LogiCORE IP AXI INTC (v1.04a)

DS747 June 19, 2013 **Product Specification**

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
- Single interrupt output
- Cascadable to provide additional interrupt inputs
- Priority between interrupt requests is determined by vector position. The least significant bit (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
- Configurable software interrupt capability

LogiCORE IP Facts Table	
Core Specifics	
Supported Device Family[1]	Zynq®-7000, Virtex®-7, Kintex®-7, Artix®-7, Virtex-6, Spartan®-6
Supported User Interfaces	AXI4-Lite
Resources	See Table 17 through Table 19.
Provided with Core	
Documentation	Product Specification
Design Files	VHDL
Example Design	Not Provided
Test Bench	Not Provided
Constraints File	Not Provided
Simulation Model	Not Provided
Supported S/W Driver[2]	Standalone
Tested Design Flows[3]	
Design Entry	ISE® Design Suite v14.6
Simulation	Mentor Graphics Questa® SIM
Synthesis	Xilinx Synthesis Technology (XST)
Support	
Provided by Xilinx @ www.xilinx.com/support	

1. For a complete list of supported derivative devices, see the Embedded Edition Derivative Device Support.
2. 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.
3. For the supported versions of the tools, see the Xilinx Design Tools: Release Notes Guide.

Functional Description

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.

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.

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 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.

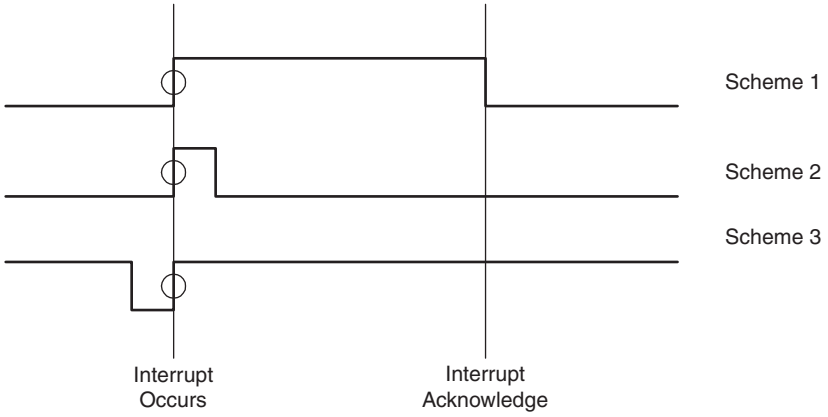


Figure 1: **Schemes for Generating Edges**

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 (IVR) 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 the 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 lower 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 interrupt request t (Irq). The Irq generated is cleared based on the `Processor_ack` signal, and the corresponding Interrupt Acknowledge Register (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 a Return from Interrupt (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.

Table 1: **Interrupt Acknowledgement Signal Actions**

Processor_ack Pattern	Action
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 interrupts. 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. Cascade Mode Interrupt describes how to set this mode and how it is used in the system.

Module Description

Figure 2 illustrates 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 the AXI INTC module
- Interrupt Controller (INTC) Core: Consists of the Int Det, Irq Gen, and Regs logical blocks

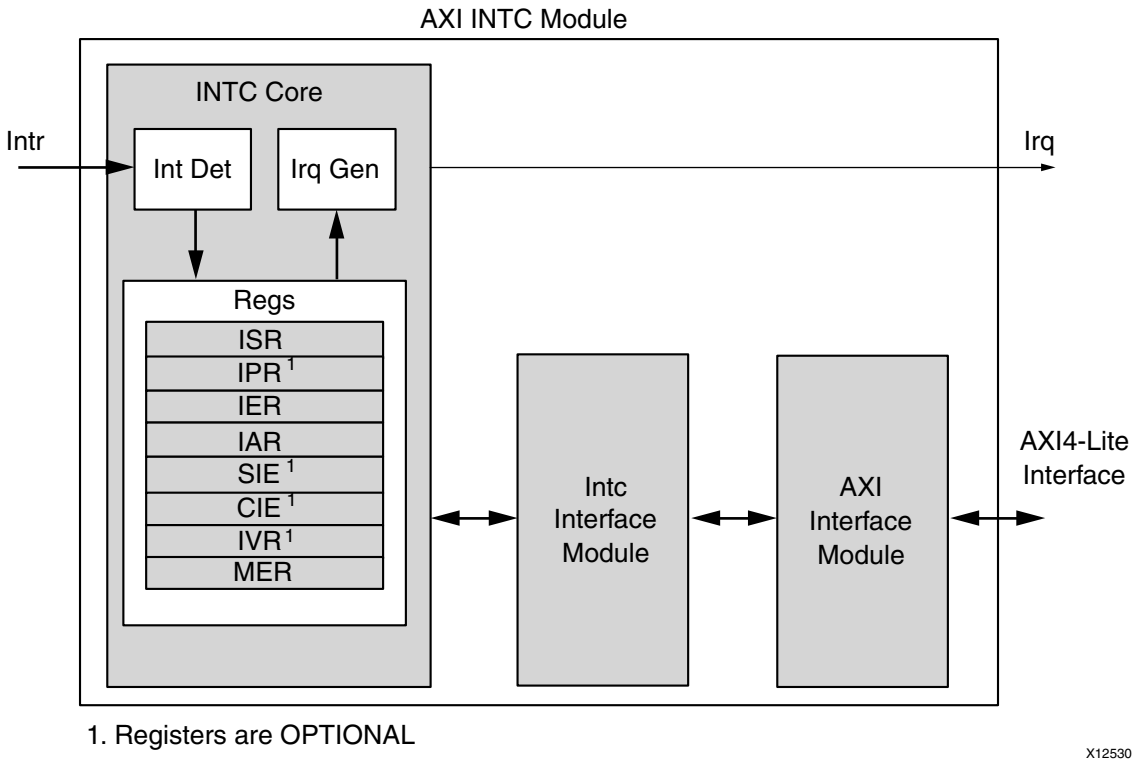


Figure 2: **AXI INTC in Normal Mode**

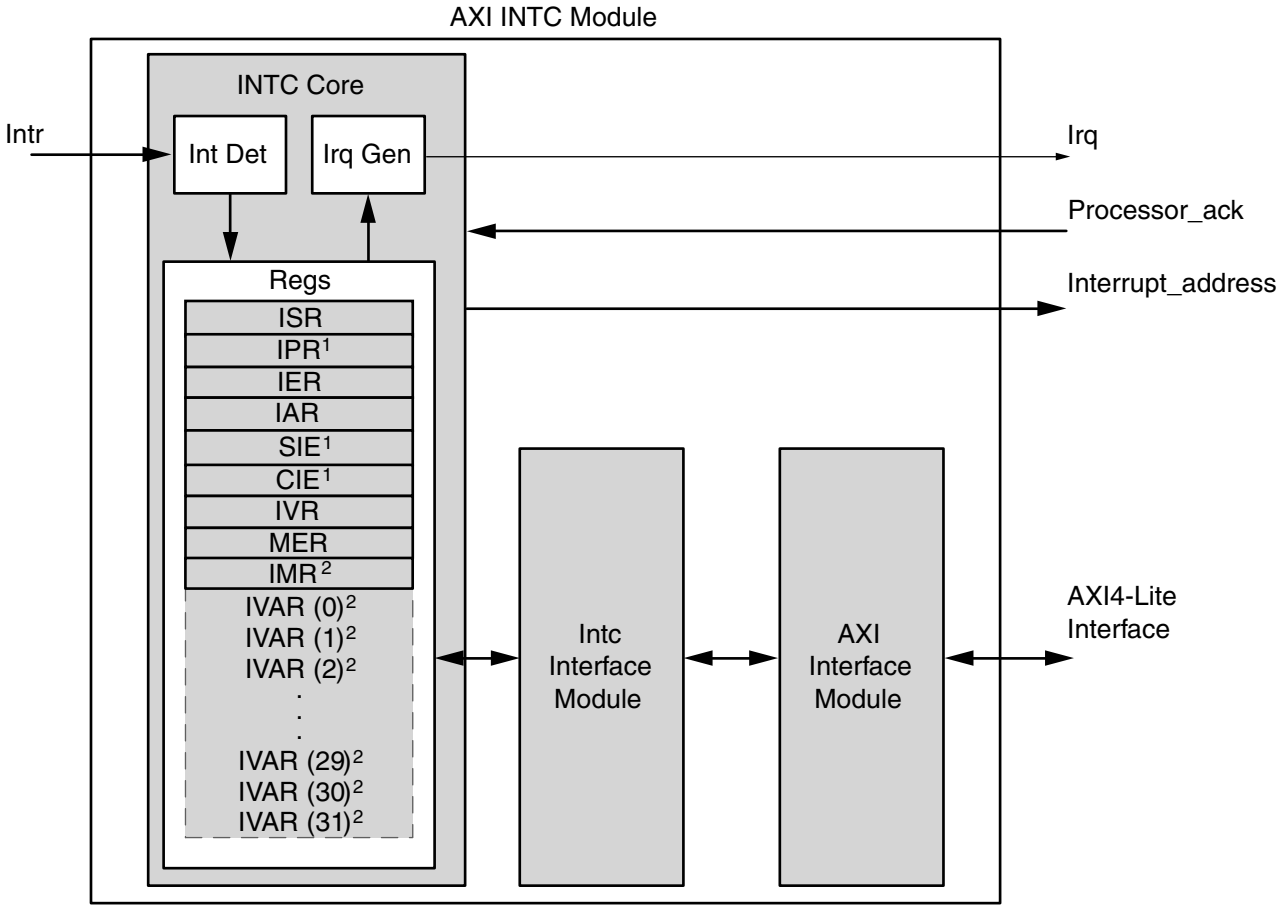


Figure 3: **AXI INTC in Fast Interrupt Mode**

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

The Interrupt Controller core consists of logical blocks as follows:

- Interrupt Detection (Int Det) - Detects the valid interrupt from all the interrupt inputs
- Programmer Registers (Regs) - Status and control registers
- Interrupt Request Generation (Irq Gen) - Generates the interrupt request output

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 see:

- Interrupt Status Register (ISR)
- Interrupt Pending Register (IPR)
- Interrupt Enable Register (IER)
- Interrupt Acknowledge Register (IAR)
- Set Interrupt Enables (SIE)
- Clear Interrupt Enables (CIE)
- Interrupt Vector Register (IVR)
- Master Enable Register (MER)
- Interrupt Mode Register (IMR)
- Interrupt Vector Address Register (IVAR)

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 the IER and MER to enable the interrupt generation. Irq Gen also resets the interrupt after an acknowledge. Irq Gen also writes the vector address of the active interrupt in the 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. See Design Parameters for more information on these parameters. This mode is optional and when default mode is used for these parameters, the core behaves with limited support of 32 interrupts. See Cascade Mode Interrupt for more information.

I/O Signals

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

Table 2: **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					
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
P14	S_AXI_ARVALID	AXI	Input	-	Read address valid. This signal indicates, when High, that the read address and control information is valid and remains stable until the address acknowledgement signal, S_AXI_ARREADY, is High.

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

Port	Signal Name	Interface	I/O	Initial State	Description
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
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 'Interrupt_address' port and available only when C_HAS_FAST=1 (for secondary instance(s) 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 the processor when the 31st bit, which is the cascaded interrupt, is served by the processor (for secondary instance(s) of the AXI INTC core .)

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 design a uniquely tailored AXI INTC core, certain features can be parameterized in the core. This allows configuration of a design that uses only required system resources, and that operates with the best possible performance. The features that can be parameterized in the AXI INTC core are listed in Table 3.

Table 3: **Design Parameters**

Generic	Feature/Description	Parameter Name	Allowable Values	Default Value	VHDL Type
System Parameter					
G1	Target FPGA family	C_FAMILY	zynq, kintex7, artix7, virtex7, spartan6, virtex6	virtex6	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 peripheral interrupt inputs	C_NUM_INTR_INPUTS	1-32	2	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	0	integer

Table 3: **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 cascade mode interrupt core instance as primary(7)	C_CASCADE_MASTERS	0 = No Cascade Mode Master 1 = Cascade Mode Master	0	integer
G19	Number of software interrupts	C_NUM_SW_INTR	0-31	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 when the processor clock is not connected to INTC core. When the processor clock is not connected, the IRQ to the processor is generated on the AXI clock. This flexibility is provided for backward compatibility of the core. The synchronizers in the design are disabled when the processor clock is not connected.
4. When the processor clock 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 the AXI INTC core (which has less than 32 interrupts) should have this parameter set to 0.
7. C_CASCADE_MASTER should be set to 1 only for 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 interfaces directly with the processor. None of the other instances of the AXI INTC core interface with processor even though these are configured in 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 4. 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 4: **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	-	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.
G17	C_EN_CASCADE_MODE	P26, P27	-	The parameter should be set only when the Cascade mode of interrupt is set.
G18	C_CASCADE_MASTERS	P26, P27	-	This parameter should be set only for the master instance of AXI INTC core.

Table 4: **Parameter-I/O Signal Dependencies *(Cont'd)***

Generic or Port	Name	Affects	Depends	Relationship Description
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 AXI interface. Each register is 32 bits (although some bits can be unused) and is accessed on a 4-byte boundary. Because AXI addresses are byte addresses, AXI INTC register offsets are located at integral multiples of four. The AXI INTC registers are read as little-endian data. The eight registers visible to the programmer are shown in Table 5 and described in this section. These points should be considered when reading and writing to registers:

- Writes to a read-only register returns an OK response with no register changes.
- Reads to a write-only register returns zero with an OK 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 return zero when read and successfully return with no register change when written.
- All registers uses C_NUM_INTR_INPUTS + C_NUM_SW_INTR to determine their width except the MER, which is always 2 bits wide and the IVR which is always 32 bits wide.

Table 5: **Register Descriptions** (1)

Address (hex)	Register Name	Access Type	Default Value (hex)	Description
0x0	ISR	Read/Write	0x0	Interrupt Status Register (ISR)
0x4	IPR	Read	0x0	Interrupt Pending Register (IPR)
0x8	IER	Read/Write	0x0	Interrupt Enable Register (IER)
0xC	IAR	Write	0x0	Interrupt Acknowledge Register (IAR)

Table 5: **Register Descriptions *(Cont'd)***[1]

Address (hex)	Register Name	Access Type	Default Value (hex)	Description
0x10	SIE	Write	0x0	Set Interrupt Enables (SIE)
0x14	CIE	Write	0x0	Clear Interrupt Enables (CIE)
0x18	IVR	Read	0xFFFF	Interrupt Vector Register (IVR)
0x1C	MER	Read/Write	0x0	Master Enable Register (MER)
0x20	IMR	Read/Write	0x0	Interrupt Mode Register (IMR)
0x100	IVAR[2]	Read/Write	0x10	Interrupt Vector Address Register (IVAR)
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	
0x138	IVAR[2]	Read/Write	0X10	Interrupt Vector Address Register (IVAR)
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 Least Significant Bit (LSB) of the ISR, IPR, IER, IAR, SIE, CIE and additional inputs correspond 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. Bits that are zero 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 the number of peripheral interrupts (C_NUM_INTR_INPUTS). The number of remaining bits is defined by the number of software interrupts 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 the number of peripheral interrupts plus the number of software interrupts (C_NUM_INTR_INPUTS + C_NUM_SW_INTR).

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

The ISR bits, up to the number of peripheral interrupts, are writable by software only 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 the 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 allows the software to generate hardware interrupts for test purposes until the HIE bit has been set, and to generate software interrupts at any time. After the HIE bit has been set (enabling the hardware interrupt inputs), then writing to the bits up to the number of peripheral interrupts in this register does nothing.

If there are fewer interrupts than the width of the data bus, writing a 1 to a non-existing interrupt does nothing and reading it returns it 0. The ISR is shown in Figure 4 and the bits are described in Table 6.

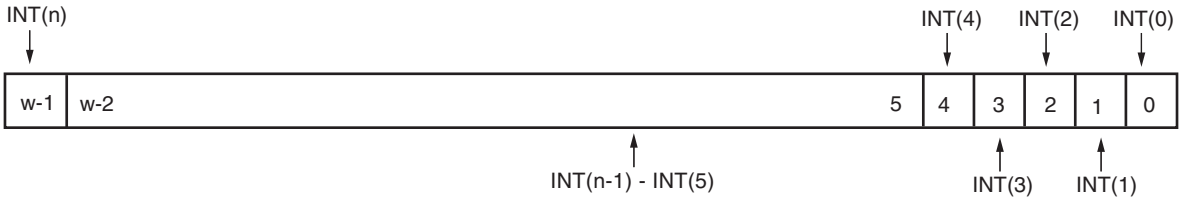


Figure 4: **Interrupt Status Register**

Table 6: **Interrupt Status Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
(w[1]-1):0	INT(n)-INT(0) ($n \leq w-1$)	Read/Write	0x0	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 core and can be parameterized out by setting C_HAS_IPR to 0. 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 interrupts than the width of the data bus, reading a non-existing interrupt returns zero. The IPR is shown in Figure 5 and the bits are described in Table 7.

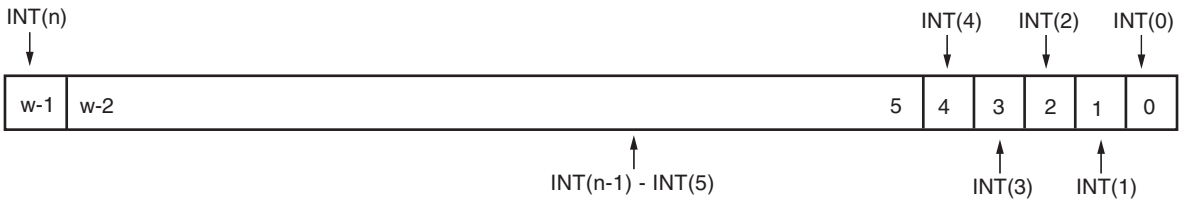


Figure 5: **Interrupt Pending Register**

Table 7: **Interrupt Pending Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
(w[(1)]-1):0	INT(n)-INT(0) ($n \leq w-1$)	Read/Write	0x0	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 the corresponding ISR bit to cause assertion of the INTC output. An IER bit set to 0 does not inhibit an interrupt condition from being captured, just from being reported. Writing a 0 to a bit disables, or masks, the generation of an interrupt output for the corresponding interrupt. Note however, that disabling an interrupt is not the same as clearing it. Disabling an active interrupt prevents that interrupt from reaching the IRQ output. When 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. Reading the IER indicates which interrupts are enabled, where a 1 indicates the interrupt is enabled and a 0 indicates the interrupt is disabled.

If there are fewer interrupts than the width of the data bus, writing a 1 to a non-existing interrupt returns a 0. The IER is shown in Figure 6 and the bits are described in Table 8.

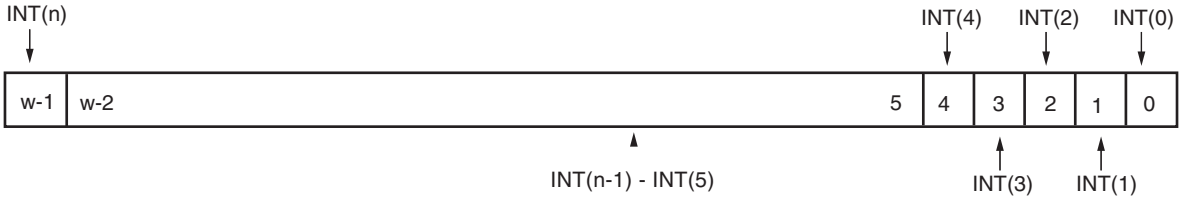


Figure 6: **Interrupt Enable Register**

Table 8: **Interrupt Enable Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
(w[(1)]-1):0	INT(n)-INT(0) ($n \leq w-1$)	Read/Write	0x0	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 location that clears the interrupt request associated with selected interrupts. Writing a 1 to a bit in the IAR clears the corresponding bit in the ISR.

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. 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 Master Enable (ME) bit in the MER is set).

Writing 0s has no effect as does writing a 1 to a bit that does not correspond to an active interrupt or for which an interrupt does not exist. The IAR is shown in Figure 7 and the bits are described in Table 9.

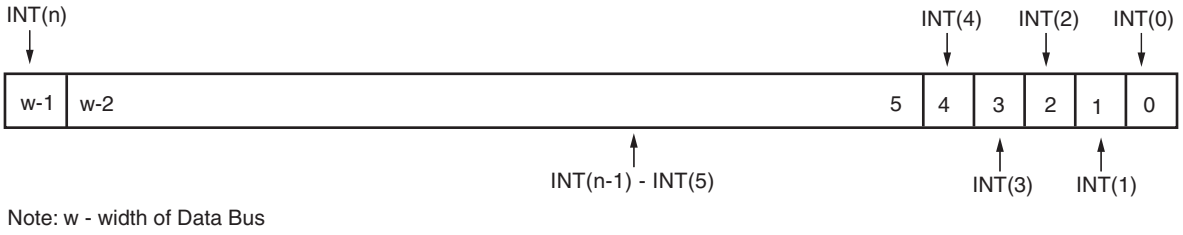


Figure 7: **Interrupt Acknowledge Register**

Table 9: **Interrupt Acknowledge Register Bit Definitions**

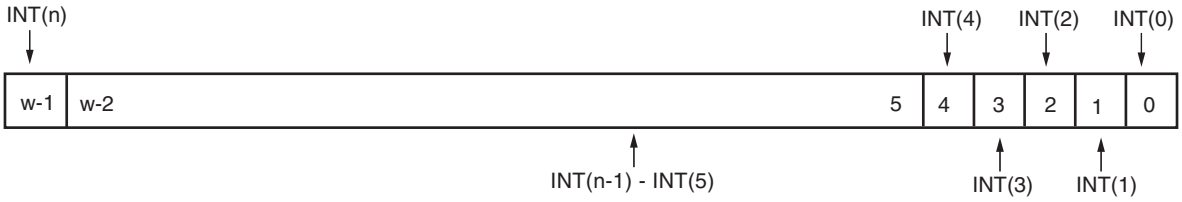
Bit(s)	Name	Core Access	Reset Value	Description
(w[1]-1):0	INT(n)-INT(0) ($n \leq w-1$)	Read/Write	0x0	Interrupt (n) - Interrupt (0) 0 - Not Active 1 - Active

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 not have any effect, as does writing a 1 to a bit location that does not correspond to an existing interrupt. The SIE is optional in the AXI INTC core and can be parametrized out by setting C_HAS_SIE to 0. The SIE register is shown in Figure 8 and the bits are described in Table 10.



Note: w - width of Data Bus

Figure 8: **Set Interrupt Enable Register**

Table 10: **Set Interrupt Enable Register Bit Definitions**

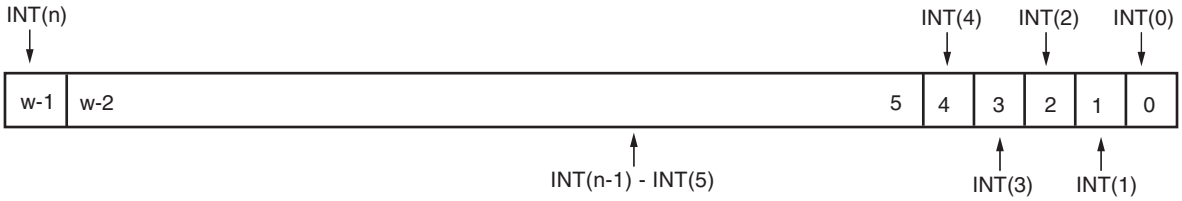
Bit(s)	Name	Core Access	Reset Value	Description
(w[1]-1):0	INT(n)-INT(0) ($n \leq w-1$)	Read/Write	0x0	Interrupt (n) - Interrupt (0) 0 - Not Active 1 - Active

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 not have any effect, as does writing a 1 to a bit location that does not correspond to an existing interrupt. The CIE is also optional in the AXI INTC core and can be parameterized out by setting C_HAS_CIE to 0. The CIE register is shown in Figure 9 and the bits are described in Table 11.



Note: w - width of Data Bus

Figure 9: **Clear Interrupt Enable Register**

Table 11: **Clear Interrupt Enable Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
(w[1]-1):0	INT(n)-INT(0) ($n \leq w-1$)	Read/Write	0x0	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. INT0 (always the LSB) is the highest priority interrupt input. Each successive input (to the left) has a corresponding lower interrupt priority.

If no interrupts are active, the IVR contains all 1s. The IVR acts as an index to provide the correct Interrupt Vector Address of the Irq. The IVR is shown in Figure 10 and described in Table 12.

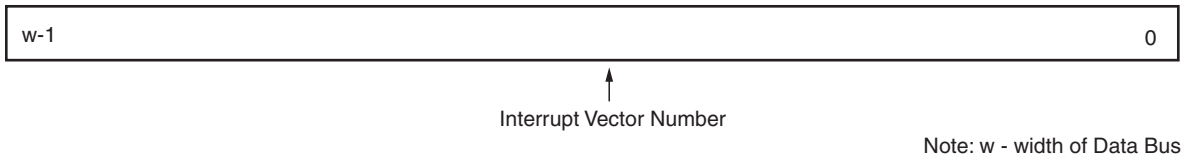


Figure 10: **Interrupt Vector Register**

Table 12: **Interrupt Vector Register 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

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 hardware 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. However, any software interrupts configured with C_NUM_SW_INTR remain writable. 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. The MER is shown in Figure 11 and is described in Table 13.

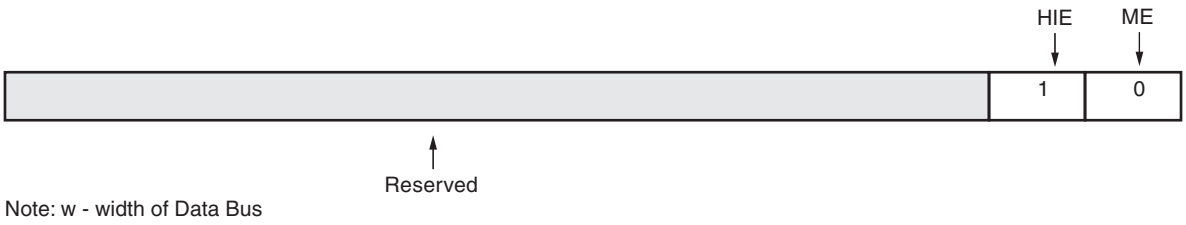


Figure 11: **Master Enable Register**

Table 13: **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 - Generating HW interrupts from SW 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. The IMR 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 the IMR. Writing 0 to any bit position sets the corresponding interrupt to normal interrupt mode. Writing 1 to any bit position sets the corresponding interrupt to fast interrupt mode. Unused bit positions in the IMR return zero.

The IMR is shown in Figure 12 and is described in Figure 14.

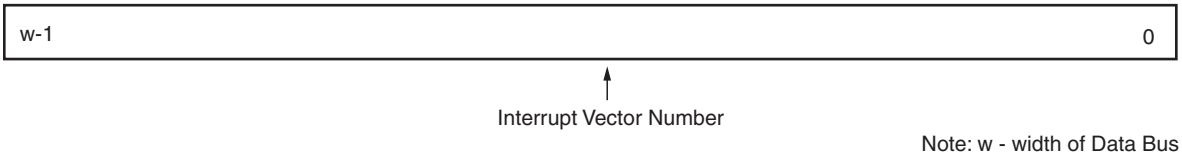


Figure 12: **Interrupt Mode Register**

Table 14: **Interrupt Mode Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
(w[1]-1):0	INT(n)-INT(0) ($n \leq w-1$)	Read/Write	0x0	Interrupt (n) - Interrupt (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 + C_NUM_SW_INTR 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`. The IVARs 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 + C_NUM_SW_INTR interrupts. The address of the interrupt with the highest priority is transmitted out along with `Irq`.

If all of the 32 interrupts are not used, any read on the unused register address returns zero. Writing 1s or 0s to any bit location has no effect.

The IVAR can be accessed through the AXI interface. All IVARs 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. Therefore, writing to these registers should be done when ME is 0.

The IVAR is shown in Figure 13 and is described in Table 15.

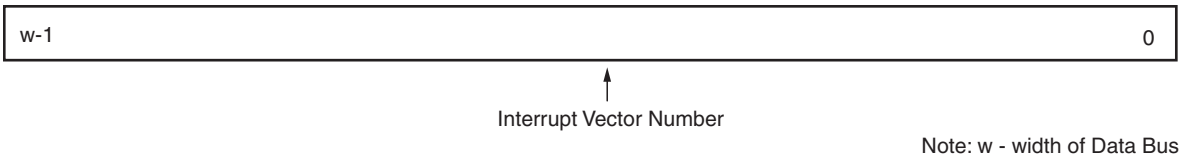


Figure 13: **Interrupt Vector Address Register**

Table 15: **Interrupt Vector Address Register Bit Definitions**

Bit(s)	Name	Core Access	Reset Value	Description
(w[(1)]-1):0	Interrupt Vector Address	Read/Write	0x0	Interrupt vector address of the corresponding interrupt

Notes:

1. w - Width of Data Bus

Cascade Mode Interrupt

This functionality is enabled when the system needs more than 32 interrupts. A single instance of the AXI INTC can handle 32 interrupts maximum. Two main parameters are required 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 instances. Table 16 shows the parameter combination used for cascade mode.

Table 16: **Parameter Combination for Cascade Interrupt Mode**

C_EN_CASCADE_MODE Default = 0	C_CASCADE_MASTER Default = 0	Operation for the AXI INTC instances
0	0	N/A. In this mode, there is no cascade interrupt feature available. It means only one instance is allowed in the system.
0	1	N/A. The C_CASCADE_MASTER parameter is allowed to set only when C_EN_CASCADE_MODE is set to 1.

Table 16: **Parameter Combination for Cascade Interrupt Mode *(Cont'd)***

C_EN_CASCADE_MODE Default = 0	C_CASCADE_MASTER Default = 0	Operation for the AXI INTC instances
1	0	This is applicable only for the lower instance of the AXI INTC module. Based on the parameter settings for this instance, the ports are connected to the primary or upstream instance of the AXI INTC core. Bit 31 of INTR is considered as the cascaded interrupt bit. No other bits are considered valid connections to use in cascade mode. (For such connections, the particular INTR pin is treated as a 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. Bit 31 of INTR is considered as the cascaded interrupt bit. No other bits are allowed to be connected to use for cascade mode.

Figure 14 shows how cascade mode interacts in a system.

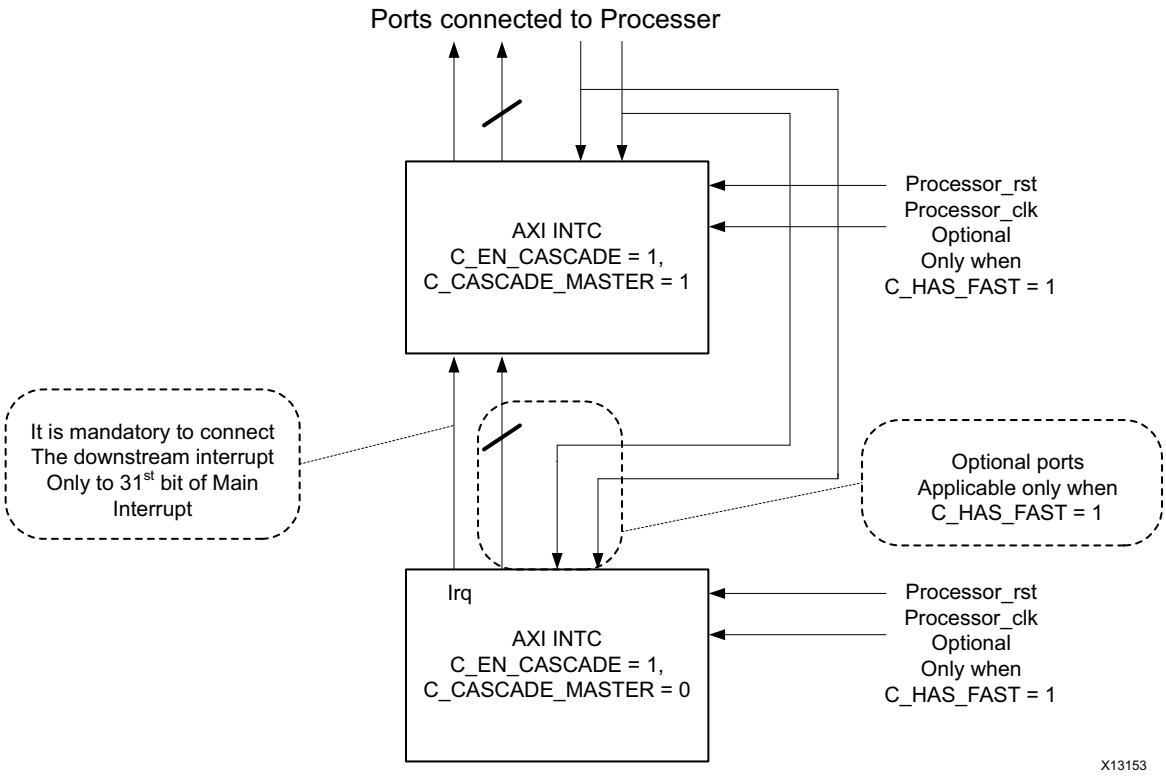


Figure 14: **Cascade Mode**

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, two new ports are enabled with each instance of the core. These ports are `Interrupt_Address_In` and `Processor_Ack_Out`.

See I/O Signals for more information about these ports.

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 16, there are three types of AXI INTC instantiations possible in cascade mode.

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 cascade mode.

Note: The use of Enable Cascade Mode Interrupt = 0 and Cascade Mode Master = 1 is not permitted, and Design Rule Check (DRC) errors are flagged.

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

This parameter set is intended only when there are more than 32 interrupts in the system. Use this parameter combination for the first instance of the AXI INTC core that 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 the cascade mode master that directly communicates with the processor. The remaining AXI INTC instances are considered secondary instances.

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

This parameter set is only for 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.

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

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

Setting the Enable Fast Interrupt and MicroBlaze Clock Connected Parameters

Note: Ensure that you assign 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 set to 1, all the AXI INTC instances must have C_HAS_FAST set to 1.
- When C_MB_CLK_NOT_CONNECTED = 0 is used (for all AXI INTC instances)
  - every AXI INTC core instance should be connected to `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 (edge or level) of one AXI INTC module should match the `Irq` behavior (edge/level) of the secondary AXI INTC module.

Note: Set the synchronizers C_DISABLE_SYNCHRONIZERS to 0 when the processor clock and core clock are different. All AXI INTC instances should receive the same AXI clock.

AXI INTC Use in Cascade Mode

Figure 15 shows how the fast interrupt of AXI INTC instances is configured in a system. This is one of the reference modes for using the core in a system.

Note: Use the same parameter configurations for all AXI INTC instances when used in cascade mode.

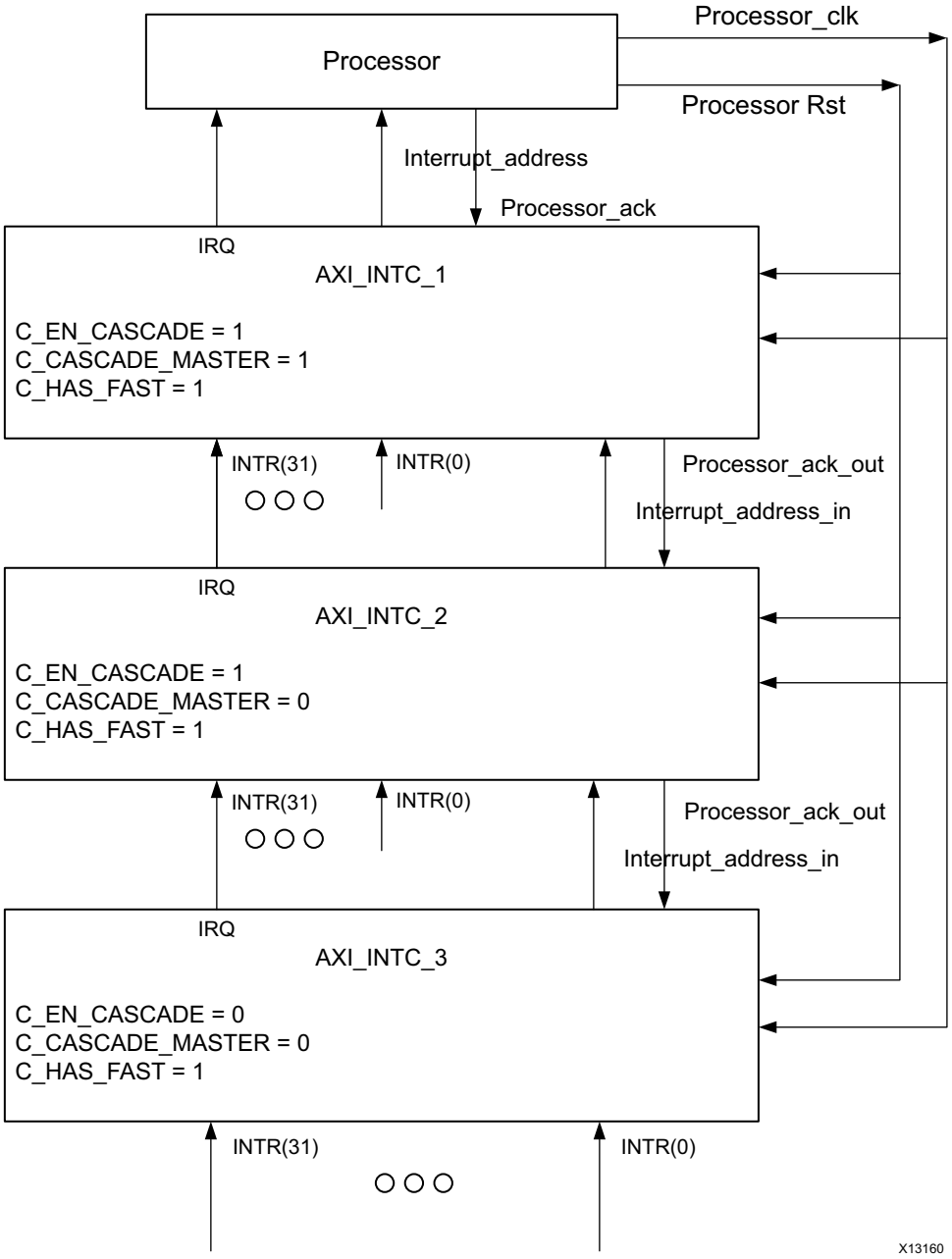


Figure 15: **Fast Interrupt of AXI INTC Instances**

Software Considerations

This section provides an overview of software initialization and communication with an AXI INTC core. The number of hardware interrupt inputs that a AXI INTC core has is set by C_NUM_INTR_INPUTS, and the number of software interrupts is set by C_NUM_SW_INTR, both described in Table 3. The first hardware interrupt input is always INT(0) and is mapped to the LSB of each register (except for the IVR and MER). The last hardware interrupt input is mapped to INT(C_NUM_INTR_INPUTS - 1). If any software interrupts are defined, the first software interrupt is mapped to INT(C_NUM_INTR_INPUTS) and the last to INT(C_NUM_INTR_INPUTS + C_NUM_SW_INTR - 1).

A valid hardware interrupt input signal is any signal that provides the correct polarity and type of interrupt input. Valid hardware interrupt inputs are rising or falling edges and high or 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 4).

Software generated hardware 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 (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 the 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 must be set to 1. This allows the AXI INTC core to begin accepting interrupt input signals and software interrupts. INT(0) has the highest priority, and it corresponds to the 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 HIE and the 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. 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 hardware 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 interrupts are active. If present, the IPR indicates which enabled interrupts are active. Reading the IVR provides the ordinal value of the highest priority interrupt that is enabled and active. In fast interrupt mode (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 clears the corresponding bit in the IAR.

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 interrupt that is to be masked. If present, SIE and CIE provide a convenient way to enable or disable (mask) an interrupt without having to read, mask off, and then write back the IER. Writing a 1 to any bit location(s) in the SIE set(s) the corresponding bit(s) in the IER without affecting any other IER bits.

Writing a 1 to any bit location(s) in the CIE clear(s) the corresponding bit(s) in the IER without affecting any other IER bits.

Timing Diagrams

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

The timing diagram in Figure 16 shows a configuration with the input interrupt (`Intr`) configured with rising edge sensitivity, the output interrupt request (`Irq`) to level sensitive (active-High), and the fast interrupt logic disabled.

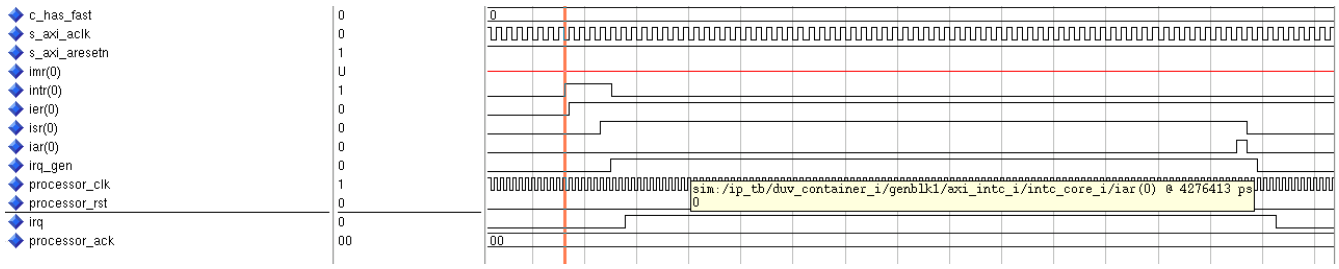


Figure 16: **Input - Rising Edge Sensitive, Output - High Level Sensitive, Fast Interrupt Logic Disabled**

The timing diagram in Figure 17 shows a configuration with the input interrupt (`Intr`) configured with rising edge sensitivity, the output interrupt request (`Irq`) to rising edge sensitivity, and the fast interrupt logic disabled.

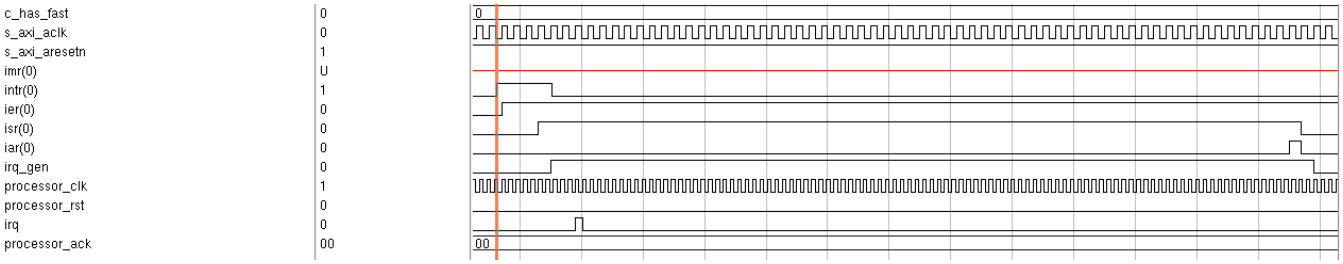


Figure 17: **Input - Rising Edge Sensitive, Output - Rising Edge Sensitive, Fast Interrupt Logic Disabled**

The timing diagram in Figure 18 shows a configuration with the input interrupt (`Intr`) configured with level sensitivity (active-High), the output interrupt request (`Irq`) with level sensitivity (active-High), and the fast interrupt logic disabled.

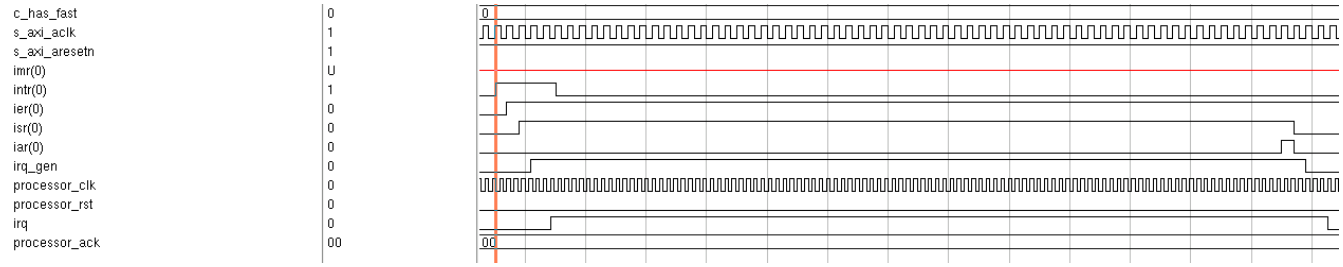


Figure 18: **Input - High Level Sensitive, Output - High Level Sensitive, Fast Interrupt Logic Disabled**

The timing diagram in Figure 19 shows a configuration with the input interrupt (Intr) configured with level sensitivity (active-High), the output interrupt request (Irq) with rising edge sensitivity, and the fast interrupt logic disabled.

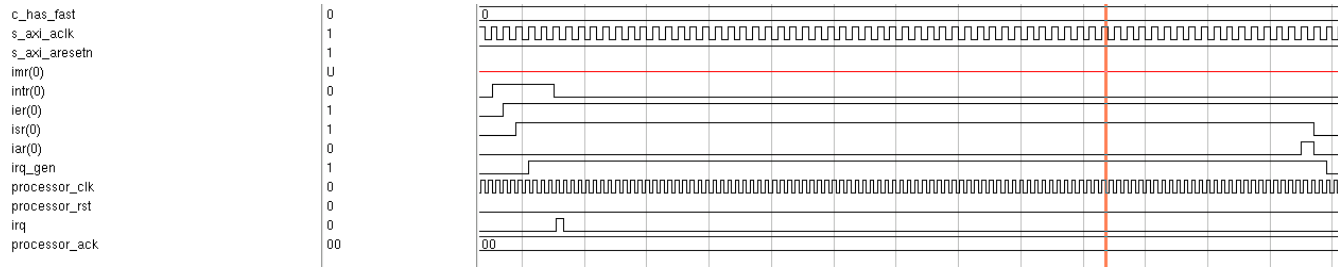


Figure 19: **Input - High Level Sensitive, Output - Rising Edge Sensitive, Fast Interrupt Logic Disabled**

The timing diagram in Figure 20 shows a configuration with the input interrupt (Intr) configured with rising edge sensitivity, and the fast interrupt logic enabled and the mode set to fast interrupt mode (IMR(i) = 1).

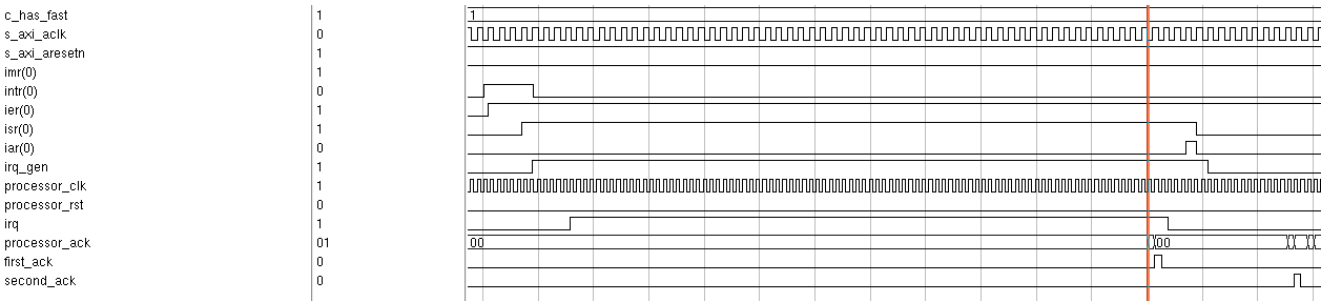


Figure 20: **Input - Rising Edge Sensitive, Fast Interrupt Logic Enabled and IMR(i) = 1**

The timing diagram in Figure 21 shows a configuration with the input interrupt (Intr) configured with level sensitivity (active-High), the fast interrupt logic enabled and the mode set to fast interrupt mode (IMR(i) = 1).

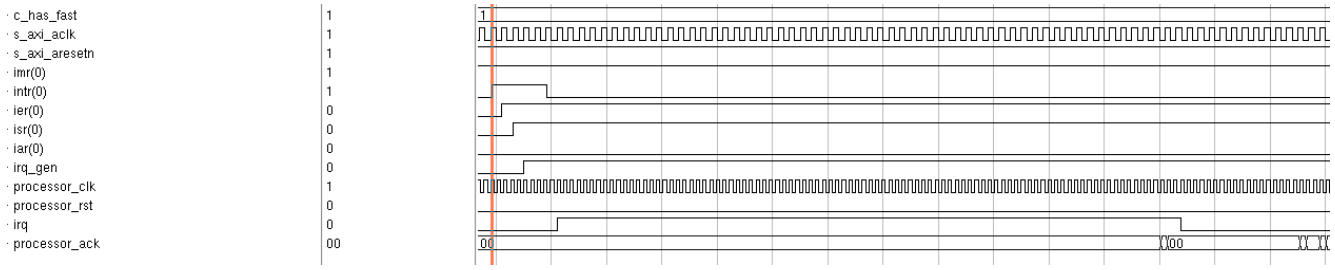


Figure 21: **Input - High Level Sensitive, Fast Interrupt Logic Enabled and IMR(i) = 1**

Design Implementation

Target Technology

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

Device Utilization and Performance Benchmarks

Core Performance

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 the AXI INTC core is combined with other designs in the system, the utilization of FPGA resources and timing of the AXI INTC design will vary from the results reported here.

The AXI INTC core resource utilization for various parameter combinations measured with a Virtex-6 device as the target is detailed in Table 17.

Table 17: **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. These numbers are reported by tools for clock period constraint of 5 ns.

The AXI INTC core resource utilization for various parameter combinations measured with a Spartan-6 device as the target is detailed in Table 18.

Table 18: **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
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. These numbers are reported by tools for clock period of constraint of 6.25 ns.

The AXI INTC core resource utilization for various parameter combinations measured with a Kintex-7 device as the target is detailed in Table 19.

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

Table 19: **Performance and Resource Utilization on Kintex-7 (XC7K410T-3-FFG676) and Zynq-7000**

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
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. These numbers are reported by tools for clock period of constraint of 5 ns.

System Performance

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 Interrupt Controller 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 FPGA resources and timing will vary from the results reported here.

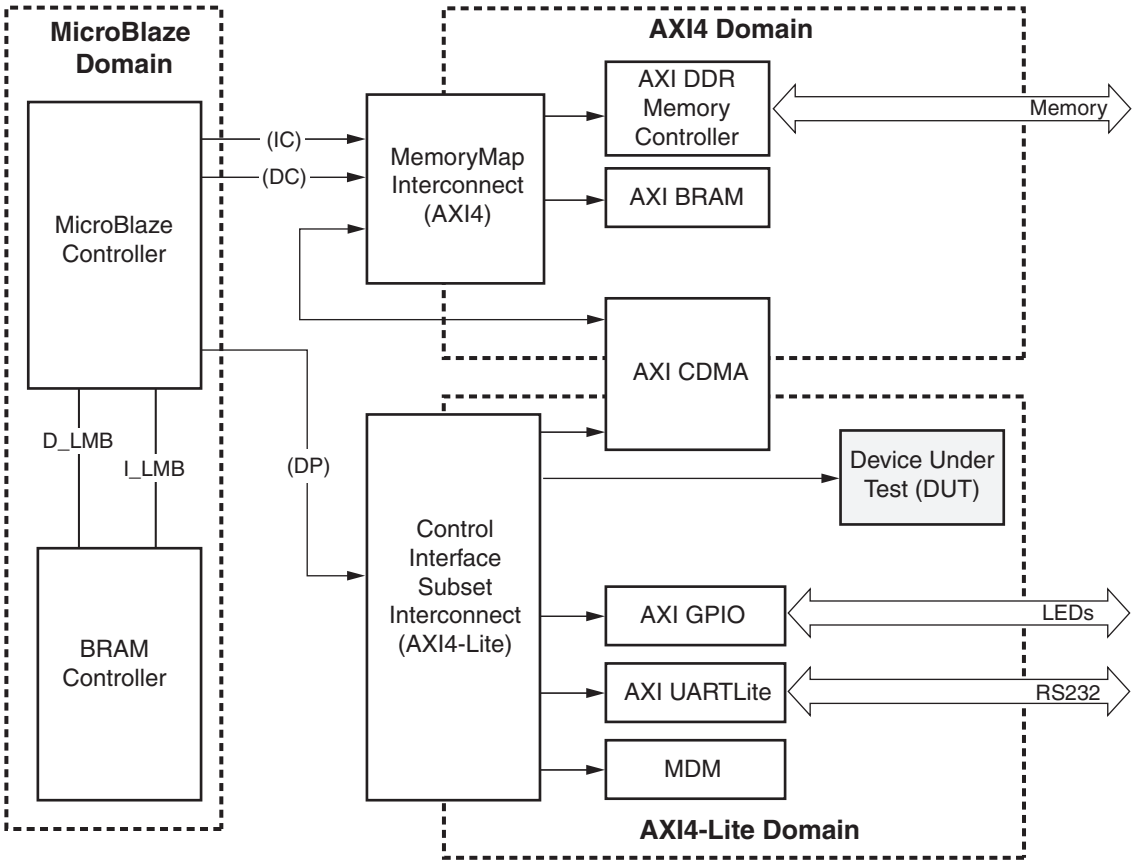


Figure 22: **Virtex-6 and Spartan-6 Devices F_{MAX} Margin System**

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 20.

Table 20: **AXI INTC Core System Performance**

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

Support

Xilinx provides technical support for this LogiCORE IP product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support of product if implemented in devices that are not defined in the documentation, if customized beyond that allowed in the product documentation, or if changes are made to any section of the design labeled *DO NOT MODIFY*.

Ordering Information

This Xilinx LogiCORE IP module is provided at no additional cost with the Xilinx ISE Design Suite Embedded Edition software 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 modules and software, contact your local Xilinx sales representative.

Revision History

The following table shows the revision history for this document:

Date	Version	Description of Revisions
09/21/2010	1.0	Initial Xilinx release.
12/14/2010	2.0	Updated for core v1.01a and ISE Design Suite v12.4.
06/22/2011	3.0	Updated for ISE Design Suite v13.2. Added support for Artix-7, Virtex-7, and Kintex-7 devices.
04/24/2012	4.0	Updated for core v1.02a and ISE Design Suite v14.1. Added Fast Interrupt Mode Control and new registers Interrupt Mode Register (IMR) and Interrupt Vector Address Register (IVAR). C_HAS_FAST parameter added.
07/25/2012	5.0	Added support for Vivado Design Suite. Added support for Zynq-7000 devices. Removed BASEADDR and HIGHADDR parameters.
10/16/2012	6.0	Updated for Vivado Design Suite 2012.3 and ISE Design Suite 14.3. Updated reset vector values.
06/19/2013	7.0	Updated for ISE Design Suite 14.6. Added description of Cascade Mode Interrupt and Software Interrupt functionality.

Notice of Disclaimer