LogiCORE IP AXI Interrupt Controller (INTC) v2.00a

Product Guide

PG099 December 18, 2012

XILINX®

XILINX

Table of Contents

SECTION I: SUMMARY

Introduction

The LogiCORE™ IP AXI Interrupt Controller (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 a slave interface for the AMBA® AXI (Advanced eXtensible Interface) specification. The number of interrupts and other aspects can be tailored to the target system. This AXI INTC core is designed to interface with the AXI4-Lite protocol.

Features

- AXI interface based on the AXI4-Lite specification.
- Each individual interrupt can be configured in Fast Interrupt mode, which improves latency. This mode enables the core to drive the interrupt vector address of the interrupt being processed and uses dedicated interrupt acknowledgement.
- Configurable number of (up to 32) interrupts inputs. Cascadable to provide additional interrupt inputs.
- 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
- Automatic edge synchronization when inputs are configured for edge sensitivity
- Output interrupt request pin is configurable for edge or level generation

LogiCORE IP Facts Table	
Core Specifics	
Supported Device Family[1]	Zynq™-7000[2], Virtex®-7, Kintex™-7, Artix™-7 Virtex-6, Spartan®-6
Supported User Interfaces	AXI4-Lite
Resources	See Resource Utilization.
Provided with Core	
Design Files	ISE: VHDL Vivado: RTL
Example Design	Not Provided
Test Bench	Not Provided
Constraints File	Not Applicable
Simulation Model	Not Applicable
Supported S/W Driver[3]	Standalone
Tested Design Flows[4]	
Design Entry	ISE® Design Suite v14.4 Vivado™ Design Suite v2012.4
Simulation	Mentor Graphics ModelSim
Synthesis	Xilinx Synthesis Technology (XST) Vivado Synthesis
Support	
Provided by Xilinx @ www.xilinx.com/support	

Notes:

1. For a complete list of supported derivative devices, see Embedded Edition Derivative Device Support.
2. Supported in ISE Design Suite implementations only.
3. Standalone driver details can be found in the EDK or SDK directory (<*install_directory*>/doc/usenglish/xilinx_drivers.htm). Linux OS and driver support information is available from //wiki.xilinx.com.
4. For the supported versions of the tools, see the Xilinx Design Tools: Release Notes Guide.

Overview

The LogiCORE™ IP AXI Interrupt Controller (AXI 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 a slave interface for the AMBA® AXI (Advanced Micro controller Bus Architecture Advanced eXtensible Interface) protocol specification [Ref 1]. The number of interrupts and other aspects can be tailored to the target system. This AXI INTC core is designed to interface with the AXI4-Lite protocol.

Feature Summary

The AXI INTC core can be used to expand the number of inputs available to the processor and an option to provide a priority encoding scheme. The output of the AXI INTC core is intended to be connected to a device that can generate interrupt conditions.

Capturing, Acknowledging and Enabling Interrupt Conditions

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 Capture Modes

Two modes are defined for the capture of interrupt inputs as interrupt conditions:

- Edge-sensitive capture
- Level-sensitive capture

Edge-sensitive capture mode 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.) Figure 1-1 illustrates three main types of edge generation schemes. In this example all schemes use rising edges for detecting an active edge. The peripheral device generating the interrupt input provides an active edge and later produces an inactive edge to detect another active edge for generating a new interrupt request.

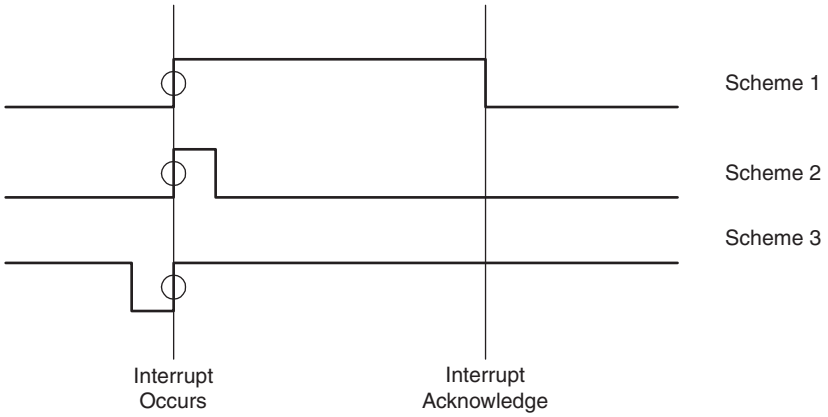

Figure 1-1: **Schemes for Generating Edges**

Level-sensitive capture mode captures 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 as per-input option.)

Observable differences between edge-sensitive and level-sensitive capture are:

- Active level (without subsequent transitions) can produce multiple interrupt conditions
- In edge-sensitive capture, the interrupt input must cycle using an inactive edge and subsequent new active edge to generate an additional interrupt condition

Interrupt Vector Register (Optional)

If an optional Interrupt Vector Register is configured in the AXI INTC module, then a priority relationship is established between the interrupt inputs (lower number, higher priority). The register returns the number attached to the individually enabled interrupt with highest priority. This can be used to expedite the speed at which software can select the appropriate interrupt service routine (software vectoring).

Fast Interrupt Mode Control

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 better latency. Fast interrupt mode is enabled by setting the corresponding bit in Interrupt Mode register (IMR). The interrupt is acknowledged through PROCESSOR_ACK ports driven by the processor for interrupts configured in fast interrupt mode. The AXI INTC core drives the interrupt vector address of the highest priority interrupt along with the IRQ. The IRQ generated is cleared based on the PROCESSOR_ACK signal, and the corresponding IAR bit is updated after acknowledgement is received by PROCESSOR_ACK. The IAR write operation is done internally by the AXI INTC core. The processor sends 0b01 on the PROCESSOR_ACK port when the interrupt is acknowledged by the processor (when branching to the interrupt service routine), and sends 0b10 when executing an RTID

instruction in the interrupt service routine. For edge-type interrupts, the corresponding bit in the ISR is cleared upon receiving 0b01 on the PROCESSOR_ACK port. For level-type interrupts, the corresponding bit in the ISR is cleared when 0b10 or 0b11 is seen on the PROCESSOR_ACK port.

Writing to the IAR is not allowed when servicing a fast interrupt. Also, the IVR might not reflect the exact interrupt that is being serviced.

Interrupt Vector Address registers (IVAR) and the Interrupt Mode register (IMR) are generally written during software initialization and must not be written when any interrupt is enabled. Interrupt acknowledgement signals are shown in Table 1-1.

Table 1-1: **Interrupt Acknowledgement Signal Actions**

PROCESSOR_ACK Pattern	Action Description
0b00	No interrupt received
0b01	Set when processor branches to interrupt routine
0b10	Set when processor returns from interrupt routine by executing RTID
0b11	Set when processor enables interrupts

Cascade Mode (Optional)

When the system requires more than 32 interrupts, it is necessary to expand the AXI INTC capability to handle more interrupt. This can be achieved by setting the parameters related to Cascade Mode in the core. Set the parameters C_EN_CASCADE_MODE and C_CASCADE_MASTER. The next sections cover setting this mode and its usage in the system.

Module Description

Figure 1-2 and Figure 1-3 illustrate the main functional units in the AXI INTC module:

- AXI Interface Module: Converts AXI transactions to the internal IPIC interface
- Intc Interface Module: Connects the INTC core to the AXI interface in AXI INTC module
- Interrupt Controller (INTC) Core: Consists of the Int Det, Irq Gen, and Regs logical blocks
- Interrupt Detection (Int Det): Detects the valid interrupt from all the interrupt inputs
- Programmer Registers (Regs): Registers used to program and control the operation of interrupt controller
- Interrupt Request Generation (Irq Gen): Generate the interrupt request output

- Fast Interrupt Mode Control: Controls the operation to be either in normal or fast interrupt mode

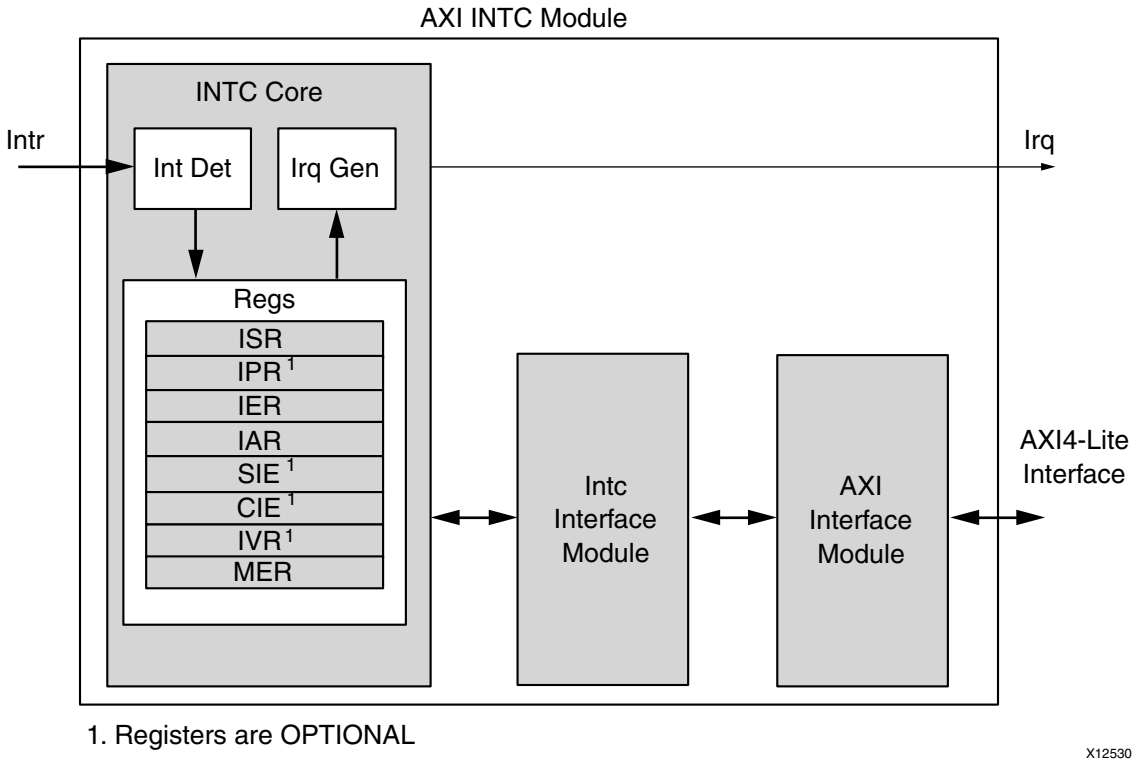

Figure 1-2: **AXI INTC Module Block Diagram**

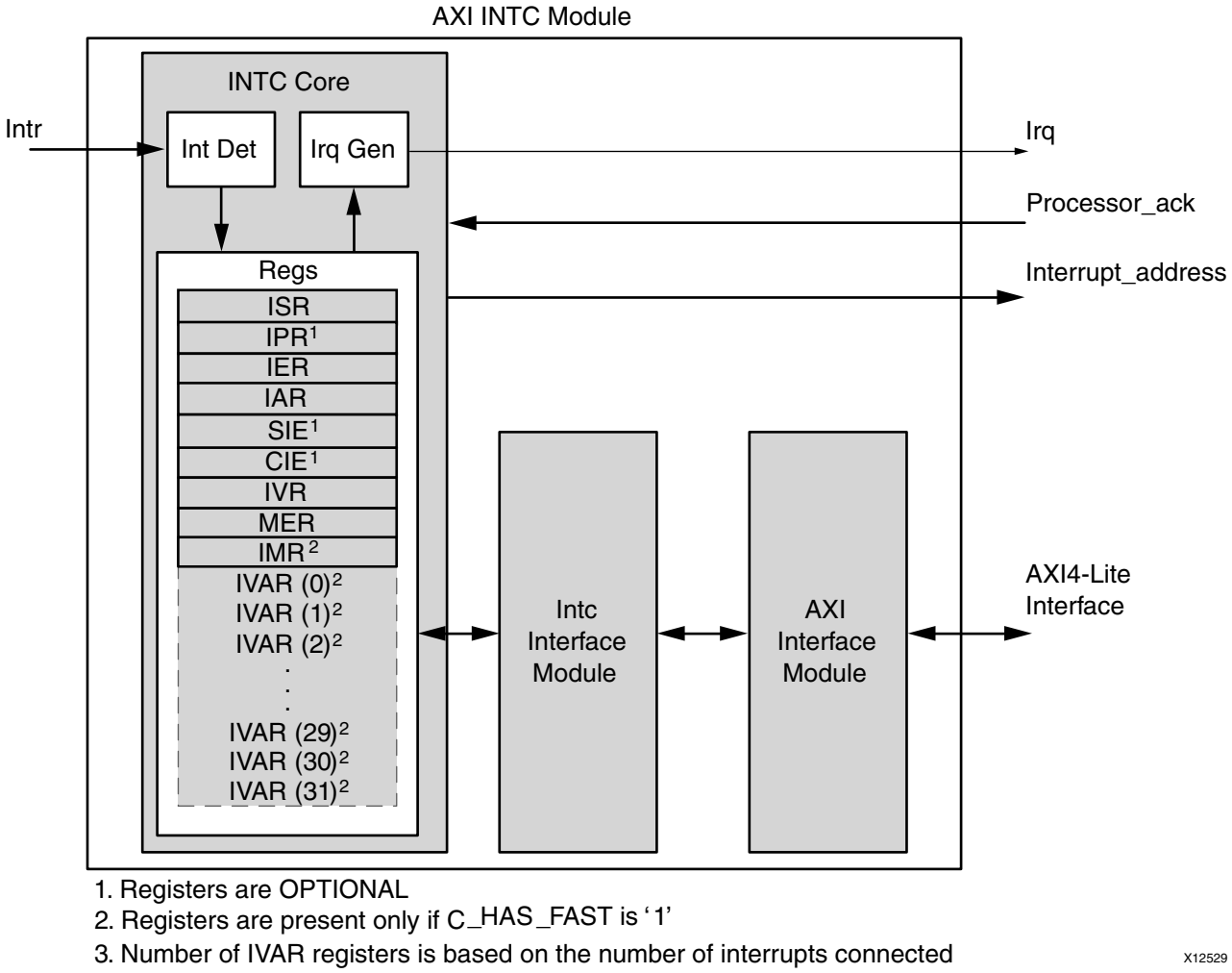

Figure 1-3: **AXI INTC Module Low Latency Block Diagram**

AXI Interface Module

The AXI interface provides a slave interface for transferring data between the INTC and the processor. The AXI INTC registers are mapped into the AXI4-Lite address space.

The register addresses are fixed on 4-byte boundaries. All the registers and the data transfers to and from them are always as wide as the data bus.

The number of interrupt inputs is configurable up to the width of the data bus, which is set by a configuration parameter. The base address for the registers is also set by a configuration parameter.

Intc Interface Module

This logical block connects the INTC core to the AXI interface module.

Interrupt Controller (INTC) Core

Interrupt Controller Core consists of logical blocks as follows:

- **Interrupt Detection (Int Det)** - For the detection of active input interrupt
- **Registers (Regs)** - Contains all status and control registers
- **Interrupt Request Generation (Irq Gen)** - Generates the final output interrupt

Interrupt Detection (Int Det)

Interrupt detection can be configured for either level or edge detection for each interrupt input. If edge detection is chosen, synchronization registers are included. Interrupt request generation is also configurable as either a pulse output for an edge sensitive request or as a level output that is cleared when the interrupt is acknowledged.

Programmer Registers (Regs)

The interrupt controller contains programmer accessible registers that allow interrupts to be enabled, queried and cleared under software control. For details refer to:

- Interrupt Status Register (ISR), page 26
- Interrupt Pending Register (IPR), page 27
- Interrupt Enable Register (IER), page 28
- Interrupt Enable Register (IER), page 28
- Interrupt Acknowledge Register (IAR), page 29
- Set Interrupt Enables (SIE), page 30
- Clear Interrupt Enables (CIE), page 30
- Interrupt Vector Register (IVR), page 31
- Master Enable Register (MER), page 31
- Interrupt Mode Register (IMR), page 32
- Interrupt Vector Address Register (IVAR), page 33

Interrupt Request Generation (Irq Gen)

The Irq Gen block generates the final output interrupt from the interrupt controller core. The output interrupt sensitivity is determined by the configuration parameters. Irq Gen checks for IER and MER to enable the interrupt generation. Irq Gen also resets the interrupt after acknowledge. Irq Gen also writes the vector address of the active interrupt in IVR and enables the IPR for pending interrupts.

Cascade Mode Generation

When the system needs more than 32 interrupts to be processed by the AXI INTC core, multiple instances of the core are needed. There are two parameters which provides this functionality. These are C_EN_CASCADE_MODE and C_CASCADE_MASTER. Refer the Parameter section for more information on these parameters. This mode is optional and with default mode of these parameters, the core behaves with limited support of 32 interrupts.

Licensing and Ordering Information

This Xilinx LogiCORE IP module is provided at no additional cost with the Xilinx Vivado™ Design Suite and ISE® Design Suite Embedded Edition tools 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 on pricing and availability of other Xilinx LogiCORE IP modules and tools, contact your local Xilinx sales representative.

Product Specification

The LogiCORE™ IP AXI Interrupt Controller (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 a slave interface for the AMBA® AXI (Advanced eXtensible Interface) specification [Ref 1]. The number of interrupts and other aspects can be tailored to the target system. This AXI INTC core is designed to interface with the AXI4-Lite protocol.

Performance

Performance numbers are included with Resource Utilization. To measure the system performance (F_{MAX}) of this core, this core was added to a system using a Virtex®-6 FPGA and a system using a Spartan®-6 FPGA as the device under test (DUT).

Because the AXI INTC core can be used with other design modules in the FPGA, the utilization and timing numbers reported in this section are estimates only. When this core is

combined with other designs in the system, the FPGA resources and timing usage of the design vary from the results reported here.

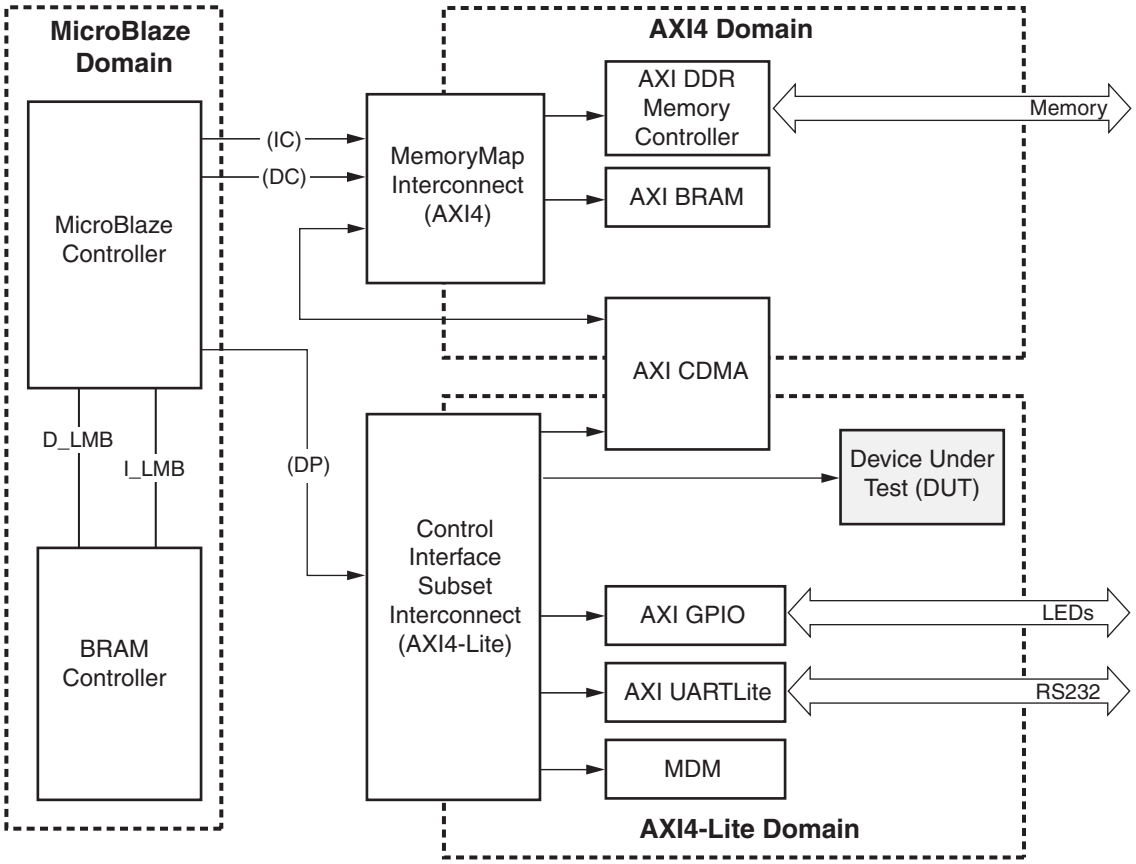

Figure 2-1: **Virtex-6 and Spartan-6 Devices F_{MAX} Margin System**

Maximum Frequencies

Because the AXI INTC core is used with other design modules in the FPGA, the utilization and timing numbers reported in this section are estimates only. When this core is combined with other designs in the system, the utilization of FPGA resources and timing of the design varies from the results reported here.

The target FPGA was filled with logic to drive the LUT and block RAM usage to approximately 70% and the I/O usage to approximately 80%. Using the default tool options and the slowest speed grade for the target FPGA, the resulting target F_{MAX} numbers are shown in Table 2-1.

Table 2-1: **AXI INTC Core System Performance**

Target FPGA	Target F_{MAX} (MHz)
Kintex-7	200
Virtex-7	200

Table 2-1: ***(Cont'd)*AXI INTC Core System Performance *(Cont'd)***

Target FPGA	Target F_{MAX} (MHz)
Virtex-6	150
Spartan-6	133

The target F_{MAX} is influenced by the exact system and is provided for guidance. It is not a guaranteed value across all systems.

Resource Utilization

The intended target technology is Zynq™-7000, 7 series, Spartan®-6, and Virtex®-6 family FPGAs.

These values were generated using the Xilinx® XPS tool v14.4 and Vivado™ Design Suite v2012.4. The utilization numbers reported here are estimates only. When the AXI INTC core is combined with other designs in the system, the utilization of FPGA resources of the AXI INTC design will vary from the results reported here.

The AXI INTC resource utilization for various parameter combinations were measured with a Kintex™-7 device (Table 2-2) and an Artix™-7 device (Table 2-3).

Note: Resources numbers for Virtex-7 devices are approximately the same as those for Kintex-7.

Note: Resources numbers for Zynq-7000 devices are expected to be similar to 7 series device numbers.

Table 2-2: **Performance and Resource Utilization For Kintex-7 (XC7K410T-3-FFG676)**

Parameter Values (other parameters at default value)					Device Resources			Performance
C_NUM_INTR_INPUTS	C_HAS_IPR	C_HAS_SIE	C_HAS_CIE	C_HAS_IVR	Slices	Slice Flip-Flops	LUTs	F_{MAX} (MHz)
32	1	1	1	1	386	588	751	265.463
8	1	1	1	1	308	422	587	224.316
8	1	1	1	1	386	588	751	265.463
8	1	1	1	1	324	486	611	215.564
32	1	1	1	1	324	486	611	215.564
8	1	1	1	1	386	588	751	265.463

Table 2-2: **Performance and Resource Utilization For Kintex-7 (XC7K410T-3-FFG676) *(Cont'd)***

Parameter Values (other parameters at default value)					Device Resources			Performance
C_NUM_INTR_INPUTS	C_HAS_IPR	C_HAS_SIE	C_HAS_CIE	C_HAS_IVR	Slices	Slice Flip-Flops	LUTs	F_{MAX} (MHz)
32	1	1	1	1	386	588	751	265.463
32	1	1	1	1	324	486	611	215.564
8	1	1	1	1	324	486	611	215.564
16	1	1	1	1	324	486	611	215.564
16	1	1	1	1	324	486	611	215.564
16	1	1	1	1	324	486	611	215.564

Notes:
1. The above numbers are reported by tools for clock period of constraint of 5 ns.

Table 2-3: **Performance and Resource Utilization Benchmarks on Artix-7 (xc7a200tffg1156-2)**

Parameter Values						Device Resources			Performance
C_NUM_INTR_INPUTS	C_HAS_IPR	C_HAS_SIE	C_HAS_CIE	C_HAS_IVR	C_HAS_FAST	Slices	Slice Flip-Flop	LUTs	FMAX(Mhz)
32	1	1	1	1	1	406	467	560	131
16	1	1	1	1	1	237	288	342	130
8	1	1	1	1	1	182	198	244	130
32	1	1	1	1	0	276	397	408	125
16	1	1	1	1	0	155	221	235	136
8	1	1	1	1	0	91	132	155	149

The AXI INTC resource utilization for various parameter combinations measured with a Virtex-6 device as the target is detailed in Table 2-4.

Table 2-4: **Performance and Resource Utilization Benchmarks on Virtex-6 (XC6VLX240T-3-FF1156)**

Parameter Values (other parameters at default value)					Device Resources			Performance
C_NUM_INTR_INPUTS	C_HAS_IPR	C_HAS_SIE	C_HAS_CIE	C_HAS_IVR	Slices	Slice Flip-Flops	LUTs	F_{MAX} (MHz)
32	1	1	1	1	352	533	585	209.118
8	1	1	1	1	219	367	402	218.531
8	1	1	1	1	219	367	402	218.531
8	1	1	1	1	318	469	576	217.675
16	1	1	1	1	237	431	434	232.072
16	1	1	1	1	237	431	434	232.072
8	1	1	1	1	318	469	576	217.675
32	1	1	1	1	237	431	434	232.072
16	1	1	1	1	318	469	576	217.675
8	1	1	1	1	237	431	434	232.072
16	1	1	1	1	318	469	576	217.675
8	1	1	1	1	237	431	434	232.072

Notes:

1. The above numbers are reported by tools for clock period constraint of 5 ns.

The AXI INTC resource utilization for various parameter combinations measured with a Spartan-6 device as the target is detailed in Table 2-5.

Table 2-5: **Performance and Resource Utilization Benchmarks on Spartan-6 (XC6SLX75-3-FGG676)**

Parameter Values (other parameters at default value)					Device Resources			Performance
C_NUM_INTR_INPUTS	C_HAS_IPR	C_HAS_SIE	C_HAS_CIE	C_HAS_IVR	Slices	Slice Flip-Flops	LUTs	F_{MAX} (MHz)
32	1	1	1	1	296	533	617	146.327
8	1	1	1	1	239	369	437	147.842

Table 2-5: **Performance and Resource Utilization Benchmarks on Spartan-6 (XC6SLX75-3-FGG676)**

Parameter Values (other parameters at default value)					Device Resources			Performance
C_NUM_INTR_INPUTS	C_HAS_IPR	C_HAS_SIE	C_HAS_CIE	C_HAS_IVR	Slices	Slice Flip-Flops	LUTs	F_{MAX} (MHz)
32	1	1	1	1	256	433	461	144.739
32	1	1	1	1	270	469	593	146.284
16	1	1	1	1	256	433	461	144.739
32	1	1	1	1	270	469	593	146.284
16	1	1	1	1	296	533	617	146.327
16	1	1	1	1	296	533	617	146.327
8	1	1	1	1	270	469	593	146.284
8	1	1	1	1	270	469	593	146.284
8	1	1	1	1	239	369	437	147.842
8	1	1	1	1	256	433	461	144.739

Notes:
1. The above numbers are reported by tools for clock period of constraint of 6.25 ns.

Port Descriptions

I/O Signals

The AXI INTC I/O signals are listed and described in Table 2-6.

Table 2-6: **I/O Signal Description**

Port	Signal Name	Interface	I/O	Initial State	Description
AXI Global System Signals					
P1	S_AXI_ACLK	AXI	Input	-	AXI Clock
P2	S_AXI_ARESETN	AXI	Input	-	AXI Reset, active-Low
AXI Write Address Channel Signals					

Table 2-6: **I/O Signal Description *(Cont'd)***

Port	Signal Name	Interface	I/O	Initial State	Description
P3	S_AXI_AWADDR[C_S_AXI_ADDR_WIDTH-1:0]	AXI	Input	-	AXI Write address. The write address bus gives the address of the write transaction
P4	S_AXI_AWVALID	AXI	Input	0x0	Write address valid. This signal indicates that valid write address and control information are available
P5	S_AXI_AWREADY	AXI	Output	0x0	Write address ready. This signal indicates that the slave is ready to accept an address and associated control signals
AXI Write Channel Signals					
P6	S_AXI_WDATA[C_S_AXI_DATA_WIDTH - 1: 0]	AXI	Input	-	Write data
P7	S_AXI_WSTB[C_S_AXI_DATA_WIDTH/8-1:0]	AXI	Input	-	Write strobes. This signal indicates which byte lanes to update in memory
P8	S_AXI_WVALID	AXI	Input	-	Write valid. This signal indicates that valid write data and strobes are available
P9	S_AXI_WREADY	AXI	Output	0x0	Write ready. This signal indicates that the slave can accept the write data
AXI Write Response Channel Signals					
P10	S_AXI_BRESP[1:0]	AXI	Output	0x0	Write response. This signal indicates the status of the write transaction "00"- OKAY "10"- SLVERR "11"- DECERR
P11	S_AXI_BVALID	AXI	Output	0x0	Write response valid. This signal indicates that a valid write response is available
P12	S_AXI_BREADY	AXI	Input	0x1	Response ready. This signal indicates that the master can accept the response information
AXI Read Address Channel Signals					
P13	S_AXI_ARADDR[C_S_AXI_ADDR_WIDTH -1:0]	AXI	Input	-	Read address. The read address bus gives the address of a read transaction

Table 2-6: **I/O Signal Description *(Cont'd)***

Port	Signal Name	Interface	I/O	Initial State	Description
P14	S_AXI_ARVALID	AXI	Input	-	Read address valid. This signal indicates, when high, that the read address and control information is valid and it remains stable until the address acknowledgement signal, S_AXI_ARREADY, is high.
P15	S_AXI_ARREADY	AXI	Output	0x1	Read address ready. This signal indicates that the slave is ready to accept an address and associated control signals.
AXI Read Data Channel Signals					
P16	S_AXI_RDATA[C_S_AXI_DATA_WIDTH -1:0]	AXI	Output	0x0	Read data
P17	S_AXI_RRESP[1:0]	AXI	Output	0x0	Read response. This signal indicates the status of the read transfer. "00"- OKAY "10"- SLVERR "11"- DECERR
P18	S_AXI_RVALID	AXI	Output	0x0	Read valid. This signal indicates that the required read data is available and the read transfer can complete
P19	S_AXI_RREADY	AXI	Input	0x1	Read ready. This signal indicates that the master can accept the read data and response information
INTC Interface Signals					
P20	Intr[C_NUM_INTR_INPUTS-1:0](1)	INTC	Input	-	Interrupt inputs
P21	Irq	INTC	Output	0x1	Interrupt request output
P22	Interrupt_address[31:0](2)	INTC	Output	0x0	Interrupt address output
P23	Processor_ack[1:0]	INTC	Input	-	Interrupt acknowledgement input
P24	Processor_clk	INTC	Input	-	MicroBlaze™ processor clock
P25	Processor_rst	INTC	Input	-	MicroBlaze processor reset, active-High

Table 2-6: **I/O Signal Description *(Cont'd)***

Port	Signal Name	Interface	I/O	Initial State	Description
P26	Interrupt_address_in[31:0]	INTC	Input	-	This port is applicable only when C_EN_CASCADE_MODE = 1 and C_HAS_FAST = 1. This port should be connected to the downstream AXI INTC instances "Ïnterrupt_address" port and available only when C_HAS_FAST = 1 (for secondary instance of AXI INTC.)
P27	Processor_ack_out[1:0]	INTC	Output	0x0	This port is applicable to the instance of AXI INTC only when C_EN_CASCADE_MODE = 1 and C_HAS_FAST = 1. The main AXI INTC instance passes these port values obtained from processor when the 31st bit, which is the cascaded interrupt, is served by the processor (for secondary instance of AXI INTC.)

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.

Design Parameters

To allow you to obtain an AXI INTC that is uniquely tailored for the system, certain features can be parameterized in the AXI INTC design. This allows you to configure a design that utilizes 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 as shown in Table 2-7.

Table 2-7: **Design Parameters**

Generic	Feature/Description	Parameter Name	Allowable Values	Default Value	VHDL Type
System Parameter					
G1	Target FPGA family	C_FAMILY	zynq, virtex7, kintex7, artix7, virtex6, spartan6	virtex6	string
AXI Parameters					
G2	AXI address bus width	C_S_AXI_ADDR_WIDTH	9	9	integer

Table 2-7: **Design Parameters *(Cont'd)***

Generic	Feature/Description	Parameter Name	Allowable Values	Default Value	VHDL Type
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 = Edge 0 = Level	C_KIND_OF_INTR	See(1)	ALL 1s	std_logic_vector
G6	Type of each edge sensitive input 1 = Rising 0 = Falling Valid if C_KIND_OF_INTR = 1s	C_KIND_OF_EDGE	See(1)	ALL 1s	std_logic_vector
G7	Type of each level sensitive input 1= High 0= Low Valid if C_KIND_OF_INTR = 0s	C_KIND_OF_LVL	See(1)	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
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

Table 2-7: **Design Parameters *(Cont'd)***

Generic	Feature/Description	Parameter Name	Allowable Values	Default Value	VHDL Type
G17	Enable Cascade mode interrupt(6)	C_EN_CASCADE_MODE	0 = Default 1 = Enable Cascade Mode	0	integer
G18	Make cascase mode interrupt core instance as primary(7)	C_CASCADE_MASTER	0 = No Cascade Mode Master 1 = Cascade Mode Master	0	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 in case the processor clock is not connected to INTC core. When processor clock is not connected, IRQ to the processor is generated on AXI clock. This flexibility is given for backward compatibility of the core. 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, adds the necessary synchronizer logic for internal signals.
6. C_EN_CASCADE_MODE should be set to 1 when there are more than 32 interrupts to handle in the system. For each successive addition of 32 more interrupts, one new AXI INTC core has to be instantiated. For each instance of AXI INTC this parameter should be set to 1. Only the last instance of AXI INTC (which is having less than the 32 interrupts to handle) should have this parameter set to 0.
7. C_CASCADE_MASTER should be set to 1, only with the master instance of the AXI INTC core. Setting of this parameter is applicable only when the C_EN_CASCADE_MODE parameter is set to 1. This instance directly interfaces with the processor. None of the other instances of the AXI INTC core interface with processor even though these are configured in the Cascade mode. 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 2-8. In addition, when certain features are parameterized out of the design, the related logic is no longer a part of the design. The unused input signals and related output signals are set to a specified value.

Table 2-8: **Parameter-I/O Signal Dependencies**

Generic or Port	Name	Affects	Depends	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	-	C_MB_CLK_NOT_CONNECTED has to be 0 when C_HAS_FAST is set to 1.

Table 2-8: **Parameter-I/O Signal Dependencies *(Cont'd)***

Generic or Port	Name	Affects	Depends	Relationship Description
G17	C_EN_CASCADE_MODE	P26, P27	-	This parameter should be set only when the Cascade mode of interrupt is set.
G18	C_CASCADE_MASTER	P26, P27	-	This parameter should be set only for the master instance of AXI INTC core.
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[31:0]	-	G15	Port is valid if the generic C_HAS_FAST = 1
P23	Processor_ack[1:0]	-	G15	Port is valid if the generic C_HAS_FAST = 1
P26	Interrupt_address_in[31:0]	-	G15, G17, G18	Port existence is depend upon the FAST Mode interrupt as well as Cascade mode interrupt
P27	Processor_ack_out[1:0]	-	G15, G17, G18	Port existence is depend upon the FAST Mode interrupt as well as Cascade mode interrupt

Register Descriptions

All AXI INTC registers are accessed through the AXI4 Lite interface. Each register is 32 bits although some bits can be unused. Each register is accessed on a 4-byte boundary.

Because AXI addresses are byte addresses, AXI INTC register offsets are located at integral multiples of four. Table 2-9 illustrates the registers. The AXI INTC registers are read as little-endian data.

The eight registers visible to the programmer are shown in Table 2-9 and described in this section. These points should be considered when reading and writing to registers:

- Writes to a read-only register returns an OKAY response with no register changes.

- Reads to a write-only register returns zero with an OKAY response.
- All registers are defined for 32-bit access only; any partial-word accesses (byte, half word) have undefined results and return a bus error (SLVERR).
- Unless stated otherwise, any register bits that are not mapped to inputs successfully returns zero when read and successfully returns with no register change when written.
- All registers uses C_NUM_INTR_INPUTS to determine its width except MER, which is always 2 bits wide and IVR which is always 32 bits wide.

Table 2-9: **Register Descriptions** (1)

Address (hex)	Register Name	Access Type	Default Value (hex)	Description
0x0	ISR	Read / Write	0x0	Interrupt Status Register
0x4	IPR	Read	0x0	Interrupt Pending Register
0x8	IER	Read / Write	0x0	Interrupt Enable Register
0xC	IAR	Write	0x0	Interrupt Acknowledge Register
0x10	SIE	Write	0x0	Set Interrupt Enable Register
0x14	CIE	Write	0x0	Clear Interrupt Enable Register
0x18	IVR	Read	0xFFFF	Interrupt Vector Register
0x1C	MER	Read / Write	0x0	Master Enable Register
0x20	IMR	Read / Write	0x0	Interrupt Mode Register
0x100	IVAR(2)	Read/Write	0x10	Interrupt Vector Address Register
0x104			0x10	
0x108			0x10	
0x10C			0x10	
0x110			0x10	
0x114			0x10	
0x118			0x10	
0x11C			0x10	
0x120			0x10	
0x124			0x10	
0x128			0x10	
0x12C			0X10	
0x130			0X10	
0x134			0X10	

Table 2-9: **Register Descriptions *(Cont'd)*[1]**

Address (hex)	Register Name	Access Type	Default Value (hex)	Description
0x138	IVAR[2]	Read/Write	0X10	Interrupt Vector Address Register
0x13C			0X10	
0x140			0X10	
0x144			0X10	
0x148			0X10	
0x14C			0X10	
0x150			0X10	
0x154			0X10	
0x158			0X10	
0x15C			0X10	
0x160			0X10	
0x164			0X10	
0x168			0X10	
0x16C			0X10	
0x170			0X10	
0x174			0X10	
0x178			0X10	
0x17C			0X10	

Notes:

1. If the number of interrupt inputs is less than the data bus width the inputs start with INT0. INT0 maps to the LSB of the ISR, IPR, IER, IAR, SIE, CIE and additional inputs corresponding sequentially to successive bits to the left.
2. IVAR(0) is at (0x100) and IVAR(31) at (0x17C).

Interrupt Status Register (ISR)

When read, the contents of this register indicate the presence or absence of an active interrupt signal. Each bit in this register that is set to a 1 indicates an active interrupt signal on the corresponding interrupt input. Bits that are 0 are not active. The bits in the ISR are independent of the interrupt enable bits in the IER. See the Interrupt Enable Register (IER), page 28 for the interrupt status bits that are masked by disabled interrupts.

The ISR register is writable by software until the Hardware Interrupt Enable (HIE) bit in the MER has been set. After that bit has been set, software can no longer write to the ISR. Given these restrictions, when this register is written to, any data bits that are set to 1 activate the corresponding interrupt just as if a hardware input became active. Data bits that are zero have no effect.

This allows software to generate interrupt to test purposes until the HIE bit has been set. After HIE has been set (enabling the hardware interrupt inputs), then writing to this register

does nothing. 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 2-2 and the bits are described in Table 2-10.

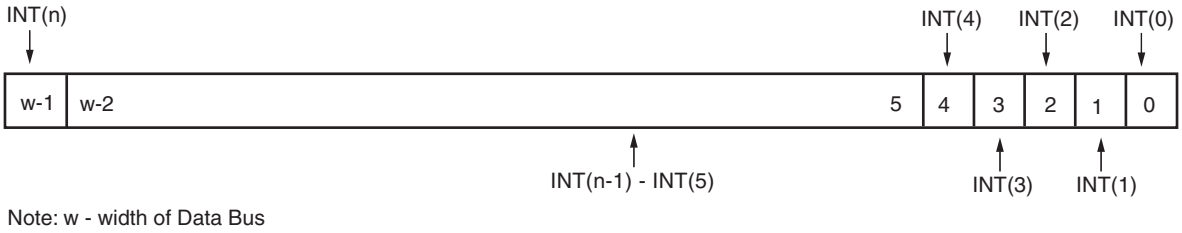

Figure 2-2: **Interrupt Status Register (ISR)**

Table 2-10: **Interrupt Status Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
(w[1]-1):0	INT(n)-INT(0) (n<= w-1)	Read / Write	0x0	Interrupt Input (n) - Interrupt Input (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 parameterized out by setting **`C_HAS_IPR = 0`**. Reading the contents of this register indicates the presence or absence of an active interrupt signal 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 input returns zero. The Interrupt Pending Register (IPR) is shown in Figure 2-3 and the bits are described in Table 2-11.

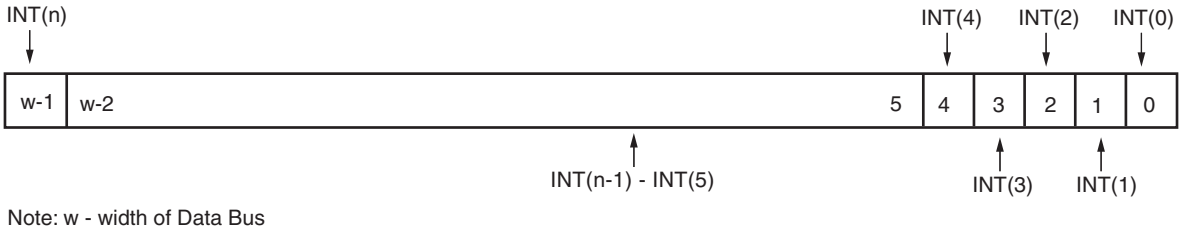

Figure 2-3: **Interrupt Pending Register**

Table 2-11: **Interrupt Pending Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
(w[(1)]-1):0	INT(n)-INT(0) (n<= w-1)	Read / Write	0x0	Interrupt Input (n) - Interrupt Input (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 the corresponding ISR bit to cause assertion of the INTC output. An IER bit set to 0 does not inhibit an interrupt condition for being captured, just reported. Writing a 0 to a bit disables, or masks, the generation of interrupt output for corresponding interrupt input signal.

IMPORTANT: *Disabling an interrupt input is not the same as clearing an interrupt.*

Disabling an interrupt means the interrupt event occurs from peripheral but does not pass to the processor. *Clearing an interrupt* means that after the peripheral generates an interrupt then pass to the processor. In return, the processor reads the Interrupt Status Register and clears the interrupt bit in order to serve the interrupt.

Disabling an active interrupt blocks that interrupt from reaching the `Irq` output, but as soon as it is re-enabled the interrupt immediately generates a request on 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 2-4 and the bits are described in Table 2-12.

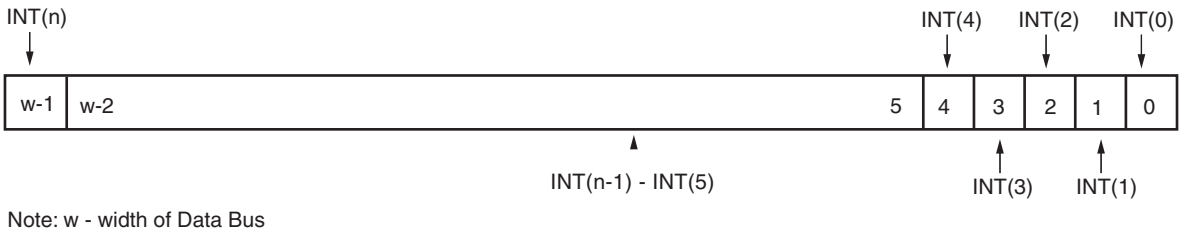

Figure 2-4: **Interrupt Enable Register**

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

Bit(s)	Name	Core Access	Reset Value	Description
(w[(1)]-1):0	INT(n)-INT(0) (n<= w-1)	Read / Write	0x0	Interrupt Input (n) - Interrupt Input (0) 0 - Not Active 1 - Active

Notes:
1. w - Width of Data Bus

Interrupt Acknowledge Register (IAR)

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

In fast interrupt mode, bits in the IAR are cleared by the sensing the acknowledgement pattern over the PROCESSOR_ACK port. In normal interrupt mode, the IAR is cleared by the acknowledgement received over 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 input 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 0s does nothing as does writing a 1 to a bit that does not correspond to an active input or for which an interrupt input does not exist. The IAR is shown in Figure 2-5 and the bits are described in Table 2-13.

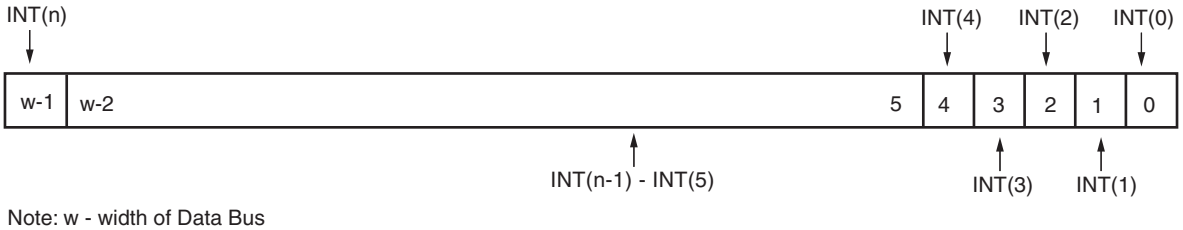

Figure 2-5: **Interrupt Acknowledge Register**

Table 2-13: **Interrupt Acknowledge Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
(w[(1)]-1):0	INT(n)-INT(0) (n<= w-1)	Read / Write	0x0	Interrupt Input (n) - Interrupt Input (0) 0 - Not Active 1 - Active

Notes:
1. w - Width of Data Bus

Set Interrupt Enables (SIE)

SIE is a location 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 0s does nothing, as does wiring a 1 to a bit location that corresponds to a non-existing interrupt input. The SIE is optional in the AXI INTC core and can be parametrized out by setting C_HAS_SIE = 0. The SIE register is shown in Figure 2-6 and the bits are described in Table 2-14.

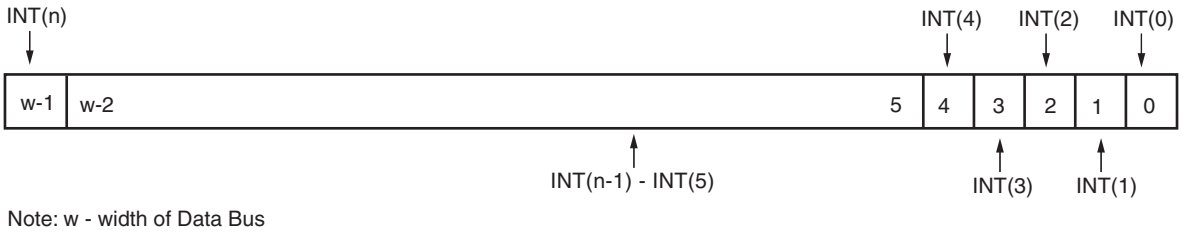

Figure 2-6: **Set Interrupt Enable (SIE) Register**

Table 2-14: **Set Interrupt Enable (SIE) Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
(w[(1)]-1):0	INT(n)-INT(0) (n<= w-1)	Read / Write	0x0	Interrupt Input (n) - Interrupt Input (0) 0 - Not Active 1 - Active

Notes:

1. w - Width of Data Bus

Clear Interrupt Enables (CIE)

CIE is a location 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 0s does nothing, as does writing a 1 to a bit location that corresponds to a non-existing interrupt input. The CIE is also optional in the AXI INTC core and can be parameterized out by setting C_HAS_CIE = 0. The Clear Interrupt Enables (CIE) register is shown in Figure 2-7 and the bits are described in Table 2-15.

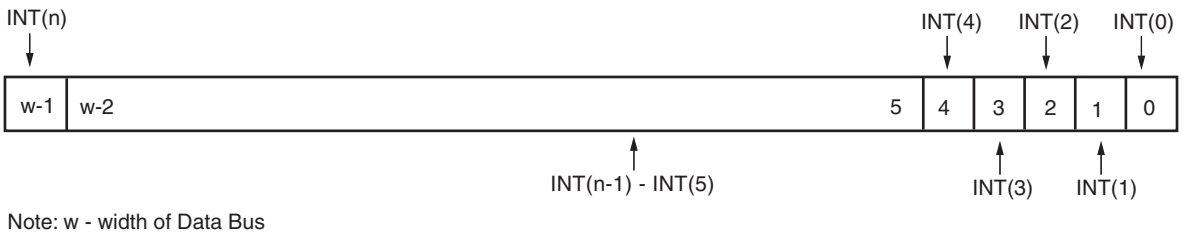

Figure 2-7: **Clear Interrupt Enable (CIE) Register**

Table 2-15: **Clear Interrupt Enable Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
(w[1]-1):0	INT(n)-INT(0) (n<= w-1)	Read / Write	0x0	Interrupt Input (n) - Interrupt Input (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 1s. This IVR acts as an index for giving the correct Interrupt Vector Address along with IRQ. The Interrupt Vector Register (IVR) is shown in Figure 2-8 and described in Table 2-16

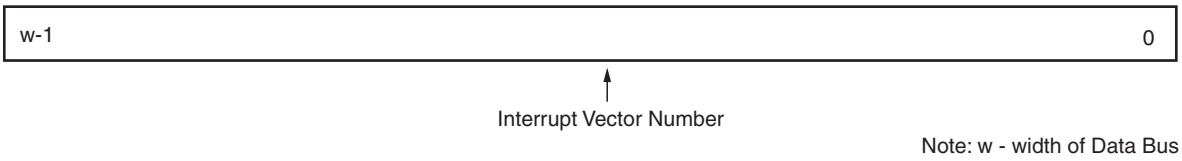

Figure 2-8: **Interrupt Vector Register (IVR)**

Table 2-16: **Interrupt Vector Register (IVR) Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
(w[1]-1):0	Interrupt Vector Number	Read	0x0	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 two bits are mapped to the two least significant bits of the location. 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 interrupts for testing purposes, and disabling any hardware interrupt inputs. Writing a 1 to this bit enables the hardware interrupt inputs and disables software generated inputs. Writing a 1 also disables any further changes to this bit until the device has been reset. Writing 1s or 0s 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 0s. All other bits read as 0. The Master Enable Register (MER) is shown in Figure 2-9 and is described in Table 2-17.

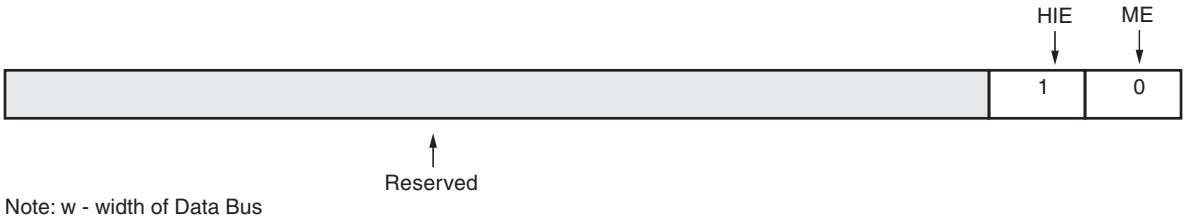

Figure 2-9: **Master Enable Register (MER)**

Table 2-17: **Master Enable Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
(w(1)-1):2	Reserved	N/A	0x0	Reserved
1	HIE	Read / Write	0	Hardware Interrupt Enable 0 = Read - SW interrupts enabled Write - No effect 1 = Read - HW interrupts enabled Write - Enable HW interrupts
0	ME	Read / Write	0	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 if parameter C_HAS_FAST is set to 1. IMR register is used to set the interrupt mode of the connected interrupts. All the interrupts can be set in any mode 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 2-10 and is described in Figure 2-18.

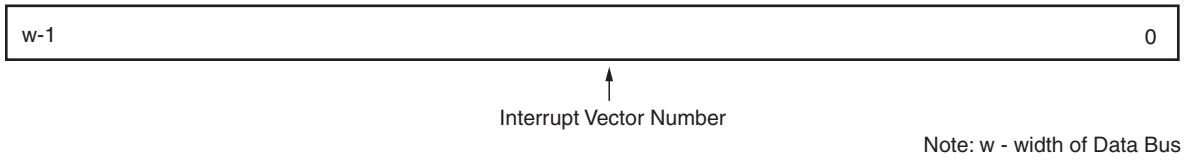

Figure 2-10: **Interrupt Mode Register (IMR)**

Table 2-18: **Interrupt Mode Register (IMR) Bit Definitions**

Bit(s)	Name	Core	Reset Value	Description
(w(1)-1):0	INT(n)-INT(0) (n<= w-1)	Read / Write	0x0	Interrupt Input (n) - Interrupt Input (0) 0 – Normal Interrupt mode 1 – Fast Interrupt mode

Notes:
1. w - Width of Data Bus

Interrupt Vector Address Register (IVAR)

These are 32-bit wide read/write registers and are C_NUM_INTR_INPUTS in number. 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 of operation (IMR(i) = 0), the interrupt vector addresses are taken by the drivers/application. In fast interrupt mode (IMR(i) = 1), the service routine address is driven by the interrupt controller along with the IRQ. IVAR registers are programmed with the corresponding peripheral interrupt vector address during software initialization.

These registers store the interrupt vector addresses of all the C_NUM_INTR_INPUTS interrupts. Address of the interrupt with highest priority is transmitted out along with IRQ.

If not all the 32 interrupts are used, any read on the unused register address returns zero. Writing 1s or 0s to any bit location does nothing.

The IVAR can be accessed through the AXI interface. IVAR registers are in the AXI clock domain and are used in the processor clock domain to give the INTERRUPT_ADDRESS along with IRQ. These are not synchronized to the processor clock domain. So, it is expected that writing to these registers should be done when ME is 0 (MER(0) = 0).

The Interrupt Vector Address Register (IVAR) is shown in Figure 2-11 and is described in Table 2-19.

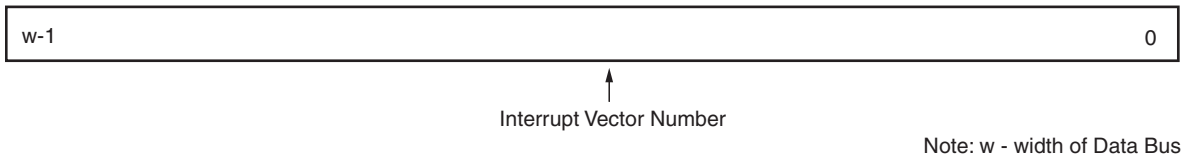

Figure 2-11: **Interrupt Vector Address Register (IVAR)**

Table 2-19: **Interrupt Mode Register (IMR) Bit Definitions**

Bit(s)	Name	Core	Reset Value	Description
(w(1)-1):0	Interrupt Vector Address	Read / Write	0x0	Interrupt vector address of the active interrupt with highest priority

Notes:
1. w - Width of Data Bus

Designing with the Core

Cascade Mode Interrupt

This functionality is enabled when the system needs more than 32 interrupts. Single instance of the AXI INTC can handle the 32 interrupts at maximum. Two main parameters need to use in this mode: C_EN_CASCADE_MODE and C_CASCADE_MASTER. In cascade mode, there are more than one AXI INTC instances in the system. It is mandatory that the 31st interrupt bit of the main AXI INTC instance be used to cascade to the lower order AXI INTC instance. Table 3-1 shows the parameter combination used for the cascade mode.

Table 3-1: **Parameter combination and its usage in Cascade Interrupt Mode**

C_EN_CASCADE_MODE Default = 0	C_CASCADE_MASTER Default = 0	Mode of operation for the AXI INTC instances
0	0	NA. In this mode, there is no cascade interrupt feature available. It means only one instance is allowed in system.
0	1	NA. The C_CASCADE_MASTER is allowed to set only when the C_EN_CASCADE_MODE is set to 1.
1	0	This is applicable only for the lower instance of the AXI INTC module. Based upon the parameter settings for this instance, the ports are connected to the primary or upstream instance of AXI INTC core. The bit 31 of INTR is considered as the cascaded interrupt bit. No other bits are considered valid to be connected to use for cascade mode. (In case of such connections, the particular INTR pin is treated as general interrupt pin).
1	1	This is applicable only when cascade mode is enabled. The parameter settings are applicable to the primary instance of AXI INTC in the system. The bit 31 of INTR is considered as the cascaded interrupt bit. No other bits are allowed to be connected to use for cascade mode.

Table 3-1 shows how Cascade Mode interacts in given system.

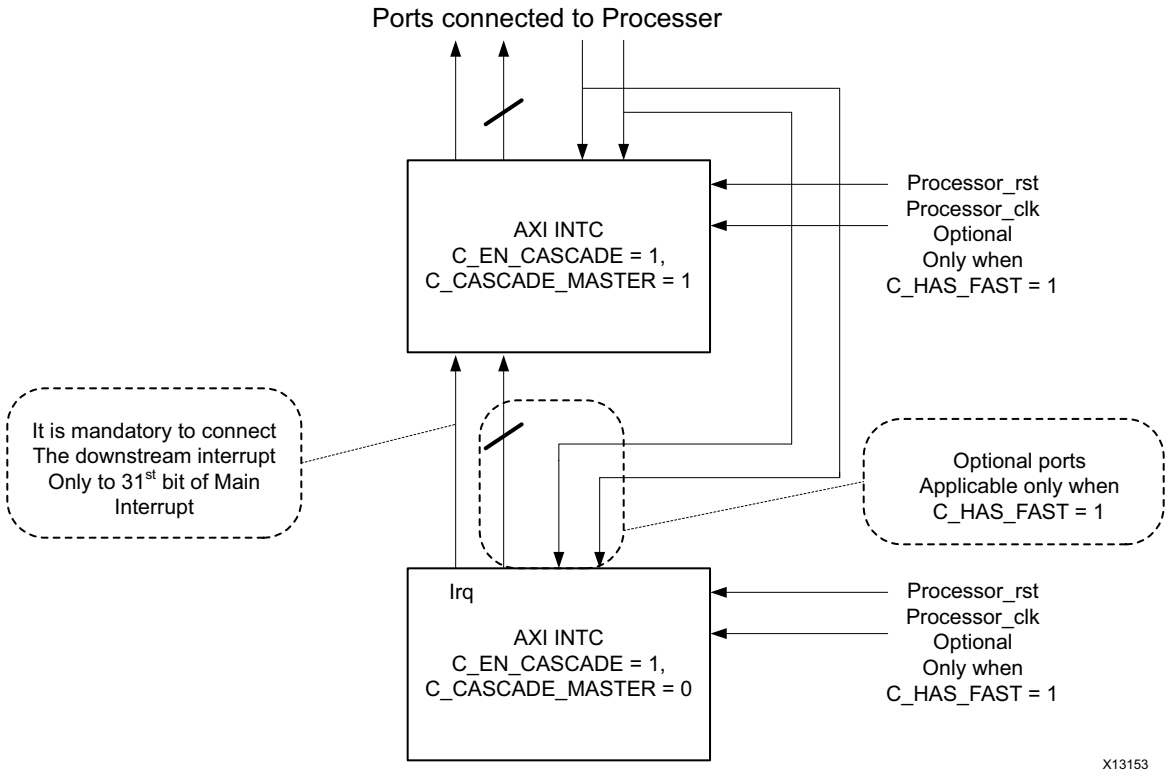

Figure 3-1: **Cascade Mode**

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

- **Cascade Mode with Standard Interrupt**: In this mode, there are no special ports enabled in the system.
- **Cascade Mode with Fast Interrupt**: In this mode, there are two new ports enabled with each instance of the core. These ports are `Interrupt_Address_In` and `Processor_Ack_Out`.

See Port Descriptions for more information about these ports.

Cascade Mode Interrupt Behavior

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

The master instance of AXI INTC first addresses all interrupts set between `INTR(0)` to `INTR(30)`. Then, it only provides service to the `INTR(31)` bit in the case of cascade mode.

TIP: *The use of C_EN_CASCADE = 0 and C_CASCADE_MASTER = 1 is illegal, and Design Rule Check (DRC) errors are flagged.*

Using C_EN_CASCADE = 1 and C_CASCADE_MASTER = 1

This parameter set is intended only when there are more than 32 interrupts present in the system. Use this parameter combination for the *first instance* of the AXI INTC core which directly communicates to the processor. No other instance of further AXI INTC should be allowed to communicate with the processor directly.

The first instance of AXI INTC is considered as Cascade Mode Master that directly communicates with the processor. The remaining AXI INTC instances are considered secondary instances.

Using C_EN_CASCADE = 1 and C_CASCADE_MASTER = 0

This parameter set is intended only when there are more than 32 interrupts present in the system. Use this parameter combination for the *second and subsequent instances* of the AXI INTC core that still have lower-level instances of AXI INTC. This core instance communicates with the top-level master AXI INTC instance as well as lower level instances of the core.

Using C_EN_CASCADE = 0 and C_CASCADE_MASTER = 0

This parameter set it intended for use only for the last instance of AXI INTC core. Use this parameter combination for the *last instance* of AXI INTC core.

Setting the C_HAS_FAST and C_MB_CLK_NOT_CONNECTED Parameters

IMPORTANT: *You are responsible for assigning the correct parameter configurations for each instance of the AXI INTC core in the system.*

- If any of the lower-level modules have `C_HAS_FAST = 1`, all the AXI INTC instances must have `C_HAS_FAST = 1`.
- When `C_MB_CLK_NOT_CONNECTED = 0` is used (for all AXI INTC instances), each AXI INTC core instance should be connected to the `Processor_clk` and `Processor_rst`.
- All AXI INTC instances should match their `INTR(31)` input pin with the `IRQ` of the lower module. The `INTR(31)` bit polarity (such as, rising edges, falling edges, high levels, and low levels) of one AXI INTC module should match the IRQ behavior (edge/level) of the lower instance AXI INTC module.

RECOMMENDED: *Assign the synchronizers `C_DISABLE_SYNCHRONIZERS = 0` when the processor clock and core clock are different. All AXI INTC instances should receive the same AXI clock.*

AXI INTC Usage in Cascade Mode

RECOMMENDED: *Use the same parameter configurations for all AXI INTC instances when used in Cascade Mode.*

Figure 3-2 shows how the fast interrupt of AXI INTC instances is configured in a given system. This is one of the reference modes for using the core in system.

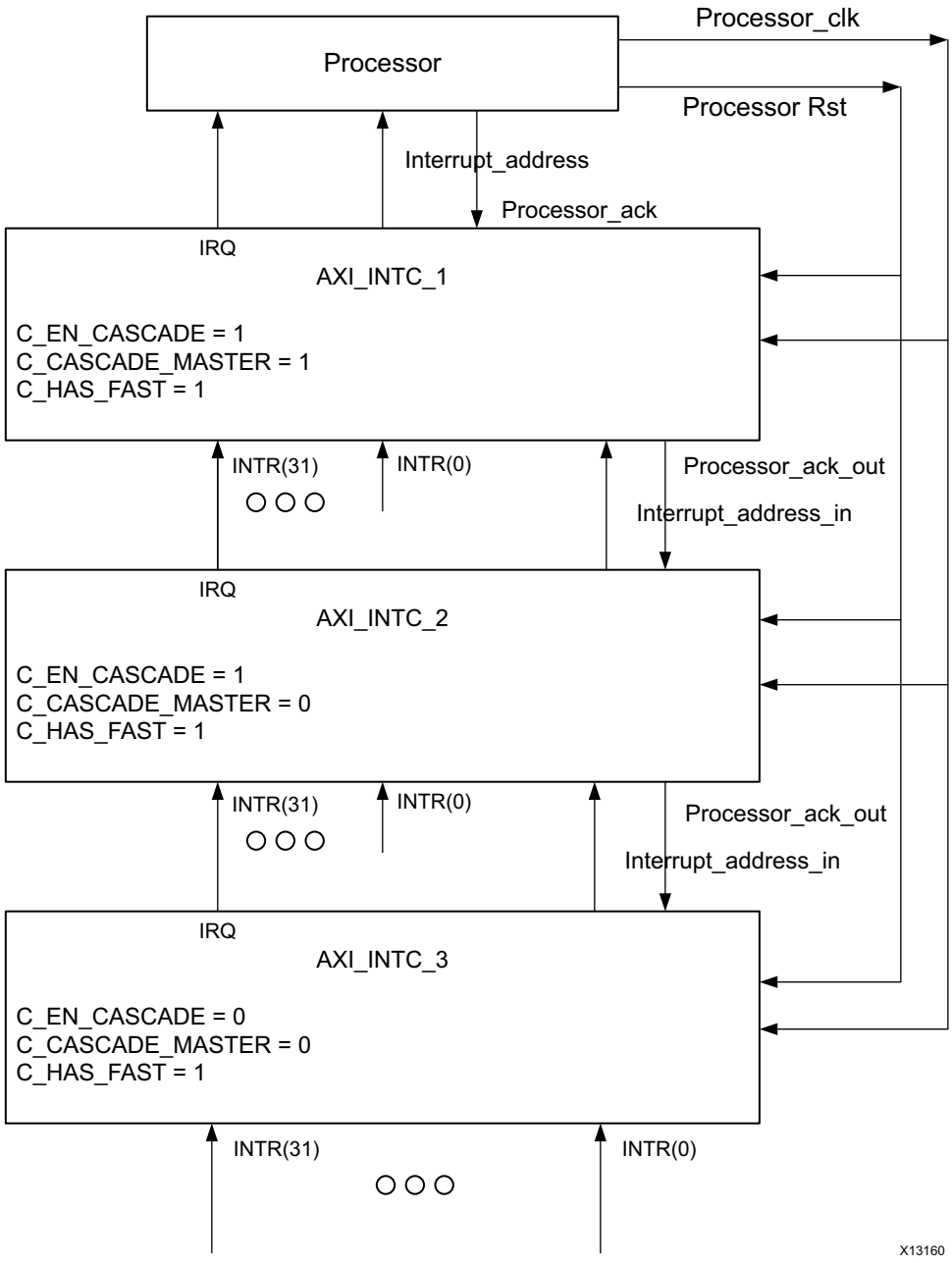

Figure 3-2: **Fast Interrupt of AXI INTC Instances**

General Design Guidelines

This section provides an overview of software initialization and communication with an AXI INTC core. The number of interrupt inputs that a AXI INTC core has is set by the C_NUM_INTR_INPUTS generic described in Table 2-7. The first input is always INT0 and is mapped to each LSB of each register (except for the IVR and MER).

A valid interrupt input signal is any signal that provides the correct polarity and type of interrupt input. Valid hardware interrupt inputs are rising edges, falling edges, high levels, and low levels. Valid interrupts can be generated if the HIE bit has not been set. The polarity and type of each hardware interrupt input is specified in the AXI INTC generics C_KIND_OF_INTR, C_KIND_OF_EDGE and C_KIND_OF_LVL (see Table 2-8).

Software interrupts do not have any polarity or type associated with them, so, until the HIE bit has been set, they are always valid. Any valid interrupt input signal that is applied to an enabled interrupt input generates a corresponding interrupt request within the AXI INTC core.

All interrupt requests are combined (an OR function) to form a single interrupt request output. Interrupts are enabled individually by dedicated interrupt enable bits and the external Irq output signal is then gated by a Master Enable (ME) bit.

During power-up or reset, the AXI INTC core is put into 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 input must be set to 1. This allows the AXI INTC core to begin accepting interrupt input signals. 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 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. A corresponding interrupt request is generated if that interrupt is enabled, and interrupt handling proceeds normally.
4. After software testing has been completed, or if software testing is not performed, a 1 is written to the HIE bit, which enables the hardware interrupt inputs and disables any further software generated interrupts.
5. After a 1 has been written to the HIE bit, any further writes to this bit have no effect. This feature prevents stray pointers from accidentally generating unwanted interrupt requests, while still allowing self-test software to perform system tests at power-up or after a reset.

Reading the ISR indicates which inputs are active. If present, the IPR indicates which enabled inputs are active. Reading the IVR provides the ordinal value of the highest priority interrupt that is enabled and active. In fast interrupt mode, that is when C_HAS_FAST = 1, reading any IVAR provides the interrupt vector address of that particular interrupt.

For example, if a valid interrupt signal has occurred on the INT3 interrupt input and nothing is active on INT2, INT1 and INT0, reading the IVR provides a value of 3. If INT0 becomes active, then reading the IVR provides a value of 0. If no interrupts are active or it is not present, reading the IVR returns all 1s.

Acknowledging an interrupt is achieved by writing a 1 to the corresponding bit location in the IAR. Disabling an interrupt by masking it (writing a 0 to the IER) does not clear the interrupt. That interrupt remains active but blocked until it is unmasked or cleared. Interrupts in fast interrupt mode are acknowledged through the PROCESSOR_ACK port, which writes to the corresponding bit in the IAR register.

An interrupt acknowledge bit clears the corresponding interrupt request. However, if a valid interrupt signal remains on that input (another edge occurs or an active level still exists on the corresponding interrupt input), a new interrupt request output is generated. Also, all interrupt requests are combined to form the Irq output so any remaining interrupt requests that have not been acknowledged cause a new interrupt request output to be generated.

The software can disable the interrupt request output at any time by writing a 0 to the ME bit in the MER. This effectively masks all interrupts for that AXI INTC core. Alternatively, interrupt inputs can be selectively masked by writing a 0 to each bit location in the IER that corresponds to an input that is to be masked. If present, SIE and CIE provide a convenient way to enable or disable (mask) and interrupt input without having to read, mask off, and then write back the IER. Writing a 1 to any bit locations in the SIE sets the corresponding bits in the IER without affecting any other IER bits.

Writing a 1 to any bit locations in the CIE clears the corresponding bits in the IER without affecting any other IER bits.

Timing Diagrams

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

- Configured Input Interrupt (Intr) for Edge sensitive (rising), Output Interrupt Request (Irq) to Level sensitive (active-High), disabled fast interrupt logic (C_HAS_FAST = 0). Timing diagram is shown in Figure 3-3.

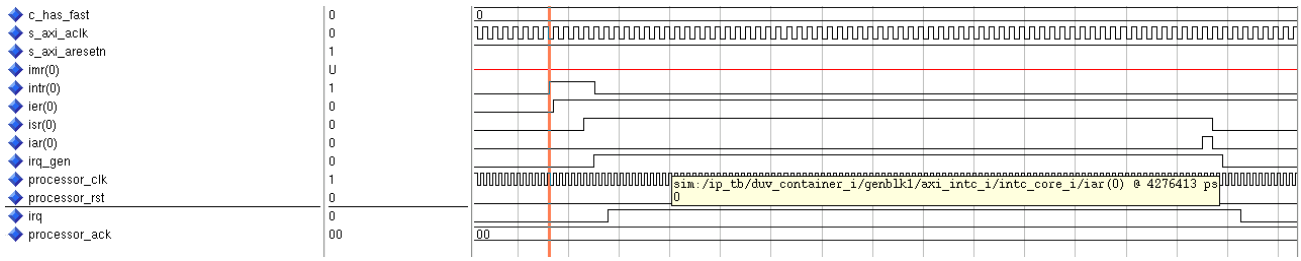

Figure 3-3: **Input Intr - Edge Sensitive (Rising), Output Irq - Level Sensitive (High), C_HAS_FAST = 0**

- Configured Input Interrupt (Intr) for Edge sensitive (rising), Output Interrupt Request (Irq) to Edge sensitive (Rising), disabled fast interrupt logic (C_HAS_FAST = 0). Timing diagram is shown in Figure 3-4.

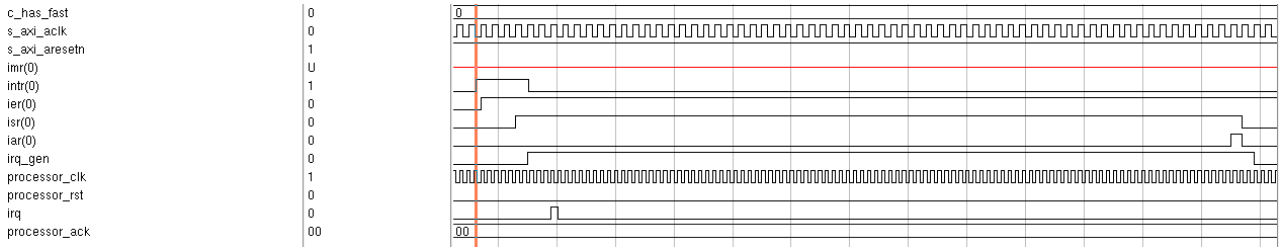

Figure 3-4: **Input Intr - Edge Sensitive (Rising), Output Irq - Edge Sensitive (Rising), C_HAS_FAST = 0**

- Configured Input Interrupt (Intr) for Level sensitive (active-High), Output Interrupt Request (Irq) to Level sensitive (active-High), disabled fast interrupt logic (C_HAS_FAST = 0). Timing diagram is shown in Figure 3-5.

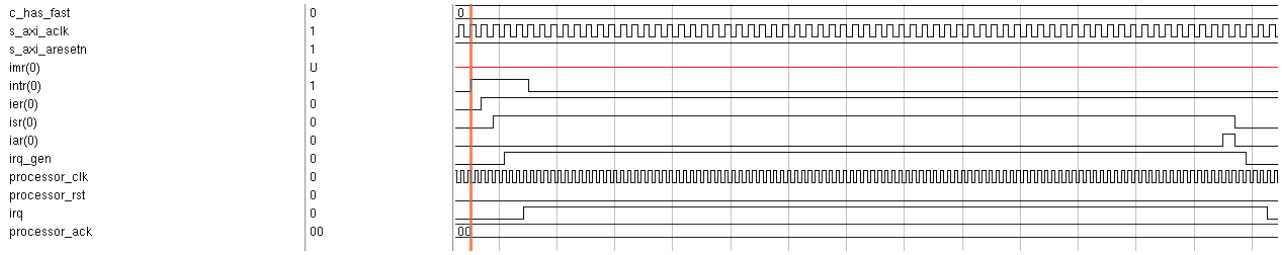

Figure 3-5: **Input Intr - Level Sensitive (High), Output Irq - Level Sensitive (High), C_HAS_FAST = 0**

- Configured Input Interrupt (Intr) for Level sensitive (active-High), Output Interrupt Request (Irq) to Edge sensitive (Rising), disabled fast interrupt logic (C_HAS_FAST = 0). Timing diagram is shown in Figure 3-6.

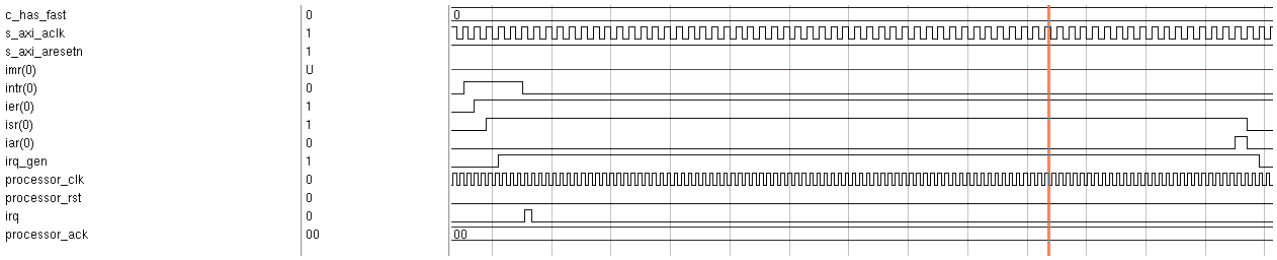

Figure 3-6: **Input Intr - Level Sensitive (High), Output Irq - Edge Sensitive (Rising), C_HAS_FAST = 0**

- Configured Input Interrupt (Intr) for Edge sensitive (rising), include fast logic to High (C_HAS_FAST = 1) and mode set to fast interrupt mode (IMR(i) = 1). Timing diagram is shown in Figure 3-7.

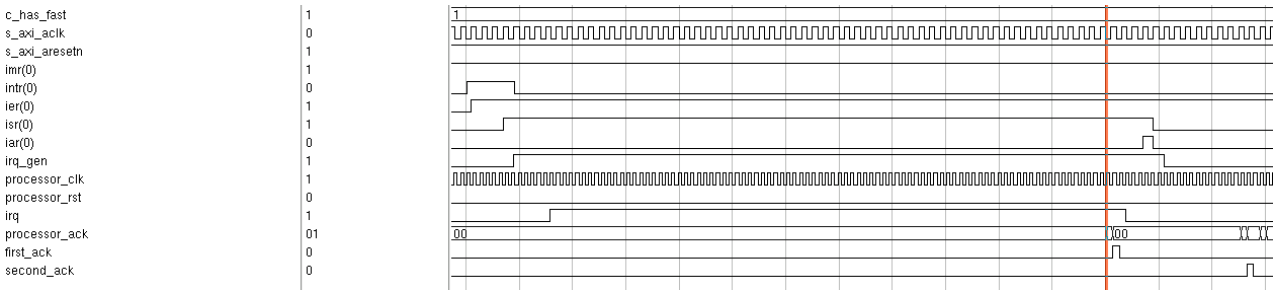

Figure 3-7: **Input Intr - Edge Sensitive (Rising), C_HAS_FAST = 1 and IMR(i) = 1**

- Configured Input Interrupt (Intr) for Level sensitive (active-High), include fast Logic to High (C_HAS_FAST = 1) and mode set to fast interrupt mode (IMR(i) = 1). Timing diagram is shown in Figure 3-8.

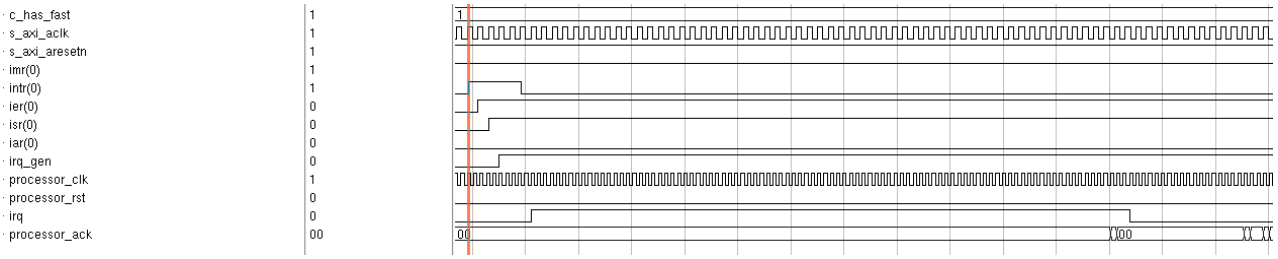

Figure 3-8: **Input Intr - Level Sensitive (active High), C_HAS_FAST = 1 and IMR(i) = '1**

Clocking

The AXI INTC core works on the AXI clock in default mode. Based on requirements, the core can be configured to operate on a MicroBlaze clock only at interrupt signal level. For the maximum supported values, see Resource Utilization.

Resets

The AXI INTC works on `AXI_ARESETn`, which is active-Low.

SECTION II: VIVADO DESIGN SUITE

Customizing and Generating the Core

Constraining the Core

Customizing and Generating the Core

This chapter includes information about how to customize and generate the core in the Vivado™ Design Suite.

GUI

The Vivado Design Suite customize options for the AXI INTC instance, shown in Figure 4-1, includes a Basic Tab and an Advanced Tab.

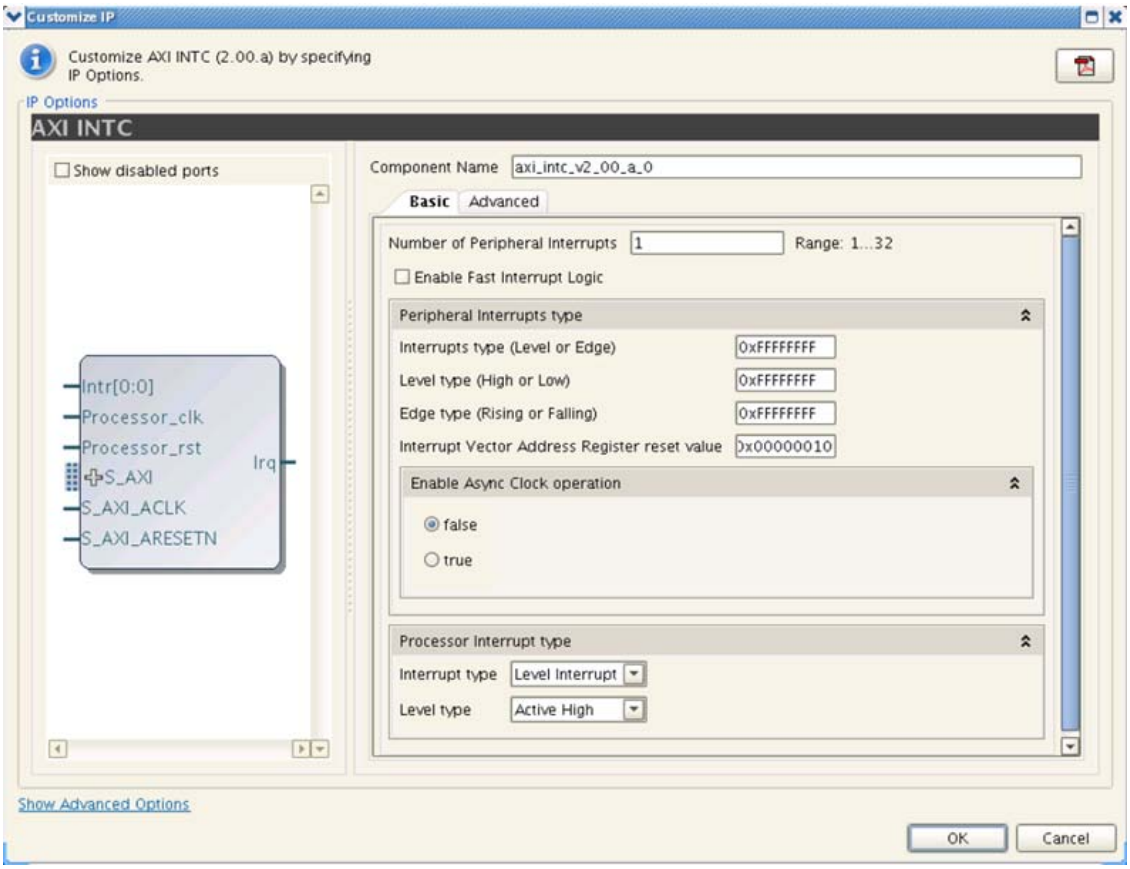

Figure 4-1: **AXI INTC**

Basic Tab

In Basic Tab, you can:

- Configure the number of interrupts.
- Enable the fast mode interrupt.
- Provide the inputs for each bit of interrupt, such as edge sensitive or level sensitive.
- Provide the Interrupt Vector Table reference value.
- Configure the mechanism of interrupt provided to the processor. The IRQ generated from the core is either edge or level sensitive.
- In standalone mode, if the processor clock and AXI clock both are present and if they are different from the origin of source, you can set the logic, which crosses these path, as false paths.

Advanced Tab

In Advanced Tab, you can:

- Configure the core in cascade mode.
- Set registers such as Set Interrupt Enable, Clear Interrupt Enable, Enable Interrupt Vector Register, Enable Interrupt Pending Registers.

Output Generation

This section provides detailed information about the files and the directory structure generated by the Xilinx Vivado tool. The *component name* of the IP generated is axi_intc_v2_00_a_0.

Note: The file output list might include some or all of the following files.

- **<project_directory>**
  Top-level project directory; name is user-defined. The output files generated from the Vivado IP Catalog are placed in the project directory.
  - <project_directory>.srcs
    - sources_1
      - ip
        - component_name
          - component_name
            AXI INTC Controller folders and files, including instantiation template files: `<component_name>.veo/.xci/.xml`.
        - <component_name>/sim
          Simulation wrapper
        - <component_name>/synth
          Synthesis wrapper
          - <component_name>/<component_name>/hdl/src/vhdl
            AXI INTC core RTL files

Constraining the Core

There are no constraints associated with this core.

SECTION III: ISE DESIGN SUITE

Customizing and Generating the Core

Constraining the Core

Customizing and Generating the Core

This chapter includes information about using Xilinx tools to customize and generate the core in the ISE® Design Suite environment.

GUI

The AXI INTC GUI includes two configuration GUI tabs: User and System.

IMPORTANT: *The Interconnect Settings for BUSIF tab is not currently used. Disregard this tab.*

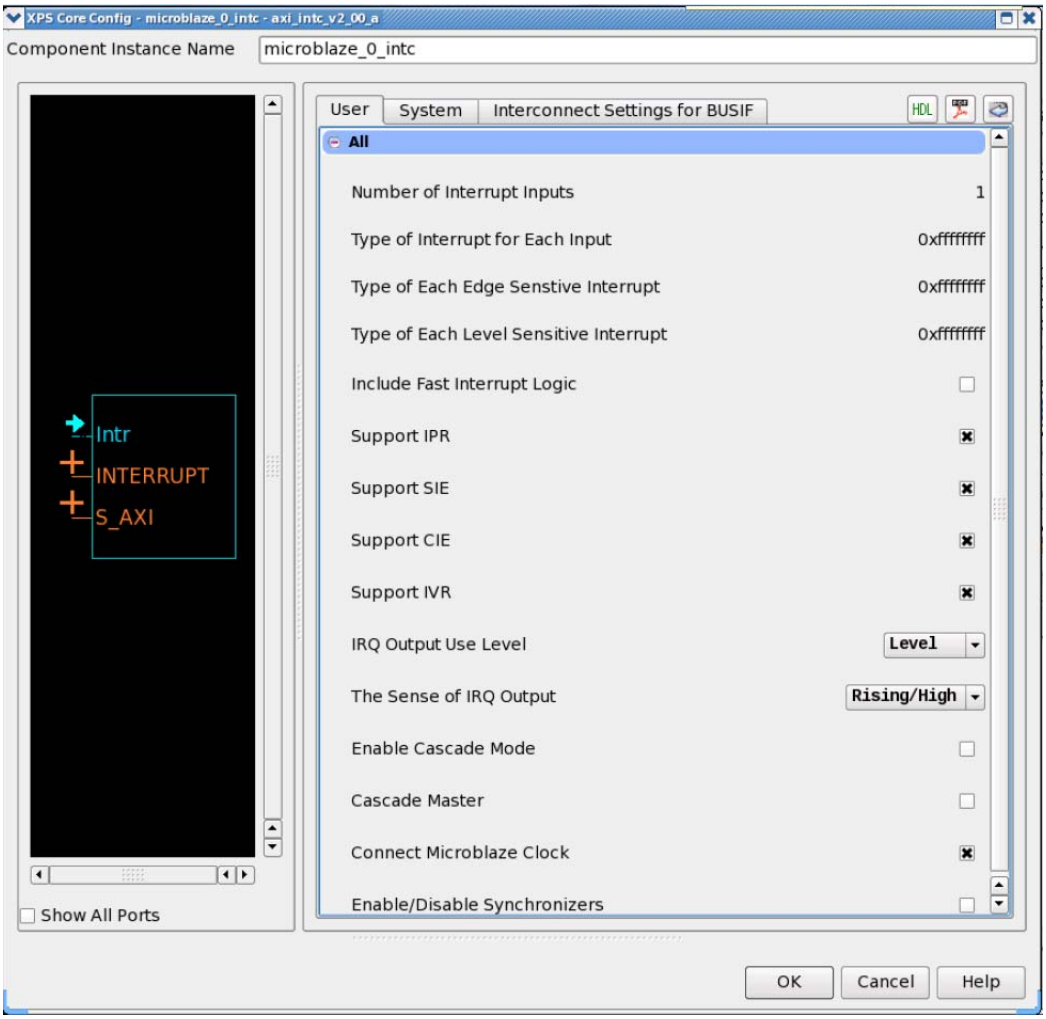

Figure 6-1: **AXI INTC XPS GUI**

User

The User tab, you can:

- Define the component name in the Component Instance Name option.
- Configure the number of interrupts.
- Enable the fast mode interrupt.
- Provide the inputs for each bit of interrupt like edge sensitive or level sensitive.
- Provide the Interrupt Vector Table reference value.
- Configure the mechanism of interrupt provided to the processor. The IRQ generated from the core is either edge or level sensitive.

- Configure the core in Cascade Mode, then include registers like Set Interrupt Enable, Clear Interrupt Enable, Enable Interrupt Vector Register, Enable Interrupt Pending Registers.

System

The System tab includes the following options:

- **AXI Base Address and AXI High Address**: This section must be configured after the IP has been added to the Base System Builder in XPS.
- **AXI**: The AXI INTC uses an AXI4-Lite protocol. The data width is set to 32 and the address width is set to 9.

Parameter Values in the XCO File

To allow you to obtain an AXI INTC that is uniquely tailored for the system, certain features can be parameterized in the AXI INTC design. This allows you to configure a design that utilizes the resources required by the system only and that operates with the best possiblem performance. The features that can be parameterized in the AXI INTC core are as shown in Table 2-7, page 21.

Dependencies Between Parameters and I/O Signals

The dependencies between the AXI INTC core design parameters and I/O signals are described in Table 2-8, page 23. In addition, when certain features are parameterized out of the design, the related logic is no longer a part of the design. The unused input signals and related output signals are set to a specified value.

Output Generation

This section provides detailed information about the files and the directory structure generated by the Xilinx XPS tool.

The output files generated from the XPS IP Catalog are placed in the project directory. Some of the main directories and the files generated are listed here for reference.

<project directory>
Top-level project directory; name is user-defined.

data
This folder contains the top level UCF file generated by the XPS tool. User can edit this file as per requirement.

- hdl

  This folder contains the wrapper file for each of the IP core instantiated in the MHS file. This folder also contains the system level wrapper file where all the IPs are instantiated.

- implementation

  This folder contains the ISE related files including the NGC, NGD, NCD, TWR, BIT files etc.

- pcores

  This folder is optional for user usage. If user has local IP which is not the part of IP catalog provided by Xilinx, then user can add his IP here as per Xilinx Standard IP Guidelines and refer it in the MHS file of the project with proper instance name and other configuration.

- synthesis

  This folder contains the synthesis related files.

<project directory>

The output files generated from the Xilinx XPS IP Catalog are placed in the project directory.

Table 6-1: **Project Directory**

Name	Description
<project_dir>	
platgen.opt	This file contains the options used in the Platform generation (Platgen) process.
Platgen	This file contains the log, info, warning, and messages for the steps carried out during the Platgen process.
system.make	This is required for the XPS tool.
system_incl.make	This is required for the XPS tool.
system.xmp	This is required for the XPS tool. This file has information about the project, and device used.

Back to Top

Constraining the Core

There are no constraints associated with this core

SECTION IV: APPENDICES

Migrating

Debugging

Additional Resources

Appendix A

Migrating

For information on migrating to the Vivado™ Design Suite, see *Vivado Design Suite Migration Methodology Guide (UG911)* [Ref 2].

Debugging

This appendix includes details about resources available on the Xilinx Support website and debugging tools. In addition, this appendix provides a step-by-step debugging process and a flow diagram to guide you through debugging the AXI INTC core.

The following topics are included in this appendix:

Finding Help on Xilinx.com

To help in the design and debug process when using the AXI INTC, the Xilinx Support web page (www.xilinx.com/support) contains key resources such as product documentation, release notes, answer records, information about known issues, and links for opening a Technical Support WebCase.

Documentation

This product guide is the main document associated with the AXI INTC core. This guide, along with documentation related to all products that aid in the design process, can be found on the Xilinx Support web page (www.xilinx.com/support) or by using the Xilinx Documentation Navigator.

Download the Xilinx Documentation Navigator from the Design Tools tab on the Downloads page (www.xilinx.com/download). For more information about this tool and the features available, open the online help after installation.

Release Notes

Known issues for all cores, including the AXI INTC are described in the IP Release Notes Guide (XTP025).

Known Issues

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 are listed below, and can also 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 message(s)
- Summary of the issue encountered

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

Answer Records for the AXI INTC

Known issues for all cores, including AXI INTC are described in the IP Release Notes Guide (XTP025).

Contacting Technical Support

Xilinx provides premier technical support for customers encountering issues that require additional assistance.

To contact Xilinx Technical Support:

1. Navigate to www.xilinx.com/support.
2. Open a WebCase by selecting the WebCase link located under Support Quick Links.

When opening a WebCase, include:

- Target FPGA including package and speed grade.
- All applicable Xilinx Design Tools and simulator software versions.
- Additional files based on the specific issue might also be required. See the relevant sections in this debug guide for guidelines about which file (or files) to include with the WebCase.

Interface Debug

AXI4-Lite Interfaces

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

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see the Xilinx Support website at:

www.xilinx.com/support.

For a glossary of technical terms used in Xilinx documentation, see:

www.xilinx.com/company/terms.htm.

References

These documents provide supplemental material useful with this product guide:

1. *AMBA AXI4-Stream Protocol Specification*
2. *Vivado™ Design Suite Migration Methodology Guide (UG911)*
3. *LogiCORE IP AXI Lite IPIF (v1.01a) Data Sheet (DS765)*
4. 7 Series FPGAs Overview (DS180)
5. Virtex-6 Family Overview (DS150)
6. Spartan-6 Family Overview (DS160)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
12/18/12	1.0	• Initial product guide release. Replaces *LogiCORE IP AXI INTC Data Sheet (DS747)*. • Added Cascade Mode.

Notice of Disclaimer