LogiCORE IP AXI INTC (v1.00a)

DS747 September 21, 2010 **Product Specification**

Introduction

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® protocol's AXI (Advanced Micro controller Bus Architecture 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 is based on the AXI4-Lite specification
- Configurable number of (up to 32) interrupt inputs
- Single interrupt output
- Easily cascaded to provide additional interrupt inputs
- 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 interrupt request output
- Each input is configurable for edge or level sensitivity:
  - Edge sensitivity can be configured for rising or falling
  - Level sensitivity can be active High or active Low
- Automatic edge synchronization when inputs are configured for edge sensitivity
- Output interrupt request pin is configurable for edge or level generation:
  - Edge generation configurable for rising or falling
  - Level generation configurable for active High or active Low

LogiCORE IP Facts Table					
Core Specifics					
Supported Device Family(1)	Spartan®-6(2) Virtex®-6(3)				
Supported User Interfaces	AXI4-Lite				
	Resources				**Frequency**
Configuration	**LUTs**	**FFs**	**DSP Slices**	**Block RAMs(4)**	**Max. Freq.(5)**
Config1	Refer to Table 13 and Table 14				
Provided with Core					
Documentation	Product Specification				
Design Files	VHDL				
Example Design	Not Provided				
Test Bench	Not Provided				
Constraints File	None				
Simulation Model	None				
Tested Design Tools					
Design Entry Tools	ISE® 12.3 software				
Simulation	ModelSim SE/EE 6.5c				
Synthesis Tools	XST 12.3 software				
Support					
Provided by Xilinx, Inc.					

Notes:

1. For a complete listing of supported devices, see the release notes for this core.
2. For more information on the Spartan-6 devices, see the *Spartan-6 Family Overview* [Ref 3]
3. For more information on the Virtex-6 devices, see the *Virtex-6 Family Overview* [Ref 4]
4. Based on 18K block RAMs (or 36K - select appropriate size).
5. Performance numbers listed are for Virtex-6 FPGAs. For more complete performance data, see Device Utilization and Performance Benchmarks, page 24.

Functional Description

Interrupt Controller Overview

AXI INTC 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 AXI INTC is intended to be connected to a device that can generate interrupt conditions.

Capturing, Acknowledging and Enabling Interrupt Conditions

Interrupt conditions are captured by AXI INTC and retained until explicitly acknowledged. Interrupts can be enabled/disabled either globally or individually.

When all interrupts are globally enabled and if at least one captured interrupt is individually enabled then processor is signalled with an interrupt condition.

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 captures a new interrupt condition any time 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 use rising edges for detecting an active edge. In all the three schemes, the peripheral device generating the interrupt input provides an active edge and at some time later it produces an inactive edge to detect another active edge for generating a new interrupt request.

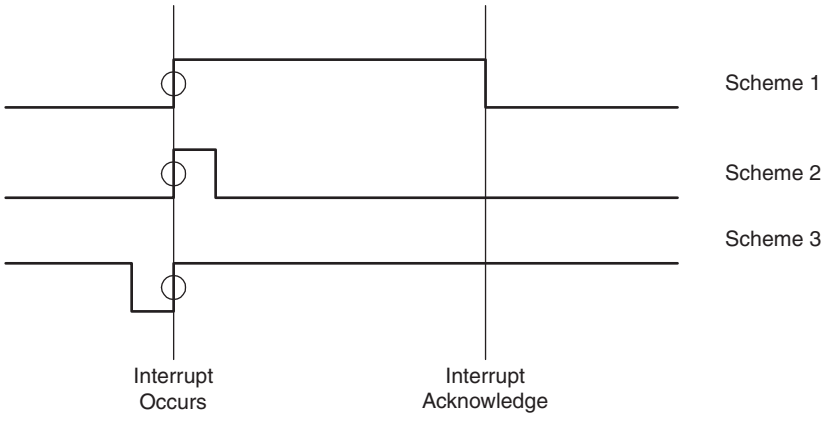

Figure 1: **Schemes for Generating Edges**

Level-sensitive capture, causes an interrupt condition to be captured any time the input is at the active level and the interrupt condition does not already exits. (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 via an inactive edge and subsequent new active edge to generate an additional interrupt condition

Interrupt Vector Register

If an optional Interrupt Vector Register is opted 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).

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: connect 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

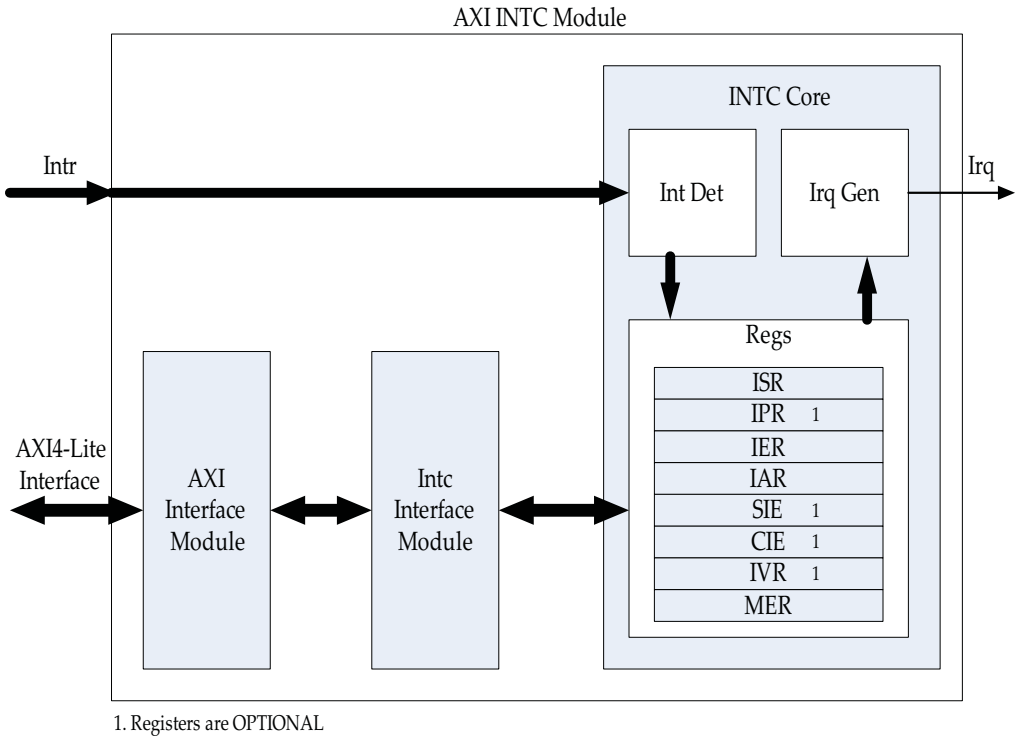

Figure 2: **AXI INTC Module Block Diagram**

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 10
- Interrupt Pending Register (IPR), page 11
- Interrupt Enable Register (IER), page 12
- Interrupt Acknowledge Register (IAR), page 13
- Set Interrupt Enables (SIE), page 14
- Clear Interrupt Enables (CIE), page 15
- Interrupt Vector Register (IVR), page 16
- Master Enable Register (MER), page 17

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.

I/O Signals

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

Table 1: **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

Table 1: **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 will remain 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

Notes:
1. Intr(0) is always the highest priority interrupt and each successive bit to the left has a corresponding lower interrupt priority.

Design Parameters

To allow the user to obtain a AXI INTC that is uniquely tailored for the system, certain features can be parameterized in the AXI INTC design. This allows the user 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.

Table 2: **Design Parameters**

Generic	Feature/Description	Parameter Name	Allowable Values	Default Value	VHDL Type
System Parameter					
G1	Target FPGA family	C_FAMILY	Spartan-6 and Virtex-6		string
AXI Parameters					
G2	AXI Base Address	C_BASEADDR	Valid Address(1)(4)	0xffffffff (3)	std_logic_vector

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

Generic	Feature/Description	Parameter Name	Allowable Values	Default Value	VHDL Type
G3	AXI High Address	C_HIGHADDR	Valid Address(2)(4)	0x00000000(3)	std_logic_vector
G4	AXI address bus width	C_S_AXI_ADDR_WIDTH	32	32	integer
G5	AXI data bus width	C_S_AXI_DATA_WIDTH	32	32	integer
INTC Parameters					
G6	Number of interrupt inputs	C_NUM_INTR_INPUTS	1-32	2	integer
G7	Type of interrupt for each input 1 = Edge 0 = Level	C_KIND_OF_INTR	See(5)	ALL 1s	std_logic_vector
G8	Type of each edge sensitive input 1 = Rising 0 = Falling Valid if C_KIND_OF_INTR = 1s	C_KIND_OF_EDGE	See(5)	ALL 1s	std_logic_vector
G9	Type of each level sensitive input 1= High 0= Low Valid if C_KIND_OF_INTR = 0s	C_KIND_OF_LVL	See(5)	ALL 1s	std_logic_vector
G10	Indicates the presence of IPR	C_HAS_IPR	0 = Not Present 1 = Present	1	integer
G11	Indicates the presence of SIE	C_HAS_SIE	0 = Not Present 1 = Present	1	integer
G12	Indicates the presence of CIE	C_HAS_CIE	0 = Not Present 1 = Present	1	integer
G13	Indicates the presence of IVR	C_HAS_IVR	0 = Not Present 1 = Present	1	integer
G14	Indicates level or edge active Irq	C_IRQ_IS_LEVEL	0 = Active Edge 1 = Active Level	1	integer
G15	indicates the sense of the Irq output	C_IRQ_ACTIVE	0 = Falling / Low 1 = Rising / High	1	std_logic

Notes:

1. The user must set the values. The C_BASEADDR must be a multiple of the range, where the range is C_HIGHADDR - C_BASEADDR + 1.
2. C_HIGHADDR - C_BASEADDR must be a power of 2 greater than equal to C_BASEADDR + 0xFFF.
3. An invalid default value will be specified to insure that the actual value is set, i.e., if the value is not set, a compiler error will be generated.
4. If the size and alignment rules are met then C_S_AXI_HIGHADR = C_BASEADDR + 2**P-1 for some p >=5; the low-order p bits of C_BASEADDR are '0'; the low-order p bits of C_HIGHADDR are '1'; and the remaining corresponding bit positions are equal. As the value of p increases the resources and time needed to decode the address range decrease and the amount of the overall system address map consumed by AXI INTC increases. For example:

   C_BASEADDR = 0x70800000

   C_HIGHADDR = 0x7080001F

   provides the maximum address decode resolution, requiring the upper 27 address bits to be decoded. Conversely,

   C_BASEADDR = 0x70000000

   C_HIGHADDRC_HIGHADDR = 0x7FFFFFFF

   will reduce the address decoding logic for an AXI INTC (only the 4 upper address bits), resulting in a smaller and faster address decode, which consumes one sixteenth of the system address map.
5. The interrupt input is a little-endian vector having the width same as the data bus and contain either a '0' or '1' in each position.

Allowable Parameter Combinations

The address range specified by C_BASEADDR and C_HIGHADDR must be a power of 2, and must be at least 0xFFF. For example, if C_BASEADDR = 0xE0000000, C_HIGHADDR must be at least = 0xE0000FFF.

Dependencies between Parameters and I/O Signals

The dependencies between the AXI INTC core design parameters and I/O signals are described in Table 3. In addition, when certain features are parameterized out of the design, the related logic will no longer be a part of the design. The unused input signals and related output signals are set to a specified value.

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

Generic or Port	Name	Affects	Depends	Relationship Description
Design Parameters				
G4	C_S_AXI_ADDR_WIDTH	P3, P13	-	Defines the width of the ports
G5	C_S_AXI_DATA_WIDTH	P6, P7, P16	-	Defines the width of the ports
I/O Signals				
P3	S_AXI_AWADDR[C_S_AXI_ADDR_WIDTH-1:0]	-	G4	Port width depends on the generic C_S_AXI_ADDR_WIDTH
P6	S_AXI_WDATA[C_S_AXI_DATA_WIDTH-1:0]	-	G5	Port width depends on the generic C_S_AXI_DATA_WIDTH
P7	S_AXI_WSTB[C_S_AXI_DATA_WIDTH/8-1:0]	-	G5	Port width depends on the generic C_S_AXI_DATA_WIDTH
P13	S_AXI_ARADDR[C_S_AXI_ADDR_WIDTH -1:0]	-	G4	Port width depends on the generic C_S_AXI_ADDR_WIDTH
P16	S_AXI_RDATA[C_S_AXI_DATA_WIDTH -1:0]	-	G5	Port width depends on the generic C_S_AXI_DATA_WIDTH

Register Descriptions

All AXI INTC registers are accessed through the AXI interface. The base address for these registers is provided by the configuration parameter, C_BASEADDR. Each register is 32 bits although some bits may be unused and is accessed on a 4-byte boundary offset from the base address.

Because AXI addresses are byte addresses, AXI INTC register offsets are located at integral multiples of four from the base address. Table 4 illustrates the registers and their offsets from the base address. The AXI INTC registers are read as little-endian data.

The eight registers visible to the programmer are shown in Table 4 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 4: **Registers and Base Address Offsets**(1)

Base Address + Offset (hex)	Register Name	Access Type	Default Value (hex)	Description
C_S_AXI_BASEDADR + 0x0	ISR	Read / Write	0x0	Interrupt Status Register
C_S_AXI_BASEDADR + 0x4	IPR	Read	0x0	Interrupt Pending Register
C_S_AXI_BASEDADR + 0x8	IER	Read / Write	0x0	Interrupt Enable Register
C_S_AXI_BASEDADR + 0xC	IAR	Write	0x0	Interrupt Acknowledge Register
C_S_AXI_BASEDADR + 0x10	SIE	Write	0x0	Set Interrupt Enable Register
C_S_AXI_BASEDADR + 0x14	CIE	Write	0x0	Clear Interrupt Enable Register
C_S_AXI_BASEDADR + 0x18	IVR	Read	0x0	Interrupt Vector Register
C_S_AXI_BASEDADR + 0x1C	MER	Read / Write	0x0	Master Enable Register

Notes:

1. If the number of interrupt inputs is less than the data bus width the inputs will 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.

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 12 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' will 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. Once 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 will return '0'. The Interrupt Status Register (ISR) is shown in Figure 3 and the bits are described in Table 5.

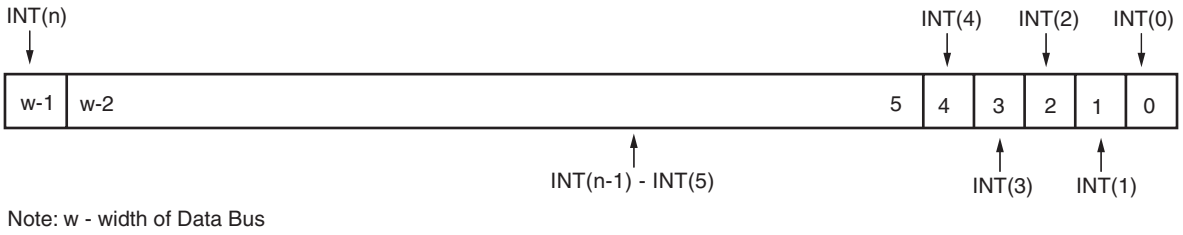

Figure 3: **Interrupt Status Register (ISR)**

Table 5: **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 will return zero. The Interrupt Pending Register (IPR) is shown in Figure 4 and the bits are described in Table 6.

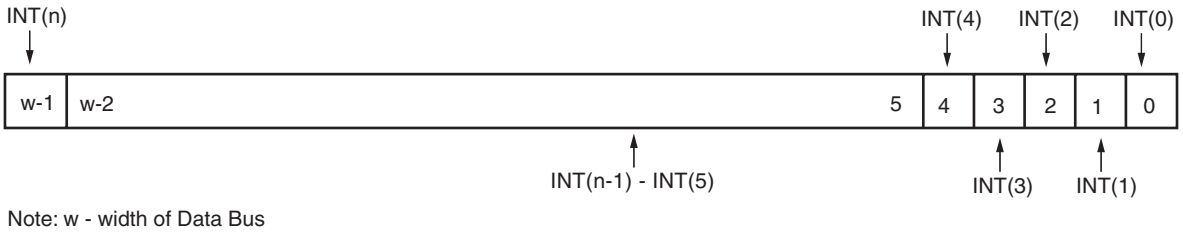

Figure 4: **Interrupt Pending Register**

Table 6: **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, will disable the generation of interrupt output for corresponding interrupt input signal. Note however, that disabling an interrupt input is not the same as clearing it. Disabling an active interrupt blocks that interrupt from reaching the Irq output, but as soon as it is re-enabled the interrupt will immediately generate 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 will return '0'. The Interrupt Enable Register (IER) is shown in Figure 5 and the bits are described in Table 7.

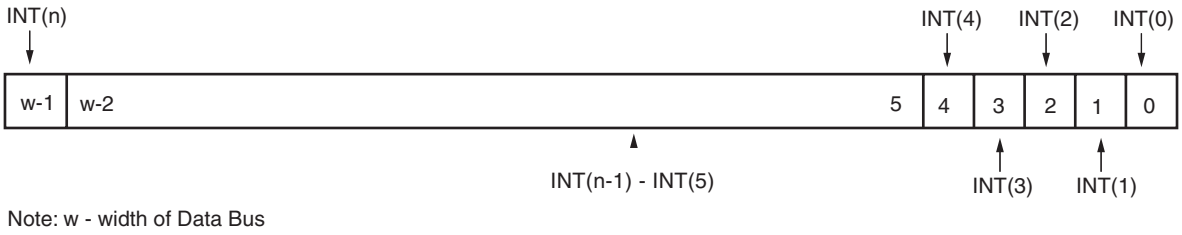

Figure 5: **Interrupt Enable Register**

Table 7: **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. Note that writing one to a bit in IAR clears the corresponding bit in ISR and also clears the same bit itself in IAR.

Writing a '1' to a bit location in the IAR will clear 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 will remain active until cleared by acknowledging it. Unmasking an active interrupt will cause 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 exits. The Interrupt Acknowledge Register (IAR) is shown in Figure 6 and the bits are described in Table 8.

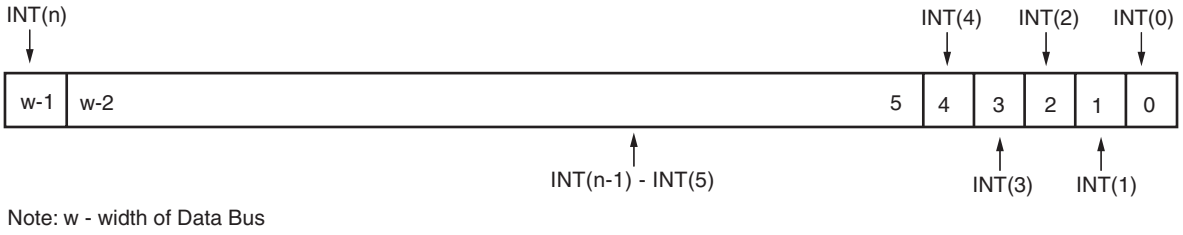

Figure 6: **Interrupt Acknowledge Register**

Table 8: **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 IER bits in a single atomic operation, rather than using a read / modify / write sequence.

Writing a '1' to a bit location in SIE will set 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 and can be parametrized out by setting C_HAS_SIE = 0.

The Set Interrupt Enable (SIE) register is shown in Figure 7 and the bits are described in Table 9.

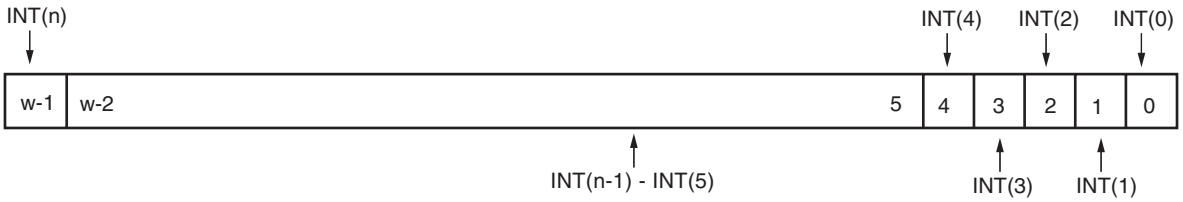

Figure 7: **Set Interrupt Enable (SIE) Register**

Table 9: **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 will clear 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 and can be parameterized out by setting C_HAS_CIE = 0.

The Clear Interrupt Enables (CIE) register is shown in Figure 8 and the bits are described in Table 10.

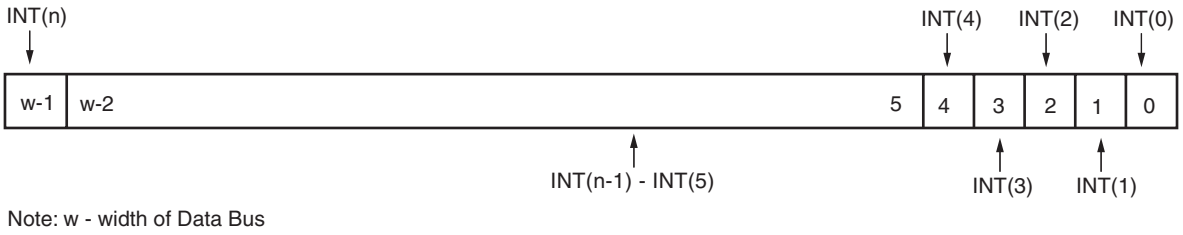

Figure 8: **Clear Interrupt Enable (CIE) Register**

Table 10: **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 will contain all 1s. The IVR is optional in the AXI INTC and can be parameterized out by setting C_HAS_IVR = 0. The Interrupt Vector Register (IVR) is shown in Figure 9 and described in Table 11

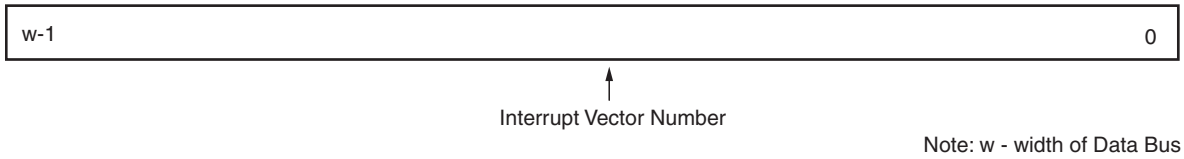

Figure 9: **Interrupt Vector Register (IVR)**

Table 11: **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 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 will reflect the state of the ME and HIE bits.

All other bits will read as 0s. The Master Enable Register (MER) is shown in Figure 10 and is described in Table 12.

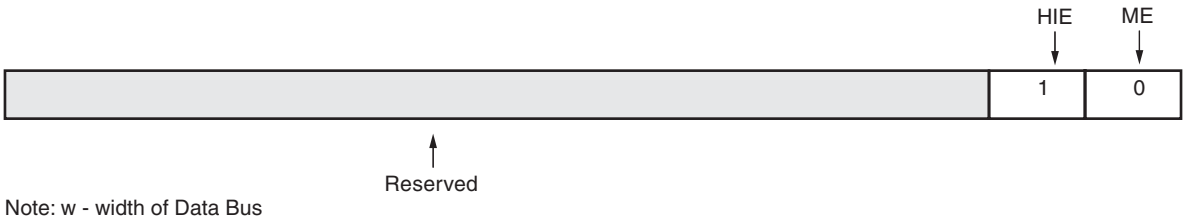

Figure 10: **Master Enable Register (MER)**

Table 12: **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

Software Considerations

This section provides an overview of software initialization and communication with an AXI INTC. The number of interrupt inputs that a AXI INTC has is set by the C_NUM_INTR_INPUTS generic described in Table 2. The first input is always INT0 and is mapped to each LSB of each register (except for 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 3).

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

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 is put into a state where all interrupt inputs and the interrupt request output are disabled. In order for the AXI INTC 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 AXI INTC 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. 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 optional IVR provides the ordinal value of the highest priority interrupt that is enabled and active.

For example, if the IVR is present, and a valid interrupt signal has occurred on the INT3 interrupt input and nothing is active on INT2, INT1 and INT0, reading the IVR will provide 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 will remain active but blocked until it is unmasked or cleared.

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 will 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. 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 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.

1. Configured Input Interrupt (Intr) for Level sensitive (active High) and Output Interrupt Request (Irq) to Edge sensitive (rising) and is shown in Figure 11.

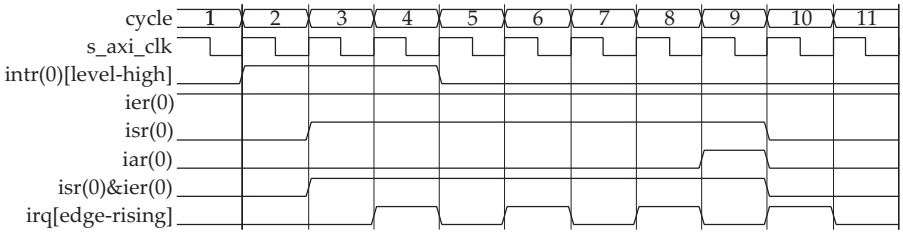

Figure 11: **Input Intr - Level Sensitive (Active High) and Output Irq - Edge Sensitive (Rising**

2. Configured Input Interrupt (Intr) for Level sensitive (active High) and Output Interrupt Request (Irq) to Edge sensitive (falling) and is shown in Figure 12.

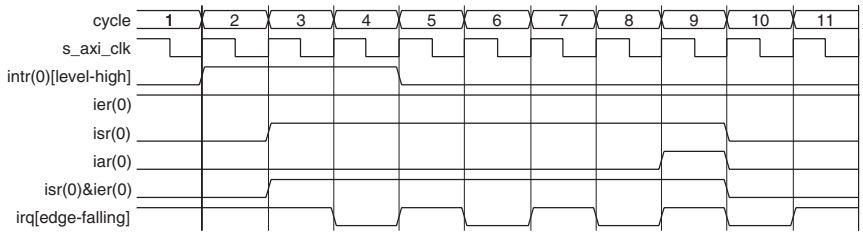

Figure 12: **Input Intr - Level Sensitive (Active High) and Output Irq - Edge Sensitive (Falling)**

3. Configured Input Interrupt (Intr) for Level sensitive (active High) and Output Interrupt Request (Irq) to Level sensitive (active High) and is shown in Figure 13.

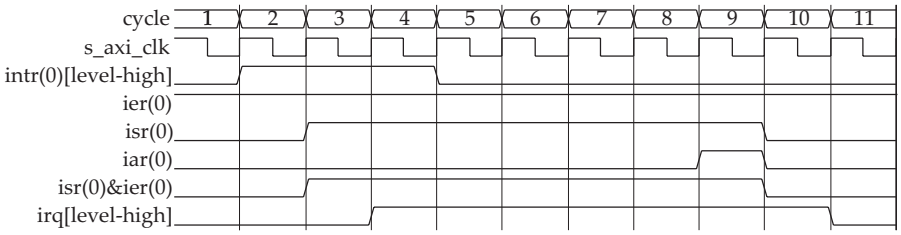

Figure 13: **Input Intr - Level Sensitive (Active High) and Output Irq - Level Sensitive (Active High)**

4. Configured Input Interrupt (Intr) for Level sensitive (active High) and Output Interrupt Request (Irq) to Level sensitive (active Low) and is shown in Figure 14.

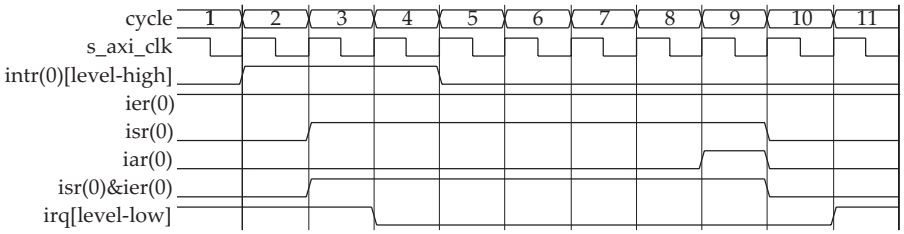

Figure 14: **Input Intr - Level Sensitive (Active High) and Output Irq - Level Sensitive (Active Low)**

5. Configured Input Interrupt (Intr) for Level sensitive (active Low) and Output Interrupt Request (Irq) to Edge sensitive (rising) and is shown in Figure 15.

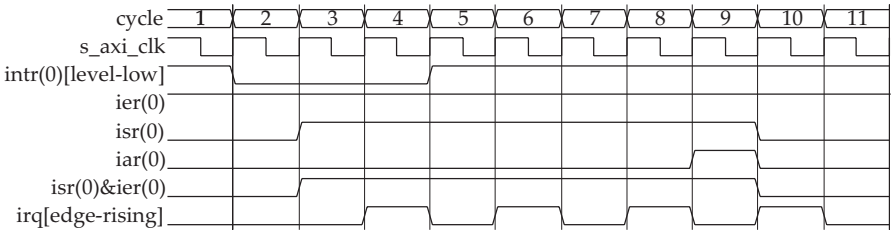

Figure 15: **Input Intr - Level Sensitive (Active Low) and Output Irq - Edge Sensitive (Rising)**

6. Configured Input Interrupt (Intr) for Level sensitive (active Low) and Output Interrupt Request (Irq) to Edge sensitive (falling) and is shown in Figure 16

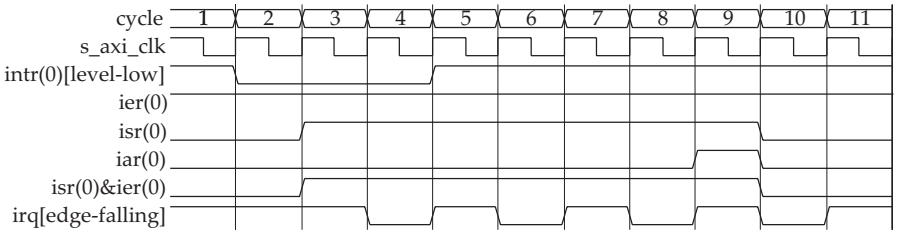

Figure 16: **Input Intr - Level Sensitive (Active Low) and Output Irq - Edge Sensitive (Falling)**

7. Configured Input Interrupt (Intr) for Level sensitive (active Low) and Output Interrupt Request (Irq) to Level sensitive (active High) and is shown in Figure 17.

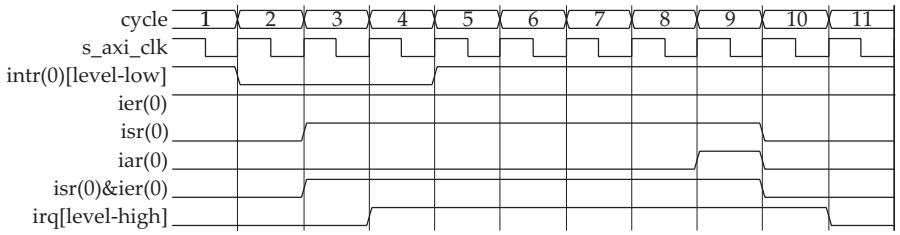

Figure 17: **Input Intr - Level Sensitive (Active Low) and Output Irq - Level Sensitive (Active High)**

8. Configured Input Interrupt (Intr) for Level sensitive (active Low) and Output Interrupt Request (Irq) to Level sensitive (active Low) and is shown in Figure 18.

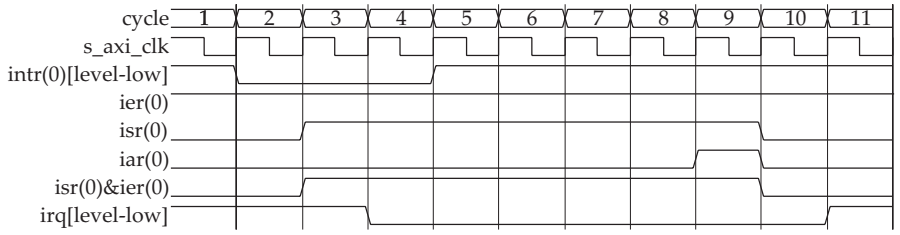

Figure 18: **Input Intr - Level Sensitive (Active Low) and Output Irq - Level Sensitive (Active Low)**

9. Configured Input Interrupt (Intr) for Edge sensitive (rising) and Output Interrupt Request (Irq) to Edge sensitive (rising) and is shown in Figure 19.

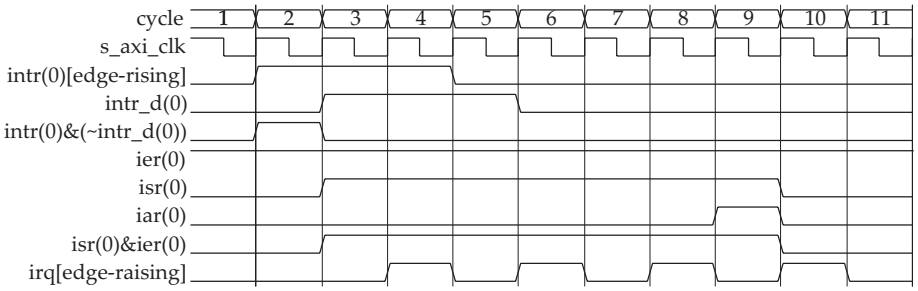

Figure 19: **Input Intr - Edge Sensitive (Rising) and Output Irq - Edge Sensitive (Rising)**

10. Configured Input Interrupt (Intr) for Edge sensitive (rising) and Output Interrupt Request (Irq) to Edge sensitive (falling) and is shown in Figure 20.

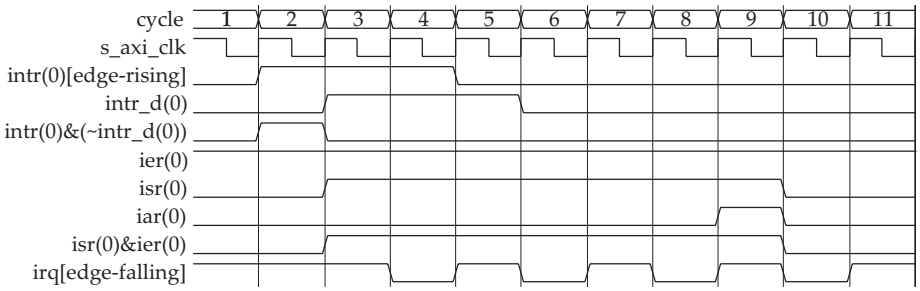

Figure 20: **Input Intr - Edge Sensitive (Rising) and Output Irq - Edge Sensitive (Falling)**

11. Configured Input Interrupt (Intr) for Edge sensitive (rising) and Output Interrupt Request (Irq) to Level sensitive (active High) and is shown in Figure 21.

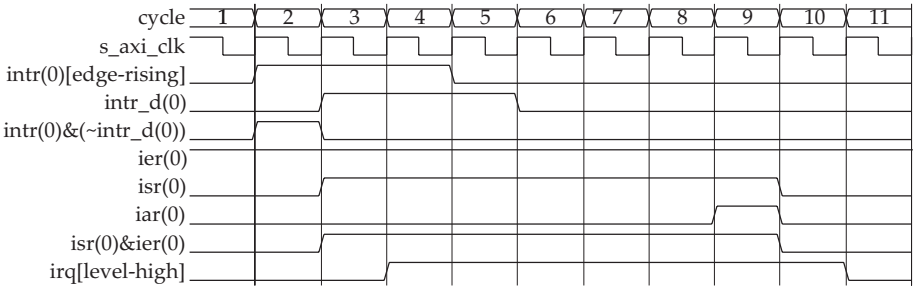

Figure 21: **Input Intr - Edge Sensitive (Rising) and Output Irq - Level Sensitive (Active High)**

12. Configured Input Interrupt (Intr) for Edge sensitive (rising) and Output Interrupt Request (Irq) to Level sensitive (active Low) and is shown in Figure 22.

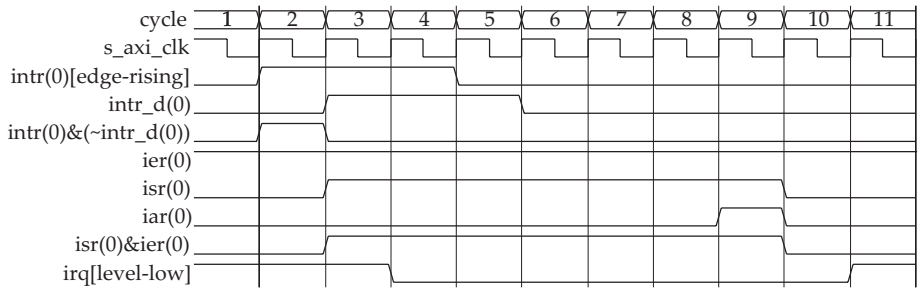

Figure 22: **Input Intr - Edge Sensitive (Rising) and Output Irq - Level Sensitive (Active Low)**

13. Configured Input Interrupt (Intr) for Edge sensitive (falling) and Output Interrupt Request (Irq) to Edge sensitive (rising) and is shown in Figure 23.

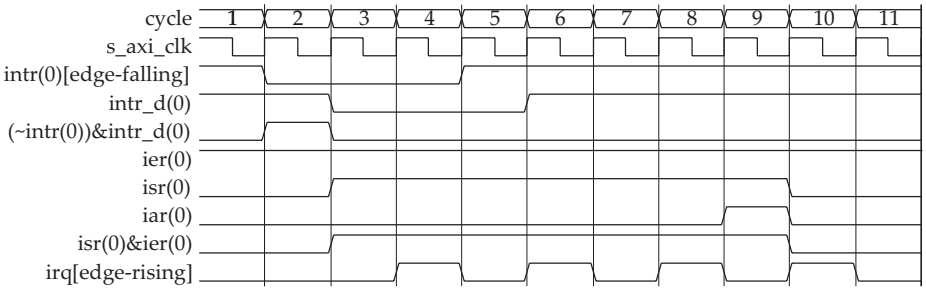

Figure 23: **Input Intr - Edge Sensitive (Falling) and Output Irq - Edge Sensitive (Rising)**

14. Configured Input Interrupt (Intr) for Edge sensitive (falling) and Output Interrupt Request (Irq) to Edge sensitive (falling) and is shown in Figure 24.

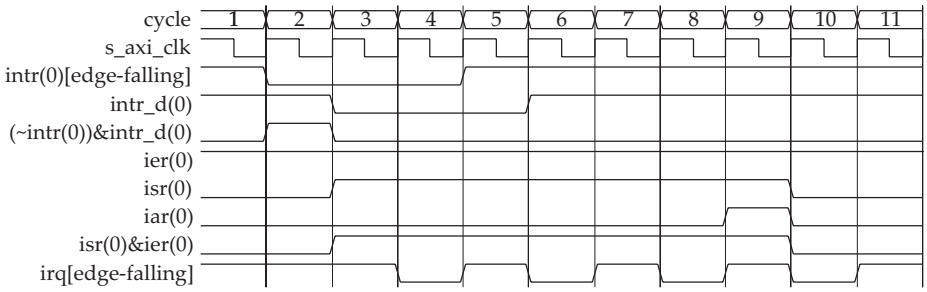

Figure 24: **Input Intr - Edge Sensitive (Falling) and Output Irq - Edge Sensitive (Falling)**

15. Configured Input Interrupt (Intr) for Edge sensitive (falling) and Output Interrupt Request (Irq) to Level sensitive (active High) and is shown in Figure 25.

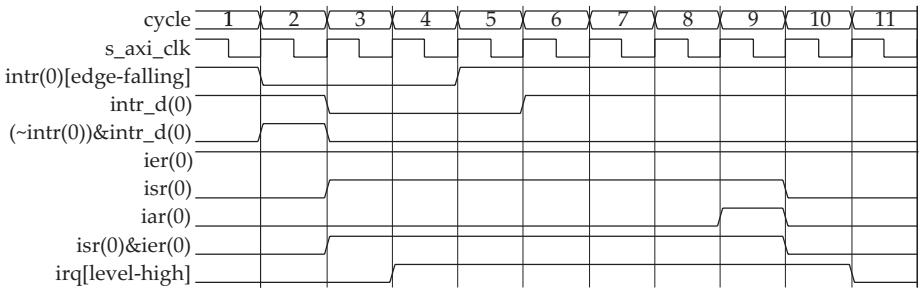

Figure 25: **Input Intr - Edge Sensitive (Falling) and Output Irq - Level Sensitive (Active High)**

16. Configured Input Interrupt (Intr) for Edge sensitive (falling) and Output Interrupt Request (Irq) to Level sensitive (active Low) and is shown in Figure 26.

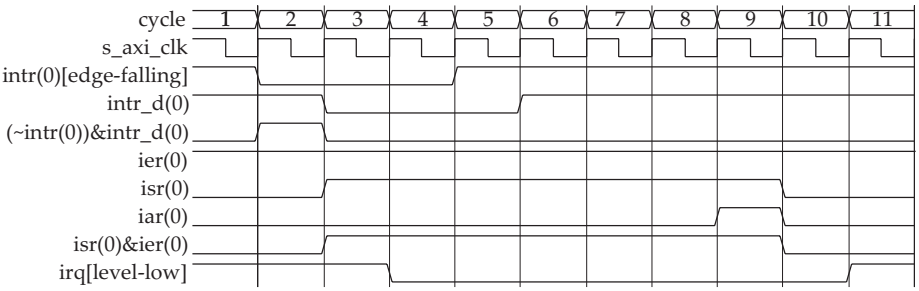

Figure 26: **Input Intr - Edge Sensitive (Falling) and Output Irq - Level Sensitive (Active Low)**

Design Implementation

Target Technology

The intended target technology is Spartan-6 and Virtex-6 family FPGAs.

Device Utilization and Performance Benchmarks

Core Performance

Because the AXI INTC core will be 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 resource utilization for various parameter combinations measured with a Virtex-6 device as the target is detailed in Table 13.

Table 13: **Performance and Resource Utilization Benchmarks on Virtex-6 (XC6VLX75T-1-FF784)**

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)
1	0	0	0	0	25	58	73	160
1	1	0	0	0	27	59	73	160
1	0	1	0	0	27	59	75	160
1	0	0	1	0	28	59	74	160
1	0	0	0	1	27	59	74	160
1	1	1	1	1	30	62	78	160
2	1	1	1	1	33	74	88	160
2	0	0	0	0	27	64	81	160
4	1	1	1	1	44	97	110	160
4	0	0	0	0	35	78	90	160
8	1	1	1	1	58	141	159	160
8	0	0	0	0	55	105	118	160
16	1	1	1	1	103	230	252	160
16	0	0	0	0	73	161	161	160
32	1	1	1	1	175	406	418	160

Notes:

1. The above numbers are reported by tools for clock period of constraint of 6.6 ns.

The AXI INTC resource utilization for various parameter combinations measured with a Spartan-6 device as the target is detailed in Table 14.

Table 14: **Performance and Resource Utilization Benchmarks on Spartan-6 (XC6SLX45T-2-FGG484)**

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)
1	0	0	0	0	36	60	72	100
1	1	0	0	0	31	60	72	100
1	0	1	0	0	38	61	74	100
1	0	0	1	0	34	61	73	100
1	0	0	0	1	31	60	73	100
1	1	1	1	1	36	63	77	100
2	1	1	1	1	42	75	87	100
2	0	0	0	0	33	67	81	100
4	1	1	1	1	48	98	110	100
4	0	0	0	0	43	81	92	100
8	1	1	1	1	66	142	154	100
8	0	0	0	0	53	106	116	100
16	1	1	1	1	108	231	243	100
16	0	0	0	0	80	162	161	100
32	1	1	1	1	173	406	407	100

Notes:

1. The above numbers are reported by tools for clock period of constraint of 100 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 will 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 design's FPGA resources and timing usage will vary from the results reported here.

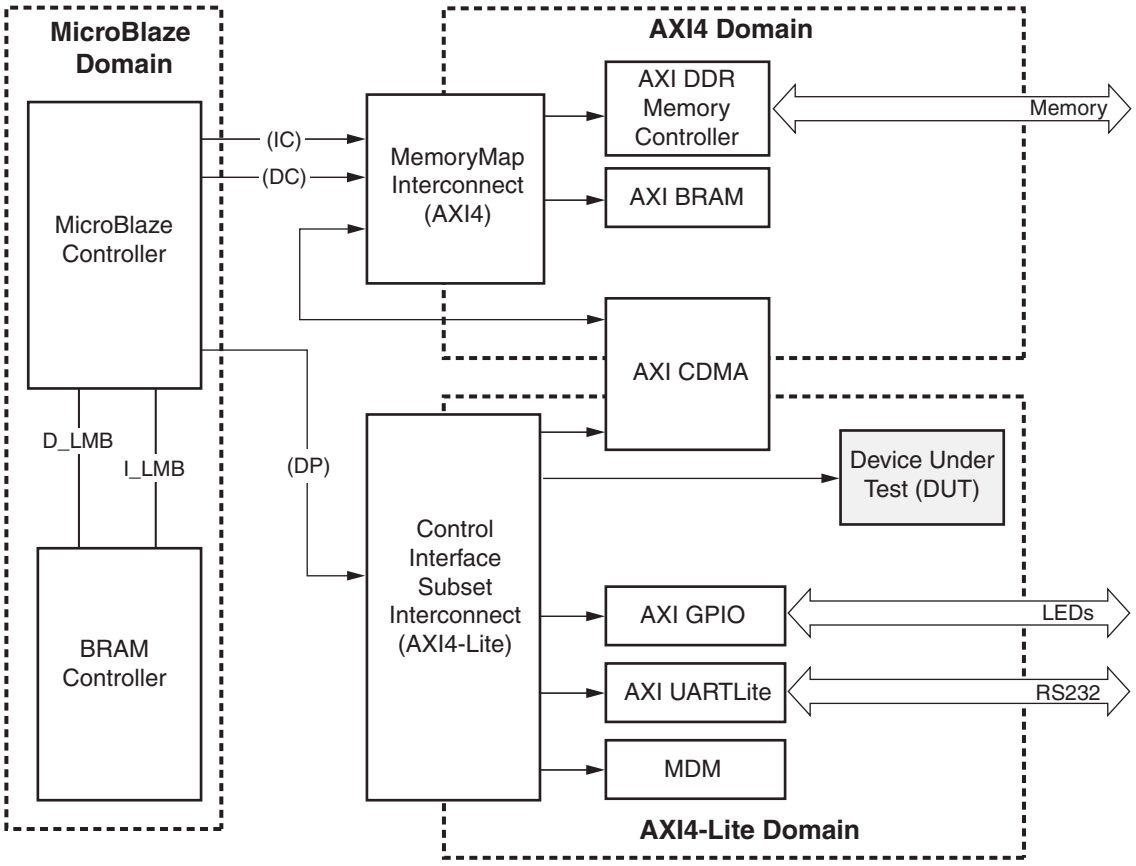

Figure 27: **Virtex-6 and Spartan-6 Devices F_{MAX} Margin System**

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 15.

Table 15: **AXI INTC System Performance**

Target FPGA	Target F_{MAX} (MHz)
Spartan-6	110
Virtex-6	150

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. The core is generated using the Xilinx ISE Embedded Edition software (EDK).

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, please contact your local Xilinx sales representative.

Reference Documents

The following documents contain reference information important to understand AXI INTC design

1. *AXI4 AMBA® AXI Protocol Version: 2.0 Specification*
2. DS765: *LogiCORE IP AXI Lite IPIF (v1.00a) Data Sheet*
3. DS160: *Spartan-6 Family Overview*
4. DS150: *Virtex-6 Family Overview*

Revision History

The following table shows the revision history for this document:

Date	Version	Description of Revisions
09/21/10	1.0	Initial Xilinx release.

Notice of Disclaimer