IP CORE MANUAL

NAVIGATOR FPGA Design Kit

Board Information Registers IP

px_brd_info_regs

PENTEK

Pentek, Inc.
One Park Way
Upper Saddle River, NJ 07458
(201) 818–5900
http://www.pentek.com/

Manual Part Number: 807.48322 Rev: 1.1 – October 23, 2017

Manual Revision History

Date	Version	Comments
12/09/16	1.0	Initial Release
10/23/17	1.1	Revised Table 4–4

Printed in the United States of America.

Table of Contents

Table of Contents

Page

Table of Contents

Page

Table of Contents

Page

This page is intentionally blank

IP Facts

Description

Pentek's Navigator™ Board Information Reg–isters Core provides byte–swap control, LED drive control, board status information, and system–level interupt generation .

This core complies with the ARM® AMBA® AXI4 Specification and also provides a con–trol/status register interface. This user manual defines the hardware interface, software inter–face, and parameterization options for the Board Information Registers Core.

Features

- Register access through AXI4–Lite interface
- Software programmable board information and code revision information
- An interrupt output can be generated, based on the user requirement, from various system level interrupt sources
- LED drive control for user LED, and system monitor LED
- Combines interrupt inputs to generate one interrupt output
- Provides byte swap control
- Seven status registers that provide board information as well as PCIe link status information

Table 1–1: IP Facts Table

Core Specifics	
Supported Design Family[a]	Kintex® Ultrascale
Supported User Interfaces	AXI4–Lite
Resources	See Table 2–1
Provided with the Core	
Design Files	VHDL
Example Design	Not Provided
Test Bench	VHDL
Constraints File	Not Provided[b]
Simulation Model	VHDL
Supported S/W Driver	HAL Software Support
Tested Design Flows	
Design Entry	Vivado® Design Suite 2017.2 or later
Simulation	Vivado VSim
Synthesis	Vivado Synthesis
Support	
Provided by Pentek fpgasupport@pentek.com	

a.For a complete list of supported devices, see the *Vivado Design Suite Release Notes.*
b.Clock constraints can be applied at the top level module of the user design.

This page is intentionally blank

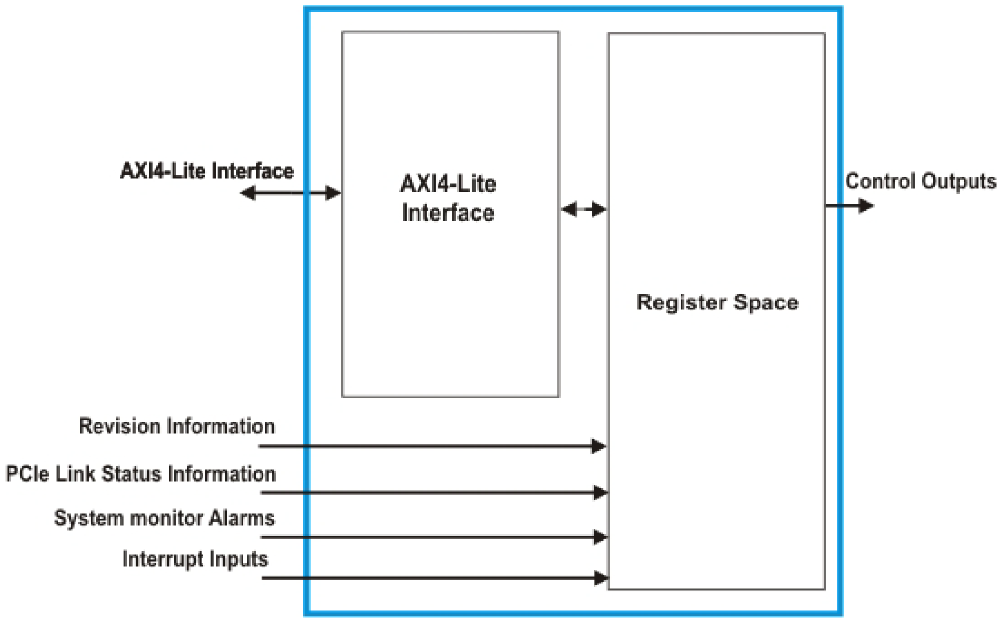
Chapter 1: Overview

1.1 Functional Description

The Board Information Registers core provides a transaction interface to the control/status registers through an AXI4–Lite Interface. The AXI4–Lite Interface acts as a slave and is connected to the Register Space as shown in Figure 1–1, below. This core provides one byte swap control register, and seven board information status registers. This core also provides LED drive control registers, and an interrupt output based on the generic parameters defined by the user. For more details on generic parameters of the Board Information Registers core, refer to Section 2.5.

Figure 1–1 is a top–level block diagram of the Pentek Board Information Registers Core. The modules within the block diagram are explained in the later sections of this manual.

Figure 1–1: Board Information Registers Core Block Diagram



1.1 Functional Description (continued)

- ❑ **AXI4–Lite Interface:** This module implements a 32–bit AXI4–Lite Slave interface to access the Register Space. For additional details about the AXI4–Lite Interface, refer to Section 3.1 AXI4–Lite Core Interfaces.

- ❑ **Register Space:** This module contains control and status registers, including Interrupt Enable, Interrupt Status and Interrupt Flag registers. Registers are accessed through the AXI4–Lite interface.

1.2 Applications

The Board Information Registers Core can be used as an interface to access control/status registers which carry board information such as board ID and code revision information, and can be used to control byte swap and LED drives (user LED and system monitor LED).

1.3 System Requirements

For a list of system requirements, see the Vivado Design Suite Release Notes.

1.4 Licensing and Ordering Information

This core is included with all Pentek Navigator FPGA Design Kits for Pentek Jade series board products. Contact Pentek for Licensing and Ordering Information (www.pentek.com).

1.5 Contacting Technical Support

Technical Support for Pentek's Navigator FPGA Design Kits is available via e–mail (fpgasupport@pentek.com) or by phone (201–818–5900 ext. 238, 9 am to 5 pm EST).

1.6 Documentation

This user manual is the main document for this IP core. The following documents provide supplemental material:

1) *Vivado Design Suite User Guide: Designing with IP*

2) *Vivado Design Suite User Guide: Programming and Debugging*

3) *ARM AMBA AXI4 Protocol Version 2.0 Specification*
http://www.arm.com/products/system–ip/amba–specifications.php

4) *Xilinx Ultrascale Devices Gen3 Intergrated Block for PCI Express Core*

Chapter 2: General Product Specifications

2.1 Standards

The Board Information Registers Core has bus a interface that complies with the *ARM AMBA AXI4–Lite Protocol Specification*.

2.2 Performance

The performance of the Board Information Registers Core is limited only by the FPGA logic speed. The values presented in this section should be used as an estimation guideline. Actual performance can vary.

2.2.1 Maximum Frequencies

The Board Information Registers Core is designed to meet a target frequency of
250 MHz on a Kintex Ultrascale –2 speed grade FPGA. 250 MHz is typically the PCI Express (PCIe®) AXI bus clock frequency.

2.3 Resource Utilization

The resource utilization of the Board Information Registers Core is shown in Table 2–1. Resources have been estimated for the Kintex Ultrascale XCKU060 –2 speed grade device. These values were generated using the Vivado Design Suite.

Table 2–1: Resource Usage and Availability

Resource	# Used
LUTs	248
Flip–Flops	403

NOTE: Actual utilization may vary based on the user design in which the Board Information Registers Core is incorporated.

2.4 Limitations and Unsupported Features

This section is not applicable to this IP core.

2.5 Generic Parameters

The generic parameters of the Board Information Registers Core are described in Table 2–2. These parameters can be set as required by the user application while customizing the core.

Table 2-2: Generic Parameters

Port/Signal Name	Type	Description
Board Information		
board_id[15:0]	std_logic_vector	**Board ID:** This parameter defines the board type of the Pentek supplied Jade series board the Board Information Registers Core is incorporated into. It supports hexadecimal and binary formats.
fpga_code_id[19:0]		**FPGA Code Type:** This parameters identifies the type of FPGA code within the board. It is the 16–bit board ID followed by a 4–bit code ID. It supports hexadecimal and binary formats.
has_fpga_size_in		**Has FPGA Size Input:** When set to True, it indicates that the core has an FPGA size input.
fpga_size	String	**FPGA Size:** This defines the size of the Kintex Ultrascale FPGA (XCKUxxx) being used. It can be “025”/ “035”/ “040”/ “060”/ “085”/ “095”/ “115”.
has_fpio_in	Boolean	**Has Front Panel Board ID Input:** When set to True, this parameter indicates that the front panel mezzanine card on the board is available to connect to the core and generates a 4–bit ID which is an input to the Board Information Registers core.
user_word[31:0]	std_logic_vector	**User Word Register Value:** This user–defined value is stored in a status register and can be read based on the user application requirement.
Code Revision Date		
has_rev_info_in	Boolean	**Has Revision Information Input:** When set to True, it indicates that the core has revision information input.
LEDs		
has_user_led_out	Boolean	**Has User LED Drive Output:** Indicates if the core has a user LED output.
has_sys_mon_led	Boolean	**Has System Monitor LED Drive Output:** Indicates if there is a system monitor LED output from the core.
Interrupts		
has_irq_out	Boolean	**Has Interrupt Output:** When set to True, this parameter indicates that the core has an interrupt output, generated by combining all the interrupt inputs.

Table 2-2: Generic Parameters (Continued)

Port/Signal Name	Type	Description
	Interrupts (continued)	
has_ext_temp_irq	Boolean	**Has External Temperature Interrupt Input:** This is set to True when the core has external temperature interrupt input.
has_sys_mon_irq	Boolean	**Has System Monitor Interrupt Input:** This is set to True when the core has system monitor interrupt input.
has_i2c1_irq	Boolean	**Has I2C Port #1 Interrupt Input:** When set to True, it indicates an interrupt request input to the core from the Port #1 I2C Bus Interface Core.
has_i2c2_irq	Boolean	**Has I2C Port #2 Interrupt Input:** When set to True, it indicates an interrupt request input to the core from the Port #2 I2C Bus Interface Core.
has_user1_irq	Boolean	**Has User–defined Interrupt #1 Input:** When set to True, it indicates a user–defined interrupt #1 input to the core.
has_user2_irq	Boolean	**Has User–defined Interrupt #2 Input:** When set to True, it indicates a user–defined interrupt #2 input to the core.
has_user3_irq	Boolean	**Has User–defined Interrupt #3 Input:** When set to True, it indicates a user–defined interrupt #3 input to the core.
has_user4_irq	Boolean	**Has User–defined Interrupt #4 Input:** When set to True, it indicates a user–defined interrupt #4 input to the core.
has_user5_irq	Boolean	**Has User–defined Interrupt #5 Input:** When set to True, it indicates a user–defined interrupt #5 input to the core.
has_user6_irq	Boolean	**Has User–defined Interrupt #6 Input:** When set to True, it indicates a user–defined interrupt #6 input to the core.
has_user7_irq	Boolean	**Has User–defined Interrupt #7 Input:** When set to True, it indicates a user–defined interrupt #7 input to the core.
has_user8_irq	Boolean	**Has User–defined Interrupt #8 Input:** When set to True, it indicates a user–defined interrupt #8 input to the core.
fpga_code_rev	Integer	**FPGA Code Revision:** This indicates the FPGA code revision number. It can range from 0 to 255.
date_month	Integer	**Month of the FPGA Code Revision Date:** This defines the month of the code revision date. It can range from 1 to 12.
date_day	Integer	**Day of the FPGA Code Revision date:** This defines the day of the code revision date. It can range from 1 to 31.
date_year	Integer	**Year of the FPGA Code Revision Date:** This defines the year of the code revision date. It can range from 0 to 99.

Table 2-2: Generic Parameters (Continued)

Port/Signal Name	Type	Description
PCIe Link Information		
has_link_stat_in	Boolean	**Has PCIe Link Status Input:** When set to True, it indicates that PCIe link status input is available to the core.

Chapter 3: Port Descriptions

This chapter provides details about the port descriptions for the following interface types:

- AXI4–Lite Core Interfaces

3.1 AXI4–Lite Core Interfaces

The Board Information Registers Core uses the Control/Status Register (CSR) interface to control, and receive status from, the user design.

3.1.1 Control/Status Register (CSR) Interface

The CSR interface is an AXI4–Lite Slave Interface that can be used to access the control and status registers in the Board Information Registers Core. Table 3–1 defines the ports in the CSR interface. See Chapter 4 for a Control/Status Register memory map and bit definitions. See the *AMBA AXI4–Lite Specification* for more details on operation of the AXI4–Lite interfaces.

Table 3-1: Control/Status Register (CSR) Interface Port Descriptions

Port	Direction	Width	Description
s_axi_csr_aclk	Input	1	**Clock**
s_axi_csr_aresetn	Input	1	**Reset:** Active low. This signal will reset all control registers to their initial states.
s_axi_csr_awaddr	Input	6	**Write Address:** Address used for write operations. It must be valid when **s_axi_csr_awvalid** is asserted and must be held until **s_axi_csr_awready** is asserted by the Board Information Registers Core.
s_axi_csr_awprot	Input	3	**Protection:** The Board Information Registers Core ignores these bits.
s_axi_csr_awvalid	Input	1	**Write Address Valid:** This input must be asserted to indicate that a valid write address is available on **s_axi_csr_awaddr**. The Board Information Registers Core asserts **s_axi_csr_awready** when it is ready to accept the address. The **s_axi_csr_awvalid** must remain asserted until the rising clock edge after the assertion of **s_axi_csr_awready**.

Table 3-1: Control/Status Register (CSR) Interface Port Descriptions (Continued)

Port	Direction	Width	Description
s_axi_csr_awready	Output	1	**Write Address Ready:** This output is asserted by the Board Information Registers Core when it is ready to accept the write address.The address is latched when **s_axi_csr_awvalid** and **s_axi_csr_awready** are high on the same cycle.
s_axi_csr_wdata	Input	32	**Write Data:** This data will be written to the address specified by **s_axi_csr_awaddr** when **s_axi_csr_wvalid** and **s_axi_csr _wready** are both asserted. The value must be valid when **s_axi_csr_wvalid** is asserted and held until **s_axi_csr_wready** is also asserted.
s_axi_csr_wstrb	Input	4	**Write Strobes:** This signal, when asserted, indicates the number of bytes of valid data on the **s_axi_csr_wdata** signal. Each of these bits, when asserted, indicate that the corresponding byte of **s_axi_csr_wdata** contains valid data. Bit 0 corresponds to the least significant byte, and bit 3 to the most significant.
s_axi_csr_wvalid	Input	1	**Write Valid:** This signal must be asserted to indicate that the write data is valid for a write operation. The value on **s_axi_csr _wdata** is written into the register at address **s_axi_csr_awaddr** when **s_axi_csr_wready** and **s_axi_csr_wvalid** are high on the same cycle.
s_axi_csr_wready	Output	1	**Write Ready:** This signal is asserted by the Board Information Registers Core when it is ready to accept data. The value on **s_axi_csr _wdata** is written into the register at address **s_axi_csr_awaddr** when **s_axi_csr_wready** and **s_axi_csr_wvalid** are high on the same cycle, assuming that the address has already or simultaneously been submitted.
s_axi_csr_bresp	Output	2	**Write Response:** The Board Information Registers Core indicates success or failure of a write transaction through this signal, which is valid when **s_axi_csr_bvalid** is asserted; 00 = Success of normal access 01 = Success of exclusive access 10 = Slave Error 11 = Decode Error Note: For more details about this signal refer to the *AMBA AXI Specification*.
s_axi_csr_bready	Input	1	**Write Response Ready:** This signal must be asserted by the user logic when it is ready to accept the Write Response.
s_axi_csr_bvalid	Output	1	**Write Response Valid:** This signal is asserted by the Board Information Registers Core when the write operation is complete and the Write Response is valid. It is held until **s_axi_csr_bready** is asserted by the user logic.

Table 3-1: Control/Status Register (CSR) Interface Port Descriptions (Continued)

Port	Direction	Width	Description
s_axi_csr_araddr	Input	6	**Read Address:** Address used for read operations. It must be valid when **s_axi_csr_arvalid** is asserted and must be held until **s_axi_csr_arready** is asserted by the Board Information Registers Core.
s_axi_csr_arprot	Input	3	**Protection:** These bits are ignored by the Board Information Registers Core
s_axi_csr_arvalid	Input	1	**Read Address Valid:** This input must be asserted to indicate that a valid read address is available on the **s_axi_csr_araddr**. The Board Information Registers Core asserts **s_axi_csr_arready** when it ready to accept the Read Address. This input must remain asserted until the rising clock edge after the assertion of **s_axi_csr_arready**.
s_axi_csr_arready	Output	1	**Read Address Ready:** This output is asserted by the Board Information Registers Core when it is ready to accept the read address. The address is latched when **s_axi_csr_arvalid** and **s_axi_csr_ arready** are high on the same cycle.
s_axi_csr_rdata	Output	32	**Read Data:** This value is the data read from the address specified by the **s_axi_csr_araddr** when **s_axi_csr_arvalid** and **s_axi_csr_arready** are high on the same cycle.
s_axi_csr_rresp	Output	2	**Read Response:** The Board Information Registers Core indicates success or failure of a read transaction through this signal, which is valid when **s_axi_csr_rvalid** is asserted; 00 = Success of normal access 01 = Success of exclusive access 10 = Slave Error 11 = Decode Error Note: For more details about this signal refer to the *AMBA AXI Specification*.
s_axi_csr_rvalid	Output	1	**Read Data Valid:** This signal is asserted by the Board Information Registers Core when the read is complete and the read data is available on **s_axi_csr_rdata**. It is held until **s_axi_csr_ rready** is asserted by the user logic.
s_axi_csr_rready	Input	1	**Read Data Ready:** This signal is asserted by the user logic when it is ready to accept the Read Data.
irq	Output	1	**Interrupt:** This is an active high, edge type interrupt output.

3.2 I/O Signals

The I/O port/ signal descriptions of the top level module of the Board Information Registers Core are discussed in the Table 3–2.

Table 3–2: I/O Signals

Port/Signal Name	Type	Direction	Description
Status Inputs			
rev[7:0]	std_logic vector_	Input	**FPGA Code Revision:** This is the FPGA code revision number. It is used when the parameter **has_rev_info_in** is True.
rev_date[23:0]	std_logic vector_	Input	**FPGA Code Revision Date:** This is the FPGA code revision date and is used when the parameter **has_rev_info_in** is True. It has a hexadecimal representation. For example, Date 08/09/2015 = x"080915".
size[11:0]	std_logic vector_	Input	**FPGA Size:** These bits indicate the FPGA size of the Xilinx Ultrascale FPGA being used. This input is used when the parameter **has_fpga_size_in** is set to True. 0x025 = XCKU025 0x035 = XCKU035 0x040 = XCKU040 0x060 = XCKU060 0x085 = XCKU085 0x095 = XCKU095 0x115 = XCKU115
fpio[3:0]	std_logic vector_	Input	**Front Panel IO ID Input:** This is front panel IO ID. This input should be made available to the Board Information Registers core when the generic parameter **has_fpio_in** is set to True.
pcie_link_status _info[31:0]	std_logic vector_	Input	**PCIe Link Status Input:** This is the PCIe link status input which should be made available to the core when the generic parameter **has_link_stat_in** is set to True.
System Monitor Alarms			
ot	std_logic	Input	**Over Temperature Alarm:** Active High. When High, it indicates that the FPGA die temperature has exceeded the OT maximum temperature.
vccint_alarm	std_logic	Input	**V_{INT} Alarm:** When high, this alarm indicates that the FPGA core voltage is outside the desired voltage range.
vccaux_alarm	std_logic	Input	**V_{AUX} Alarm:** When high, this alarm indicates that the FPGA's external auxiliary input voltage is outside the desired voltage range.

Table 3–2: I/O Signals (Continued)

Port/Signal Name	Type	Direction	Description
System Monior Alarms (continued)			
user_temp_ alarm	std_logic	Input	**User Temperature Alarm:** When high, this alarm indicates that the temperature of the device, generating this alarm, in the user application has exceeded the maximum temperature limit.
Interrupt Inputs			
ext_temp_irq_n	std_logic	Input	**External Temperature Interrupt Input:** Active Low. When this input is available to the core, the parameter **has_ext_temp_irq** must be set to True.
sys_mon_irq	std_logic	Input	**System Monitor Interrupt Input:** Active High. When this input is available to the core, the parameter **has_sys_mon_irq** must be set to True.
i2c1_irq	std_logic	Input	**I2C Port #1 Interrupt Input:** When this input is available to the core, the parameter **has_i2c1_irq** must be set to True.
i2c2_irq	std_logic	Input	**I2C Port #2 Interrupt Input:** When this input is available to the core, the parameter **has_i2c2_irq** must be set to True.
user1_irq	std_logic	Input	**User–defined Interrupt #1 Input:** When this input is available to the core, the parameter **has_user1_irq** must be set to True.
user2_irq	std_logic	Input	**User–defined Interrupt #2 Input:** When this input is available to the core, the parameter **has_user2_irq** must be set to True.
user3_irq	std_logic	Input	**User–defined Interrupt #3 Input:** When this input is available to the core, the parameter **has_user3_irq** must be set to True.
user4_irq	std_logic	Input	**User–defined Interrupt #4 Input:** When this input is available to the core, the parameter **has_user4_irq** must be set to True.
user5_irq	std_logic	Input	**User–defined Interrupt #5 Input:** When this input is available to the core, the parameter **has_user5_irq** must be set to True.
user6_irq	std_logic	Input	**User–defined Interrupt #6 Input:** When this input is available to the core, the parameter **has_user6_irq** must be set to True.
user7_irq	std_logic	Input	**User–defined Interrupt #7 Input:** When this input is available to the core, the parameter **has_user7_irq** must be set to True.

Table 3–2: I/O Signals (Continued)

Port/Signal Name	Type	Direction	Description
Interrupt Inputs (continued)			
user8_irq	std_logic	Input	**User–defined Interrupt #8 Input:** When this input is available to the core, the parameter **has_user8_irq** must be set to True.
Control Outputs			
byte_swap	std_logic	Output	**Byte Swap:** This is byte swap control output. Active High.
user_led_n	std_logic	Output	**User LED Drive Output:** Active Low.
sys_mon_led_n	std_logic	Output	**System Monitor LED Drive Output:** Active Low. When this signal is Low, it indicates that either the internal FPGA temperature is over the limit or a voltage is outside the valid range.

NOTE: User–defined interrupts must be positive–edge type.

Chapter 4: Register Space

This chapter provides the memory map and register descriptions for the register space of the Board Information Registers Core. The memory map is provided in Table 4–1.

Table 4–1: Register Space Memory Map

Register Name	Address (Base Address +)	Access	Description
Byte Swap Control	0x00	R/W	Controls swapping of data bytes.
User LED Control	0x04	R/W	Controls user LED output.
System Monitor LED Control	0x08	R/W	Controls system monitor LED output.
Board Status	0x0C	R	Indicates the board type.
FPGA Code Status	0x10	R	Indicates the FPGA code type.
FPGA Revision Status	0x14	R	Indicates FPGA code revision.
FPGA Revision Date Status	0x18	R	Indicates FPGA code revision date.
Front Panel ID Status	0x1C	R	Indicates front panel I/O ID.
User Word Status	0x20	R	Indicates the user word.
PCIe Link Status	0x24	R	Indicates the PCIe link status.
Interrupt Enable Register	0x28	R/W	Interrupt enable bits
Interrupt Status Register	0x2C	R	Interrupt source status bits
Interrupt Flag Register	0x30	R/Clr	Interrupt flag bits

4.1 Byte Swap Control Register

This register controls the byte swap output of the core. The Byte Swap Control Register is illustrated in Figure 4–1 and described in Table 4–2.

Figure 4–1: Byte Swap Control Register

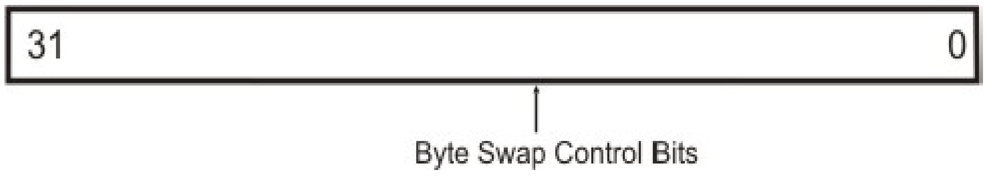


Table 4–2: Byte Swap Control Register (Base Address + 0x00)

Bits	Field Name	Default Value	Access Type	Description
31:0	byte_swap	0x00000000	R/W	**Byte Swap Control Bits:** These bits are used to control the byte swap output of the core. Setting any bit to ‘1’ will enable byte swap i.e., set the byte swap control output to ‘1’. 0x00000000 = Byte swap disabled 0x00000001 – 0xFFFFFFFF = Byte swap enabled

4.2 User LED Control Register

This control register is used to control the user LED output of the Board Information Registers Core. The User LED Control Register is illustrated in Figure 4–2 and described in Table 4–3.

Figure 4–2: User LED Control Register

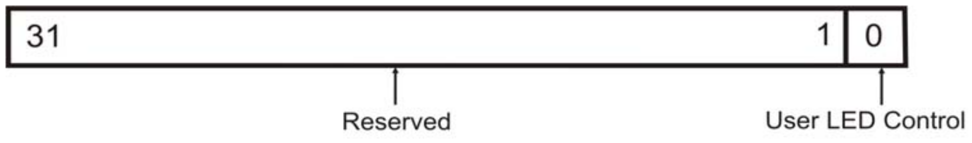


Table 4–3: User LED Control Register (Base Address + 0x04)

Bits	Field Name	Default Value	Access Type	Description
31:1	Reserved	N/A	N/A	**Reserved**
0	user_led	0	R/W	**User LED Control:** This bit controls the user LED output signal of the core. 0 = Off 1 = On

4.3 System Monitor LED Control Register

This register controls the source of the system monitor LED output of the Board Information Registers Core. The System Monitor LED Control Register is illustrated in Figure 4–3 and described in Table 4–4.

Figure 4–3: System Monitor LED Control Register

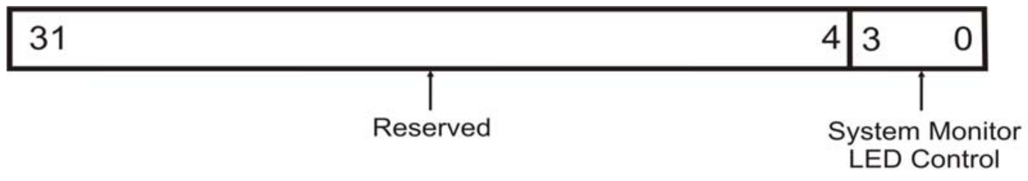


Table 4–4: System Monitor LED Control Register (Base Address + 0x08)				
Bits	**Field Name**	**Default Value**	**Access Type**	**Description**
31:4	Reserved	N/A	N/A	**Reserved**
3:0	ctl2_out	0000	R/W	**Control Bits of the System Monitor LED:** These bits enable (or disable) the system monitor alarms to generate an indication on the system monitor LED. Each alarm is enabled by assigning a '1' to the corresponding control bit. Each of the bits control the following alarms: ctl2_out[3] – FPGA auxiliary supply voltage (VCCAUX) alarm ctl2_out[2] – FPGA core voltage (VCCINT) alarm ctl2_out[1] – User Temperature alarm ctl2_out[0] – Over Temperature alarm

4.4 Board Status Register

This register indicates the board ID information. The Board Status Register is illustrated in
Figure 4–4 and described in Table 4–5.

Figure 4–4: Board Status Register

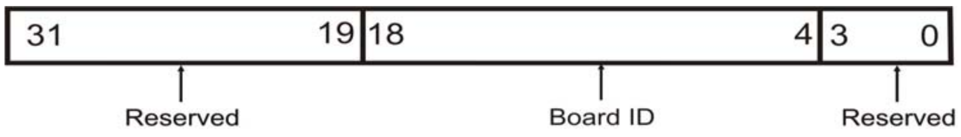


Table 4–5: Board Status Register (Base Address + 0x0C)				
Bits	**Field Name**	**Default Value**	**Access Type**	**Description**
31:19	Reserved	N/A	N/A	**Reserved**
18:4	board_id	0x0000	R	**Board ID:** This value is defined by the user in the generic parameter **board_id**.
3:0	Reserved	N/A	N/A	**Reserved**

4.5 FPGA Code Status Register

This register indicates the FPGA code ID. The FPGA Code Status Register is illustrated in Figure 4–5 and described in Table 4–6.

Figure 4–5: FPGA Code Status Register

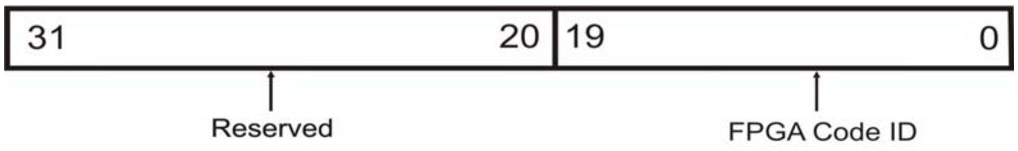


Table 4–6: FPGA Code Status Register (Base Address + 0x10)

Bits	Field Name	Default Value	Access Type	Description
31:19	Reserved	N/A	N/A	**Reserved**
19:0	fpga_code_id	0x00000	R	**FPGA Code ID:** This value is defined by the user in the generic parameter **fpga_code_id**.

4.6 FPGA Revision Status Register

This register indicates the FPGA code revision number. The FPGA Revision Status Reg–ister is illustrated in Figure 4–6 and described in Table 4–7.

Figure 4–6: FPGA Revision Status Register

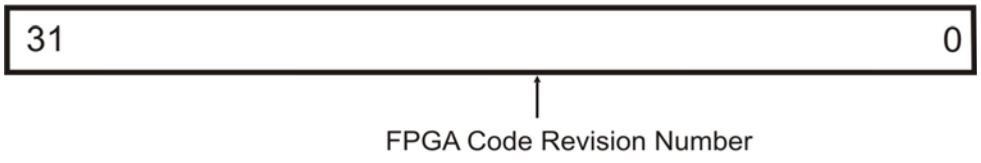


Table 4–7: FPGA Revision Status Register (Base Address + 0x14)

Bits	Field Name	Default Value	Access Type	Description
31:0	rev_num	0x00000000	R	**FPGA Code Revision Number:** This value is taken from either the input revision number or the generic parameter **fpga_code_rev** defined by the user.

4.7 FPGA Revision Date Status Register

This register indicates the FPGA code revision date. The FPGA Revision Date Status Register is illustrated in Figure 4–7 and described in Table 4–8.

Figure 4–7: FPGA Revision Date Status Register

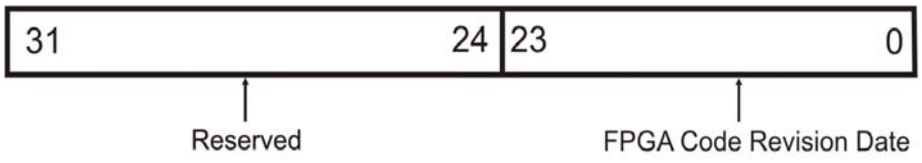


Table 4–8: FPGA Revision Date Status Register (Base Address + 0x18)				
Bits	**Field Name**	**Default Value**	**Access Type**	**Description**
31:24	Reserved	N/A	N/A	**Reserved**
23:0	rev_date	0x000000	R	**FPGA Code Revision Date:** This value is taken from either the input revision date or the generic parameters defined by the user. It has a hexadecimal representation. For example, Date 08/09/2015 = x"080915".

4.8 Front Panel ID Status Register

This register indicates the FPGA size, and front panel I/O ID information. The Front Panel ID Status Register is illustrated in Figure 4–8 and described in Table 4–9.

Figure 4–8: Front Panel ID Status Register

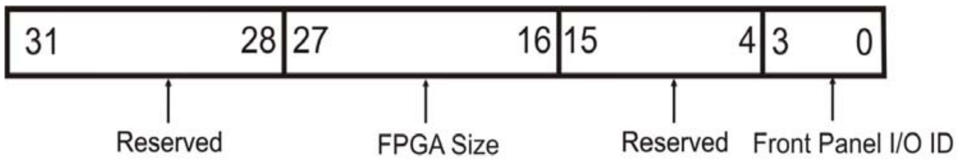


Table 4–9: Front Panel ID Status Register (Base Address + 0x1C)				
Bits	**Field Name**	**Default Value**	**Access Type**	**Description**
31:28	Reserved	N/A	N/A	**Reserved**
27:16	fpga_size _num	0x000	R	**FPGA Size:** This value is taken from either the input FPGA size or the generic parameter **fpga_size** defined by the user. 0x025 = XCKU025 0x035 = XCKU035 0x040 = XCKU040 0x060 = XCKU060 0x085 = XCKU085 0x095 = XCKU095 0x115 = XCKU115
15:4	Reserved	N/A	N/A	**Reserved**
3:0	fpio	0000	R	**Front Panel IO ID:** This value is defined by the input (**fpio**) to the core when the generic parameter **has_fpio_in** is set to True. When **has_fpio_in** is False, these bits are set to zero.

4.9 User Word Status Register

This register indicates the user word defined by the generic parameter of the Board Information Registers Core. The User Word Status Register is illustrated in Figure 4–9 and described in Table 4–10.

Figure 4–9: User Word Status Register

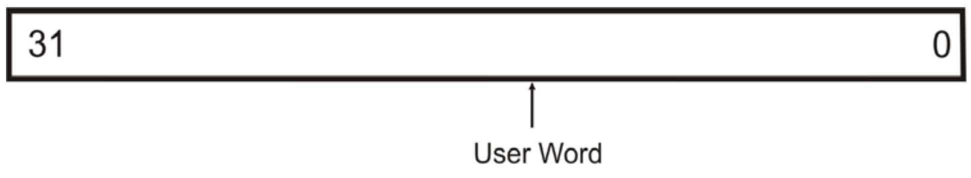


Table 4–10: User Word Status Register (Base Address + 0x20)				
Bits	**Field Name**	**Default Value**	**Access Type**	**Description**
31:0	user_word	0x00000000	R	**User Word:** This is the user word defined by the user in the generic parameter **user_word**.

4.10 PCIe Link Status Register

This register indicates the PCI Express link status information. The PCIe Link Status Register is illustrated in Figure 4–10 and described in Table 4–11.

Figure 4–10: PCIe Link Status Register

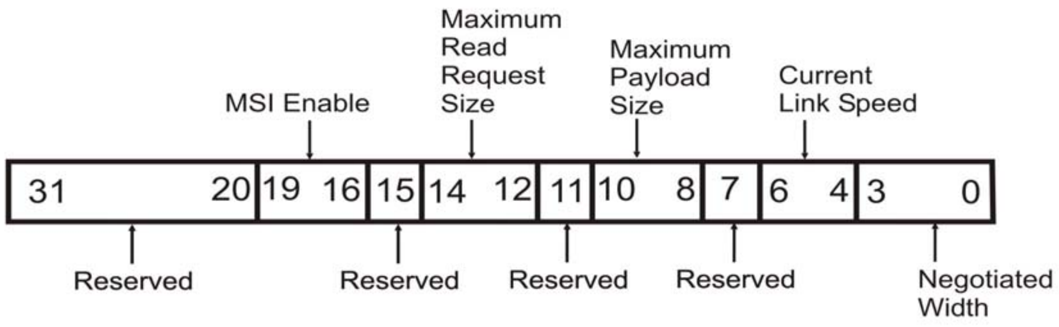


Table 4–11: PCIe Link Status Register (Base Address + 0x24)				
Bits	**Field Name**	**Default Value**	**Access Type**	**Description**
31:20	Reserved	N/A	N/A	**Reserved**
19:16	msi_en	0000	R	**MSI Enable:** These bits indicate whether the Message Signaling Interrupt is enabled per function. Bits [19:18] are reserved. 00 = Legacy Interrupts 01 = Message Signaled Interrupts Other Combinations = undefined
15	Reserved	N/A	N/A	Reserved
14:12	max_rd_rqst	000	R	**Maximum Read Request Size:** These bits indicate the maximum read request size of the PCIe link. 000 = 128 Bytes 001 = 256 Bytes 010 = 512 Bytes 011 = 1024 Bytes 100 = 2048 Bytes 101 = 4096 Bytes Other combinations = undefined
11	Reserved	N/A	N/A	**Reserved**

Table 4–11: PCIe Link Status Register (Base Address + 0x24) (Continued)

Bits	Field Name	Default Value	Access Type	Description
10:8	max_payload	000	R	**Maximum Payload Size:** These bits indicate the maximum read request size of the PCIe link. 000 = 128 Bytes 001 = 256 Bytes 010 = 512 Bytes 011 = 1024 Bytes 100 = 2048 Bytes 101 = 4096 Bytes Other combinations = undefined
7	Reserved	N/A	N/A	**Reserved**
6:4	link_speed	000	R	**Current Link Speed:** These bits indicate the current link speed of the PCI Express Link. 001 = 2.5 GT/s PCI Express Link 010 = 5.0 GT/s PCI Express Link 100 = 8.0 GT/s PCI Express Link Other combinations = undefined
3:0	neg_width	0000	R	**Negotiated Width:** These bits indicate the negotiated width of the PCI Express link. 0000 = x1 0010 = x2 0100 = x4 1000 = x8 Other combinations = undefined

4.11 Interrupt Enable Register

The bits in the interrupt enable register are used to enable (or disable) the generation of interrupts based on the condition of certain circuit elements, known as interrupt sources. When a bit in this register associated with a given interrupt source is High, an interrupt will be generated by the rising edge of that source's Interrupt Status Register bit (See Section 4.12). This register is illustrated in Figure 4–11 and described in Table 4–12.

Figure 4–11: Interrupt Enable Register

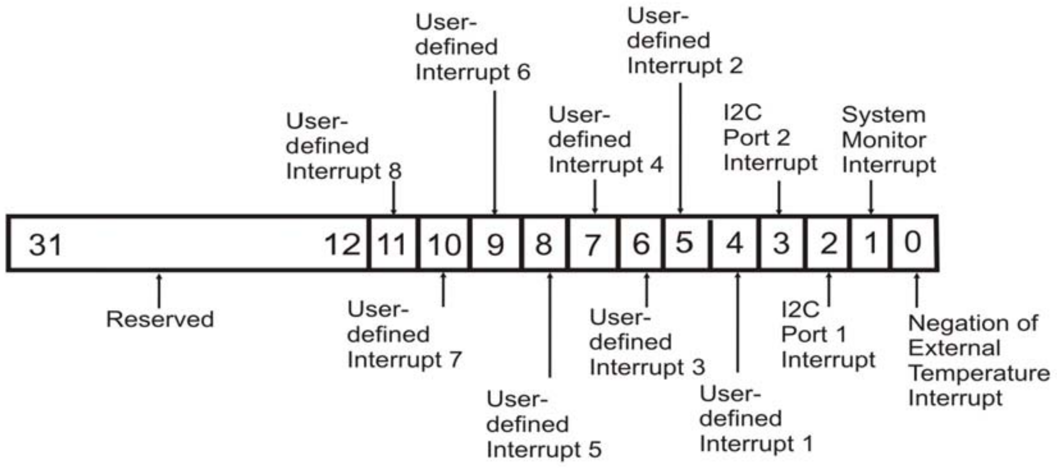


Table 4–12: Interrupt Enable Register (Base Address + 0x28)				
Bits	**Field Name**	**Default Value**	**Access Type**	**Description**
31:12	Reserved	N/A	N/A	**Reserved**
11	user8_irq	0	R/W	**User–defined Interrupt #8**: This bit enables/ disables the user–defined interrupt #8 source. 0 = Disable interrupt 1 = Enable interrupt
10	user7_irq			**User–defined Interrupt #7**: This bit enables/ disables the user–defined interrupt #7 source. 0 = Disable interrupt 1 = Enable interrupt

Table 4–12: Interrupt Enable Register (Base Address + 0x28) (Continued)

Bits	Field Name	Default Value	Access Type	Description
9	user6_irq	0	R/W	**User–defined Interrupt #6**: This bit enables/ disables the user–defined interrupt #6 source. 0 = Disable interrupt 1 = Enable interrupt
8	user5_irq			**User–defined Interrupt #5**: This bit enables/ disables the user–defined interrupt #5 source. 0 = Disable interrupt 1 = Enable interrupt
7	user4_irq			**User–defined Interrupt #4**: This bit enables/ disables the user–defined interrupt #4 source. 0 = Disable interrupt 1 = Enable interrupt
6	user3_irq			**User–defined Interrupt #3**: This bit enables/ disables the user–defined interrupt #3 source. 0 = Disable interrupt 1 = Enable interrupt
5	user2_irq			**User–defined Interrupt #2** : This bit enables/ disables the user–defined interrupt #2 source. 0 = Disable interrupt 1 = Enable interrupt
4	user1_irq			**User–defined Interrupt #1**: This bit enables/ disables the user–defined interrupt #1 source. 0 = Disable interrupt 1 = Enable interrupt
3	i2c2_irq			**I2C Port 2 Interrupt**: This bit enables/ disables the I2C bus interface port 2 interrupt source. 0 = Disable interrupt 1 = Enable interrupt
2	i2c1_irq			**I2C Port 1 Interrupt**: This bit enables/ disables the I2C bus interface port 1 interrupt source. 0 = Disable interrupt 1 = Enable interrupt

Table 4–12: Interrupt Enable Register (Base Address + 0x28) (Continued)

Bits	Field Name	Default Value	Access Type	Description
1	sys_mon_irq	0	R/W	**System Monitor Interrupt:** This bit enables/ disables the system monitor interrupt source. 0 = Disable interrupt 1 = Enable interrupt
0	ext_temp_irq	0	R/W	**Negation of External Temperature Interrupt:** This bit enables/ disables the negated external temperature interrupt source. 0 = Disable interrupt 1 = Enable interrupt

4.12 Interrupt Status Register

The Interrupt Status Register has read–only access associated with each interrupt condition. A status bit changes to ‘1’ when the source interrupt occurs. When a status bit in this register changes to ‘1’ the corresponding flag bit in the Interrupt Flag Register is set to ‘1’. A status bit in this register clears to ‘0‘ when that interrupt condition clears, whereas the associated flag bit in the Interrupt Flag Register remains at logic ‘1’ until it is explicitly cleared by the user.

Some of the interrupt sources are transient and so may not appear in the Interrupt Status Register at the time it is read. In such cases use the Interrupt Flag Register to see the interrupt conditions that have occurred. This Interrupt Status Register is illustrated in Figure 4–12 and described in Table 4–13.

Figure 4–12: Interrupt Status Register

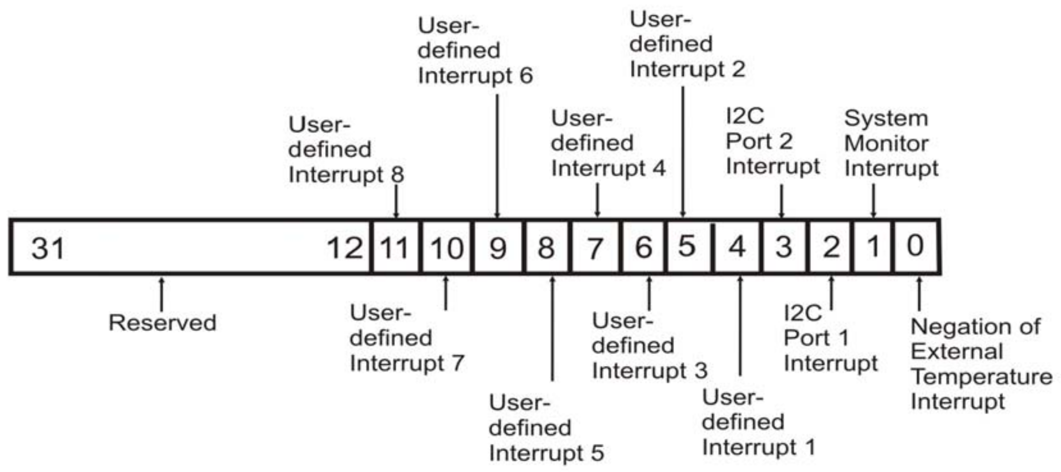


Table 4–13: Interrupt Status Register (Base Address + 0x2C)

Bits	Field Name	Default Value	Access Type	Description
31:12	Reserved	N/A	N/A	**Reserved**
11	user8_irq	0	R	**User–defined Interrupt #8**: This bit indicates the status of the user–defined interrupt #8 source. 0 = No interrupt 1 = Interrupt condition asserted

Table 4–13: Interrupt Status Register (Base Address + 0x2C) (Continued)

Bits	Field Name	Default Value	Access Type	Description
10	user7_irq	0	R	**User–defined Interrupt #7**: This bit indicates the status of the user–defined interrupt #7 source. 0 = No interrupt 1 = Interrupt condition asserted
9	user6_irq			**User–defined Interrupt #6**: This bit indicates the status of the user–defined interrupt #6 source. 0 = No interrupt 1 = Interrupt condition asserted
8	user5_irq			**User–defined Interrupt #5**: This bit indicates the status of the user–defined interrupt #5 source. 0 = No interrupt 1 = Interrupt condition asserted
7	user4_irq			**User–defined Interrupt #4**: This bit indicates the status of the user–defined interrupt #4 source. 0 = No interrupt 1 = Interrupt condition asserted
6	user3_irq			**User–defined Interrupt #3**: This bit indicates the status of the user–defined interrupt #3 source. 0 = No interrupt 1 = Interrupt condition asserted
5	user2_irq			**User–defined Interrupt #2** : This bit indicates the status of the user–defined interrupt #2 source. 0 = No interrupt 1 = Interrupt condition asserted
4	user1_irq			**User–defined Interrupt #1**: This bit indicates the status of the user–defined interrupt #1 source. 0 = No interrupt 1 = Interrupt condition asserted
3	i2c2_irq			**I2C Port 2 Interrupt**: This bit indicates the status of the I2C bus interface port 2 interrupt source. 0 = No interrupt 1 = Interrupt condition asserted

Table 4–13: Interrupt Status Register (Base Address + 0x2C) (Continued)

Bits	Field Name	Default Value	Access Type	Description
2	i2c1_irq	0	R	**I2C Port 1 Interrupt**: This bit indicates the status of the I2C bus interface port 1 interrupt source. 0 = No interrupt 1 = Interrupt condition asserted
1	sys_mon_irq			**System Monitor Interrupt:** This bit indicates the status of the system monitor interrupt source. 0 = No interrupt 1 = Interrupt condition asserted
0	ext_temp_irq			**Negation of External Temperature Interrupt:** This bit indicates the status of the negated external temperature interrupt source. 0 = No interrupt 1 = Interrupt condition asserted

4.13 Interrupt Flag Register

The Interrupt Flag Register has read/clear access associated with each interrupt condition. When reset, this register has all bits set to '0' (cleared). Each flag bit in this register latches an interrupt occurrence. A '1' in any flag bit in this register indicates that an interrupt has occurred.

Note that when any status bit in the Interrupt Status Register, changes to '1' the corresponding flag bit in this register will also be set to '1'. However, when a status bit in the Interrupt Status Register clears from '1' to '0', the corresponding latched flag bit in this register does not clear, but remains at '1'. To clear the flag bits, write '1's to the desired bits. The flags are not affected by the enable register. This Interrupt Flag Register is illustrated in Figure 4–13 and described in Table 4–14.

Figure 4–13: Interrupt Flag Register

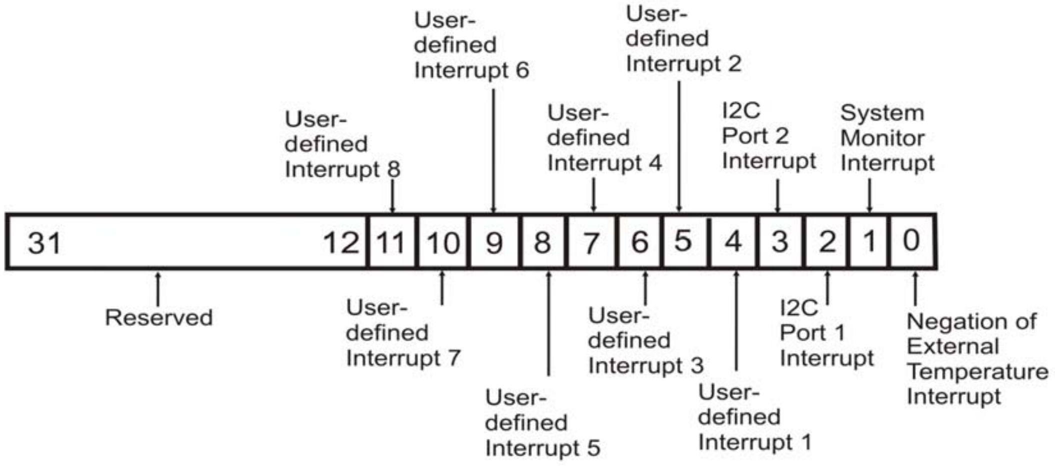


Table 4–14: Interrupt Flag Register (Base Address + 0x30)

Bits	Field Name	Default Value	Access Type	Description
31:12	Reserved	N/A	N/A	**Reserved**
11	user8_irq	0	R/Clr	**User–defined Interrupt #8**: This bit indicates the user–defined interrupt #8 flag. **Read:** 0 = No interrupt 1 = Interrupt latched **Clear:** 1 = Clear latch

Table 4–14: Interrupt Flag Register (Base Address + 0x30) (Continued)

Bits	Field Name	Default Value	Access Type	Description
10	user7_irq	0	R/Clr	**User–defined Interrupt #7**: This bit indicates the user–defined interrupt #7 flag. **Read:** 0 = No interrupt 1 = Interrupt latched **Clear:** 1 = Clear latch
9	user6_irq			**User–defined Interrupt #6**: This bit indicates the user–defined interrupt #6 flag. **Read:** 0 = No interrupt 1 = Interrupt latched **Clear:** 1 = Clear latch
8	user5_irq			**User–defined Interrupt #5**: This bit indicates the user–defined interrupt #5 flag. **Read:** 0 = No interrupt 1 = Interrupt latched **Clear:** 1 = Clear latch
7	user4_irq			**User–defined Interrupt #4**: This bit indicates the user–defined interrupt #4 flag. **Read:** 0 = No interrupt 1 = Interrupt latched **Clear:** 1 = Clear latch
6	user3_irq			**User–defined Interrupt #3**: This bit indicates the user–defined interrupt #3 flag. **Read:** 0 = No interrupt 1 = Interrupt latched **Clear:** 1 = Clear latch
5	user2_irq			**User–defined Interrupt #2** : This bit indicates the user–defined interrupt #2 flag. **Read:** 0 = No interrupt 1 = Interrupt latched **Clear:** 1 = Clear latch
4	user1_irq			**User–defined Interrupt #1**: This bit indicates the user–defined interrupt #1 flag. **Read:** 0 = No interrupt 1 = Interrupt latched **Clear:** 1 = Clear latch

Table 4–14: Interrupt Flag Register (Base Address + 0x30) (Continued)

Bits	Field Name	Default Value	Access Type	Description
3	i2c2_irq	0	R/Clr	**I2C Port 2 Interrupt**: This bit indicates the I2C bus interface port 2 interrupt flag. **Read:** 0 = No interrupt 1 = Interrupt latched **Clear:** 1 = Clear latch
2	i2c1_irq			**I2C Port 1 Interrupt**: This bit indicates the I2C bus interface port 1 interrupt flag. **Read:** 0 = No interrupt 1 = Interrupt latched **Clear:** 1 = Clear latch
1	sys_mon_irq			**System Monitor Interrupt:** This bit indicates the system monitor interrupt flag. **Read:** 0 = No interrupt 1 = Interrupt latched **Clear:** 1 = Clear latch
0	ext_temp_irq			**Negation of External Temperature Interrupt:** This bit indicates the negated external temperature interrupt flag. **Read:** 0 = No interrupt 1 = Interrupt latched **Clear:** 1 = Clear latch

This page is intentionally blank

Chapter 5: Designing with the Core

This chapter includes guidelines and additional information to facilitate designing with the Board Information Registers Core.

5.1 General Design Guidelines

The Board Information Registers Core provides control and status registers which are accessed through an AXI4–Lite Interface. This core allows the user to define the board information and code revision information. It also supports interrupt inputs from other modules including up to 8 user–defined interrupt inputs. An interrupt output can be generated based on the user application requirement.

5.2 Clocking

Main Clock: **`s_axi_csr_aclk`**

This clock is used to clock all ports of the core.

5.3 Resets

Main reset: **`s_axi_csr_aresetn`**

This is an active low reset synchronous with **`s_axi_csr_clk`**. When asserted, the control/status registers and the interrupt registers are reset.

5.4 Interrupts

This core has an edge–type (rising edge–triggered) interrupt output. It is synchronous with the **`s_axis_aclk`**. On the rising edge of any interrupt signal, a one clock cycle wide pulse is output from the core on it's **`irq`** output. Each interrupt event is stored in two registers, accessible on the **`s_axi_csr`** bus.

5.4 Interrupts (continued)

The Interrupt Status Register always reflects the current state of the interrupt condition, which may have changed since the generation of the interrupt. The Interrupt Flag Register latches the occurrence of each interrupt, in a bit that retains its state until explicitly cleared. The Interrupt flags can be cleared by writing '1' to the associated bit's location. All interrupt sources that are enabled (via the Interrupt Enable Register) are "OR ed" onto the **`irq`** output.

NOTE: All interrupt sources are latched in the interrupt flag register, even when an interrupt source is not enabled to create an interrupt.

NOTE: Because this core uses edge–triggered interrupts, the fact that an interrupt condition may remain active after servicing will not cause the generation of a new interrupt. A new interrupt will only be generated by another rising edge on an interrupt source.

5.5 Interface Operation

CSR Interface: This is the Control/Status Register Interface and is associated with **`s_axis_aclk`**. It is a standard AXI4–Lite Slave Interface. See Chapter 4 for the control register memory map, which provides more details on the registers that can be accessed through this interface.

5.6 Programming Sequence

This section briefly describes the programming sequence of registers in the Board Information Registers Core. When interrupt output is desired from the core the following programming sequence is followed.

1) Ensure that the interrupt flag register is cleared.

2) Enable the interrupt enable bits based on the user design requirement.

3) After the interrupt output is generated, clear the interrupt flag register.

5.7 Timing Diagrams

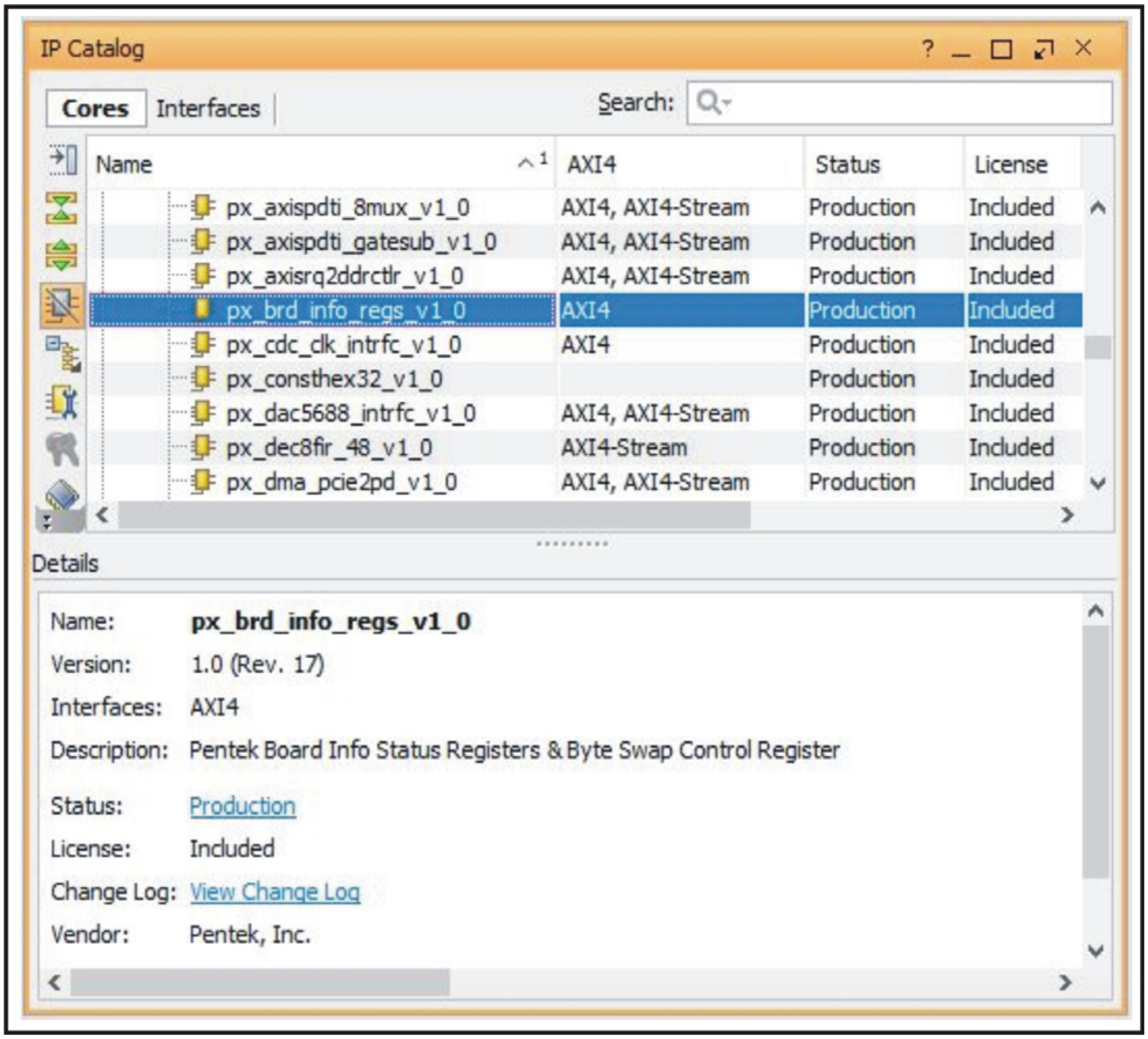
The timing diagrams for the Board Information Registers Core are obtained by running the simulation of the test bench of the core in Vivado VSim environment. For more details about the test bench, refer to Section 6.5.

Chapter 6: Design Flow Steps

6.1 Pentek IP Catalog

This chapter describes customization and generation of the Pentek Board Information Registers Core. It also includes simulation, synthesis, and implementation steps that are specific to this IP core. This core can be generated from the Vivado IP Catalog when the Pentek IP Repository has been installed. It will appear in the IP Catalog list as **px_brd_info_regs_v1_0** as shown in Figure 6–1.

Figure 6–1: Board Information Registers Core in Pentek IP Catalog



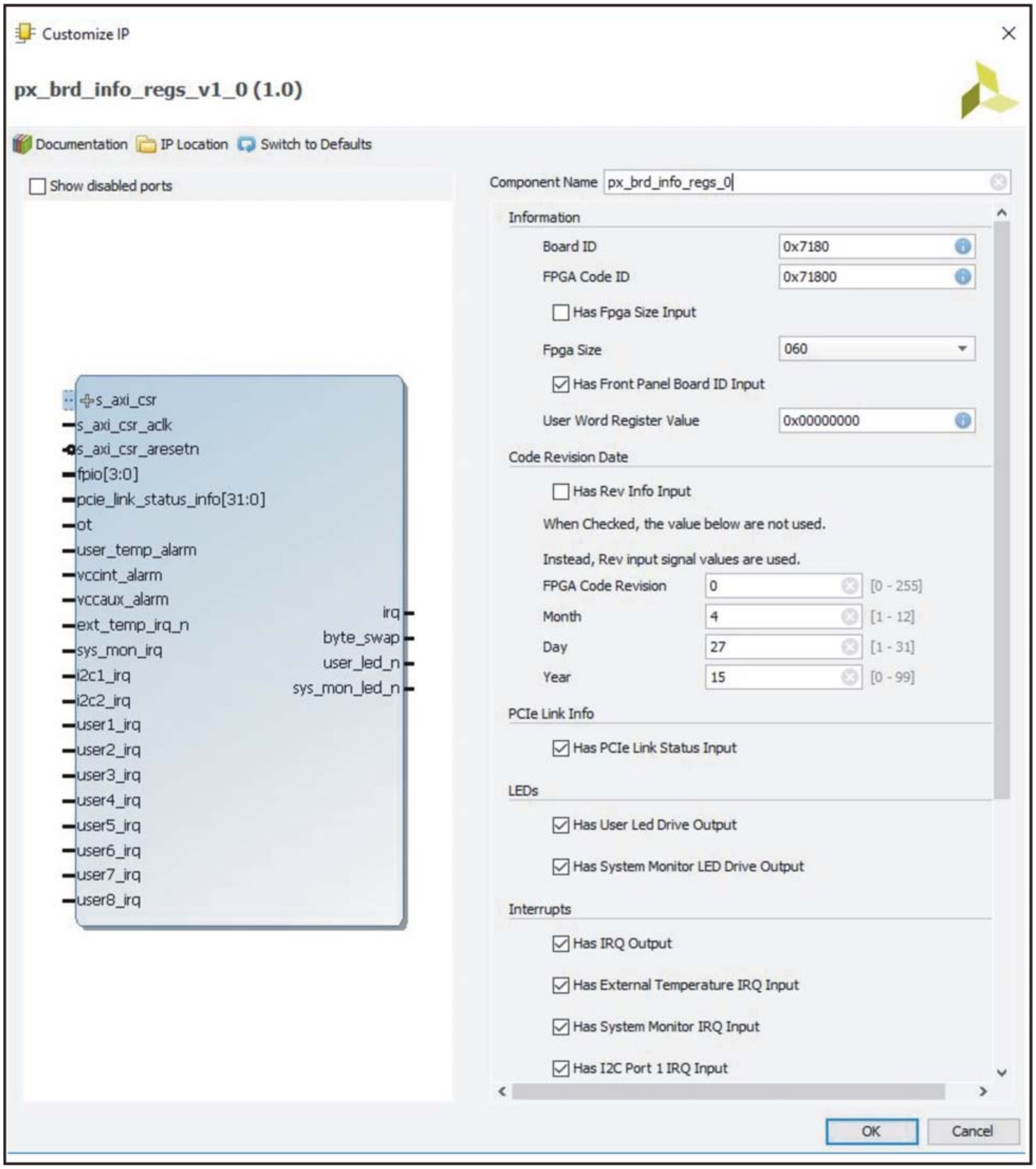
Name	AXI4	Status	License
px_axispdti_8mux_v1_0	AXI4, AXI4-Stream	Production	Included
px_axispdti_gatesub_v1_0	AXI4, AXI4-Stream	Production	Included
px_axisrq2ddrctlr_v1_0	AXI4, AXI4-Stream	Production	Included
px_brd_info_regs_v1_0	AXI4	Production	Included
px_cdc_clk_intrfc_v1_0	AXI4	Production	Included
px_consthex32_v1_0		Production	Included
px_dac5688_intrfc_v1_0	AXI4, AXI4-Stream	Production	Included
px_dec8fir_48_v1_0	AXI4-Stream	Production	Included
px_dma_pcie2pd_v1_0	AXI4, AXI4-Stream	Production	Included

Name:	**px_brd_info_regs_v1_0**
Version:	1.0 (Rev. 17)
Interfaces:	AXI4
Description:	Pentek Board Info Status Registers & Byte Swap Control Register
Status:	Production
License:	Included
Change Log:	View Change Log
Vendor:	Pentek, Inc.

6.1 Pentek IP Catalog (continued)

When you select the **px_brd_info_regs_v1_0** core, a screen appears that shows the core's symbol and the core's parameters (see Figure 6–2). The core's symbol is the box on the left side.

Figure 6–2: Board Information Registers Core IP Symbol



px_brd_info_regs_v1_0 (1.0)

Component Name: px_brd_info_regs_0

Information

Board ID	0x7180
FPGA Code ID	0x71800
☐ Has Fpga Size Input	
Fpga Size	060
☑ Has Front Panel Board ID Input	
User Word Register Value	0x00000000

Code Revision Date

☐ Has Rev Info Input

When Checked, the value below are not used.

Instead, Rev input signal values are used.

FPGA Code Revision	0	[0 - 255]
Month	4	[1 - 12]
Day	27	[1 - 31]
Year	15	[0 - 99]

PCIe Link Info

☑ Has PCIe Link Status Input

LEDs

☑ Has User Led Drive Output

☑ Has System Monitor LED Drive Output

Interrupts

☑ Has IRQ Output

☑ Has External Temperature IRQ Input

☑ Has System Monitor IRQ Input

☑ Has I2C Port 1 IRQ Input

6.2 User Parameters

The user parameters for this core are explained in the Section 2.5 of this user manual.

6.3 Generating Output

For more details about generating and using IP in the Vivado Design Suite, refer to the *Vivado Design Suite User Guide – Designing with IP*.

6.4 Constraining the Core

This section contains information about constraining the Board Information Registers Core in Vivado Design Suite.

Required Constraints

The XDC constraints are not provided with the Board Information Registers Core. Necessary constraints can be applied in the top level module of the user design.

Device, Package, and Speed Grade Selections

This IP works for the Kintex Ultrascale FPGAs.

Clock Frequencies

The clock frequency (**`s_axi_csr_aclk`**) for this IP core is 250 MHz.

Clock Management

This section is not applicable for this IP core.

Clock Placement

This section is not applicable for this IP core.

Banking and Placement

This section is not applicable for this IP core.

Transceiver Placement

This section is not applicable for this IP core.

I/O Standard and Placement

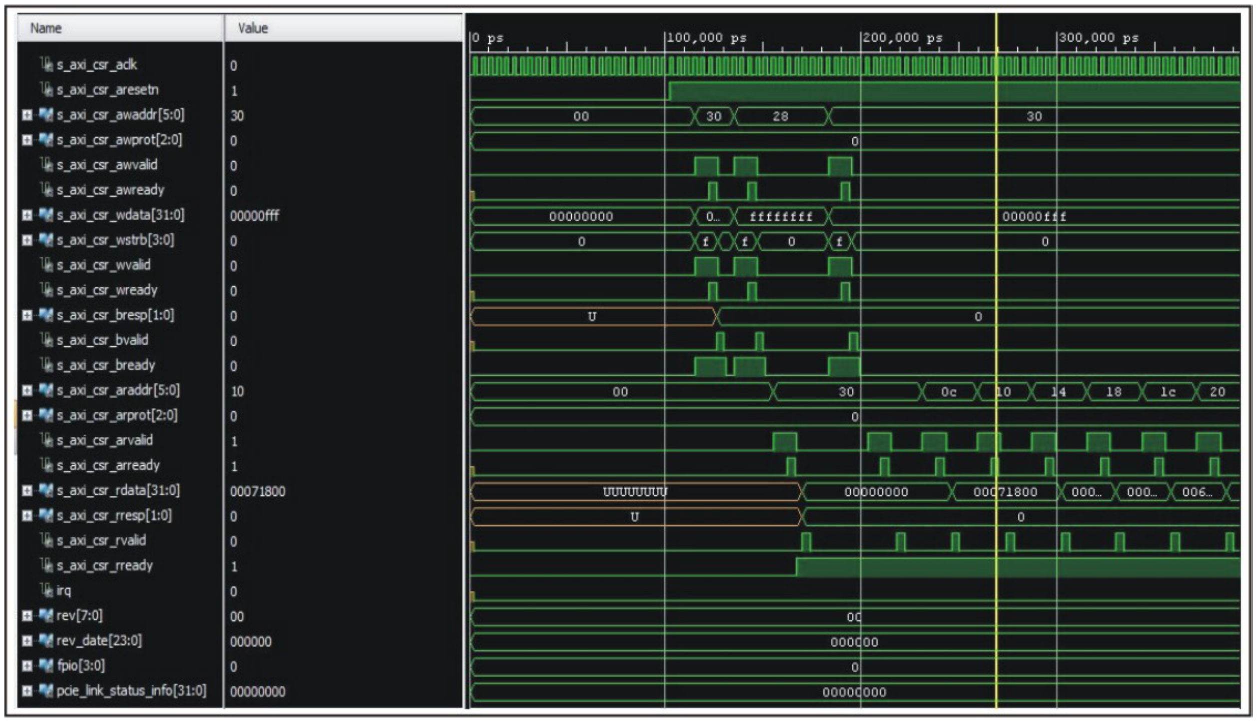
This section is not applicable for this IP core.

6.5 Simulation

The Board Information Registers IP has a test bench which generates the output waveforms using the Vivado VSim environment.

The test bench is designed to run at 250 MHz input clock frequency. The generic parameters for the core are defined in the test bench. The test bench has the core set to generate an interrupt output. It also has revision information provided, and input interrupts. The programming procedure is the same as described in Section 5.6. The test bench prints the information within the Status Registers in the Tcl Console of the Vivado VSim environment. When run, the simulation produces the results shown in Figures 6–3 and 6–4.

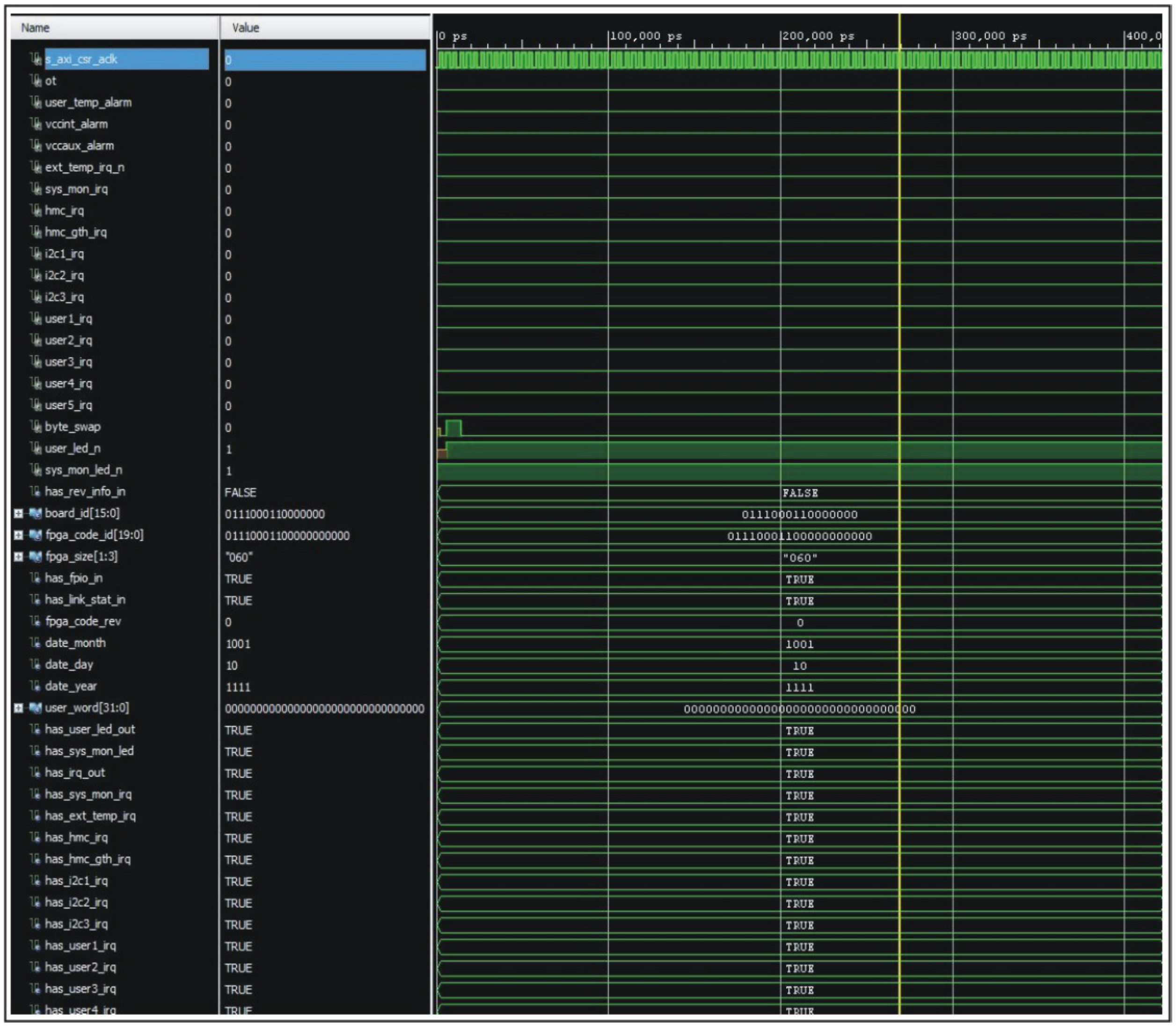
Figure 6–3: Board Information Registers Core Test Bench Simulation Output – Part 1



Name	Value
s_axi_csr_aclk	0
s_axi_csr_aresetn	1
s_axi_csr_awaddr[5:0]	30
s_axi_csr_awprot[2:0]	0
s_axi_csr_awvalid	0
s_axi_csr_awready	0
s_axi_csr_wdata[31:0]	00000fff
s_axi_csr_wstrb[3:0]	0
s_axi_csr_wvalid	0
s_axi_csr_wready	0
s_axi_csr_bresp[1:0]	0
s_axi_csr_bvalid	0
s_axi_csr_bready	0
s_axi_csr_araddr[5:0]	10
s_axi_csr_arprot[2:0]	0
s_axi_csr_arvalid	1
s_axi_csr_arready	1
s_axi_csr_rdata[31:0]	00071800
s_axi_csr_rresp[1:0]	0
s_axi_csr_rvalid	0
s_axi_csr_rready	1
irq	0
rev[7:0]	00
rev_date[23:0]	000000
fpio[3:0]	0
pcie_link_status_info[31:0]	00000000

6.5 **Simulation** (continued)

Figure 6–4: Board Information Registers Core Test Bench Simulation Output – Part 2



Name	Value
s_axi_csr_aclk	0
ot	0
user_temp_alarm	0
vccint_alarm	0
vccaux_alarm	0
ext_temp_irq_n	0
sys_mon_irq	0
hmc_irq	0
hmc_gth_irq	0
i2c1_irq	0
i2c2_irq	0
i2c3_irq	0
user1_irq	0
user2_irq	0
user3_irq	0
user4_irq	0
user5_irq	0
byte_swap	0
user_led_n	1
sys_mon_led_n	1
has_rev_info_in	FALSE
board_id[15:0]	0111000110000000
fpga_code_id[19:0]	01110001100000000000
fpga_size[1:3]	"060"
has_fpio_in	TRUE
has_link_stat_in	TRUE
fpga_code_rev	0
date_month	1001
date_day	10
date_year	1111
user_word[31:0]	00000000000000000000000000000000
has_user_led_out	TRUE
has_sys_mon_led	TRUE
has_irq_out	TRUE
has_sys_mon_irq	TRUE
has_ext_temp_irq	TRUE
has_hmc_irq	TRUE
has_hmc_gth_irq	TRUE
has_i2c1_irq	TRUE
has_i2c2_irq	TRUE
has_i2c3_irq	TRUE
has_user1_irq	TRUE
has_user2_irq	TRUE
has_user3_irq	TRUE
has_user4_irq	TRUE

6.6 Synthesis and Implementation

For details about synthesis and implementation see the *Vivado Design Suite User Guide – Designing with IP*.