Interrupt acknowledgment input. 00 - No interrupt received 01 - Set when processor branches to interrupt routine 10 - Set when processor returns from interrupt routine by executing RTID 11 - Set when processor enables interrupts
processor_clk	INTC	I	-	Processor clock
processor_rst	INTC	I	-	Processor reset, active-High
irq_in	INTC	I	-	This port is applicable only when **Enable Cascade Interrupt Mode** is selected in the Vivado IDE, and should be connected from the downstream AXI INTC instance irq port.
interrupt_address_in[w - 1:0]	INTC	I	-	This port is applicable only when **Enable Cascade Interrupt Mode** and **Enable Fast Interrupt Logic** are selected in the Vivado IDE, and should be connected from the downstream AXI INTC interrupt_address port (w = C_ADDR_WIDTH, 32 to 64 bits).
processor_ack_out[1:0]	INTC	O	0x0	This port is applicable only when **Enable Cascade Interrupt Mode** and **Enable Fast Interrupt Logic** are selected in the Vivado IDE, and should be connected to the downstream processor_ack port.

Notes:

1. Intr(0) is always the highest priority interrupt and each successive bit to the left has a corresponding lower interrupt priority.
2. Interrupt_address always drives the vector address of highest priority interrupt.
3. Each of the interrupt inputs is treated as synchronous to the AXI clock unless the corresponding bit in the parameter C_ASYNC_INTR is set. In that case, the input is synchronized with the number of flip-flops defined by the parameter C_NUM_SYNC_FF.

Design Parameters

To allow you to obtain an AXI INTC core that is uniquely tailored for your system, certain features can be parameterized in the AXI INTC design. This allows you to configure a design that uses the resources required by the system only and that operates with the best possible performance. The features that can be parameterized in the AXI INTC core are shown in Table 2-2.

When certain features are disabled by parameterization, the corresponding logic is removed. Unused input ports are ignored, and unused output ports are set to constant values.

Table 2-2: **Design Parameters**

Generic	Feature/Description	Parameter Name	Allowable Values	Default Value	VHDL Type
System Parameter					
G1	Target FPGA family	C_FAMILY	Supported architectures	kintex7	string
AXI Parameters					
G2	AXI address bus width	C_S_AXI_ADDR_WIDTH	9	9	integer
G3	AXI data bus width	C_S_AXI_DATA_WIDTH	32	32	integer
INTC Parameters					
G4	Number of interrupt inputs	C_NUM_INTR_INPUTS	1-32	1	integer
G5	Type of interrupt for each input(1)	C_KIND_OF_INTR	1 = Edge 0 = Level	ALL 1s	std_logic_vector
G6	Type of each edge sensitive input(1)	C_KIND_OF_EDGE	1 = Rising 0 = Falling Valid if C_KIND_OF_INTR = 1s	ALL 1s	std_logic_vector
G7	Type of each level sensitive input(1)	C_KIND_OF_LVL	1 = High 0 = Low Valid if C_KIND_OF_INTR = 0s	ALL 1s	std_logic_vector
G8	Indicates the presence of IPR	C_HAS_IPR	0 = Not Present 1 = Present	1	integer
G9	Indicates the presence of SIE	C_HAS_SIE	0 = Not Present 1 = Present	1	integer
G10	Indicates the presence of CIE	C_HAS_CIE	0 = Not Present 1 = Present	1	integer
G11	Indicates the presence of IVR	C_HAS_IVR	0 = Not Present 1 = Present	1	integer
G12	Indicates level or edge active Irq	C_IRQ_IS_LEVEL	0 = Active Edge 1 = Active Level	1	integer
G13	Indicates the sense of the Irq output	C_IRQ_ACTIVE	0 = Falling/Low 1 = Rising/High	1	std_logic

Table 2-2: **Design Parameters *(Cont'd)***

Generic	Feature/Description	Parameter Name	Allowable Values	Default Value	VHDL Type
G14	Indicates if processor clock is connected to INTC(3)(4)	C_MB_CLK_NOT_ CONNECTED	0 = Connected 1 = Not Connected	1	integer
G15	Indicates the presence of FAST INTERRUPT logic(4)	C_HAS_FAST	0 = Not Present 1 = Present	0	integer
G16	Use synchronizers in design(5)	C_DISABLE_ SYNCHRONIZERS	1 = Not Used 0 = Used	1	integer
G17	Enable Cascade mode interrupt(6)	C_EN_CASCADE_MODE	0 = Default 1 = Enable Cascade Mode	0	integer
G18	Define cascade mode interrupt core instance as primary(7)	C_CASCADE_MASTER	0 = No Cascade Mode Master 1 = Cascade Mode Master	0	integer
G19	Number of software interrupts	C_NUM_SW_INTR	0-31	0	integer
G20	Asynchronous interrupt for each input(1)	C_ASYNC_INTR	0=Synchronous 1=Asynchronous	All 1s	std_logic_ vector
G21	Number of synchronization flip-flops	C_NUM_SYNC_FF	0-7	2	integer
G22	Indicates the presence of ILR, to support nested interrupts	C_HAS_ILR	0 = Not Present 1 = Present	0	integer
G23	Selects Bus or Single interrupt output connection	C_IRQ_CONNECTION	0 = Bus 1 = Single	0	integer
G24	Width of interrupt address	C_ADDR_WIDTH	32-64	32	integer

Notes:

1. The interrupt input is a little-endian vector with the same width as the data bus and contains either a 0 or 1 in each position.
2. Synchronizers in the design can be disabled if the processor clock and AXI clock are identical. This reduces the core area and latencies introduced by the synchronizers in passing the IRQ to the processor.
3. C_MB_CLK_NOT_CONNECTED should be set to 1 when the processor clock is not connected to the AXI INTC core. When the processor clock is not connected, the IRQ to the processor is generated on the AXI clock. The synchronizers in the design are disabled when the processor clock is not connected.
4. When processor_clk is connected, a DRC error is generated if the same clock is not connected to both the processor and the AXI INTC core.
5. The C_DISABLE_SYNCHRONIZERS parameter, by default, defines if necessary synchronization logic is added for internal signals.
6. C_EN_CASCADE_MODE can be set to 1 when there are more than 32 interrupts to handle in the system. For each successive set of 31 interrupts, one additional AXI INTC core has to be instantiated. For the primary and each intermediate instance of AXI INTC this parameter should be set to 1. Only the final instance of AXI INTC should have this parameter set to 0. See Cascade Mode Interrupt for more information.
7. C_CASCADE_MASTER should only be set to 1 for the primary instance of the AXI INTC core. This parameter is only available when the C_EN_CASCADE_MODE parameter is set to 1. The primary instance directly interfaces with the processor. See Cascade Mode Interrupt for more information.

Dependencies Between Parameters and I/O Signals

The dependencies between the AXI INTC core design parameters and I/O signals are described in Table 2-3.

Table 2-3: **Parameter-I/O Signal Dependencies**

Generic or Port	Name	Affects	Depends On	Relationship Description
Design Parameters				
G2	C_S_AXI_ADDR_WIDTH	P3, P13	-	Defines the width of the ports
G3	C_S_AXI_DATA_WIDTH	P6, P7, P16	-	Defines the width of the ports
G15	C_HAS_FAST	P22, P23, P26, P27	-	Ports are valid if the generic C_HAS_FAST = 1
G15	C_HAS_FAST	G14, G23	G23	C_MB_CLK_NOT_CONNECTED and C_IRQ_CONNECTION have to be 0 when C_HAS_FAST is set to 1.
G17	C_EN_CASCADE_MODE	P26, P27	-	This parameter should be set for primary and intermediate cores when using cascade mode.
G18	C_CASCADE_MASTER	P26, P27	-	This parameter should be set for the primary core when using cascade mode.
G23	C_IRQ_CONNECTION	G15	G15	C_HAS_FAST has to be 0 when C_IRQ_CONNECTION is set to 1.
I/O Signals				
P3	S_AXI_AWADDR[C_S_AXI_ADDR_WIDTH-1:0]	-	G2	Port width depends on the generic C_S_AXI_ADDR_WIDTH
P6	S_AXI_WDATA[C_S_AXI_DATA_WIDTH-1:0]	-	G3	Port width depends on the generic C_S_AXI_DATA_WIDTH
P7	S_AXI_WSTB[C_S_AXI_DATA_WIDTH/8-1:0]	-	G3	Port width depends on the generic C_S_AXI_DATA_WIDTH
P13	S_AXI_ARADDR[C_S_AXI_ADDR_WIDTH -1:0]	-	G2	Port width depends on the generic C_S_AXI_ADDR_WIDTH
P16	S_AXI_RDATA[C_S_AXI_DATA_WIDTH -1:0]	-	G3	Port width depends on the generic C_S_AXI_DATA_WIDTH
P22	Interrupt_address[C_ADDR_WIDTH-1:0]	-	G15, G24	Port is valid if the generic C_HAS_FAST = 1 Port width depends on the generic C_ADDR_WIDTH
P23	Processor_ack[1:0]	-	G15	Port is valid if the generic C_HAS_FAST = 1

Table 2-3: **Parameter-I/O Signal Dependencies *(Cont'd)***

Generic or Port	Name	Affects	Depends On	Relationship Description
P26	Interrupt_address_in[C_ADDR_WIDTH-1:0]	-	G15, G17, G18, G24	Port existence is depend upon the FAST Mode interrupt as well as Cascade mode interrupt Port width depends on the generic C_ADDR_WIDTH
P27	Processor_ack_out[1:0]	-	G15, G17, G18	Port existence is depend upon the FAST Mode interrupt as well as Cascade mode interrupt
P28	Irq_in	-	G17	Port existence is dependent on Cascade mode interrupt

Register Space

All AXI INTC registers listed in Table 2-4 are accessed through the AXI4-Lite interface. Each register is accessed on a 4-byte boundary. The AXI INTC registers are read as little-endian data.

Table 2-4: **Register Address Mapping**

Address Offset	Register Name	Description
00h	ISR	Interrupt Status Register (ISR)
04h	IPR	Interrupt Pending Register (IPR)
08h	IER	Interrupt Enable Register (IER)
0Ch	IAR	Interrupt Acknowledge Register (IAR)
10h	SIE	Set Interrupt Enables (SIE)
14h	CIE	Clear Interrupt Enables (CIE)
18h	IVR	Interrupt Vector Register (IVR)
1Ch	MER	Master Enable Register (MER)
20h	IMR	Interrupt Mode Register (IMR)
24h	ILR	Interrupt Level Register (ILR)
100h to 17Ch	IVAR	Interrupt Vector Address Register (IVAR)
200h to 2FCh	IVEAR	Interrupt Vector Extended Address Register (IVEAR)

Interrupt Status Register (ISR)

When read, the contents of this register indicate the presence or absence of an active interrupt signal. Bits that are 0 are not active. Each bit in this register that is set to a 1 indicates an active interrupt. This is an active interrupt signal on the corresponding hardware interrupt input for bits up to **Number of Peripheral Interrupts** defined in the Customize IP dialog box in the Vivado Design Suite (parameter C_NUM_INTR_INPUTS). The number of remaining bits is defined by **Number of Software Interrupts** in the Customize IP dialog box in the Vivado Design Suite (parameter C_NUM_SW_INTR). These bits provide the ability to generate software interrupts by writing to the ISR. The total number of bits in the register is **Number of Peripheral Interrupts** + **Number of Software Interrupts**.

The bits in the ISR are independent of the interrupt enable bits in the IER. See the Interrupt Enable Register (IER), page 18 for the interrupt status bits that are masked by disabled interrupts.

The ISR register bits up to **Number of Peripheral Interrupts** is writable by software until the Hardware Interrupt Enable (HIE) bit in the MER has been set, whereas the remaining bits (if any) can still be set by software. Given these restrictions, when this register is written to, any data bits that are set to 1 activate the corresponding interrupt. For the bits up to

Number of Peripheral Interrupts, this has the same effect as if a hardware input became active. Data bits that are zero have no effect.

This functionality allows the software to generate interrupts for test purposes until the HIE bit has been set, and to generate software interrupts at any time. After HIE has been set (enabling the hardware interrupt inputs), then setting the bits up to **Number of Peripheral Interrupts** in this register has no effect.

If there are fewer interrupt inputs than the width of the data bus, writing a 1 to a non-existing interrupt input does nothing and reading it returns 0.

The Interrupt Status Register (ISR) is shown in Figure 2-1 and the bits are described in Table 2-5.

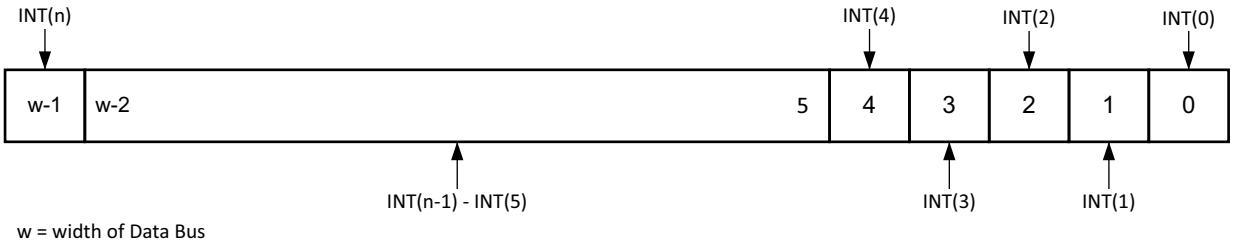

Figure 2-1: **Interrupt Status Register (ISR)**

Table 2-5: **Interrupt Status Register Bit Definitions**

Bits	Name	Reset Value	Access	Description
(w[(1)]-1):0	INT(n)-INT(0) (n ≤ w-1)	0x0	Read / Write	Interrupt (n) - Interrupt (0) 0 - Not Active 1 - Active

Notes:

1. w - Width of Data Bus

Interrupt Pending Register (IPR)

This is an optional read-only register in the AXI INTC and can be set in Vivado Design Suite Customize IP dialog box by checking **Enable Interrupt Pending Register** (parameter C_HAS_IPR). Reading the contents of this register indicates the presence or absence of an active interrupt that is also enabled. This register is used to reduce interrupt processing latency by reducing the number of reads of the INTC by one.

Each bit in this register is the logical AND of the bits in the ISR and the IER. If there are fewer interrupt inputs than the width of the data bus, reading a non-existing interrupt returns zero. The Interrupt Pending Register (IPR) is shown in Figure 2-2 and the bits are described in Table 2-6.

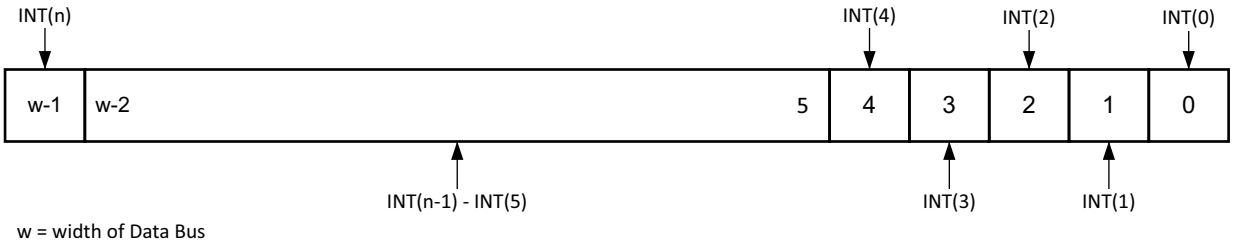

Figure 2-2: **Interrupt Pending Register**

Table 2-6: **Interrupt Pending Register Bit Definitions**

Bits	Name	Reset Value	Access	Description
(w[(1)]-1):0	INT(n)-INT(0) (n ≤ w-1)	0x0	Read / Write	Interrupt (n) - Interrupt (0) 0 - Not Active 1 - Active

Notes:

1. w - Width of Data Bus

Interrupt Enable Register (IER)

This is a read-write register. Writing a 1 to a bit in this register enables, or unmasks, the corresponding ISR bit, allowing it to affect the `irq` output. An IER bit set to 0 does not inhibit an interrupt condition from being captured, just passing it to the processor. Writing a 0 to a bit disables, or masks, the generation of interrupt output for the corresponding interrupt.

When an interrupt is disabled, the interrupt event occurs but is not passed to the processor. Disabling an active interrupt prevents that interrupt from affecting the `irq` output, but as soon as it is re-enabled the interrupt immediately sets the `irq` output.

An interrupt must be cleared by writing to the Interrupt Acknowledge Register as described below. Reading the IER indicates which interrupt inputs are enabled, where a 1 indicates the input is enabled and a 0 indicates the input is disabled.

If there are fewer interrupt inputs than the width of the data bus, writing a 1 to a non-existing interrupt input does nothing and reading it returns 0. The Interrupt Enable Register (IER) is shown in Figure 2-3 and the bits are described in Table 2-7.

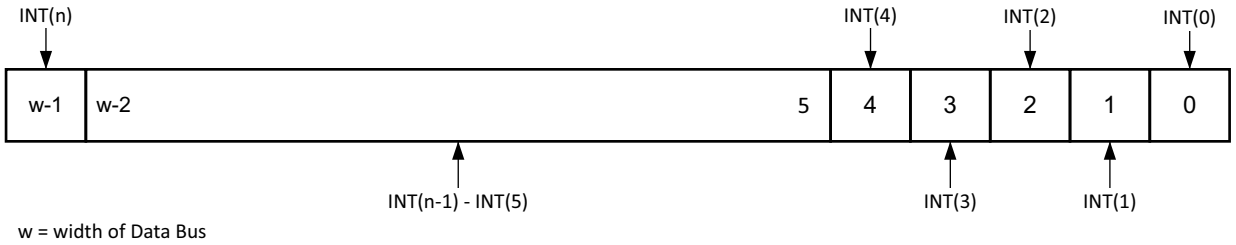

Figure 2-3: **Interrupt Enable Register**

Table 2-7: **Interrupt Enable Register Bit Definitions**

Bits	Name	Reset Value	Access	Description
(w[(1)]-1):0	INT(n)-INT(0) (n ≤ w-1)	0x0	Read / Write	Interrupt (n) - Interrupt (0) 0 - Not Active 1 - Active

Notes:

1. w - Width of Data Bus

Interrupt Acknowledge Register (IAR)

The IAR is a write-only register that clears the interrupt request associated with selected interrupts. Writing 1 to a bit in IAR clears the corresponding bit in the ISR, and also clears the bit itself in IAR.

In fast interrupt mode, bits in the IAR are automatically cleared by using the information from the `processor_ack` port. In normal interrupt mode, bits in the IAR are cleared by writing to the register via the AXI interface.

Writing a 1 to a bit location in the IAR clears the interrupt request that was generated by the corresponding interrupt input. An interrupt that is active and masked by writing a 0 to the corresponding bit in the IER remains active until cleared by acknowledging it. Unmasking an active interrupt causes an interrupt request output to be generated (if the ME bit in the MER is set).

Writing 0 does nothing as does writing 1 to a bit that does not correspond to an active input, or for which an interrupt does not exist. The IAR is shown in Figure 2-4 and the bits are described in Table 2-8.

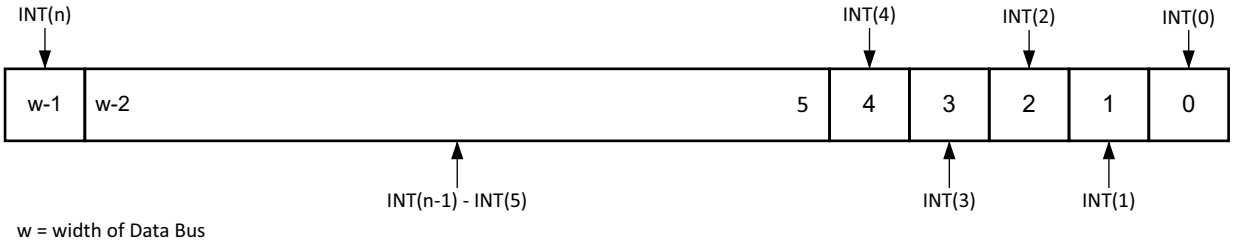

Figure 2-4: **Interrupt Acknowledge Register**

Table 2-8: **Interrupt Acknowledge Register Bit Definitions**

Bits	Name	Reset Value	Access	Description
(w[(1)]-1):0	INT(n)-INT(0) (n ≤ w-1)	0x0	Read / Write	Interrupt (n) - Interrupt (0) 0 - Not Active 1 - Active

Notes:

1. w - Width of Data Bus

Set Interrupt Enables (SIE)

SIE is a register used to set the IER bits in a single atomic operation, rather than using a read-modify-write sequence. Writing a 1 to a bit location in SIE sets the corresponding bit in the IER. Writing 0 does nothing, as does writing 1 to a bit location that corresponds to a non-existing interrupt.

The SIE is optional in the AXI INTC core and can be enabled by selecting **Enable Set Interrupt Enable Register** in the Vivado Design Suite Customize IP dialog box (parameter C_HAS_SIE).

The SIE register is shown in Figure 2-5 and the bits are described in Table 2-9.

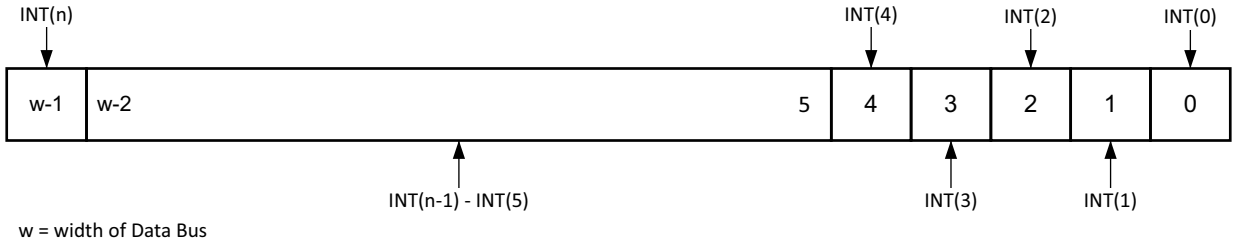

Figure 2-5: **Set Interrupt Enable (SIE) Register**

Table 2-9: **Set Interrupt Enable (SIE) Register Bit Definitions**

Bits	Name	Reset Value	Access	Description
(w[(1)]-1):0	INT(n)-INT(0) (n ≤ w-1)	0x0	Read / Write	Interrupt (n) - Interrupt (0) 0 - Not Active 1 - Active

Notes:

1. w - Width of Data Bus

Clear Interrupt Enables (CIE)

CIE is a register used to clear IER bits in a single atomic operation, rather than using a read-modify-write sequence. Writing a 1 to a bit location in CIE clears the corresponding bit in the IER. Writing 0 does nothing, as does writing 1 to a bit location that corresponds to a non-existing interrupt.

The CIE is optional in the AXI INTC core and can be enabled by selecting **Enable Clear Interrupt Enable Register** in the Vivado Design Suite Customize IP dialog box (parameter C_HAS_CIE).

The CIE register is shown in Figure 2-6 and the bits are described in Table 2-10.

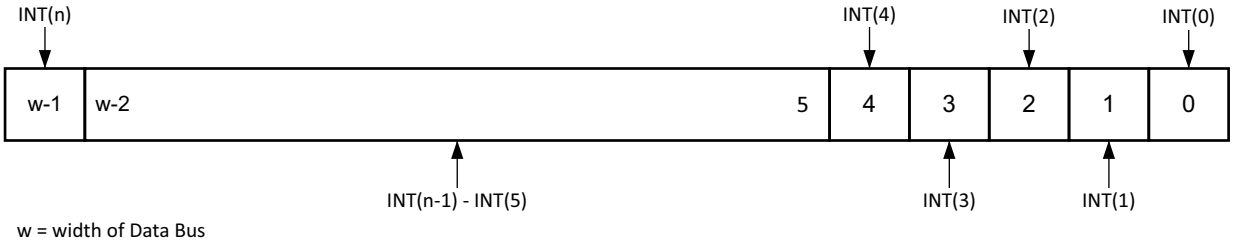

Figure 2-6: **Clear Interrupt Enable (CIE) Register**

Table 2-10: **Clear Interrupt Enable Register Bit Definitions**

Bits	Name	Reset Value	Access	Description
(w[(1)]-1):0	INT(n)-INT(0) (n ≤ w-1)	0x0	Read / Write	Interrupt (n) - Interrupt (0) 0 - Not Active 1 - Active

Notes:

1. w - Width of Data Bus

Interrupt Vector Register (IVR)

The IVR is a read-only register and contains the ordinal value of the highest priority, enabled, and active interrupt input. INT0 (always the LSB) is the highest priority interrupt input and each successive input to the left has a correspondingly lower interrupt priority. If no interrupt inputs are active then the IVR contains all ones. The IVR acts as an index to the correct Interrupt Vector Address.

The Interrupt Vector Register (IVR) is shown in Figure 2-7 and described in Table 2-11.

Figure 2-7: **Interrupt Vector Register (IVR)**

Table 2-11: **Interrupt Vector Register (IVR) Bit Definitions**

Bits	Name	Reset Value	Access	Description
(w[(1)]-1):0	Interrupt Vector Number	0x0	Read	Ordinal of highest priority, enabled, active interrupt input

Notes:

1. w - Width of Data Bus

Master Enable Register (MER)

This is a 2-bit read-write register. The least significant bit contains the Master Enable (ME) bit and the next bit contains the Hardware Interrupt Enable (HIE) bit. Writing a 1 to the ME bit enables the Irq output signal. Writing a 0 to the ME bit disables the Irq output, effectively masking all interrupt inputs.

The HIE bit is a write-once bit. At reset, this bit is reset to 0, allowing the software to write to the ISR to generate hardware interrupts for testing purposes, and disabling any hardware interrupt inputs. Writing a 1 to this bit enables the hardware interrupt inputs and disables the software generated inputs. However, any software interrupts configured with C_NUM_SW_INTR remain writable. Writing a 1 also disables any further changes to this bit until a reset occurs. Writing to any other bit location does nothing. When read, this register reflects the state of the ME and HIE bits. All other bits read as 0.

The Master Enable Register (MER) is shown in Figure 2-8 and is described in Table 2-12.

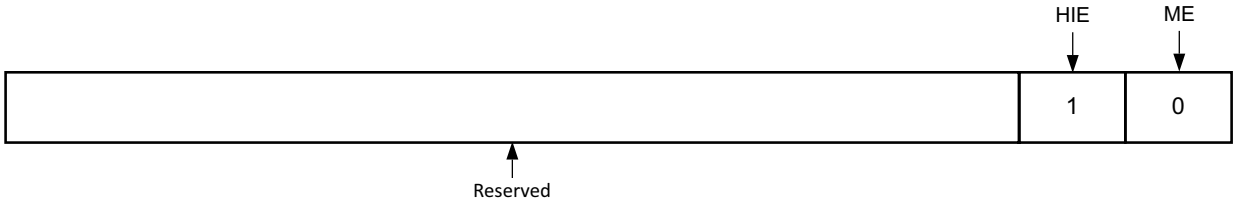

Figure 2-8: **Master Enable Register (MER)**

Table 2-12: **Master Enable Register Bit Definitions**

Bits	Name	Reset Value	Access	Description
(w[(1)]-1):2	Reserved	0x0	N/A	Reserved
1	HIE	0	Read / Write	Hardware Interrupt Enable 0 = Read - Generating HW interrupts from SW enabled Write - No effect 1 = Read - HW interrupts enabled Write - Enable HW interrupts
0	ME	0	Read / Write	Master IRQ Enable 0 = Irq disabled - All interrupts disabled 1 = Irq enabled - All interrupts can be enabled

Notes:

1. w - Width of Data Bus

Interrupt Mode Register (IMR)

This register exists only when **Enable Fast Interrupt Mode Logic** is selected in the Customize IP dialog box in the Vivado Design Suite (parameter C_HAS_IMR). The IMR register is used to set the interrupt mode of the connected interrupts. All the interrupts can be individually configured by setting the corresponding interrupt bit position in IMR. Writing 0 to any bit position processes the corresponding interrupt in normal interrupt mode. Writing 1 to any bit position processes the corresponding interrupt in fast interrupt mode. Unused bit positions in the IMR register return zero.

The Interrupt Mode Register (IMR) is shown in Figure 2-9 and is described in Figure 2-13.

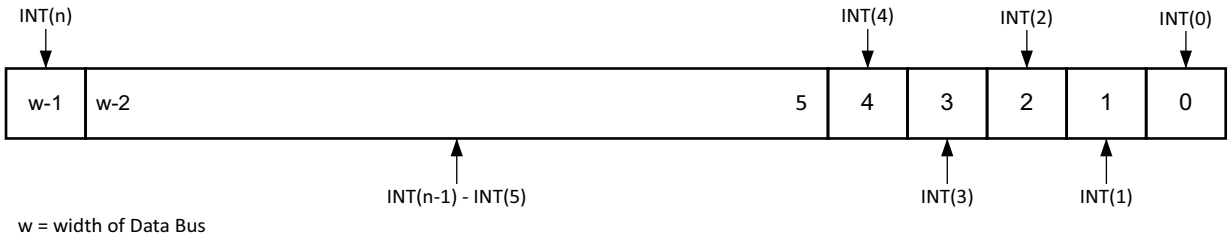

Figure 2-9: **Interrupt Mode Register (IMR)**

Table 2-13: **Interrupt Mode Register (IMR) Bit Definitions**

Bits	Name	Reset Value	Access	Description
(w[(1)]-1):0	INT(n)-INT(0) (n ≤ w-1)	0x0	Read / Write	Interrupt (n) - Interrupt (0) 0 – Normal Interrupt mode 1 – Fast Interrupt mode

Notes:

1. w - Width of Data Bus

Interrupt Level Register (ILR)

The Interrupt Level Register (ILR) is a read-write register that contains the ordinal value of the highest priority interrupt prevented from generating a processor IRQ. The ILR provides a method to block lower priority interrupts in order to support nested interrupt handling.

When the ILR is 0, no interrupt is allowed to generate IRQ, when the ILR is 1 only INT(0) is allowed to generate IRQ, etc. If all interrupts are allowed to generate IRQ, the ILR should contain all ones.

The ILR is shown in Figure 2-10 and described in Table 2-14.

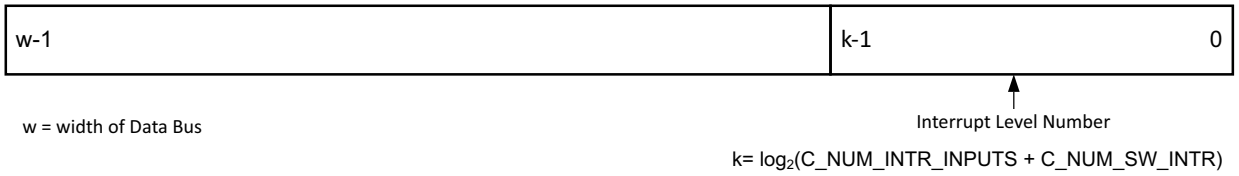

Figure 2-10: **Interrupt Level Register (ILR)**

Table 2-14: **Interrupt Level Register (ILR) Bit Definitions**

Bits	Name	Reset Value	Access	Description
(w[(1)]-1):0	Interrupt Level Number	0xFFFFFFFF	Read / Write	Ordinal of highest priority interrupt not allowed to generate IRQ

Notes:

1. w - Width of Data Bus

Interrupt Vector Address Register (IVAR)

Interrupt Vector Extended Address Register (IVEAR)

These are read-write registers with the number of registers defined by **Number of Peripheral Interrupts** + **Number of Software Interrupts** in the Customize IP dialog box in the Vivado Design Suite (parameters C_NUM_INTR_INPUTS and C_NUM_SW_INTR). The registers are only available when **Enable Fast Interrupt Logic** is selected (parameter C_HAS_FAST).

Note: IVAR registers are 32-bits wide, and IVEAR registers are up to 64-bits wide, depending on the setting of the parameter C_ADDR_WIDTH. Software should use IVAR for 32-bit vector addresses, and IVEAR for extended vector addresses with more than 32 bits.

Each interrupt connected to the Interrupt controller has a unique Interrupt vector address that the processor jumps to for servicing that particular interrupt. In normal interrupt mode (C_HAS_FAST = 0), the interrupt vector addresses are determined by the software drivers or application. In fast interrupt mode (C_HAS_FAST = 1), the service routine address is driven by the interrupt controller along with the IRQ. IVAR or IVEAR registers are programmed with the corresponding peripheral interrupt vector address during initialization. When a particular interrupt is not handled as a fast interrupt (IMR(i) = 0), the corresponding IVAR or IVEAR register should be programmed with the normal interrupt mode processor interrupt vector address.

These registers store the interrupt vector addresses of all the **Number of Peripheral Interrupts** + **Number of Software Interrupts**. The address of the interrupt with highest priority is passed to the processor.

If not all 32 interrupts are used, reading the unused register address returns zero. Writing to any unused register does nothing.

IVAR or IVEAR is accessed through the AXI interface. The registers are in the AXI clock domain and are used in the processor clock domain to provide the interrupt vector address. Since the registers are not synchronized to the processor clock domain, registers should only be changed when the corresponding interrupt is disabled.

An Interrupt Vector Address Register (IVAR) or Interrupt Vector Extended Address Register (IVEAR) is shown in Figure 2-11 and is described in Table 2-15.

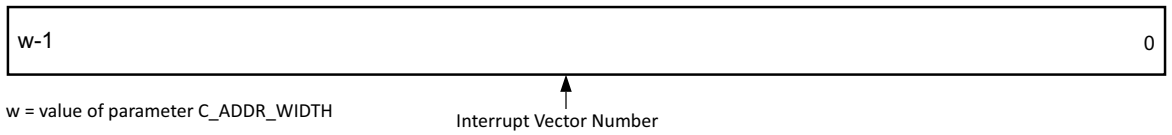

Figure 2-11: **Interrupt Vector Address Register or Interrupt Vector Extended Address Register**

Table 2-15: **IVAR or IVEAR Bit Definitions**

Bits	Name	Reset Value	Access	Description
(w[(1)]-1):0	Interrupt Vector Address	0x0	Read / Write	Interrupt vector address of the active interrupt with highest priority

Notes:

1. w - Value of C_ADDR_WIDTH parameter

Designing with the Core

Clocking

The AXI INTC core uses the AXI clock in default mode. When the processor clock is connected, the interrupt output is synchronized to the processor clock.

Resets

The AXI INTC uses `axi_aresetn`, which is active-Low. When the processor clock is connected, part of the AXI INTC logic gets reset through the `processor_rst` input signal.

Programming Sequence

During power-up or reset, the AXI INTC core is initialized to a state where all interrupt inputs and the interrupt request output are disabled. In order for the AXI INTC core to accept interrupts and request service, the following steps are required:

1. Each bit in the IER corresponding to an interrupt must be set to 1. This allows the AXI INTC core to begin accepting interrupt input signals and software interrupts. INT0 has the highest priority, and it corresponds to the least significant bit (LSB) in the IER.
2. The MER must be programmed based on the intended use of the AXI INTC core. There are two bits in the MER: the Hardware Interrupt Enable (HIE) and the Master IRQ Enable (ME). The ME bit must be set to enable the interrupt request output.
3. If software testing of hardware interrupts is to be performed, the HIE bit must remain at its reset value of 0. Software testing can now proceed by writing a 1 to any bit position in the ISR that corresponds to an existing interrupt input or software interrupt. A corresponding interrupt request is generated if that interrupt is enabled, and interrupt handling proceeds normally.
4. After software testing of hardware interrupts has been completed, or if testing is not performed, a 1 must be written to the HIE bit, which enables the hardware interrupt inputs and disables any further software generated hardware interrupts.
5. After 1 is written to the HIE bit, any further writes to this bit have no effect.

Cascade Mode Interrupt

Overview

The cascade mode functionality can be used when the processor has more than 32 interrupts. A single instance of the AXI INTC can handle a maximum of 32 interrupts. In cascade mode, there are two or more AXI INTC instances connected to a processor.

Both **Enable Cascade Interrupt Mode** (parameter C_EN_CASCADE_MODE) and **Cascade Mode Master** (parameter C_CASCADE_MASTER) have to be defined in this mode.

IMPORTANT: *Either the `irq_in` port or the 31st interrupt bit of the primary AXI INTC instance must be used to cascade from the secondary AXI INTC instances.*

Table 3-1 shows the parameter combinations used for cascade mode.

Table 3-1: **Parameter Combination and Use in Cascade Interrupt Mode**

Enable Cascade Interrupt Mode	Cascade Mode Master	Mode of Operation for the AXI INTC Instances
0	0	In this mode, there is no cascade interrupt feature available. Only one instance is allowed per processor.
0	1	Invalid. Cascade Mode Master can be set only when Enable Cascade Interrupt Mode is set.
1	0	This is applicable only for intermediate instances of the AXI INTC core. The interrupt interface ports are connected to the primary or upstream instance of the AXI INTC core. Either the irq_in port or bit 31 of intr is the cascaded interrupt port
1	1	This is applicable only when cascade mode is enabled. This parameter setting must only be used for the primary instance of AXI INTC. Either the irq_in port or bit 31 of intr is the cascaded interrupt port.

CAUTION! *The combination of Enable Cascade Interrupt Mode=0 and Cascade Mode Master=1 is not valid, and Design Rule Check (DRC) errors are issued.*

Figure 3-1 shows how cascade mode interacts with a processor, using two AXI INTC instances to handle up to 63 interrupts.

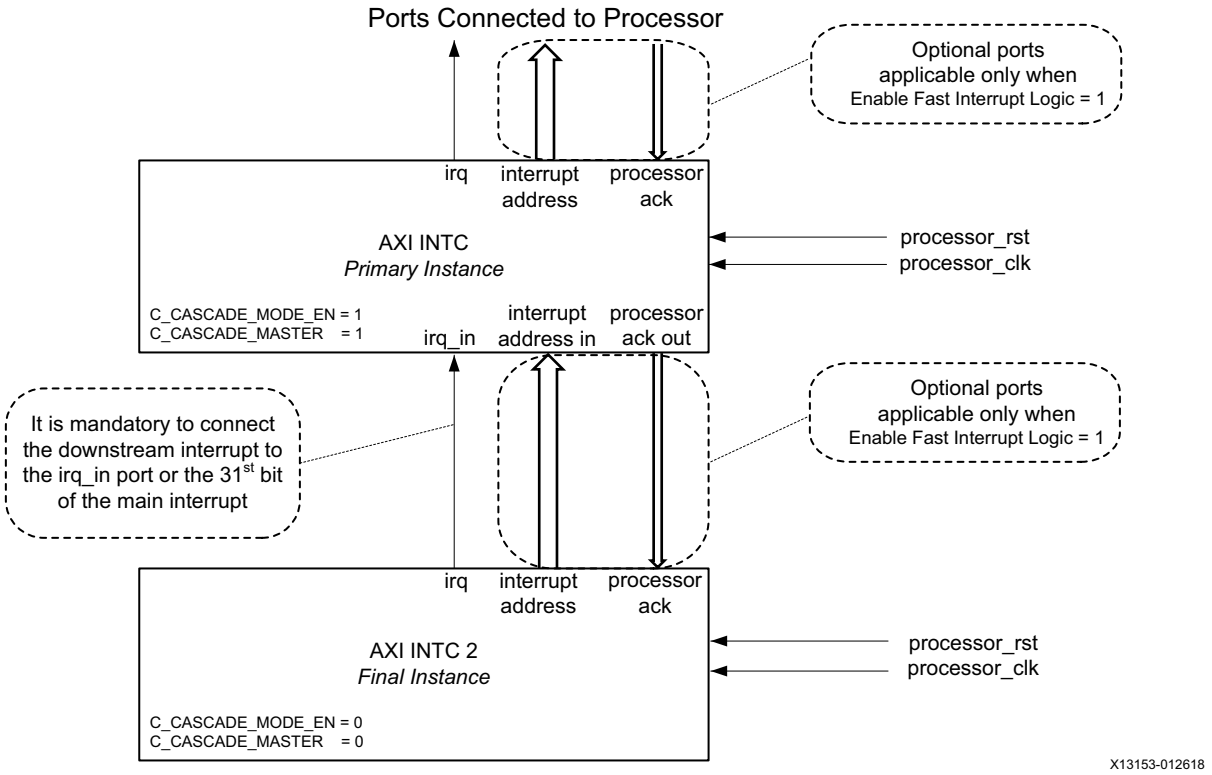

Figure 3-1: **Cascade Mode**

Depending on the type of interrupt chosen, such as Standard Mode or Fast Interrupt Mode, additional signals are enabled in the core instances. These modes are defined as follows:

- **Cascade Mode with Standard Interrupt**: In this mode, there is one new port enabled for each instance of the core. This is the `irq_in` port.
- **Cascade Mode with Fast Interrupt**: In this mode, there are three new ports enabled for each instance of the core. These ports are `irq_in`, `interrupt_address_in` and `processor_ack_out`.

See Port Descriptions for more information about these ports.

RECOMMENDED: *For Cascade Mode interrupts, Xilinx recommends that all AXI INTC core instances use the same parameter settings, in particular either Standard Mode or Fast Interrupt Mode.*

Cascade Mode Interrupt Behavior

The cascade mode of interrupts can be set by using the Enable Cascade Interrupt Mode and Cascade Mode Master parameters. As described in Table 3-1, there are three types of AXI INTC instantiations possible when cascade mode is considered.

In cascade mode the primary and any intermediate instances of AXI INTC first handle all active interrupts from `intr(0)` to `intr(30)`. Following that, they handle active inputs on the `intr(31)` bit.

Enable Cascade Interrupt Mode = 1 and Cascade Mode Master = 1

This parameter combination is only used when there are more than 32 interrupts. Use this parameter combination for the primary instance of the AXI INTC core, which directly communicates with the processor.

This instance of the AXI INTC core has the `cascade_interrupt` bus interface consisting of the ports `irq_in`, `interrupt_address_in` and `processor_ack_out`, which can be directly connected from the interrupt bus interface of a secondary instance of the AXI INTC core. In this case, the `intr(31)` bit is not available.

The primary instance has 31 available interrupt inputs.

Enable Cascade Interrupt Mode = 1 and Cascade Mode Master = 0

This parameter combination is only used when there are more than 63 interrupts. Use this parameter combination for the intermediate instances of the AXI INTC core.

The intermediate instances of the AXI INTC core also have the `cascade_interrupt` bus interface, which can be directly connected from the interrupt bus interface of the cascaded instances. In this case, the `intr(31)` bit is not available.

The intermediate instances have 31 available interrupt inputs.

Enable Cascade Interrupt Mode = 0 and Cascade Mode Master = 0

This parameter set is intended for use only for the final instance of the AXI INTC core.

The final instance has from 1 to 32 available interrupt inputs.

Fast Interrupt Connection in Cascade Mode

Figure 3-2 shows an example of how the fast interrupt of AXI INTC instances is configured, using three AXI INTC instances to handle up to 94 interrupts.

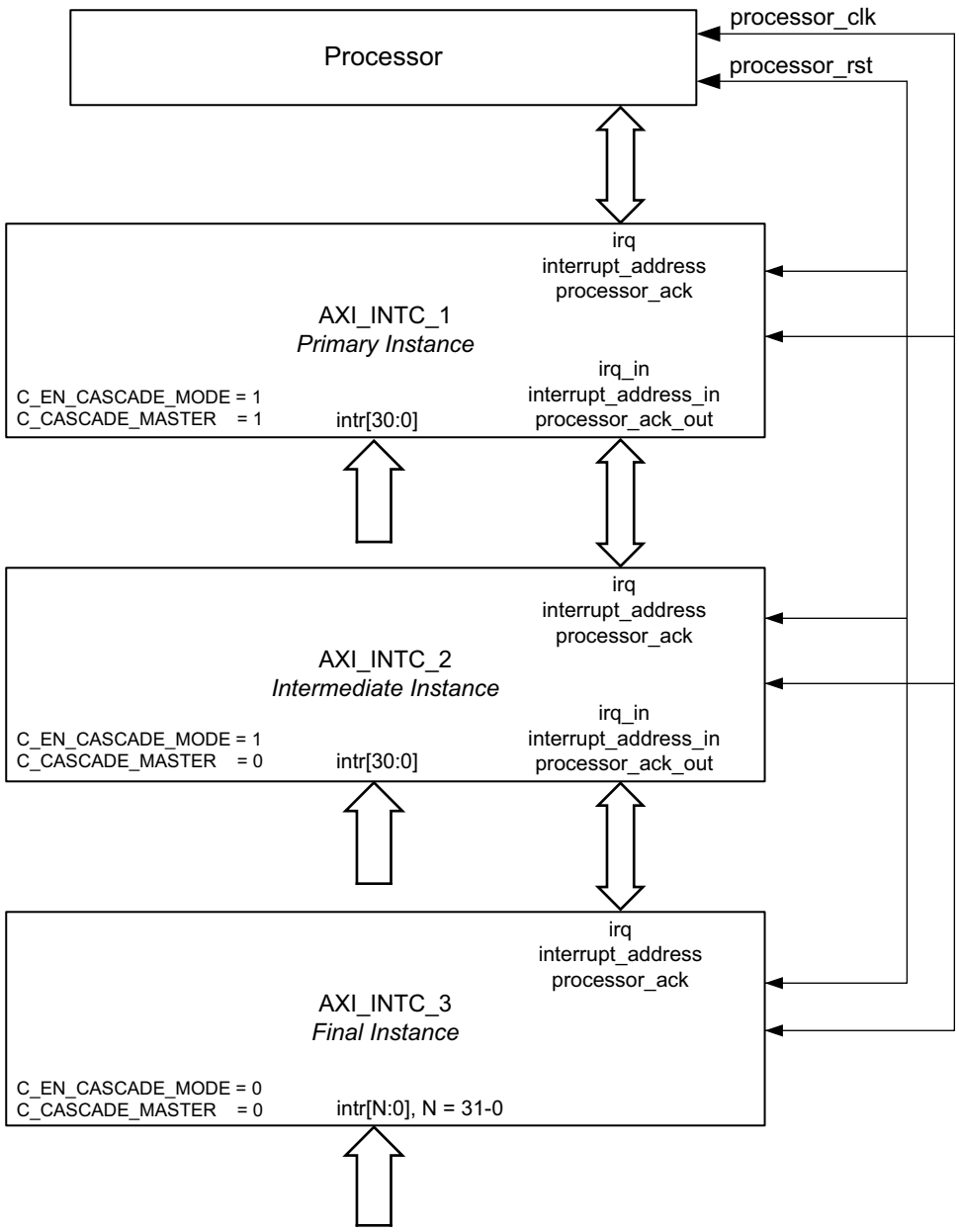

Figure 3-2: **Fast Interrupt of AXI INTC Instances**

Timing Diagrams

The timing diagrams in this section illustrate the functionality of the core.

Figure 3-3 shows the timing diagram with the following core settings:

- Configured Input Interrupt (INTR) for edge sensitive (rising)
- Output Interrupt Request (IRQ) to level sensitive (active-High)
- Disabled fast interrupt logic

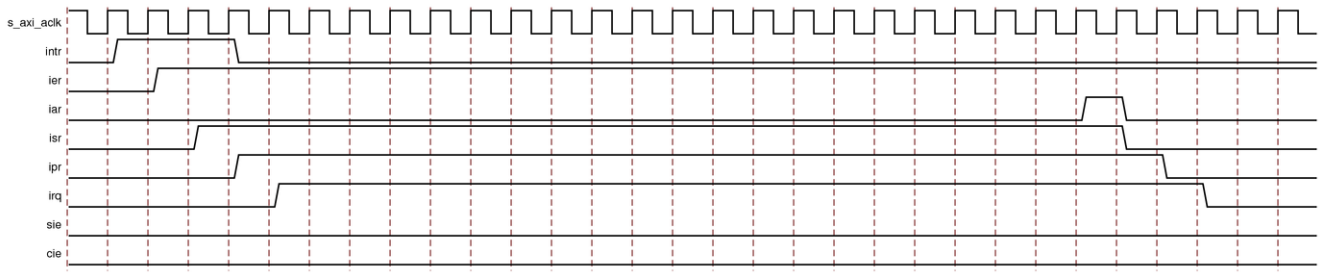

Figure 3-3: **Input - Rising Edge Sensitive, Output - High Level Sensitive**

Figure 3-4 shows the timing diagram with the following core settings:

- Configured Input Interrupt (INTR) for Edge sensitive (rising)
- Output Interrupt Request (IRQ) to Edge sensitive (rising)
- Disabled fast interrupt logic.

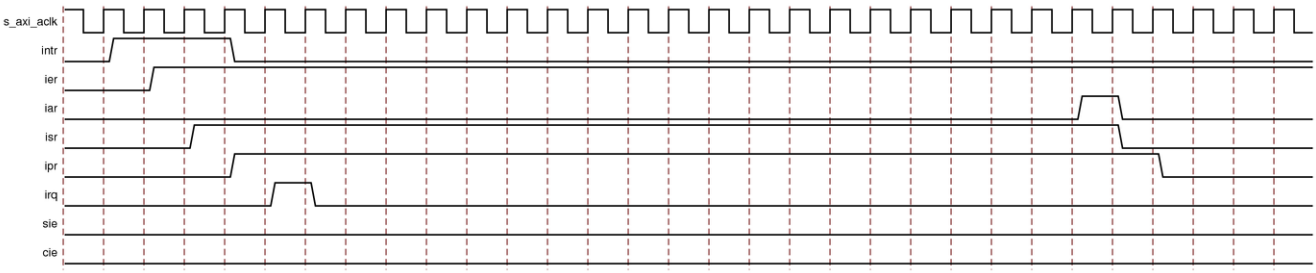

Figure 3-4: **Input - Rising Edge Sensitive, Output - Rising Edge Sensitive, Fast Interrupt Logic Disabled**

Figure 3-5 shows the timing diagram with the following core settings:

- Configured Input Interrupt (INTR) for Edge sensitive (rising)
- Enable fast interrupt logic
- Mode set to fast interrupt mode (IMR(i) = 1)

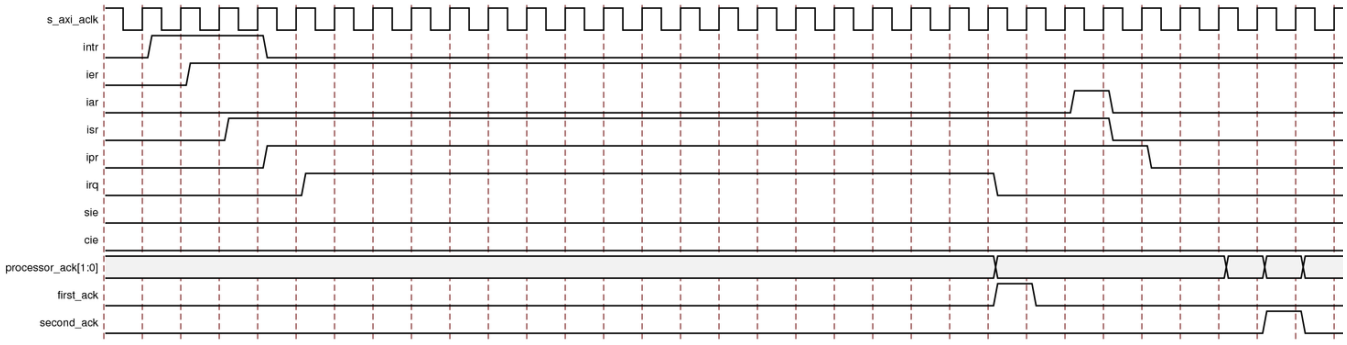

Figure 3-5: **Input - Rising Edge Sensitive, Fast Interrupt Logic Enabled and IMR(i) = 1**

Design Flow Steps

This chapter describes customizing and generating the core, constraining the core, and the simulation, synthesis and implementation steps that are specific to this IP core. More detailed information about the standard Vivado® design flows and the IP integrator can be found in the following Vivado Design Suite user guides:

- *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 3]
- *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 1]
- *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 4]
- *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 5]

Customizing and Generating the Core

This section includes information about using Xilinx tools to customize and generate the core in the Vivado Design Suite.

If you are customizing and generating the core in the Vivado IP integrator, see the *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 3] for detailed information. IP integrator might auto-compute certain configuration values when validating or generating the design. To check whether the values do change, see the description of the parameter in this chapter. To view the parameter value, run the `validate_bd_design` command in the Tcl Console.

You can customize the IP for use in your design by specifying values for the various parameters associated with the IP core using the following steps:

1. Select the IP from the Vivado IP catalog.
2. Double-click the selected IP or select the **Customize IP** command from the toolbar or right-click menu.

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 1] and the *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 3].

Note: Figures in this chapter are illustrations of the Vivado Integrated Design Environment (IDE). The layout depicted here might vary from the current version.

Basic Tab

The parameters in the Basic tab are shown in Figure 4-1 and are described in this section.

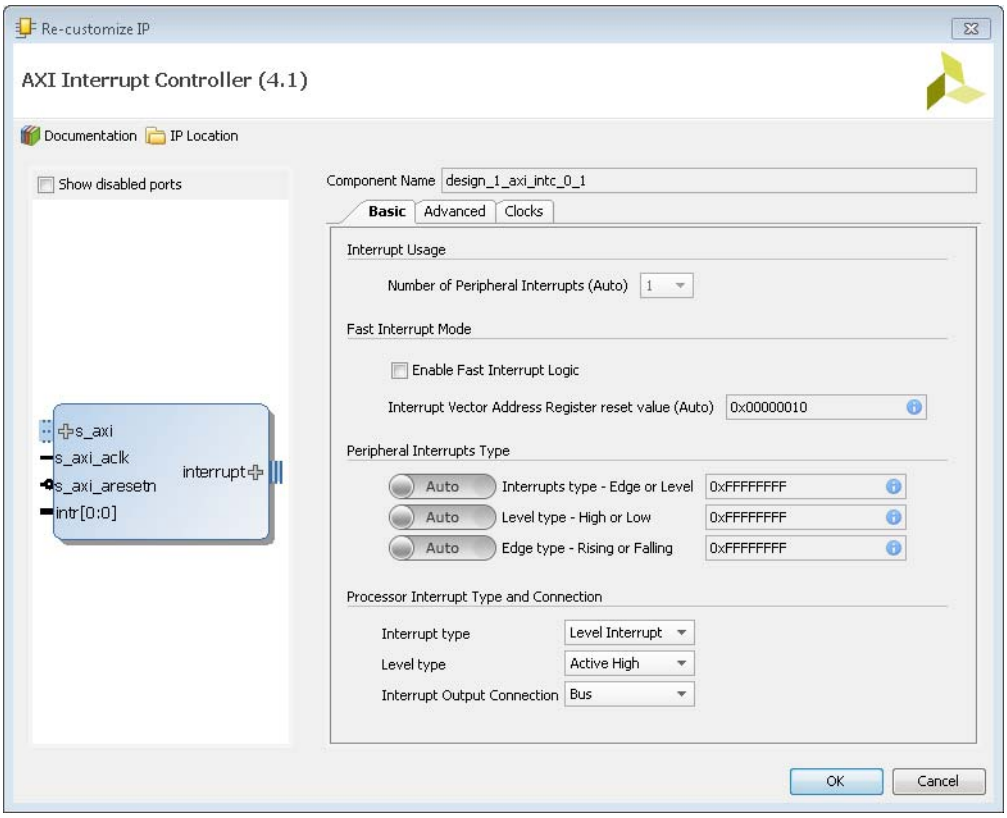

Figure 4-1: **AXI INTC Basic Tab**

Number of Peripheral Interrupts

This option enables selection of number of peripheral interrupt inputs. In IP Integrator, this value is automatically determined from the number of connected interrupt signals.

Enable Fast Interrupt Logic

This option enables AXI INTC to work in Fast Interrupt mode. In this mode, AXI INTC provides the interrupt vector address using the `interrupt_address` signal, and the processor acknowledges an interrupt through the `processor_ack` signal. Fast Interrupt Mode is not available when selecting Single interrupt output connection.

Peripheral Interrupts Type

Interrupts type - Edge or Level

This option is used to set the input interrupts to be either Edge or Level type.

- 0 - Level
- 1 - Edge

Width of this field is equal to **Number of Peripheral Interrupts** selected. This 32-bit field is directly mapped to interrupt inputs. For example, setting bit 0 (the least significant bit) affects `intr(0)`, setting bit 1 affects `Intr(1)`, and so on.

In IP Integrator, this value is normally automatically determined from the connected interrupt signals, but can be set manually if necessary.

Level type - High or Low

This option is used to set the input Level type interrupts to be either High or Low.

- 0 - Low
- 1 - High

Width of this field is equal to **Number of Peripheral Interrupts** selected. This 32-bit field is directly mapped to interrupt inputs. For example, setting bit 0 (the least significant bit) affects `intr(0)`, setting bit 1 affects `intr(1)`, and so on.

In IP Integrator, this value is normally automatically determined from the connected interrupt signals, but can be set manually if necessary.

Edge type - Rising or Falling

This option is used to set the input Edge type interrupts to be either Rising or Falling edge.

- 0 - Falling edge
- 1 - Rising edge

Width of this field is equal to **Number of Peripheral Interrupts** selected. This 32-bit field is directly mapped to interrupt inputs. For example, setting bit 0 (the least significant bit) affects `intr(0)`, setting bit 1 affects `intr(1)`, and so on.

In IP Integrator, this value is normally automatically determined from the connected interrupt signals, but can be set manually if necessary.

Processor Interrupt Type and Connection

Interrupt type

This option is used to set the output interrupt to be either Edge or Level type.

- 0 - Level
- 1 - Edge

CAUTION! *Xilinx recommends that this setting is not changed from its default value (0 - Level) when AXI INTC is connected to a MicroBlaze™ processor.*

Level type

This option is used to set the output Level type interrupts to be either High or Low. It is shown when **Interrupt type** is set to Level.

- 0 - Active-Low
- 1 - Active-High

Edge type

This option is used to set the output Edge type interrupts to be either Rising or Falling edge. It is shown when **Interrupt type** is set to Edge.

- 0 - Falling edge
- 1 - Rising edge

Interrupt Output Connection

Select interrupt output connection bus interface. Normally **Bus** is used when connecting to MicroBlaze and cascaded AXI Interrupt Controllers. Otherwise, **Single** can be used when Fast Mode Interrupt is not enabled, and the target has a single interrupt input.

- 0 - Bus
- 1 - Single

Advanced Tab

The parameters in the Advanced tab are shown in Figure 4-2 and are described in this section.

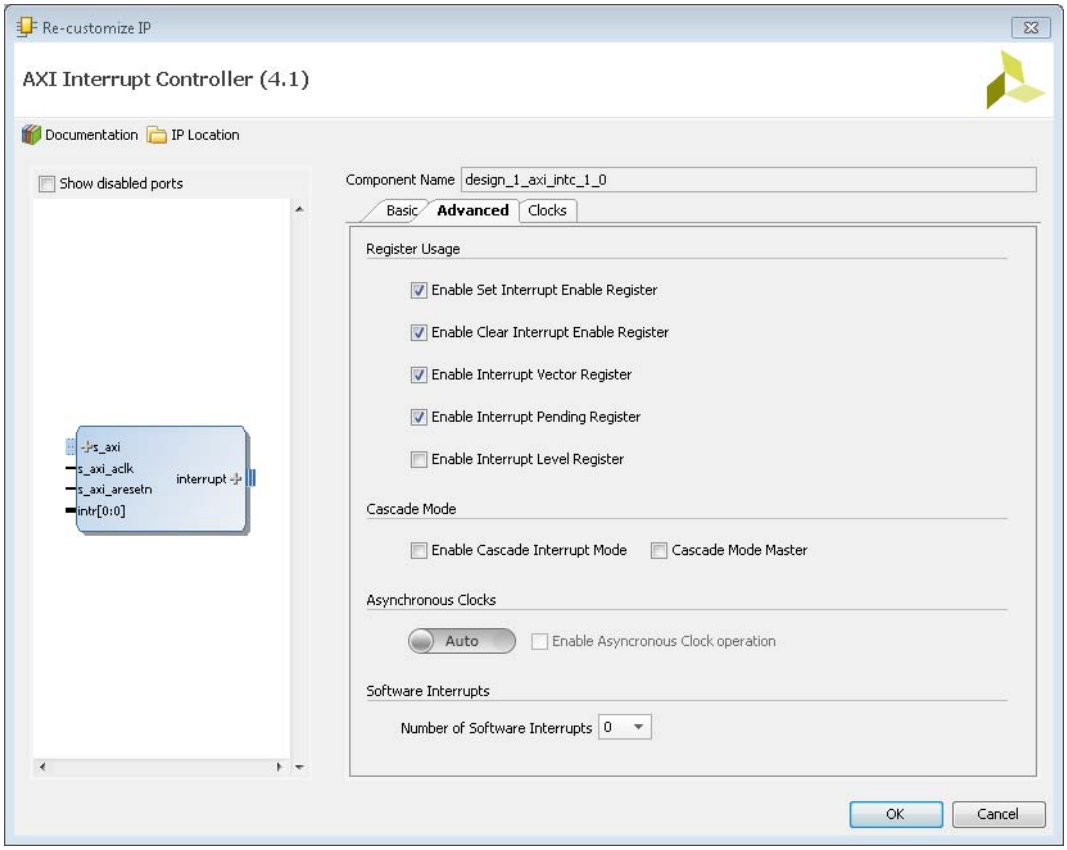

Figure 4-2: **AXI INTC Advanced Tab**

Enable Set Interrupt Enable Register

Setting this option includes Set Interrupt Enable Register.

Enable Clear Interrupt Enable Register

Setting this option includes Clear Interrupt Enable Register.

Enable Interrupt Vector Register

Setting this option includes Interrupt Vector Register.

Enable Interrupt Pending Register

Setting this option includes Interrupt Pending Register.

Enable Interrupt Level Register

Setting this option includes Interrupt Level Register, to support nested interrupts.

Enable Cascade Interrupt Mode

Setting this option enables AXI INTC cascade mode.

Cascade Mode Master

Setting this option defines AXI INTC as a cascade mode master.

Enable Asynchronous Clock Operation

Enabling this option allows the AXI clock and processor clk to run asynchronously.

- Check box selected - asynchronous mode
- Check box not selected - synchronous mode

In IP Integrator, this value is automatically determined depending on the connected clocks, but the user can override the value if necessary.

Number of Software Interrupts

This option enables selection of number of software interrupts. The maximum number of interrupts, **Number of Peripheral Interrupts** + **Number of Software Interrupts**, is 32.

Clocks Tab

The parameters in the Clocks tab are shown in Figure 4-3.

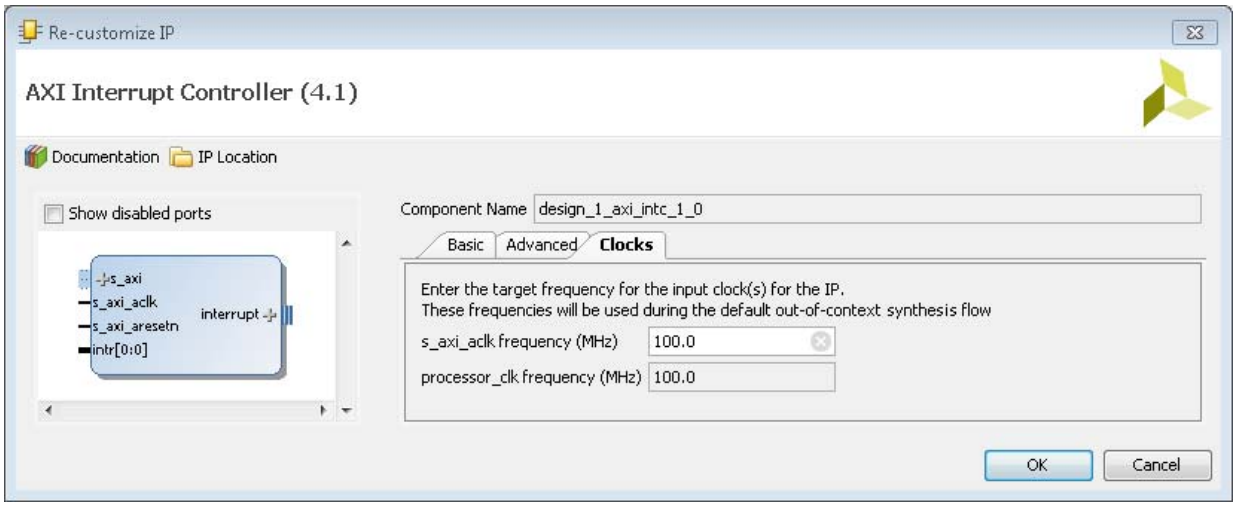

Figure 4-3: **Clocks Parameter Tab**

- **s_axi_aclk frequency (MHz)**- Sets the frequency for the input AXI clock
- **processor_clk frequency (MHz)**- Sets the frequency for the input processor clock

User Parameters

Table 4-1 shows the relationship between the fields in the Vivado IDE and the User Parameters (which can be viewed in the Tcl Console).

Table 4-1: **Vivado IDE Parameter to User Parameter Relationship**

Vivado IDE Parameter	User Parameter	Default Value
Number of Peripheral Interrupts	C_NUM_INTR_INPUTS	1
Enable Fast Interrupt Logic	C_HAS_FAST	0
Interrupt Vector Address Register reset value	C_IVAR_RESET_VALUE	0x00000010
Interrupts type - Edge or Level	C_KIND_OF_INTR	0xFFFFFFFF
Level type - High or Low	C_KIND_OF_LVL	0xFFFFFFFF
Edge type - Rising or Falling	C_KIND_OF_EDGE	0xFFFFFFFF
Interrupt type	C_IRQ_IS_LEVEL	1
Level type	C_IRQ_ACTIVE	0x1
Interrupt Output Connection	C_IRQ_CONNECTION	0
Enable Set Interrupt Enable Register	C_HAS_SIE	1
Enable Clear Interrupt Enable Register	C_HAS_CIE	1
Enable Interrupt Vector Register	C_HAS_IVR	1

Table 4-1: **Vivado IDE Parameter to User Parameter Relationship *(Cont'd)***

Vivado IDE Parameter	User Parameter	Default Value
Enable Interrupt Pending Register	C_HAS_IPR	1
Enable Interrupt Level Register	C_HAS_ILR	0
Enable Cascade Interrupt Mode	C_EN_CASCADE_MODE	0
Cascade Mode Master	C_CASCADE_MASTER	0
Enable Asynchronous Clock operation	C_ENABLE_ASYNC	0
Number of Software Interrupts	C_NUM_SW_INTR	0

Output Generation

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 1].

Constraining the Core

This section contains information about constraining the core in the Vivado Design Suite.

Required Constraints

This section is not applicable for this IP core.

Device, Package, and Speed Grade Selections

This section is not applicable for this IP core.

Clock Frequencies

This section is not applicable for this IP core.

Clock Management

This section is not applicable for this IP core.

Clock Placement

This section is not applicable for this IP core.

Banking

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.

Simulation

For comprehensive information about Vivado simulation components, as well as information about using supported third-party tools, see the *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 5].

IMPORTANT: *For cores targeting 7 series or Zynq-7000 devices, UNIFAST libraries are not supported. Xilinx IP is tested and qualified with UNISIM libraries only.*

Synthesis and Implementation

For details about synthesis and implementation, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 1].

Migrating and Upgrading

This appendix contains information about migrating a design from ISE® to the Vivado® Design Suite, and for upgrading to a more recent version of the IP core. For customers upgrading in the Vivado Design Suite, important details (where applicable) about any port changes and other impact to user logic are included.

Migrating to the Vivado Design Suite

For information on migrating from the Xilinx ISE® Design Suite tools to the Vivado® Design Suite, see the *ISE to Vivado Design Suite Migration Guide* (UG911) [Ref 6].

Upgrading in the Vivado Design Suite

This section provides information about any changes to the user logic or port designations that take place when you upgrade to a more current version of this IP core in the Vivado Design Suite.

When upgrading to AXI Interrupt Controller v4.1 from v2.00.a, v3.1 or v3.2, the ports `processor_clk` and `processor_rst` are disabled when C_HAS_FAST is not set, because they are not needed in this case, and should not be connected.

Debugging

This appendix includes details about resources available on the Xilinx Support website and debugging tools.

Finding Help on Xilinx.com

To help in the design and debug process when using the AXI INTC core, the Xilinx Support web page contains key resources such as product documentation, release notes, answer records, information about known issues, and links for obtaining further product support.

Documentation

This product guide is the main document associated with the AXI INTCcore. This guide, along with documentation related to all products that aid in the design process, can be found on the Xilinx Support web page or by using the Xilinx Documentation Navigator.

Download the Xilinx Documentation Navigator from the Downloads page. For more information about this tool and the features available, open the online help after installation.

Answer Records

Answer Records include information about commonly encountered problems, helpful information on how to resolve these problems, and any known issues with a Xilinx product. Answer Records are created and maintained daily ensuring that users have access to the most accurate information available.

Answer Records for this core a can be located using the Search Support box on the main Xilinx support web page. To maximize your search results, use proper keywords such as

- Product name
- Tool messages
- Summary of the issue encountered

A filter search is available after results are returned to further target the results.

Master Answer Record for the AXI INTC Core

AR: 54423

Technical Support

Xilinx provides technical support at the Xilinx Support web page for this LogiCORE™ IP product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support if you do any of the following:

- Implement the solution in devices that are not defined in the documentation.
- Customize the solution beyond that allowed in the product documentation.
- Change any section of the design labeled DO NOT MODIFY.

To contact Xilinx Technical Support, navigate to the Xilinx Support web page.

Debug Tools

There are many tools available to address AXI INTC core design issues. It is important to know which tools are useful for debugging various situations.

Vivado Design Suite Debug Feature

The Vivado® Design Suite debug feature inserts logic analyzer and virtual I/O cores directly into your design. The debug feature also allows you to set trigger conditions to capture application and integrated block port signals in hardware. Captured signals can then be analyzed. This feature in the Vivado IDE is used for logic debugging and validation of a design running in Xilinx devices.

The Vivado logic analyzer is used to interact with the logic debug IP cores, including:

- ILA 2.0 (and later versions)
- VIO 2.0 (and later versions)

See the *Vivado Design Suite User Guide: Programming and Debugging* (UG908) [Ref 7].

AXI4-Lite Interface Debug

Read from a register that does not have all 0s as a default to verify that the interface is functional. Output `s_axi_arready` asserts when the read address is valid, and output `s_axi_rvalid` asserts when the read data/response is valid. If the interface is unresponsive, ensure that the following conditions are met:

- The `s_axi_aclk` input is connected and toggling.
- The interface is not being held in reset, and `s_axi_aresetn` is an active-Low reset.
- The main core clocks are toggling and that the enables are also asserted.
- All required interrupts are connected to the INTR input of the core and the IRQ (and other Fast Mode signals) are tied to the interrupt interface of the processor.
- The AXI INTC core is configured properly for the target application.

To debug the AXI INTC core, read all the application registers of the core to verify that all are functioning correctly.

Appendix C

Additional Resources and Legal Notices

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see Xilinx Support.

References

These documents provide supplemental material useful with this product guide:

1. *Vivado® Design Suite User Guide: Designing With IP* (UG896)
2. *Vivado Design Suite AXI Reference Guide* (UG1037)
3. *Vivado Design Suite User Guide: Designing IP Subsystems Using IP Integrator* (UG994)
4. *Vivado Design Suite User Guide: Getting Started* (UG910)
5. *Vivado Design Suite User Guide - Logic Simulation* (UG900)
6. *ISE® to Vivado Design Suite Migration Guide* (UG911)
7. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
11/14/2018	4.1	Added extended address interrupt vector registers, to support up to 64 bit fast interrupt vector addresses.
04/04/2018	4.1	Updated and clarified description of cascade mode.
10/04/2017	4.1	Added information on using fast interrupt mode with nested interrupts.
04/06/2016	4.1	• Allow selection of single interrupt output interface • Added Cascade Interrupt Mode bus interface to simplify cascade mode connection
11/18/2015	4.1	Added support for UltraScale+ families.
06/24/2015	4.1	• Moved performance and resource utilization data to the web • Clarified interrupt vector generation for fast interrupt mode • Clarified edge- and level-sensitive interrupt input behavior
12/18/2013	4.1	• Updated to add description of Interrupt Level Register used for nested interrupt handling
10/02/2013	4.0	• processor_clk and processor_rst pins hidden when fast interrupt is not enabled • synchronization flip-flops added on asynchronous interrupt inputs.
06/19/2013	3.1	• Updated for core v3.1. • Added description of software interrupt.
03/20/2013	2.0	• Updated for core v3.0 and Vivado Design Suite only support. • Updated signal names, and timing diagrams.
12/18/2012	1.0	• Initial product guide release. Replaces *LogiCORE IP AXI INTC Data Sheet (DS747)*. • Added Cascade Mode.

Please Read: Important Legal Notices

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Xilinx shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in connection with, the Materials (including your use of the Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the Materials without prior written consent. Certain products are subject to the terms and conditions of Xilinx's limited warranty, please refer to Xilinx's Terms of Sale which can be viewed at https://www.xilinx.com/legal.htm#tos; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx products are not designed or intended to be fail-safe or for use in any application requiring fail-safe performance; you assume sole risk and liability for use of Xilinx products in such critical applications, please refer to Xilinx's Terms of Sale which can be viewed at https://www.xilinx.com/legal.htm#tos.

AUTOMOTIVE APPLICATIONS DISCLAIMER

AUTOMOTIVE PRODUCTS (IDENTIFIED AS "XA" IN THE PART NUMBER) ARE NOT WARRANTED FOR USE IN THE DEPLOYMENT OF AIRBAGS OR FOR USE IN APPLICATIONS THAT AFFECT CONTROL OF A VEHICLE ("SAFETY APPLICATION") UNLESS THERE IS A SAFETY CONCEPT OR REDUNDANCY FEATURE CONSISTENT WITH THE ISO 26262 AUTOMOTIVE SAFETY STANDARD ("SAFETY DESIGN"). CUSTOMER SHALL, PRIOR TO USING OR DISTRIBUTING ANY SYSTEMS THAT INCORPORATE PRODUCTS, THOROUGHLY TEST SUCH SYSTEMS FOR SAFETY PURPOSES. USE OF PRODUCTS IN A SAFETY APPLICATION WITHOUT A SAFETY DESIGN IS FULLY AT THE RISK OF CUSTOMER, SUBJECT ONLY TO APPLICABLE LAWS AND REGULATIONS GOVERNING LIMITATIONS ON PRODUCT LIABILITY.