XILINX® LogiCORE™

Interrupt Control (v2.00a)

DS516 February 27, 2007 **Product Specification**

Introduction

The Interrupt Control service is a continuation of the Xilinx family of IBM CoreConnect compatible LogiCORE products. It provides interrupt capture support for internal IPIF sub-block as well as support for the connected IP function.

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™ Facts		
Core Specifics		
Supported Device Family	Virtex™, Virtex-E, Virtex-II, and Virtex-II Pro, Spartan-II, Spartan-IIE, Spartan-3, Spartan-3E	
Version of Core	interrupt_control	v2.00a
Resources Used		
	Min	Max
Slices	28	132
LUTs	19	53
FFs	43	240
Block RAMs	0	
Provided with Core		
Documentation	Product Specification	
Design File Formats	VHDL	
Constraints File	None	
Verification	EDK Simulation Support	
Instantiation Template	Wizard Support	
Reference Designs	N/A	
Design Tool Requirements		
Xilinx Implementation Tools	Xilinx EDK8.1	
Verification	ModelSim SE 5.7e or later	
Simulation	ModelSim SE 5.7e or later	

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 is used to collect interrupts from a user device. These will be 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 utilize to read interrupt status, set up masking criteria, and perform interrupt clearing for the individual interrupts.

Interrupts may 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 is generally quite different from IP to IP. 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 1 is based on the interrupt source controller, sometimes referred to simply as an ISC. An interrupt source controller is a function that captures a number of 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.

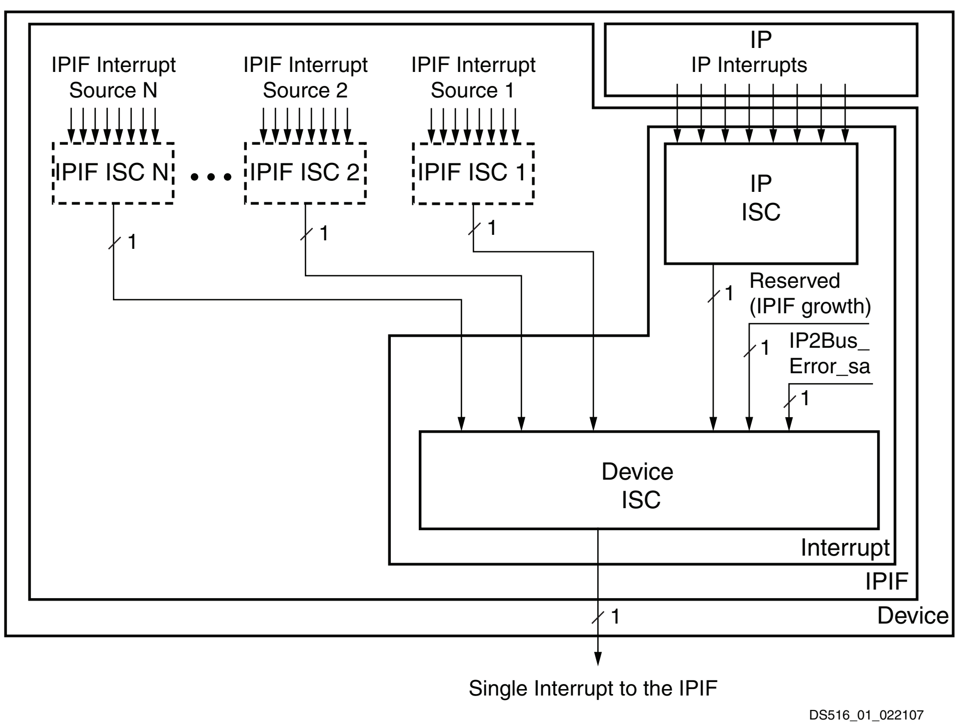

Figure 1: **Interrupt Control Hierarchy**

An interrupt-event signal is a transient condition that needs to be captured by an interrupt source controller and held until there is an explicit acknowledgement (actually 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 hierarch. An interrupt structure for an example user device that is maximally populated with interrupts is shown in Figure 2.

In the lower half of the diagram, the device interrupt source controller is shown. It is the function of the device ISC to output the single device interrupt signal to the system interrupt controller in the microprocessor via the Dev_Intr_Out output port. It is the highest level of interrupt hierarchy for the device. The device registers also provide control and status information that are used to mask and discover the source of interrupts within the device, or both.

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.

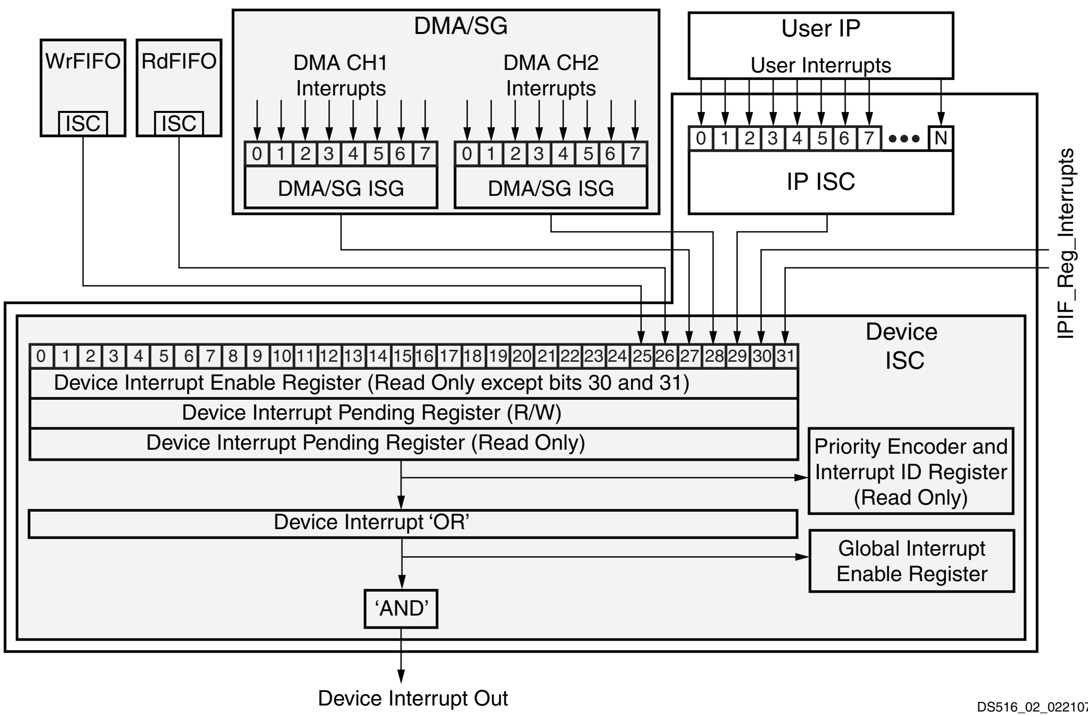

Figure 2: **Example User Device Interrupt Hierarchy**

The device ISE may be parameterized out of the interrupt control service by setting C_INCLUDE_DEV_ISE = 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.

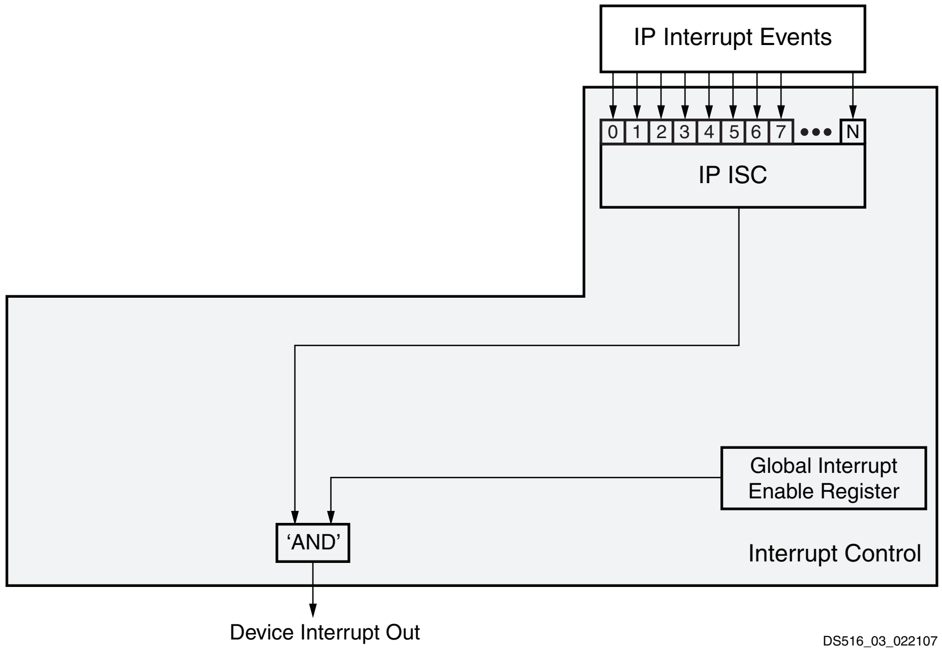

Figure 3: **Example with Device ISE Removed**

User IP Interrupt Capture Mode

There are 6 User IP interrupt capture modes. These are summarized in Table 1. The interrupt capture mode is set via the C_IP_INTR_MODE_ARRAY parameter.

Table 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's 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's 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 2 Bus2IP_Clk cycles to be captured by the ISC.
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 2 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.

Interrupt Control Parameters

The Interrupt Control service can be parameterized for individual application. Table 2 shows the Interrupt Control parameters.

Table 2: **Interrupt control Design Parameters**

Generic	Parameter Name	Feature / Description	Allowable Values	Default Value	VHDL Type
G1	C_NUM_CE	Sizes the interrupt_RdCE and Interrupt_WrCe ports. Specifies the number of chip enables requiered for registers.	16 for C_IPIF_DBUS_ WIDTH = 32 8 for C_IPIF_DBUS_ WIDTH = 64 4 for C_IPIF_DBUS_ WIDTH = 128	4	Integer
G2	C_NUM_IPIF_ IRPT_SRC	Number of device Level Interrupts	4 to 128(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	Device Priority Encoder feature Inclusion/Omission	true=Include Priority Encoder(1) false=Omit Priority Encoder	false	Boolean
G5	C_INCLUDE_DEV _ISC	Device ISC feature Inclusion/Omission	true=Include Device ISC false=Omit Device ISC	false	Boolean
G6	C_IPIF_DBUS_ WIDTH	IPIF Data Bus Width	32, 64, or 128	128	Integer

Note:
1. C_INCLUDE_DEV_PENCODER is only valid if the device ISC is included, i.e. C_INCLUDE_DEV_ISC=true

Allowable Parameter Combinations

Table 3: **Interrupt Control 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 = 7. If G6 = 128, set G1 = 4.

Interrupt Control I/O Signal

The Interrupt Control service has 2 interfaces. These are the Host Bus Interface (IPIF), and the User IP interface (IP). The I/O signals for the design are listed in Table 4.

Table 4: **Interrupt Control 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
P6	Interrupt_WrCE(0 to **C_INTERRUPT_REG_NUM** - 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_DBUS_WIDTH** - 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'.

Parameter - Port Dependencies

The Interrupt control parameterization has effects on some of the I/O port sizes. These are indicated in the port definitions presented in Table 4.

Interrupt Control Register Summary

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

Table 5: **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
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 OPB or PLB IPIF utilizing the Interrupt Control service.
2. Toggle each bit position to which a 1 is written

Interrupt Control Register Description

Device Interrupt Status Register (offset 0x00)

The device Interrupt Status Register shown in Figure 4 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 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 once 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 user parameter, C_NUM_IPIF_IRPT_SRC. 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_DBUS_WIDTH**-2.

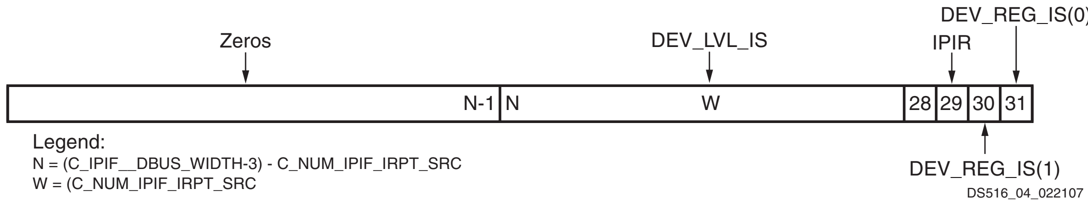

Figure 4: **Device Interrupt Status Register**

Table 6: **Device Interrupt Status Register Description**

Bit(s)	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 via 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 5 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 will report only captured interrupts that are also enabled by the corresponding bit in the Interrupt Enable Register.

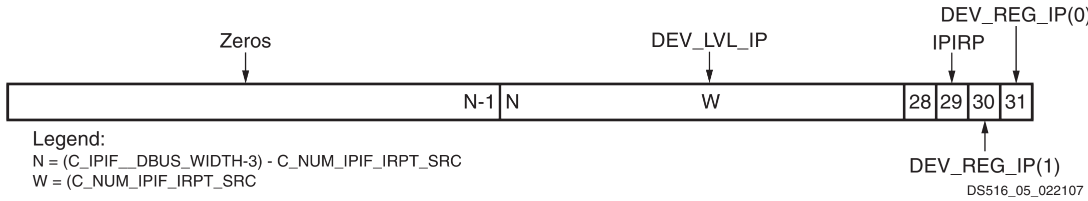

Figure 5: **Device Interrupt Pending Register**

Table 7: **Device Interrupt Pending Register Register**

Bit(s)	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

Note:
1. N = **(C_IPIF_DBUS_WIDTH**-3) - **C_NUM_IPIF_IRPT_SRC**. **C_NUM_IPIF_IRPT_SRC** must be less than or equal to **C_IPIF_DBUS_WIDTH**-3.

Device Interrupt Enable Register (offset 0x08)

The Device Interrupt Enable Register shown in Figure 6 determines which interrupt sources in the Device Interrupt Status Register are allowed to generate interrupts to the system.

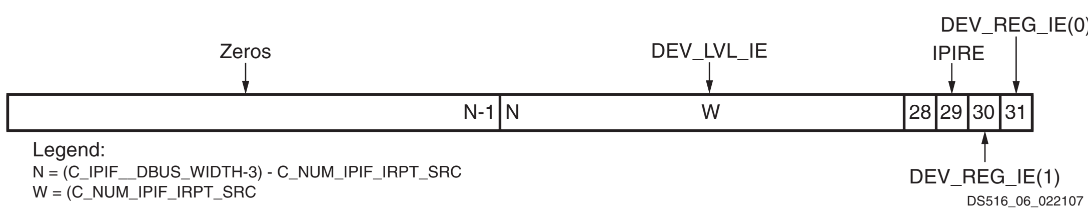

Figure 6: **Device Interrupt Enable Register**

Table 8: **Device Interrupt Enable Register**

Bit(s)	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

Note:
1. N = **(C_IPIF_DBUS_WIDTH**-3) - **C_NUM_IPIF_IRPT_SRC**. **C_NUM_IPIF_IRPT_SRC** must be less than or equal to **C_IPIF_DBUS_WIDTH**-3.

Device Interrupt ID Register (offset 0x18)

The Device Interrupt ID Register shown in Figure 7 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 (an increases in reported ID value) for each successively more significant position (i.e. going left).r

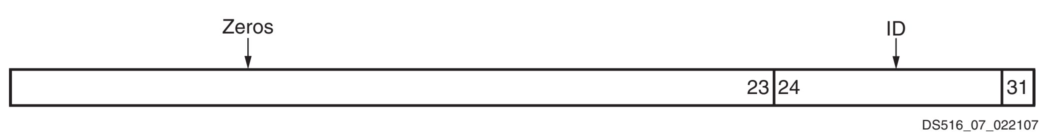

Figure 7: **Device Interrupt ID Register**

Table 9: **Device Interrupt ID Register Description**

Bit(s)	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 8 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.

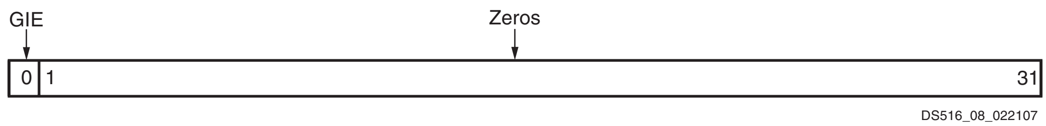

Figure 8: **Device Global Interrupt Enable Register**

Table 10: **Device Global Interrupt Enable Register Description**

Bit(s)	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 9 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 user entries for the parameter C_IP_INTR_MODE_ARRAY. Bits of IP2Bus_IntrEvent are assigned in increasing order, starting with 0, to decreasing bit positions in the status register, starting with 31.

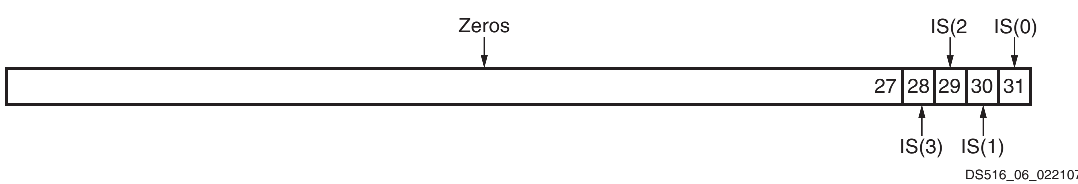

Figure 9: **IP Interrupt Status Register**

Table 11: **IP Interrupt Status Register Description**

Bit(s)	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

Note:

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 over-rides 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 0x24)

The IP Interrupt Enable Register shown in Figure 10 has an enable bit for each defined bit of the IP Interrupt Status Register.

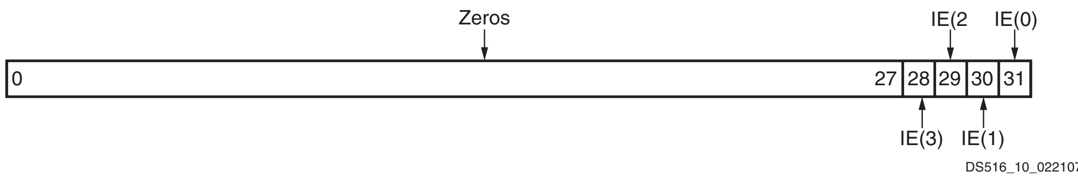

Figure 10: **IP Interrupt Enable Register**

Table 12: **IP Interrupt Enable Register Description**

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

User Application Topics

FPGA Design Application Hints

Single Entry in Unconstrained Array Parameters

The DMA parameterization employes generics that are defined as unconstrained arrays, such as arrays whose size is left unbound at declaration and fixed later by the user. This is the underlying VHDL mechanism that allows the DMA to grow or shrink to the size required by the application. The size of the unconstrained array and its element values are fixed simultaneously by the user by assigning to the array a constant aggregate which is a list of values enclosed in parentheses and separated by commas.

The list can take either of two forms, *positional association*, in which the values at indices in the array are populated by the aggregate elements as they appear, left to right, or *named association*, in which each value populates an index to which it is explicitly assigned by being proceeded by "INDX =>". Thus, the following positional and named aggregates are identical: (4,3,9) and (0=>4, 1=>3, 2=>9).

For aggregates with a single element, positional association is not allowed because the syntax would be ambiguous with a parenthesized expression. The following example shows the incorrect and the correct way to associate a single element to an unconstrained array.

Incorrect:

C_IP_INTR_MODE_ARRAY **=> (1);** --VHDL **positional association** NOT allowed because it would be ambiguous.

Correct:

C_IP_INTR_MODE_ARRAY **=> (0=> 1);** -- VHDL **named association** instead.

Target Technology

This Interrupt Control is targeted for the Virtex-II Pro and Virtex4-FX devices

Device Utilization and Performance Benchmarks

The Interrupt Control benchmarks shown in Table 13 are for a Virtex-II Pro -7 FPGA.

Table 13: **Interrupt Control FPGA Performance and Resource Utilization Benchmarks**

Parameter Values				Device Resources			
C_INCLUDE_DEV_ISC	C_INCLUDE_DEV_PENCODER	C_NUM_IPIF_IRPT_SRC	C_IPIF_DWIDTH	Slices	Slice Flip-Flops	4-input LUTs	f_{MAX_REG} (1)
false	false	4	32	28	19	43	162.3
false	false	29	32	28	19	43	162.3
false	false	29	64	28	19	41	210.3
true	false	29	64	93	53	164	156.4
true	true	29	64	140	53	249	197.4
true	true	29	128	132	53	240	146.5

Notes:
1. Fmax represents the maximum frequency of the Interrupt Control service in a standalone configuration. The actual maximum frequency will depend on the entire system and may be greater or less than what is recorded in this table. Thus Fmax should be used purely as a reference and a rough measure of the relative affects various configurations have on the operating frequency.
2. C_IP_INTR_MODE_ARRAY = (1, 2, 3, 4, 5, 6) for all configurations

Specification Exceptions

None

Interrupt Control Signaling Exceptions

Reference Documents

None

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
2/9/06	1.0	Initial Xilinx release
2/27/07	1.1	Converted to new data sheet template; images produced to graphic standard