LogiCORE IP Interrupt Control v3.0

Product Guide for Vivado Design Suite

PG166 October 2, 2013

XILINX®

XILINX

Table of Contents

Introduction

The Xilinx LogiCORE™ IP Interrupt Control core provides interrupt capture support for internal IP Interface (IPIF) sub-block, as well as support for the connected IP function. This core is a helper core, which provides the facility to enable, store and generate a local interrupt mechanism to other cores that instantiate this core.

Features

- Parameterized number of interrupts needed by the IP
- Provides both Interrupt Status Register (ISR) and Interrupt Enable Register (IER) functions for the user IP.
- Inclusion/Omission of Priority Encoder
- Inclusion/Omission of Device ISC
- Parameterized number of local IPIF generated interrupt sources.
- Provides both Interrupt Status Register (ISR) and Interrupt Enable Register (IER) functions for the Device level interrupts.
- Supports 32, 64, and 128-bit IPIF data bus widths
- Global Enable/Disable for final interrupt output to the System Interrupt Controller
- Parameterized user IP interrupt capture mode
  - Pass Through (non-inverting)
  - Pass Through (inverting)
  - Registered Level (non-inverting)
  - Registered Level (inverting)
  - Positive edge detect
  - Negative edge detect

LogiCORE IP Facts Table	
Core Specifics	
Supported Device Family	Helper Core (Not applicable)
Supported User Interfaces	Helper Core (Not applicable)
Resources	See Table 2-1.
Provided with Core	
Design Files	RTL
Example Design	Not Provided
Test Bench	Not Provided
Constraints File	Not Provided
Simulation Model	Not Provided
Supported S/W Driver	N/A
Tested Design Flows(1)	
Design Entry	Vivado® Design Suite
Simulation	For supported simulators, see the Xilinx Design Tools: Release Notes Guide
Synthesis	Vivaco Synthesis
Support	
Provided by Xilinx @ www.xilinx.com/support	

Notes:

1. For the supported versions of the tools, see the Xilinx Design Tools: Release Notes Guide.

Overview

Feature Summary

See Features for a list of features for this core.

Applications

There is no information currently provided for this core.

Unsupported Features

There are no unsupported features for this core.

Licensing and Ordering Information

This Xilinx LogiCORE™ IP module is provided at no additional cost with the Xilinx Vivado® Design Suite under the terms of the Xilinx End User License.

Information about this and other Xilinx LogiCORE IP modules is available at the Xilinx Intellectual Property page. For information on pricing and availability of other Xilinx LogiCORE IP modules and tools, contact your local Xilinx sales representative.

Product Specification

Performance

Performance data for this core is provided in Resource Utilization.

Maximum Frequencies

Maximum frequency data for this core is provided in Resource Utilization.

Latency

One clock cycle is the time at which the interrupt condition is received at input and actual interrupt is generated from the core. This latency is calculated for Registered Level interrupt configurations.

Throughput

There is no information currently provided for this core.

Power

There is no information currently provided for this core.

Resource Utilization

Resources required for the Interrupt Control core have been estimated for Zynq®-7000, Virtex®-7, Kintex®-7, and Artix®-7 devices (Table 2-1). These values were generated using Vivado® Design Suite. They are derived from post-synthesis reports, and might change during Map and PAR.

Table 2-1: **Resource Utilization**

Device	Parameters[1]			Device Resources			
	C_NUM_CE	C_NUM_IPIF_IRPT_SRC	C_IPIF_DWIDTH	LUTs	FFs	Slices	Fmax (MHz)
Artix-7 and Zynq-7000 [2]	16	4	32	13	7	5	200
Kintex-7 and Zynq-7000 (Kintex-7 Based) [3]	16	4	32	13	7	5	200
Virtex-7 [4]	16	4	32	13	7	5	200

Notes:

1. All other parameters use default values.
2. Resource estimates for Artix-7 and Zynq-7000 devices are generated using xc7a200tfbg676-2.
3. Resources estimates for Kintex-7 and Zynq-7000 (Kintex-7 based) devices are generated using xc7k325tffg900-3.
4. Resource estimates for Virtex-7 device are generated using xc7vx485tffg1761-3.

Port Descriptions

I/O Signals

The Interrupt Control device has two interfaces:

- Host Bus Interface (IPIF)
- User IP Interface (IP)

The I/O signals and associated interface for the design are provided in Table 2-2.

Table 2-2: **I/O Signals**

Port	Signal Name	Interface	I/O	Description
P1	Bus2IP_Reset	IPIF	I	Active-High reset signal.
P2	Bus2IP_Clk	IPIF	I	Input synchronization clock from the IPIF Bus clock.
P3	Bus2IP_Data(0 to C_IPIF_WIDTH - 1)	IPIF	I	Input Data Bus used for manipulating the Interrupt Registers.
P4	Bus2IP_BE(0 to C_IPIF_WIDTH/8 - 1)	IPIF	I	IPIF Byte Enable bus.
P5	Interrupt_RdCE(0 to C_NUM_CE - 1)	IPIF	I	Active-High read chip enable.

Table 2-2: **I/O Signals *(Cont'd)***

Port	Signal Name	Interface	I/O	Description
P6	Interrupt_WrCE(0 to C_NUM_CE - 1	IPIF	I	Active-High write chip enable.
P7	IPIF_Reg_Interrupts(0 to 1)	IPIF	I	Active-High interrupt inputs from the IPIF internal functions that need to be latched by the device Interrupt Source Controller.
P8	IPIF_Lvl_Interrupts(0 to C_NUM_IPIF_IRPT_SRC - 1)	IPIF	I	Active-High level interrupt inputs from IPIF internal functions to the device Interrupt Source Controller.
P9	IP2Bus_IntrEvent(0 to C_IP_INTR_MODE_ARRAY'length - 1)	IP	I	Interrupt signals from the user IP to the IP Interrupt Source Controller.
P10	Intr2Bus_DevIntr	IPIF	O	Active-High interrupt output to be sent to the System Interrupt Controller (INTC).
P11	Intr2Bus_DBus(0 to C_IPIF_DWIDTH - 1)	IPIF	O	Output Data Bus used during register read operations.
P12	Intr2Bus_WrAck	IPIF	O	Active-High signal indicating that the requested write operation has completed using the data input on the `Bus2IP_Data` bus.
P13	Intr2Bus_RdAck	IPIF	O	Active-High signal indicating that the requested read data is being output on the `Intr2Bus_DBus`.
P14	Intr2Bus_Error	IPIF	O	This signal is always set to 0.
P15	Intr2Bus_Retry	IPIF	O	This signal is always set to 0.
P16	Intr2Bus_ToutSup	IPIF	O	This signal is always set to 0.

Register Space

Register Summary

The following section discusses the user application interface to the registers provided by the Interrupt Controller.

Table 2-3: **Interrupt Control Channel Register Summary**

Register Name	Abbreviation	Address Offset from Base Address Assignment[(1)]	Access
Device Interrupt Source Controller			
Device Interrupt Status Register	DEVICE_ISR	0x00	Read/Toggle on Write[(2)]
Device Interrupt Pending Register	DEVICE_IPR	0x04	Read

Table 2-3: **Interrupt Control Channel Register Summary *(Cont'd)***

Register Name	Abbreviation	Address Offset from Base Address Assignment[1]	Access
Device Interrupt Enable Register	DEVICE_IER	0x08	Read/Write
Device Interrupt ID Register (Priority Encoder)	DEVICE_IIR	0x18	Read
Global Interrupt Enable	DEVICE_GIE	0x1C	Read/Write
User IP Interrupt Source Controller			
IP Interrupt Status Register	IPISR	0x20	Read/Toggle on Write[2]
IP Interrupt Enable Register	IPIER	0x28	Read/Write

Notes:

1. The Base Address is assigned by C_ARD_ADDR_RANGE_ARRAY generic for the AXI-Lite IPIF utilizing the Interrupt Control service.
2. Toggle each bit position to which a 1 is written

Register Description

Device Interrupt Status Register (offset 0x00)

The Device Interrupt Status Register shown in Figure 2-1 gives the interrupt status for the device. This register is fixed at 32 bits wide and each utilized bit within the register is set to 1 whenever the corresponding interrupt input (`IPIF_Lvl_Interrupts` and `IPIF_Reg_Interrupts`) has met the interrupt capture criteria. Unlike the IP Interrupt Status Register, the capture mode for this register is fixed. DEV_REG_IS(0) and DEV_REG_IS(1) (mapped from the `IPIF_Reg_Interrupts(0)` and `IPIF_Reg_Interrupts(1)` ports, respectively) are captured with a sample and hold high mode. This means that if the input interrupt is sampled to be 1 at a rising edge of a `Bus2IP_Clk` pulse, the register bit is set to a 1 and held until the user interrupt service routine clears it to a 0. The remaining bits within the register, DEV_LVL_IS(), (mapped from `IPIF_Lvl_Interrupts` ports) are passed through. Any additional sample and hold operation is not necessary in this register because after being asserted, the bits are held by the source of the interrupt. These interrupts must be cleared at the source function.

The number of active bits in the DEVICE_ISR allocated for level interrupts is determined by the C_NUM_IPIF_IRPT_SRC parameter. Bits of `IPIF_Lvl_Interrupts` are assigned in

increasing order, starting with 0, to decreasing bit positions in the status register, starting with C_IPIF_DWIDTH-2.

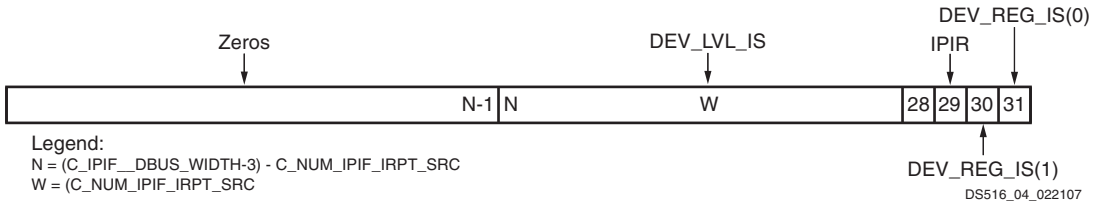

Figure 2-1: **Device Interrupt Status Register**

Table 2-4: **Status Register Description**

Bits	Name	Core Access	Reset Value	Description
31	DEV_REG_IS(0)	Read/Toggle on Write	0	Device Registered Interrupt Status 0: 0 = No interrupt is captured 1 = Interrupt is captured
30	DEV_REG_IS(1)	Read/Toggle on Write	0	Device Registered Interrupt Status 1: 0 = No interrupt is captured 1 = Interrupt is captured
29	IPIR	Read	0	IP Interrupt Request: This interrupt indicates that a User IP interrupt input on the `IP2Bus_IntrEvent` bus has been captured in the IP Interrupt Status Register and is enabled by the IP Interrupt Enable Register 0 = No interrupt is captured 1 = IP interrupt is captured
(N[(1)]) to 28	DEV_LVL_IS	Read	zeros	Device Level Interrupts: 0 = No interrupt is asserted 1 = Interrupt is asserted
0 to (N[(1)]-1)	Unused	Read	zeros	Reserved

Notes:

1. N = 29 - C_NUM_IPIF_IRPT_SRC.
2. Writing a 1 to a bit position within the register changes the corresponding bit position in the register to the toggle state. This mechanism avoids the requirement on the user interrupt service routing to perform a Read/Modify/Write operation to clear a single bit within the register.

Device Interrupt Pending Register (offset 0x04)

The device Interrupt Pending Register shown in Figure 2-2 is a read-only value that is the logical AND of the Device Interrupt Status Register and the Device Interrupt Enable Register on a bit-by-bit basis. Therefore, the Interrupt Pending Register reports only captured interrupts that are also enabled by the corresponding bit in the Interrupt Enable Register.

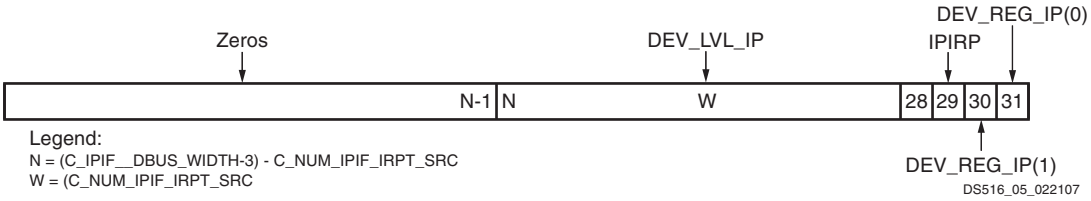

Figure 2-2: **Device Interrupt Pending Register**

Table 2-5: **Device Interrupt Pending Register Register**

Bits	Name	Core Access	Reset Value	Description
31	DEV_REG_IP(0)	Read	0	Device Registered Interrupt Pending 0: 0 = No enabled interrupt is pending 1 = Enabled Interrupt is pending
30	DEV_REG_IP(1)	Read	0	Device Registered Interrupt Pending 1: 0 = No enabled interrupt is pending 1 = Enabled Interrupt is pending
29	IPIRP	Read	0	IP Interrupt Request Pending: 0 = No enabled interrupt is pending 1 = Enabled Interrupt is pending
(N[(1)]) to 28	DEV_LVL_IP	Read	zeros	Level Interrupts Pending: 0 = No enabled interrupt is pending 1 = Enabled Interrupt is pending
0 to (N[(1)]-1)	Unused	Read	zeros	Reserved

Notes:

1. N = (C_IPIF_DWIDTH-3) - C_NUM_IPIF_IRPT_SRC. C_NUM_IPIF_IRPT_SRC must be less than or equal to C_IPIF_DWIDTH-3.

Device Interrupt Enable Register (offset 0x08)

The Device Interrupt Enable Register shown in Figure 2-3 determines which interrupt sources in the Device Interrupt Status Register are allowed to generate interrupts to the system.

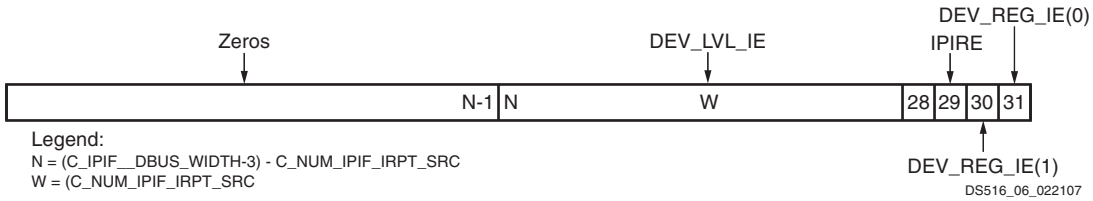

Figure 2-3: **Device Interrupt Enable Register**

Table 2-6: **Device Interrupt Enable Register**

Bits	Name	Core Access	Reset Value	Description
31	DEV_REG_IE(0)	Read/Write	0	Device Registered Interrupt Enable 0: 0 = Mask Interrupt 1 = Enabled Interrupt
30	DEV_REG_IE(1)	Read/Write	0	Device Registered Interrupt Enable 1: 0 = Mask Interrupt 1 = Enabled Interrupt
29	IPIRE	Read/Write	0	IP Interrupt Request Enable 0 = Mask Interrupt 1 = Enabled Interrupt
(N[(1)]) to 28	DEV_LVL_IE	Read/Write	zeros	Device Level Interrupts Enable: 0 = Mask Interrupt 1 = Enabled Interrupt
0 to (N[(1)]-1)	Unused	Read	zeros	Reserved

Notes:

1. N = (C_IPIF_DWIDTH-3) - C_NUM_IPIF_IRPT_SRC. C_NUM_IPIF_IRPT_SRC must be less than or equal to C_IPIF_DWIDTH-3.

Device Interrupt ID Register (offset 0x18)

The Device Interrupt ID Register shown in Figure 2-4 is an ordinal value output of a priority encoder. The value indicates which interrupt source, if any, has a pending interrupt. A value of `0x80` indicates that there are no pending interrupts, otherwise, the value gives the bit position in the Device Interrupt Pending Register (DIPR) of the highest priority interrupt that is pending.

The priority is highest for the interrupt bit in the LSB position (bit31), which reports as ID value `0x00`, and decreases in priority (and increases in reported ID value) for each successively more significant position (that is, going left).

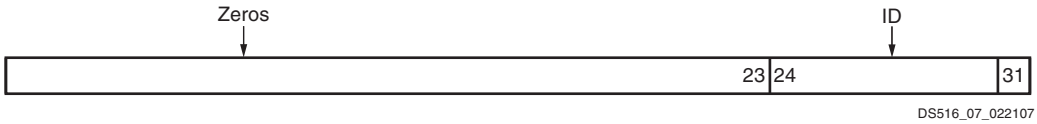

Figure 2-4: **Device Interrupt ID Register**

Table 2-7: **Device Interrupt ID Register Description**

Bits	Name	Core Access	Reset Value	Description
24 - 31	IID	Read	0x80	Interrupt ID: `0x80` - The DIPR has no pending interrupts Otherwise - The ordinal ID of the highest-priority pending interrupt in the DIPR
0 - 23		Read	zeros	Unused

Device Global Interrupt Enable Register (offset 0x1C)

The Global Interrupt Enable Interrupt Register shown in Figure 2-5 has a single defined bit, in the high-order position, that is used to globally enable the final interrupt output form the Interrupt Control service to the `Dev_Intr_out` output port.

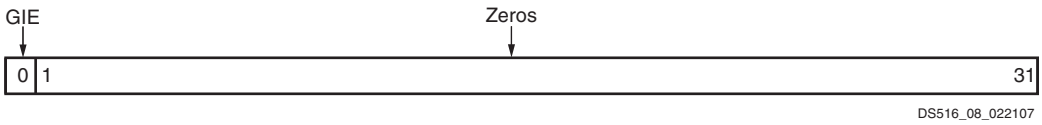

Figure 2-5: **Device Global Interrupt Enable Register**

Table 2-8: **Device Global Interrupt Enable Register Description**

Bits	Name	Core Access	Reset Value	Description
0	GIE	Read/Write	0	Global Interrupt Enable 0 = Interrupts disabled; no interrupt possible from this device. 1 = Interrupts enabled
1 - 31		Read	zeros	Unused

IP Interrupt Status Register (offset = 0x20)

The IP Interrupt Status Register (IPISR) shown in Figure 2-6 is the interrupt capture register for the user IP. It is part of the interrupt service. The IPISR captures interrupts input from the user IP on the `IP2Bus_IntrEvent` input port. The number of active bits in the IPISR, as well as the capture mode for each, is determined by the entries for the C_IP_INTR_MODE_ARRAY parameter. Bits of `IP2Bus_IntrEvent` are assigned in increasing order, starting with 0, to decreasing bit positions in the status register, starting with 31.

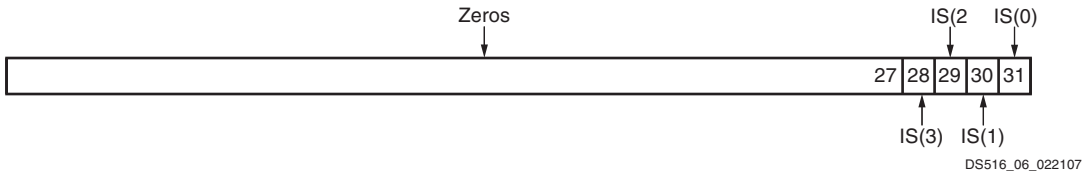

Figure 2-6: **IP Interrupt Status Register**

Table 2-9: **IP Interrupt Status Register Description**

Bits	Name	Core Access	Reset Value	Description
0 - 31	IS(i)	Read/Toggle on Write(1)	zeros	Interrupt Status: 1 = Interrupt Captured 0 = Interrupt Not Captured

Notes:

1. Writing a 1 to a bit position within the register changes the corresponding bit position in the register to the toggle state. This mechanism overrides the requirement on the user interrupt service routing to perform a Read/Modify/Write operation to clear a single bit within the register.

IP Interrupt Enable Register (offset 0x28)

The IP Interrupt Enable Register shown in Figure 2-7 has an enable bit for each defined bit of the IP Interrupt Status Register.

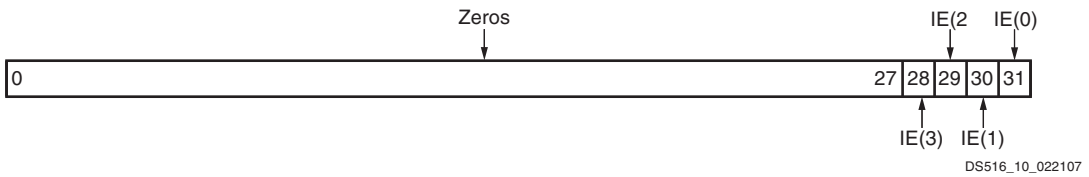

Figure 2-7: **IP Interrupt Enable Register**

Table 2-10: **IP Interrupt Enable Register Description**

Bits	Name	Core Access	Reset Value	Description
0 - 31	IE(i)	Read/Write	zeros	Interrupt Status: 1 = Interrupt Enabled 0 = Interrupt Masked

Designing with the Core

This chapter includes guidelines and additional information to facilitate designing with the core.

General Design Guidelines

Functional Description

Most microprocessor systems require peripheral devices to request the attention of the microprocessor through the assertion of interrupt signals. Generally, a central interrupt controller is used to collect the interrupts from various sources and then apply prioritization and masking functions to them per user application programming. The interrupt control service is a simple interrupt controller function that collects interrupts from a user device. These interrupts are generated by device services and the user IP. The interrupt service captures and coalesces these various interrupt signals into a single interrupt output signal that is sent to the system interrupt controller in the microprocessor. The service also provides local registers that the user application can use to read interrupt status, set up masking criteria, and perform interrupt clearing for the individual interrupts.

Interrupts can be generated within a device by the user IP and/or other device services. The number of user IP interrupts that need to be captured depends on the function of the IP and varies from IP core to IP core. Rather than attempting to accommodate this variable number of interrupt bits into a single register, a hierarchical interrupt capture and reporting scheme is used that is coupled with user parameterization.

The hierarchical interrupt reporting structure shown in Figure 3-1 is based on the Interrupt Source Controller (ISC). An ISC is a function that captures many interrupt input signals and,

using masking and logic, coalesces the captured interrupts into a single output single that is sent to the next higher level of interrupt hierarchy.

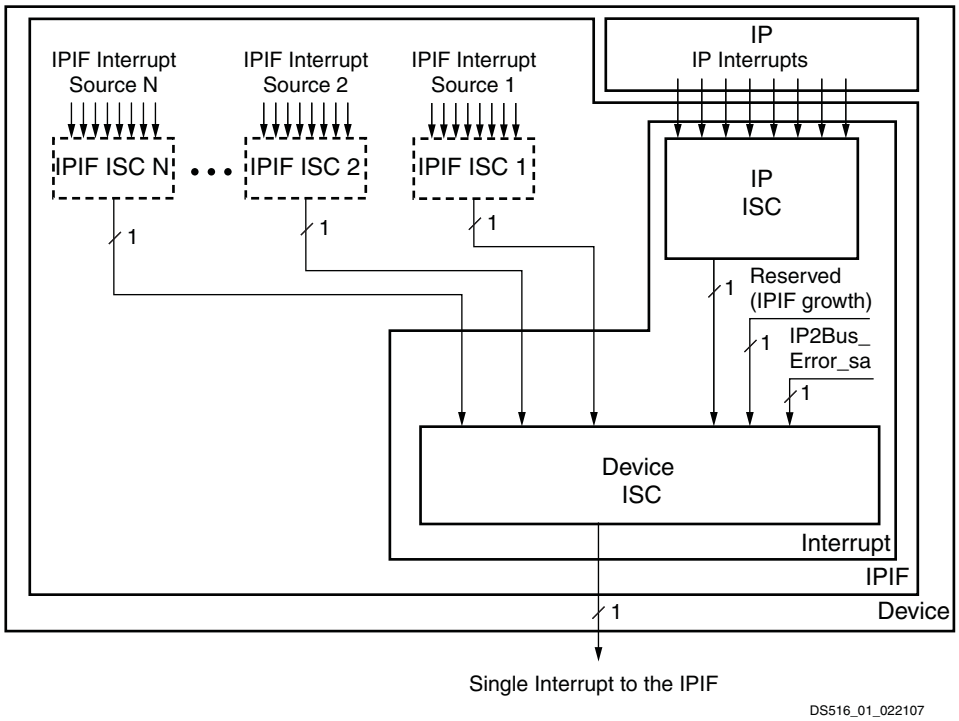

Figure 3-1: **Interrupt Control Hierarchy**

An interrupt-event signal is a transient condition captured by an interrupt source controller and held until there is an explicit acknowledgement (a clear operation) by the user application. An interrupt-active signal is defined as an interrupt signal that is captured and held (until actively cleared) by a lower-level ISC. Interrupt active signals do not need to be recaptured at the next higher level of interrupt hierarchy. An interrupt structure for an example device that is populated with a maximum number of interrupts is shown in Figure 3-2.

The device interrupt source controller is shown in the lower half of the diagram. IThe device ISC outputs the single device interrupt signal to the system interrupt controller in the microprocessor through the `Dev_Intr_Out` output port. This is the highest level of interrupt hierarchy for the device. The device registers also provide control and status information to mask and discover the source of interrupts within the device.

User interrupts are generally captured and controlled in the IP ISC. The IP ISC captures interrupt events directly from the user IP per the capture mode specified by the C_IP_INTR_MODE_ARRAY parameter. The number of user IP interrupts needed (N) is inferred from the number of entries in the C_IP_INTR_MODE_ARRAY parameter. The IP ISC then

coalesces the IP interrupts into a single interrupt active signal that is output the to the Device ISC.

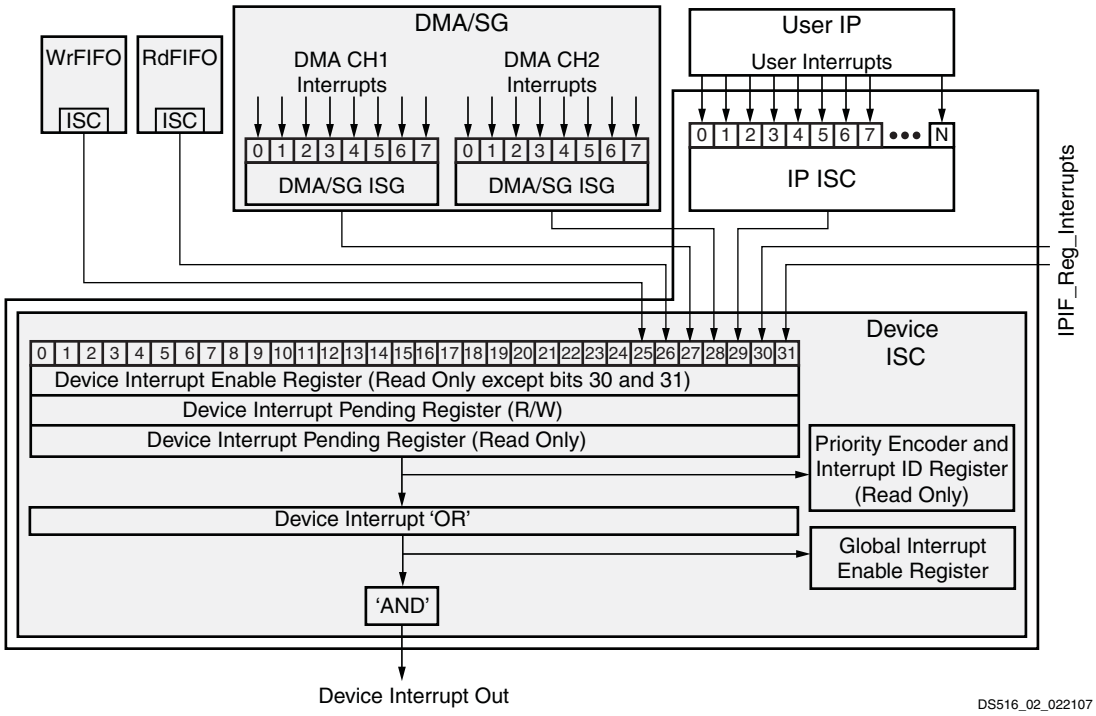

Figure 3-2: **Example User Device Interrupt Hierarchy**

The device ISC can be parameterized out of the interrupt control service by setting C_INCLUDE_DEV_ISC = FALSE. In this case the only source of interrupts is the IP ISC. This option, which reduces hardware cost and software accesses, is shown in Figure 3-3.

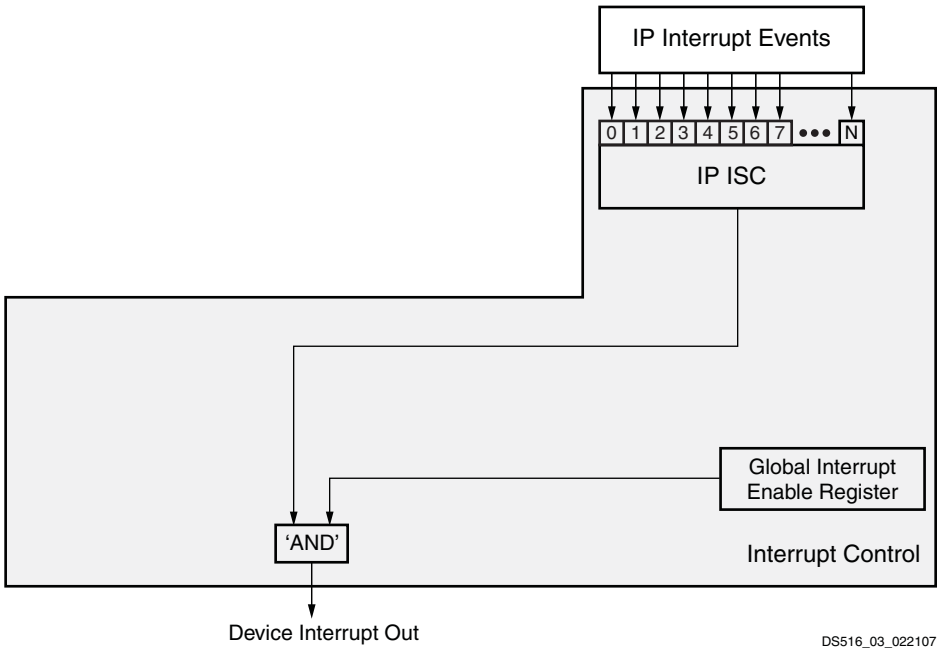

Figure 3-3: **Example with Device ISC Removed**

User IP Interrupt Capture Mode

There are six user IP interrupt capture modes. These are summarized in Table 3-1. The interrupt capture mode is set with the C_IP_INTR_MODE_ARRAY parameter.

Table 3-1: **User IP Capture Mode**

Capture Mode	Description
1	**Pass Through (non-inverting)**: This mode passes through the interrupt event without capturing the event. This mode is used for situations where the interrupt event is actually captured in the user IP. In this mode, to clear the interrupt, the event must be cleared at the user IP.
2	**Pass Through (inverting)**: This mode passes through the interrupt and as it is doing so, inverts the level of the interrupt event. This mode is used for active-Low interrupt events. Like in mode 1, this mode is used for situation where the interrupt event is actually captured in the user IP. In this mode, to clear the interrupt, the event must be cleared at the user IP.
3	**Registered Level (non-inverting)**: This mode captures the interrupt event based on the level input from the user IP. This interrupt mode does not require the interrupt event to be captured in the user IP, but does require the interrupt event to maintain the active level (logical 1) a minimum of two `Bus2IP_Clk` cycles to be captured by the ISC.

Table 3-1: **User IP Capture Mode *(Cont'd)***

Capture Mode	Description
4	**Registered Level (inverting)**: This mode captures the interrupt event based on the level input from the user IP. This interrupt mode does not require the interrupt event to be captured in the user IP, but does require the interrupt event to maintain the active level (logical 0) a minimum of two `Bus2IP_Clk` cycles to be captured by the ISC.
5	**Positive Edge Detect**: This mode captures the interrupt event on the rising edge of the event. This interrupt mode does not require the interrupt event to be captured in the user IP.
6	**Negative Edge Detect**: This mode captures the interrupt event on the falling edge of the event. This interrupt mode does not require the interrupt event to be captured in the user IP.

Parameters

The Interrupt Control device can be parameterized for an individual application. Table 3-2 shows the Interrupt Control parameters.

Table 3-2: **Interrupt Control Design Parameters**

Generic	Parameter Name	Feature / Description	Allowable Values	Default Value	VHDL Type
GI	C_NUM_CE	Sizes the `Interrupt_RdCE` and `Interrupt_WrCe` ports. Specifies the number of chip enables required for registers.	16 for C_IPIF_DWIDTH = 32 8 for C_IPIF_DWIDTH = 64 4 for C_IPIF_DWIDTH = 128	4	Integer
G2	C_NUM_IPIF_IRPT_SRC	Number of device Level Interrupts	1 to 29(4)	4	Integer
G3	C_IP_INTR_MODE_ARRAY	Capture mode for IP interrupts. This array also sets the number of IP interrupts to capture(1).	1 = Pass Through (non-inverting) 2 = Pass Through (inverting) 3 = Registered Level (non-inverting) 4 = Registered Level (inverting) 5 = Positive Edge Detect 6 = Negative Edge Detect	(1,2)	INTEGER_ARRAY_TYPE
G4	C_INCLUDE_DEV_PENCODER(1)	Device Priority Encoder feature Inclusion/ Omission	true = Include Priority Encoder(1) false = Omit Priority Encoder	false	Boolean

Table 3-2: **Interrupt Control Design Parameters**

Generic	Parameter Name	Feature / Description	Allowable Values	Default Value	VHDL Type
G5	C_INCLUDE_DEV_ISC	Device ISC feature Inclusion/Omission	true = Include Device ISC false = Omit Device ISC	false	Boolean
G6	C_IPIF_DWIDTH	IPIF Data Bus Width	32, 64, or 128	128	Integer

Notes:

1. C_INCLUDE_DEV_PENCODER is only valid if the device ISC is included, that is, C_INCLUDE_DEV_ISC=true.

Allowable Parameter Combinations

Table 3-3: **Allowable Parameter Combinations**

Dependent Parameter		Affected Parameter		Dependency Description
C_INCLUDE_DEV_PENCODER	G4	C_INCLUDE_DEV_ISC	G5	If G5 is set to false, G4 is not used
C_NUM_CE	G1	C_IPIF_WIDTH	G6	If G6 = 32, set G1 = 16 If G6 = 64, set G1 = 8 If G6 = 128, set G1 = 4

Clocking

The Interrupt Control core works on the AXI clock or the clock provided by the core that instantiates this core.

Resets

The Interrupt Control core works with an active-High reset.

Protocol Description

There is no information currently provided for this core.

Customizing and Generating the Core

This chapter includes information about using the Vivado® Design Suite to customize and generate the core.

Vivado Integrated Design Environment

The Interrupt Control core is not available in the Vivado Integrated Design Environment.

This helper core is instantiated in the master IP, which requires interrupt generation and logic handling. For information on configuring the core, see the parameters defined in Parameters in Chapter 3.

Output Generation

The master core into which the Interrupt Control core is instantiated provides the generated output files.

Constraining the Core

There are no constraints associated with this core. You can add constraints to meet design requirements.

Simulation

This chapter contains information about simulating in the Vivado® Design Suite environment. For comprehensive information about Vivado simulation components, as well as information about using supported third party tools, see the *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 4].

Synthesis and Implementation

This chapter contains information about synthesizing and implementing IP in the Vivado® Design Suite environment.

For details about synthesis and implementation, see "Synthesizing IP" and "Implementing IP" in the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 3].

Migrating and Upgrading

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

Migrating to the Vivado Design Suite

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

Upgrading in the Vivado Design Suite

Interrupt Control is a helper core. For information on upgrading from older versions of the core to the current release, see the product guide for the core that instantiates Interrupt Control.

Debugging

This appendix includes details about resources available on the Xilinx Support website and debugging tools to guide you through debugging the Interrupt Control core.

Finding Help on Xilinx.com

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

Documentation

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

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

Answer Records

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

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

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

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

Master Answer Record for the Interrupt Control

AR: 56143

Contacting Technical Support

Xilinx provides technical support at www.xilinx.com/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.

To contact Xilinx Technical Support:

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

When opening a WebCase, include:

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

Note: Access to WebCase is not available in all cases. Login to the WebCase tool to see your specific support options.

Debug Tools

Vivado Lab Tools

Vivado® lab tools inserts logic analyzer (ILA) and virtual I/O (VIO) cores directly into your design. Vivado lab tools allow you to set trigger conditions to capture application and integrated block port signals in hardware. Captured signals can then be analyzed. This feature represents the functionality in the Vivado Integrated Design Environment that is used for logic debugging and validation of a design running in Xilinx FPGA devices in hardware.

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

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

See *Vivado Design Suite User Guide: Programming and Debugging* (UG908) [Ref 5].

Interface Debug

Core Interfaces

The Interrupt Control core has an IP Interconnect (IPIC) interface and it should be used in the other master core. The core interface mainly includes signals such as `Bus2IP_*` and `Intr2Bus_*`. The `Bus2IP_*` signals are used to transfer the AXI transactions in the core. While doing so, the AXI transactions can use the user-defined logic for signal conversion in `Bus2IP_*` and for reading through `Intr2Bus_*` signal interface format. The internal registers of the core can be accessed for writing using appropriate `Bus2IP_WrCe` signal along with the `Bus2IP_Data` and `Bus2IP_BE` signal. The registers can be read using the appropriate `Bus2IP_RdCE` where the register data is available on the `Intr2Bus_DBus` signal.

If the interface is unresponsive, ensure that the following conditions are met:

- The AXI clock is available, and the `Bus2IP_Clk` inputs are connected and toggled.
- The interface is not being held in reset, and `Bus2IP_Reset` is an active-High reset.
- The interface is enabled, and `Bus2IP_Clk` is active-High (if used).
- The main core clocks are toggling and that the enables are also asserted.

- If the simulation has been run, verify in simulation and/or a Lab Tools debugging tool capture that the waveform is correct for accessing the IPIC interface.

RECOMMENDED: *Xilinx recommends that you operate this core in 32-bit data width mode.*

- All `Bus2IP_*` signals should be driven from the core where this core is instantiated.
- For Read transactions from AXI to this core, the appropriate lane in the `Bus2IP_RdCE` signal is activated along with `Bus2IP_CS`.
- For Write transactions from AXI to this core, the appropriate lane in the `Bus2IP_WrCE` signal is activated along with `Bus2IP_CS`.
- For read or write transactions, keep check on all `IP2Bus_*` signals.
- All read transactions should be completed with `IP2Bus_RdAck` while all the write transactions should be completed with `IP2Bus_WrAck`. The `IP2Bus_Error` signal will be activated along with the read or write `ack` in case of error scenarios.
- The `interrupt` signal coming out of this core is observed on `IP2Bus_IntrEvent`.

Additional Resources

Xilinx Resources

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

www.xilinx.com/support

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

www.xilinx.com/company/terms.htm

References

These documents provide supplemental material useful with this product guide:

1. *LogiCORE IP AXI4-Lite IPIF Product Guide* (PG155)
2. *ISE to Vivado Design Suite Migration Methodology Guide* (UG911)
3. *Vivado Design Suite User Guide: Designing with IP* (UG896)
4. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
5. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
06/19/2013	3.0	Initial Xilinx release as a product guide. Replaces *LogiCORE IP Interrupt Control Data Sheet* (DS516).
10/02/2013	3.0	Re-released the core v2.0 product guide, with the following changes made: • Added statement to add constraints based on design requirements. • Added further conditions to check for in the Interface Debug section (Core Interfaces). • Added Simulation, and Synthesis and Implementation chapters. • Added additional resources in the References section.

Notice of Disclaimer