XPS Interrupt Controller (v2.00a)

Introduction

The XPS Interrupt Controller (XPS INTC) concentrates multiple interrupt inputs from peripheral devices to a single interrupt output to the system processor. The registers for checking, enabling and acknowledging interrupts are accessed through a slave interface for the Processor Local Bus (PLB V4.6). The number of interrupts and other aspects can be tailored to the target system.

Features

- Connects as a 32-bit slave on PLB V4.6 bus of 32, 64 and 128-bit data width
- 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 disabling individual interrupt inputs
- Master Enable Register for disabling 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 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 low

LogiCORE™ Facts		
Core Specifics		
Supported Device Family	Spartan®-3A/3A DSP, Spartan-3, Spartan-3E, Automotive Spartan 3/3E/3A/3A DSP, Spartan-6, Virtex®-4 /4Q/4QV, Virtex-5/5FX, Virtex-6/6CX	
Version of Core	xps_intc	v2.00a
Resources Used		
	Min	Max
Slices	Refer to the Table 13, Table 14, Table 15, Table 16, Table 17and Table 18	
FFs		
LUTs		
Block RAMs	N/A	
Provided with Core		
Documentation	Product Specification	
Design File Formats	VHDL	
Constraints File	N/A	
Verification	N/A	
Instantiation Template	N/A	
Reference Designs & Application Notes	N/A	
Design Tool Requirements		
Xilinx Implementation Tools	ISE® 11.4 or later	
Verification	ModelSim PE/SE 6.4b or later	
Simulation	ModelSim PE/SE 6.4b or later	
Synthesis	XST	
Support		
Provided by Xilinx, Inc.		

Functional Description

Interrupt Controller Overview

Most CPUs provide one or more interrupt request input pins that allow external devices to request service. XPS INTC can be used to expand the number of interrupt inputs available to the CPU and, optionally, to provide a priority encoding scheme. The output of XPS INTC is intended to be connected to a CPU interrupt input and each of the interrupt inputs to XPS 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 XPS INTC and retained until explicitly cleared (also referred to as acknowledged). Facilities are also provided to enable or disable interrupts, globally and individually. If all interrupts are globally enabled and if at least one captured interrupt is individually enabled, an interrupt condition is signaled upstream to the CPU.

Edge and Level-Sensitive Capture Modes

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

The first, 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 the three main types of edge generation schemes, using rising edges for the active edge in this example. In all three schemes, the device generating the interrupt provides an active edge and some time later it produces an inactive edge in preparation for generating a new interrupt request.
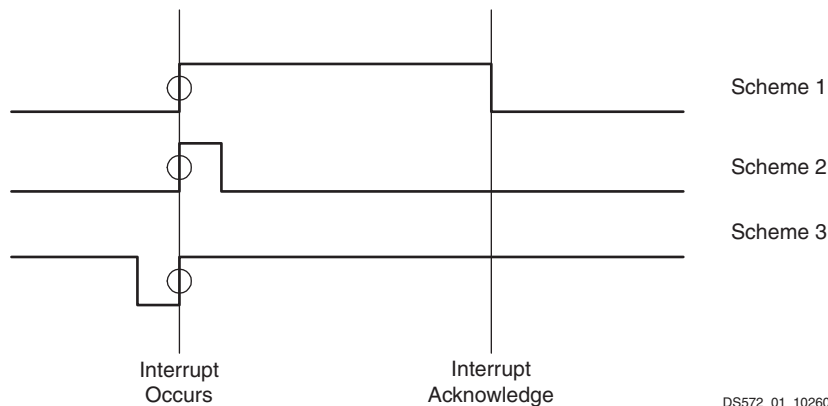

Figure 1: **Schemes for Generating Edges**

The second mode, level-sensitive capture, causes an interrupt condition to be captured any time the input is at the active level and an interrupt condition does not already exist. (The polarity of the active level, high or low, is an per-input option.)

An observable difference between edge-sensitive and level-sensitive capture is:

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

Timing Requirement

There is a timing requirement on interrupt input signaling. After an edge for an edge-sensitive interrupt input signal or after the establishment of the active level for a level-sensitive interrupt input signal, the signal must remain stable for a minimum interval to guarantee capture. If the device driving the interrupt input is on the same clock domain as XPS INTC, then this interval is one clock period with standard synchronous setup and hold requirements. If the driving device is on an unrelated clock, the interval must be at least an XPS INTC clock period plus a suitable margin. The margin is chosen by specification to be 20% of the XPS INTC clock period.

Interrupt Vector Register

If an optional Interrupt Vector Register is opted in the XPS INTC, 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).

XPS INTC Organization

The XPS INTC is organized mainly into the following functional units as shown in Figure 2:

• PLB interface - Provides the interface to Processor Local Bus

• Interrupt Detect (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

• Request Generation (Irq Gen) - Generate the output interrupt request
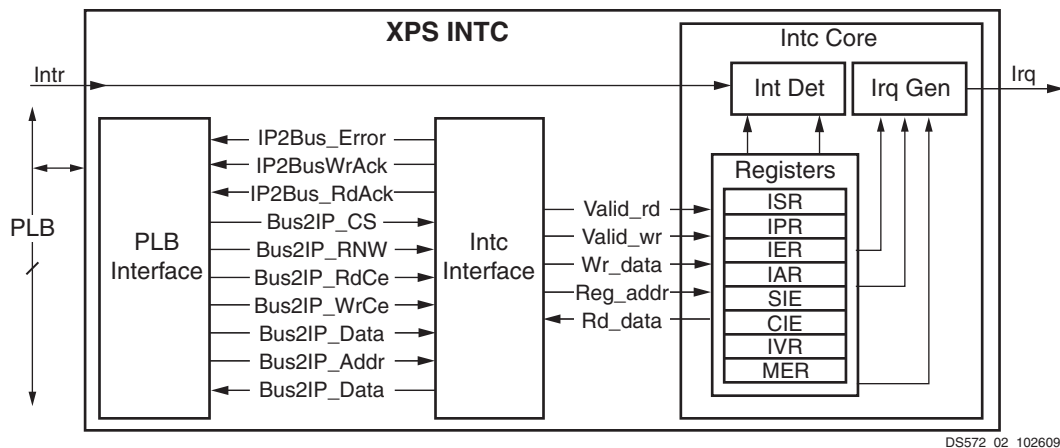

DS572_02_102609

Figure 2: **XPS INTC Block Diagram**

PLB Interface

PLB interface provides a slave interface on the PLB for transferring data between the INTC and the processor. The XPS INTC registers are memory mapped into the PLB address space and data transfers occur using PLB byte enables.

The register addresses are fixed on four 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

This is the logical block to connect the Intc Core to the PLB Interface in XPS INTC

Interrupt Controller Core (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.

Registers (Regs)

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

- "Interrupt Status Register (ISR)" on page 10
- "Interrupt Pending Register (IPR)" on page 11
- "Interrupt Enable Register (IER)" on page 12
- "Interrupt Acknowledge Register (IAR)" on page 12
- "Set Interrupt Enables (SIE)" on page 13
- "Clear Interrupt Enables (CIE)" on page 14
- "Interrupt Vector Register (IVR)" on page 14
- "Master Enable Register (MER)" on page 15

Interrupt request Generation (Irq Gen)

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

XPS INTC I/O Signals

The XPS INTC signals are listed and described in Table 1.

Table 1: **XPS INTC I/O Signal Description**

Port	Signal Name	Interface	I/O	Initial State	Description
System Signals					
P1	SPLB_Clk	PLB	I	-	PLB clock
P2	SPLB_Rst	PLB	I	-	PLB reset, active high
PLB Interface Signals					
P3	PLB_ABus[0 : 31]	PLB	I	-	PLB address bus
P4	PLB_PAValid	PLB	I	-	PLB primary address valid
P5	PLB_masterID [0 : C_SPLB_MID_WIDTH - 1]	PLB	I	-	PLB current master identifier
P6	PLB_RNW	PLB	I	-	PLB read not write
P7	PLB_BE [0 : (C_SPLB_DWIDTH/8) - 1]	PLB	I	-	PLB byte enables
P8	PLB_size[0 : 3]	PLB	I	-	PLB size of requested transfer
P9	PLB_type[0 : 2]	PLB	I	-	PLB transfer type
P10	PLB_wrDBus [0 : C_SPLB_DWIDTH - 1]	PLB	I	-	PLB write data bus
Unused PLB Interface Signals					
P11	PLB_UABus[0 : 31]	PLB	I	-	PLB upper address bits
P12	PLB_SAValid	PLB	I	-	PLB secondary address valid
P13	PLB_rdPrim	PLB	I	-	PLB secondary to primary read request indicator
P14	PLB_wrPrim	PLB	I	-	PLB secondary to primary write request indicator
P15	PLB_abort	PLB	I	-	PLB abort bus request
P16	PLB_busLock	PLB	I	-	PLB bus lock
P17	PLB_MSize[0 : 1]	PLB	I	-	PLB data bus width indicator
P18	PLB_lockErr	PLB	I	-	PLB lock error
P19	PLB_wrBurst	PLB	I	-	PLB burst write transfer
P20	PLB_rdBurst	PLB	I	-	PLB burst read transfer
P21	PLB_wrPendReq	PLB	I	-	PLB pending bus write request
P22	PLB_rdPendReq	PLB	I	-	PLB pending bus read request
P23	PLB_wrPendPri[0 : 1]	PLB	I	-	PLB pending write request priority
P24	PLB_rdPendPri[0 : 1]	PLB	I	-	PLB pending read request priority
P25	PLB_reqPri[0 : 1]	PLB	I	-	PLB current request priority

Table 1: **XPS INTC I/O Signal Description** *(Contd)*

Port	Signal Name	Interface	I/O	Initial State	Description
P26	PLB_TAttribute[0 : 15]	PLB	I	-	PLB transfer attribute
PLB Slave Interface Signals					
P27	Sl_addrAck	PLB	O	0	Slave address acknowledge
P28	Sl_SSize[0 : 1]	PLB	O	0	Slave data bus size
P29	Sl_wait	PLB	O	0	Slave wait
P30	Sl_rearbitrate	PLB	O	0	Slave bus rearbitrate
P31	Sl_wrDAck	PLB	O	0	Slave write data acknowledge
P32	Sl_wrComp	PLB	O	0	Slave write transfer complete
P33	Sl_rdDBus[0 : C_SPLB_DWIDTH - 1]	PLB	O	0	Slave read data bus
P34	Sl_rdDAck	PLB	O	0	Slave read data acknowledge
P35	Sl_rdComp	PLB	O	0	Slave read transfer complete
P36	Sl_MBusy [0 : C_SPLB_NUM_MASTERS - 1]	PLB	O	0	Slave busy
P37	Sl_MWrErr [0 : C_SPLB_NUM_MASTERS - 1]	PLB	O	0	Slave write error
P38	Sl_MRdErr [0 : C_SPLB_NUM_MASTERS - 1]	PLB	O	0	Slave read error
Unused PLB Slave Interface Signals					
P39	Sl_wrBTerm	PLB	O	0	Slave terminate write burst transfer
P40	Sl_rdWdAddr[0 : 3]	PLB	O	0	Slave read word address
P41	Sl_rdBTerm	PLB	O	0	Slave terminate read burst transfer
P42	Sl_MIRQ [0 : C_SPLB_NUM_MASTERS - 1]	PLB	O	0	Master interrupt request
INTC Interface Signals					
P43	Intr[(C_NUM_INTR_INPUTS-1) : 0][1]	INTC	I	-	Interrupt inputs
P44	Irq	INTC	0	0	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

XPS INTC Design Parameters

To allow the user to create a XPS INTC that is uniquely tailored for the user's system, certain features are parameterizable in the XPS INTC design. This allows the user to have a design that utilizes only the resources required by the system and runs at the best possible performance. The features that are parameterizable in the XPS INTC core are as shown in Table 2.

Table 2: **XPS INTC Design Parameters**

Gen eric	Feature / Description	Parameter Name	Allowable Values	Default Value	VHDL Type
		System Parameter			
G1	Target FPGA family	C_FAMILY	spartan3, spartan3e, spartan3a, spartan3adsp, aspartan3, aspartan3e, aspartan3a, aspartan3adsp,spartan6, virtex4, qvirtex4, qvvirtex4, virtex5, virtex5fx, virtex6, virtex6cx		string
		PLB Parameters			
G2	XPS INTC Base Address	C_BASEADDR[1]	Valid Address[2]	None[3]	std_logic _vector
G3	XPS INTC High Address	C_HIGHADDR[1]	Valid Address[4]	None[3]	std_logic _vector
G4	PLB address width	C_SPLB_AWIDTH	32	32	integer
G5	PLB data width	C_SPLB_DWIDTH	32, 64, 128	32	integer
G6	Selects point-to-point or shared Bus topology	C_SPLB_P2P	0 = Shared Bus Topology[5]	0	integer
G7	Number of PLB Masters	C_SPLB_NUM_ MASTERS	1 - 16	1	integer
G8	PLB Master ID Bus Width	C_SPLB_MID_ WIDTH	$\log_2$(C_SPLB_NUM_ MASTERS) with a minimum value of 1	1	integer
G9	Width of the Slave Data Bus	C_SPLB_NATIVE_ DWIDTH	32	32	integer
G10	Burst support	C_SPLB_ SUPPORT_ BURSTS	0[6]	0	integer
		INTC Parameters			
G11	Number of interrupt inputs	C_NUM_INTR_ INPUTS	1 - C_SPLB_NATIVE_ DWIDTH	2	integer

Table 2: **XPS INTC Design Parameters** *(Contd)*

Gen eric	Feature / Description	Parameter Name	Allowable Values	Default Value	VHDL Type
G12	Type of interrupt for each input 1 = Edge 0 = Level	C_KIND_OF_INTR	See[7]	ALL 1's	std_logic_vector
G13	Type of each edge sensitive input 1 = Rising 0 = Falling Valid if C_KIND_OF_INTR = 1's	C_KIND_OF_EDGE	See[7]	ALL 1's	std_logic_vector
G14	Type of each level sensitive input 1 = High 0 = Low Valid if C_KIND_OF_INTR = 0's	C_KIND_OF_LVL	See[7]	ALL 1's	std_logic_vector
G15	Indicates the presence of IPR	C_HAS_IPR	0 = Not Present 1 = Present	1	integer
G16	Indicates the presence of SIE	C_HAS_SIE	0 = Not Present 1 = Present	1	integer
G17	Indicates the presence of CIE	C_HAS_CIE	0 = Not Present 1 = Present	1	integer
G18	Indicates the presence of IVR	C_HAS_IVR	0 = Not Present 1 = Present	1	integer
G19	Indicates level or edge active Irq	C_IRQ_IS_LEVEL	0 = Active Edge 1 = Active Level	1	integer
G20	Indicates the sense of the Irq output	C_IRQ_ACTIVE	0 = Falling / Low 1 = Rising / High	1	std_logic

Notes:

1. C_BASEADDR and C_HIGHADDR give the first and last addresses, respectively, of the address range assigned to the XPS INTC. The size in bytes, C_HIGHADDR - C_BASEADDR + 1, and the alignment of the address range must be a power of two (to allow simple decoding) and greater than or equal to 32 (to accommodate the eight 32-bit XPS INTC registers)

2. C_BASEADDR must be aligned to the address range size. In other words, it must be a multiple of the address range size

3. No default value is specified to ensure 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_HIGHADDR = 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 XPS 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_HIGHADDR = 0x7FFFFFFF
will reduce the address decoding logic for an XPS 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. Point-to-point bus topology is not allowed in this version of XPS INTC

6. Burst is not supported for this version of XPS INTC

7. A little-endian vector of width same as the data bus containing a 0 or 1 in each position corresponding to an interrupt input

XPS INTC Parameter - Port Dependencies

The dependencies between the XPS 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: **XPS INTC Parameter-Port Dependencies**

Generic or port	Name	Affects	Depends	Relationship Description
Design Parameters				
G5	C_SPLB_DWIDTH	P7, P10, P33	-	Affects number of bits in data bus
G7	C_SPLB_NUM_MASTERS	P36, P37, P38, P42, G8	-	Affects the width of busy and error signals
G8	C_SPLB_MID_WIDTH	P5	G7	Affects the width of current master identifier signals and depends on $\log_2$(C_SPLB_NUM_MASTERS) with a minimum value of 1
G9	C_SPLB_NATIVE_DWIDTH	G11	-	C_NUM_INTR_INPUTS <= C_SPLB_NATIVE_DWIDTH
G11	C_NUM_INTR_INPUTS	P43	G9	Affects number of bits in Intr. C_NUM_INTR_INPUTS <= C_SPLB_NATIVE_DWIDTH
I/O Signals				
P5	PLB_masterID[0 : C_SPLB_NUM_MASTERS - 1]	-	G7	Width varies with the parameter C_SPLB_NUM_MASTERS
P7	PLB_BE[0 : (C_SPLB_DWIDTH/8) - 1]	-	G5	Width varies with the parameter C_SPLB_DWIDTH
P10	PLB_wrDBus [0 : C_SPLB_DWIDTH - 1]	-	G5	Width varies with the parameter C_SPLB_DWIDTH
P33	Sl_rdDBus[0 : C_SPLB_DWIDTH - 1]	-	G5	Width varies with the parameter C_SPLB_DWIDTH
P36	Sl_MBusy[0 : C_SPLB_NUM_MASTERS - 1]	-	G7	Width varies with the parameter C_SPLB_NUM_MASTERS
P37	Sl_MWrErr[0 : C_SPLB_NUM_MASTERS - 1]	-	G7	Width varies with the parameter C_SPLB_NUM_MASTERS
P38	Sl_MRdErr[0 : C_SPLB_NUM_MASTERS - 1]	-	G7	Width varies with the parameter C_SPLB_NUM_MASTERS
P42	Sl_MIRQ[0 : C_SPLB_NUM_MASTERS - 1]	-	G7	Width varies with the parameter C_SPLB_NUM_MASTERS
P43	Intr[(C_NUM_INTR_INPUTS-1) : 0]	-	G11	Width varies with the parameter C_NUM_INTR_INPUTS

Programming Model

Register Data Types and Organization

All XPS INTC registers are accessed through the PLB 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 PLB addresses are byte addresses, XPS 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 XPS INTC registers are read as big-endian data.

Registers

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:

- Write of a read-only register has no affect

- Read of a write-only register returns zero

- All registers are defined for 32-bit access only; any partial-word accesses (byte, halfword) have undefined results and return a bus error

- Unless stated otherwise, any register bits that are not mapped to inputs return zero when read and do nothing when written

- All registers use C_NUM_INTR_INPUTS except MER which is always 2-bit wide and IVR which is always 32-bit wide

Table 4: **XPS INTC Registers and Base Address Offsets**

| Register Name	Base Address + Offset (Hex)	Access Type	Abbreviation	Reset Value
Interrupt Status Register	C_BASEADDR + 0x0	Read / Write	ISR	All Zeros
Interrupt Pending Register	C_BASEADDR + 0x4	Read	IPR	All Zeros
Interrupt Enable Register	C_BASEADDR + 0x8	Read / Write	IER	All Zeros
Interrupt Acknowledge Register	C_BASEADDR + 0xC	Write	IAR	All Zeros
Set Interrupt Enable Bits	C_BASEADDR + 0x10	Write	SIE	All Zeros
Clear Interrupt Enable Bits	C_BASEADDR + 0x14	Write	CIE	All Zeros
Interrupt Vector Register	C_BASEADDR + 0x18	Read	IVR	All Ones
Master Enable Register	C_BASEADDR + 0x1C	Read / Write	MER	All Zeros

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 correspond 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)" on page 12 for interrupt status bits that is masked by disabled interrupts.

The ISR register is writable by software until the Hardware Interrupt Enable (HIE) bit in the MER has been set. Once 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 interrupts for 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 zero. The Interrupt Status Register (ISR) is shown in Figure 3 and the bits are described in Table 5.
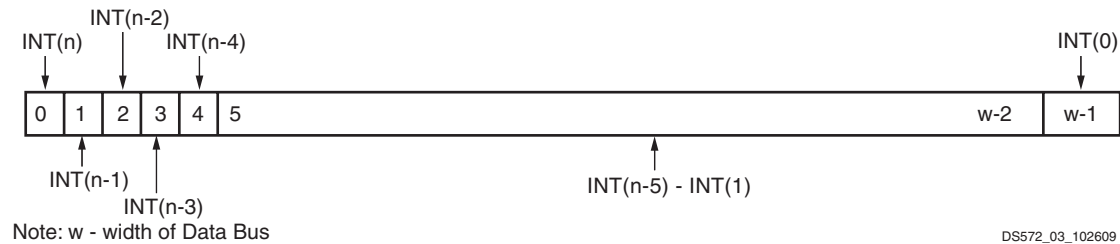
Figure 3: **Interrupt Status Register (ISR)**

Table 5: **Interrupt Status Register**

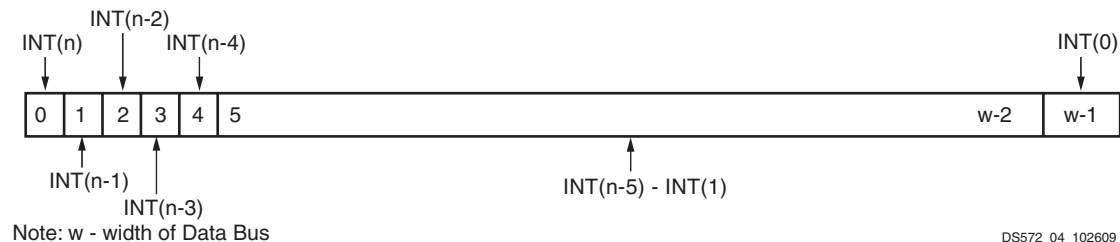| Bits	Name	Core Access	Reset Value	Description
$0 : (w^{(1)} - 1)$	INT(n) – INT(0) $(n \leq w - 1)$	Read / Write	All Zeros	**Interrupt Input (n) – Interrupt Input (0)** '0' = Not active '1' = Active

Notes:
1. w - Width of Data Bus

Interrupt Pending Register (IPR)

This is an optional read only register in the XPS 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.

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 (IPR)**

Table 6: **Interrupt Pending Register**

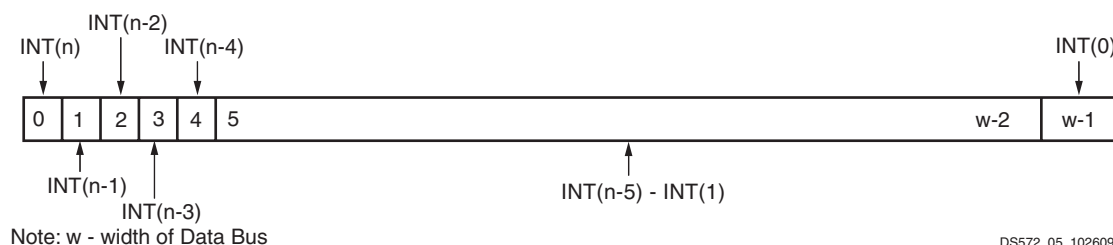| Bits	Name	Core Access	Reset Value	Description
$0 : (w^{(1)} - 1)$	$INT(n) - INT(0)$ $(n \leq w - 1)$	Read	All Zeros	**Interrupt Input (n) – Interrupt Input (0)** '0' = Not active '1' = Active

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 from 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 one indicates the input is enabled and a zero 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 zero. The Interrupt Enable Register (IER) is shown in Figure 5 and the bits are described in Table 7.

Note: w - width of Data Bus

DS572_05_102609

Figure 5: **Interrupt Enable Register (IER)**

Table 7: **Interrupt Enable Register**

Bits	Name	Core Access	Reset Value	Description
$0 : (w^{(1)} - 1)$	$INT(n) - INT(0)$ $(n \leq w - 1)$	Read / Write	All Zeros	**Interrupt Input (n) – Interrupt Input (0)** '1' = Interrupt enabled '0' = Interrupt disabled

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 one 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 zeros does nothing as does writing a one to a bit that does not correspond to an active input or for which an interrupt input does not exist. The Interrupt Acknowledge Register (IAR) is shown in Figure 6 and the bits are described in Table 8.
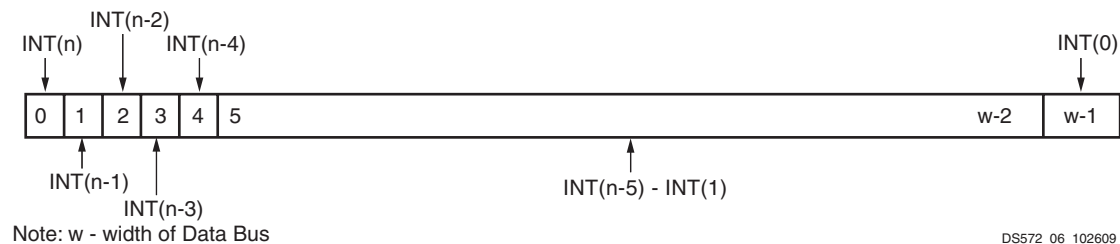

Note: w - width of Data Bus

DS572_06_102609

Figure 6: **Interrupt Acknowledge Register (IAR)**

Table 8: **Interrupt Acknowledge Register**

Bits	Name	Core Access	Reset Value	Description
$0 : (w^{(1)} - 1)$	INT(n) – INT(0) $(n \leq w - 1)$	Write	All Zeros	**Interrupt Input (n) – Interrupt Input (0)** '1' = Clear Interrupt '0' = No action

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 one to a bit location in SIE will set the corresponding bit in the IER.

Writing zeros does nothing, as does writing a one to a bit location that corresponds to a non-existing interrupt input. The SIE is optional in the XPS INTC and can be parameterized out by setting C_HAS_SIE = 0.

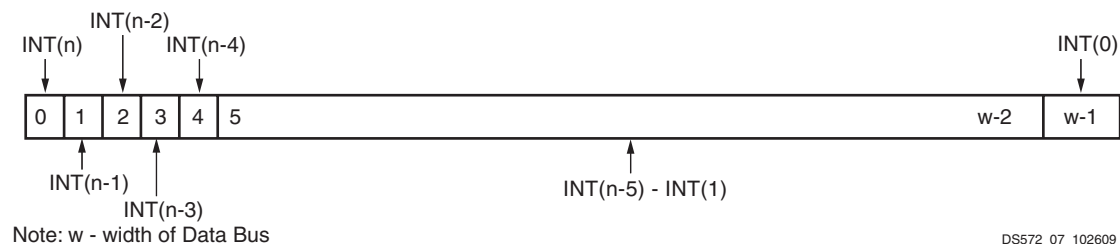The Set Interrupt Enables (SIE) register is shown in Figure 7 and the bits are described in Table 9.

Note: w - width of Data Bus

DS572_07_102609

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

Product Specification

Table 9: **Set Interrupt Enables**

Bits	Name	Core Access	Reset Value	Description
$0 : (w^{(1)} - 1)$	$INT(n) - INT(0)$ $(n \le w - 1)$	Write	All Zeros	**Interrupt Input (n) – Interrupt Input (0)** '1' = Set IER bit '0' = No action

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 one to a bit location in CIE will clear the corresponding bit in the IER.

Writing zeros does nothing, as does writing a one to a bit location that corresponds to a non-existing interrupt input. The CIE is also optional in the XPS 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.
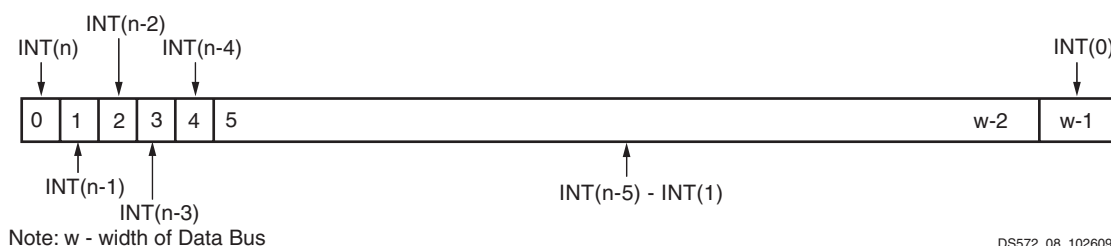

Figure 8: **Clear Interrupt Enables (CIE) Register**

Table 10: **Clear Interrupt Enables**

Bits	Name	Core Access	Reset Value	Description
$0 : (w^{(1)} - 1)$	$INT(n) - INT(0)$ $(n \le w - 1)$	Write	All Zeros	**Interrupt Input (n) – Interrupt Input (0)** '1' = Clear IER bit '0' = No action

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, 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 ones. The IVR is optional in the XPS 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.
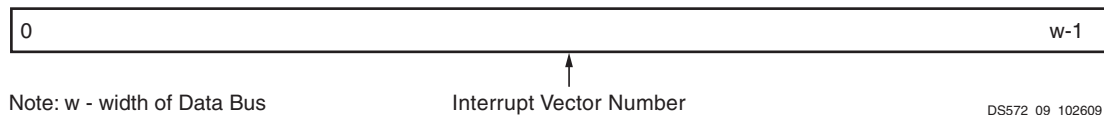
0	w-1

Note: w - width of Data Bus   Interrupt Vector Number   DS572_09_102609

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

Table 11: **Interrupt Vector Register**

Bits	Name	Core Access	Reset Value	Description
$0 : (w^{(1)} - 1)$	Interrupt Vector Number	Read	All Ones	Ordinal of highest priority, enabled, active interrupt input

Notes:
1. w - Width of Data Bus

Master Enable Register (MER)

This is a two 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 zero, allowing software to write to the ISR to generate interrupts for testing purposes, and disabling any hardware interrupt inputs.

Writing a one to this bit enables the hardware interrupt inputs and disables software generated inputs. Writing a one also disables any further changes to this bit until the device has been reset.

Writing ones or zeros 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 zeros. The Master Enable Register (MER) is shown in Figure 10 and is described in Table 12.
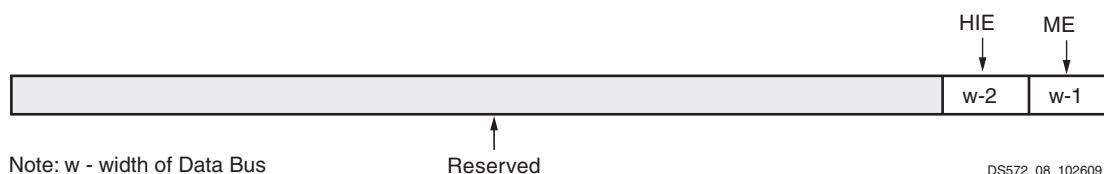
		HIE	ME
		w-2	w-1

Note: w - width of Data Bus   Reserved   DS572_08_102609

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

Table 12: **Master Enable Register**

Bits	Name	Core Access	Reset Value	Description
0 : (w[1] – 3)	Reserved	N/A	All Zeros	**Reserved**

(w[1] – 2)	HIE	Read / Write	'0'	Hardware Interrupt Enable
				'0' = Read – SW interrupts enabled
				      Write – No effect
				'1' = Read – HW interrupts enabled
				      Write – Enable HW interrupts

(w[1] – 1)	ME	Read / Write	'0'	Master IRQ Enable '0' = IRQ disabled – All interrupts disabled '1' = IRQ enabled – All interrupts enabled

Notes:
1. w - Width of Data Bus

Programming XPS INTC

This section provides an overview of software initialization and communication with an XPS INTC.

The number of interrupt inputs that a XPS 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 the LSB of the registers (except IVR and MER).

A valid interrupt input signal is any signal that provides the correct polarity and type of interrupt input. Examples of valid interrupt inputs are rising edges, falling edges, high levels, and low levels (hardware interrupts), or software interrupts if HIE has not been set. The polarity and type of each hardware interrupt input is specified in the XPS INTC generics C_KIND_OF_INTR, C_KIND_OF_EDGE, and C_KIND_OF_LVL (see Table 2).

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 XPS 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 collectively by a master interrupt enable bit.

During power-up or reset, the XPS INTC is put into a state where all interrupt inputs and the interrupt request output are disabled. In order for the XPS 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 a one. This allows the XPS 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 XPS 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 zero. Software testing can now proceed by writing a one 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. Once software testing has been completed, or if software testing is not performed, a one is written to the HIE bit, which enables the hardware interrupt inputs and disables any further software generated interrupts.

5. After a one 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 three. If Int0 becomes active then reading the IVR provides a value of zero.

If no interrupts are active or it is not present, reading the IVR returns all ones.

Acknowledging an interrupt is achieved by writing a one to the corresponding bit location in the IAR. Note that disabling an interrupt by masking it (writing a zero 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 zero to the ME bit in the MER. This effectively masks all interrupts for that XPS INTC. Alternatively, interrupt inputs can be selectively masked by writing a zero 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) an interrupt input without having to read, mask off, and then write back the IER. Writing a one to any bit location(s) in the SIE sets the corresponding bit(s) in the IER without affecting any other IER bits.

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

Design Implementation

The target technology is an FPGA listed in the Supported Device Family field of the LogiCORE IP Facts table.

Device Utilization and Performance Benchmarks

Core Performance

Since the XPS INTC core will be used with other design modules in the FPGA, the utilization and timing numbers reported in this section are estimates only, and will vary from the results reported here.

The XPS INTC resource utilization for various parameter combinations measured with Virtex-4 FPGA as the target device is detailed in Table 13.

Table 13: **FPGA Performance and Resource Utilization Benchmarks for the Virtex-4 FPGA (xc4vfx60-ff1152-11)**

Parameter Values					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	79	56	91	159
1	1	0	0	0	81	57	92	154
1	0	1	0	0	75	57	92	156
1	0	0	1	0	78	57	92	155
1	0	0	0	1	66	57	94	183
1	1	1	1	1	87	60	99	150
2	1	1	1	1	94	69	107	161
2	0	0	0	0	79	61	101	199
4	1	1	1	1	122	84	128	196
4	0	0	0	0	105	69	111	168
8	1	1	1	1	175	113	173	198
8	0	0	0	0	133	85	134	189
16	1	1	1	1	263	170	241	162
16	0	0	0	0	177	117	177	181
32	1	1	1	1	464	282	393	131
32	0	0	0	0	308	180	252	159

Notes:
1. Above numbers are reported by tool for clock of period constraint of 10ns

The XPS INTC resource utilization for various parameter combinations measured with Virtex-5 FPGA as the target device is detailed in Table 14.

Table 14: **FPGA Performance and Resource Utilization Benchmarks the Virtex-5 FPGA (xc5vfx70t-1-ff1136**

Parameter Values					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	67	60	72	172
1	1	0	0	0	75	62	73	170
1	0	1	0	0	66	61	73	195
1	0	0	1	0	71	61	73	173
1	0	0	0	1	67	61	73	167
1	1	1	1	1	79	65	76	172
2	1	1	1	1	87	77	85	152
2	0	0	0	0	76	68	76	170
4	1	1	1	1	105	100	104	172
4	0	0	0	0	94	84	85	179
8	1	1	1	1	169	145	138	167
8	0	0	0	0	133	117	104	164
16	1	1	1	1	284	234	215	184
16	0	0	0	0	211	181	140	189
32	1	1	1	1	460	410	372	156

Notes:
1. Above numbers are reported by tool for clock of period constraint of 6.6ns

The XPS INTC resource utilization for various parameter combinations measured with Spartan-3E as the target device is detailed in Table 18.

Table 15: **FPGA Performance and Resource Utilization Benchmarks the Spartan-3 FPGA (xc3s200-4-ft256)**

Parameter Values					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	71	56	78	106
1	1	0	0	0	65	57	79	110
1	0	1	0	0	72	57	79	107
1	0	0	1	0	71	57	79	107
1	0	0	0	1	71	57	78	109
1	1	1	1	1	69	60	82	114
2	1	1	1	1	83	68	89	100
2	0	0	0	0	76	60	80	106
4	1	1	1	1	101	83	108	126
4	0	0	0	0	90	68	94	102
8	1	1	1	1	145	112	149	108
8	0	0	0	0	112	84	117	102
16	1	1	1	1	230	169	240	108
16	0	0	0	0	142	116	158	105
32	1	1	1	1	389	281	391	100
32	0	0	0	0	251	179	245	100

Notes:
1. Above numbers are reported by tool for clock of period constraint of 10ns

The XPS INTC resource utilization for various parameter combinations measured with Virtex-6 lxt as the target device is detailed in Table 16.

Table 16: **FPGA Performance and Resource Utilization Benchmarks the Virtex-6 FPGA (xc6vlx130t-1-ff1156)**

Parameter Values					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	43	63	85	199
1	1	0	0	0	49	64	86	224
1	0	1	0	0	50	64	85	184
1	0	0	1	0	42	64	87	172
1	0	0	0	1	43	64	84	199
1	1	1	1	1	47	67	91	165
2	1	1	1	1	55	79	105	185
2	0	0	0	0	47	71	90	163
4	1	1	1	1	65	102	125	166
4	0	0	0	0	60	87	104	199
8	1	1	1	1	95	147	187	182
8	0	0	0	0	82	119	135	169
16	1	1	1	1	159	236	286	178
16	0	0	0	0	127	183	187	182
32	1	1	1	1	280	412	499	162

Notes:
1. Above numbers are reported by tool for clock of period constraint of 6.2ns

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

Table 17: **FPGA Performance and Resource Utilization Benchmarks the Virtex-6 cxt FPGA (xc6vcx75t-1-ff484)**

Parameter Values					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	49	63	85	174
1	1	0	0	0	50	64	86	178
1	0	1	0	0	49	64	85	169
1	0	0	1	0	40	64	87	192
1	0	0	0	1	40	64	84	165
1	1	1	1	1	48	67	92	184
2	1	1	1	1	54	79	105	168
2	0	0	0	0	48	71	91	166
4	1	1	1	1	65	102	125	179
4	0	0	0	0	60	87	104	162
8	1	1	1	1	101	147	187	177
8	0	0	0	0	74	119	137	187
16	1	1	1	1	158	236	288	174
16	0	0	0	0	127	183	189	181
32	1	1	1	1	291	412	495	164

Notes:
1. Above numbers are reported by tool for clock of period constraint of 6.2ns

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

Table 18: **FPGA Performance and Resource Utilization Benchmarks the Spartan-6 FPGA (xc6slx45t-2-fgg484)**

Parameter Values					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	55	63	80	102
1	1	0	0	0	50	64	79	100
1	0	1	0	0	51	64	79	102
1	0	0	1	0	50	64	80	101
1	0	0	0	1	55	64	80	101
1	1	1	1	1	57	67	84	102
2	1	1	1	1	60	79	96	100
2	0	0	0	0	52	71	83	100
4	1	1	1	1	75	102	117	100
4	0	0	0	0	60	87	95	170
8	1	1	1	1	139	147	171	100
8	0	0	0	0	114	119	129	100
16	1	1	1	1	211	236	268	191
16	0	0	0	0	117	183	182	174
32	1	1	1	1	422	412	463	102
32	0	0	0	0	325	307	224	101

Notes:
1. Above numbers are reported by tool for clock of period constraint of 10ns

System Performance

To measure the system performance (F_{MAX}) of this core, this core was added to a Virtex-4 FPGA system, a Virtex-5 FPGA system, and a Spartan-3A FPGA system as the Device Under Test (DUT) as shown in Figure 11, Figure 12, and Figure 13.

Because the XPS 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 utilization of FPGA resources and timing of the core design will vary from the results reported here.
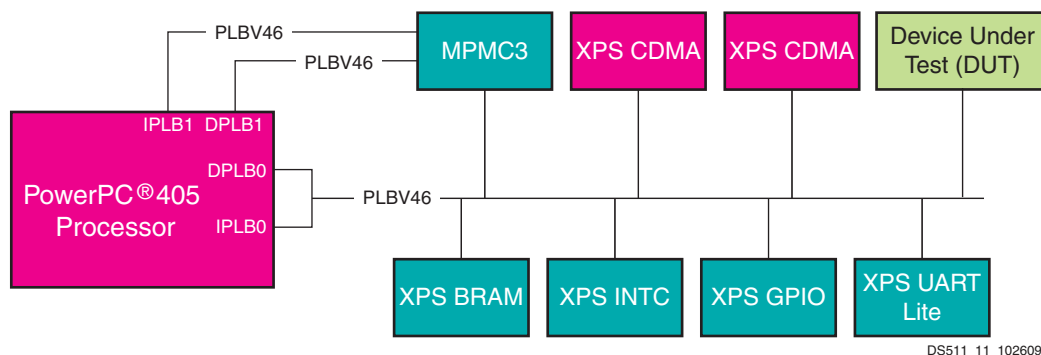

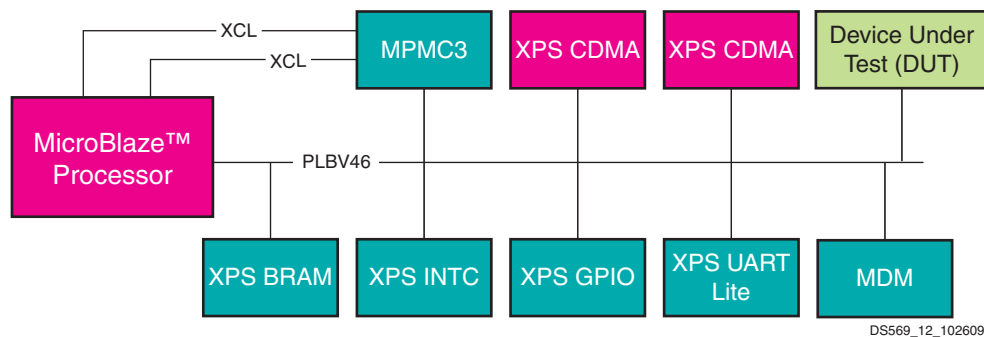*Figure 11:* **Virtex-4 FX FPGA System with the XPS INTC as the DUT**

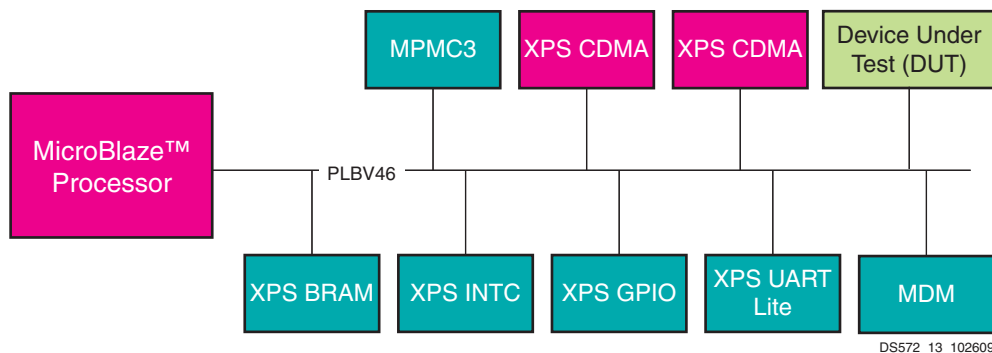*Figure 12:* **Virtex-5 LX FPGA System with the XPS INTC as the DUT**

Figure 13: **Spartan-3A FPGA System with the XPS INTC as the DUT**

The target FPGA was then filled with logic to drive the LUT and BRAM utilization to approximately 70% and the I/O utilization 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 19.

Table 19: **XPS Interrupt Controller System Performance**

Target FPGA	Target F_{MAX} (MHz)
S3A700 -4	90
V4FX60 -10	100
V5LXT50 -1	120

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

Specification Exceptions

N/A

Support

Xilinx provides technical support for this LogiCORE 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.*

Reference Documents

The following documents contain reference information important to understand the XPS INTC design.

1.   IBM CoreConnect 128-Bit Processor Local Bus: Architecture Specifications version 4.6

Notice of Disclaimer

Xilinx is providing this design, code, or information (collectively, the "Information") to you "**AS-IS**" with no warranty of any kind, express or implied. Xilinx makes no representation that the Information, or any particular implementation thereof, is free from any claims of infringement. You are responsible for obtaining any rights you may require for any implementation based on the Information. All specifications are subject to change without notice. XILINX EXPRESSLY DISCLAIMS ANY WARRANTY WHATSOEVER WITH RESPECT TO THE ADEQUACY OF THE INFORMATION OR ANY IMPLEMENTATION BASED THEREON, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OR REPRESENTATIONS THAT THIS IMPLEMENTATION IS FREE FROM CLAIMS OF INFRINGEMENT AND ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Except as stated herein, none of the Information may be copied, reproduced, distributed, republished, downloaded, displayed, posted, or transmitted in any form or by any means including, but not limited to, electronic, mechanical, photocopying, recording, or otherwise, without the prior written consent of Xilinx.

Revision History

Date	Version	Revision
4/18/07	1.0	Initial release.
4/20/07	1.1	Added Spartan-3 support.
6/29/07	1.2	Corrected features list - as design does not support edge generation for output signal Irq.
8/13/07	1.3	Register IVR is changed to 32-bit to fix CR #445886.
9/26/07	1.4	Added FMax Margin System Performance section.
11/27/07	1.5	Added SP-3A DSP support.
1/14/08	1.6	Added Virtex-II Pro support.
4/16/08	1.7	Added Automotive Spartan-3, Automotive Spartan-3E, Automotive Spartan-3A, and Automotive Spartan-3A DSP support.
7/22/08	1.8	Added QPro Virtex-4 Hi-Rel and QPro Virtex-4 Rad Tolerant FPGA support.
9/18/08	2.0	Changed to version V2_00_a; added parameter C_IRQ_IS_LEVEL; removed support for Virtex-2p
9/30/08	2.1	Updated for review comments
10/16/08	2.2	Updated resource utilization numbers
4/24/09	2.3	Replaced references to supported device families and tool name(s) with hyperlink to PDF file.
7/20/09	2.4	Added resource utilization table for Virtex-6 lxt, Virtex-6 cxt, Spartan-6.
12/2/09	2.5	Listed supported devices families and design tool requirements in LogiCORE Facts Table; updated images.