IP CORE MANUAL

NAVIGATOR

FPGA Design Kit

AXI4–Stream VITA 49 Packetizer IP

px_vita49_pkt

PENTEK

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

Manual Part Number: 807.48402 Rev: 1.4 – November 14, 2018

Manual Revision History

Date	Version	Comments
8/17/17	1.0	Initial Release
9/14/17	1.1	Revised Sect 1.1, Table 2–1, Table 4–3, Sect 5.1, and Sect 5.2.
10/3/17	1.2	Revised Figure 1–4 and Sect 1.1.
10/16/17	1.3	Revised Table 4–1 and Sect 4.2.
11/14/18	1.4	Revised IP Facts, Sect 1.1, Sect 2.4, and Table 3–2. Added Sect 2.5.

Legal Notices

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Pentek products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Pentek hereby DISCLAIMS ALL WARRANTIES AND CONDITIONS, EXPRESS, IMPLIED, OR STATUTORY, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY, NON–INFRINGEMENT, OR FITNESS FOR ANY PARTICULAR PURPOSE; and (2) Pentek shall not be liable (whether in contract or tort, including negligence, or under any other theory of liability) for any loss or damage of any kind or nature related to, arising under, or in conjunction with, the Materials (including your use of Materials), including for any direct, indirect, special, incidental, or consequential loss or damage (including loss of data, profits, goodwill, or any type of loss or damage suffered as a result of any action brought by a third party) even if such damage and loss was reasonably foreseeable or Pentek had been advised of the possibility of the same. Pentek assumes no obligation to correct any error contained in the Materials or to notify you of updates to the Materials or to product specifications. You may not reproduce, modify, distribute, or publicly display the materials without prior written consent. Certain products are subject to the terms and conditions of Pentek's limited warranty, please refer to Pentek's Ordering and Warranty information which can be viewed at http://www.pentek.com/contact/customerinfo.cfm; IP cores may be subject to warranty and support terms contained in a license issued to you by Pentek. Pentek products are not designed or intended to be fail–safe or for use in any application requiring fail–safe performance; you assume sole risk and liability for the use of Pentek products in such critical applications.

Copyright

Trademarks

Pentek, Jade, and Navigator are trademarks or registered trademarks of Pentek, Inc.

ARM and AMBA are registered trademarks of ARM Limited. PCI, PCI Express, PCIe, and PCI–SIG are trademarks or registered trademarks of PCI–SIG. Xilinx, Kintex UltraScale, Vivado, and Platform Cable USB are registered trademarks of Xilinx Inc., of San Jose, CA.

Printed in the United States of America.

Table of Contents

Page

IP Facts

Chapter 1: Overview

Chapter 2: General Product Specifications

Chapter 3: Port Descriptions

Table of Contents

Page

Chapter 4: Register Space

Table of Contents

Page

Chapter 5: Designing with the Core

Chapter 6: Design Flow Steps

Table of Contents

Page

This page is intentionally blank

IP Facts

Description

Pentek's Navigator™ AXI4–Stream VITA 49 Packetizer Core takes a Pentek PPKT type packetized data stream and packages it into a VITA 49 Data stream. VITA 49 packet size is programmable up to 4096. Timestamp data is extracted from the PPKT type input stream and inserted into the VITA 49 headers.

This core complies with the ARM® AMBA® AXI4 Specification and also provides a control/status register interface. This user manual defines the hardware interface, software interface, and parameterization options for the AXI4–Stream VITA 49 Packetizer Core.

Features

- Adds VITA 49 header to packetized Pentek PPKT type AXI4–Stream data stream
- Maximum packet data length is programmable up to 4096 32–bit words.
- Programmable time mode can be "free run mode" or "in seconds mode" depending on the timestamp format of the input stream.
- Supports AXI4–Stream input bytes in widths of 4, 8, 16, or 32 controlled by a VHDL generic parameter

Table 1–1: IP Facts Table

Core Specifics	
Supported Design Family[a]	Kintex® Ultrascale
Supported User Interfaces	AXI4–Lite and AXI4–Stream
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

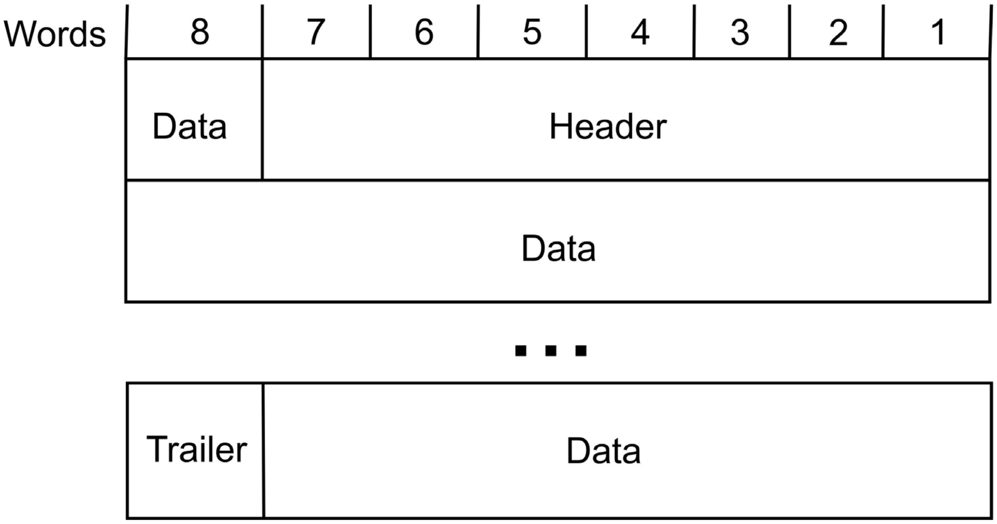
Chapter 1: Overview

1.1 Functional Description

The VITA 49 Packetizer Core (AXI4–Stream VITA 49 Packetizer Core) takes a Pentek PPKT type AXI4–Stream packetized data stream and packages it into a VITA 49 Data stream. Packet size is programmable up to 4096. Timestamp values are extracted from the input stream and inserted into the VITA 49 headers.

Figure 1–1 is a diagram showing the format of packets created by the Pentek AXI4–Stream VITA 49 Packetizer Core. Figure 1–2 shows further details of the packet format. Figure 1–3 shows the Packet Header details. Figure 1–4 is a top–level block diagram of this core. The modules within the block diagram are explained in the later sections of this user manual.

Figure 1–1: AXI4–Stream VITA 49 Packetizer Packet Format Diagram



1.1 Functional Description (continued)

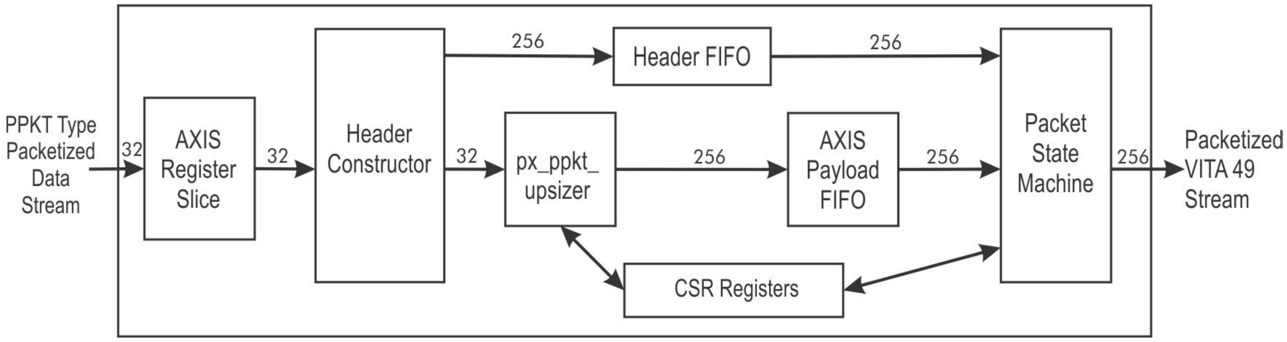
Figure 1−2: VITA 49 Packetizer Packet Format Detail

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Header (1 Word)																															
Stream Identifier (1 Word)																															
Class Identifier (2 Words)																															
Integer-seconds Timestamp (1 Word)																															
Fractional-seconds Timestamp (2 Words)																															
Data Payload (Variable)																															
Trailer (1 Word)																															

Figure 1−3: VITA 49 Packetizer Packet Header Detail

Bits	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Description	Pkt Type 0 0 0 X				C	T	RR		TSI		TSF		Packet Count				Packet Size															
Default	0001				1	1	00		-		-		-				-															

Figure 1−4: AXI4−Stream VITA 49 Packetizer Core Block Diagram



1.1 Functional Description (continued)

□ **Register Slice:** The AXI4–Stream Interconnect is a key Interconnect Infrastructure IP which enables connection of heterogeneous master/slave AMBA® AXI4–Stream protocol compliant endpoint IP. The AXI4–Stream Interconnect routes connection from one or more AXI4–Stream master channels to one or more AXI4–Stream slave channels.

□ **Header Constructor:** This module constructs the header and trailer for the VITA 49 packet. It also creates a VITA 49 packet length based on the programmable data maximum length.

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

□ **AXIS Payload FIFO:** This FIFO module stores incoming data.

□ **Header FIFO:** This FIFO module stores each header. When one header is available, one whole packet is available in the payload FIFO.

□ **Packetizer Upsizer:** This module converts the incoming generic data width of 32, 64, 128, or 256 bit data into 256 bit data.

□ **Packetizer State Machine:** This module takes a header and a full data payload and outputs the VITA 49 packet 256 bits at a time.

1.2 Applications

The VITA 49 Packetizer Core can be used with a Pentek PPKT type packetized AXI4–Stream data stream to add a VITA 49 header to the data.

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) VITA radio Transport (VRT) Draft Standard
http://shop.vita.com/ANSI–VITA–490–2015–VITA–Radio–Transport–VRT–Standard–AV490.htm

Chapter 2: General Product Specifications

2.1 Standards

The AXI4–Stream VITA 49 Packetizer Core has bus interfaces that comply with the *ARM AMBA AXI4–Lite Protocol Specification* and the *AMBA AXI4–Stream Protocol Specification*. This core also complies with VITA–49.0 – 2015.

2.2 Performance

The emergence of high–speed interconnects enables the transmission of signals, such as digitized IF, over packet networks. However, packetization comes at the expense of some overhead. The VRT protocol can minimize this overhead by allowing the volume of data, the format of data, and the type of data to be configured for optimal link utilization. Thus the overhead of the protocol can typically be configured to be a small fraction of the overall signal data bandwidth, providing the same efficiency as proprietary implementations.

2.3 Resource Utilization

The resource utilization of the VITA 49 Packetizer 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	1285
Flip–Flops	1108

NOTE: Actual utilization may vary based on the user design in which the VITA 49 Packetizer Core is incorporated.

2.4 Limitations and Unsupported Features

- Maximum VITA 49 packet length is 4096. This core does not support the full 64K maximum packet length allowed by the VITA 49 specification.
- This core only supports 32, 64, 128, and 256–bit PPKT input streams and only 256–bit PPKT output streams.
- Packet lengths must be a multiple of 8 32–bit words (including headers).
- If the input data stream ends at a length that is not a multiple of a 32–bit word, the last 16–bits of the last 32–bit word of the last packet will be invalid data.

2.5 Generic Parameters

The generic parameters of the 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
input_bytes	Integer	**Input Bytes:** This parameter selects the AXI4–Stream input byte width. The width can be set to 4, 8, 16, or 32 bytes.

Chapter 3: Port Descriptions

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

- AXI4–Lite Core Interfaces
- AXI4–Stream Core Interface

3.1 AXI4–Lite Core Interfaces

The AXI4–Stream VITA 49 Packetizer 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 VITA 49 Packetizer 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	7	**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 VITA 49 Packetizer Core. Note that the Register Space registers occupy an address range of [Base Address + (0x00 to 0x1C)].
s_axi_csr_awprot	Input	3	**Protection:** The VITA 49 Packetizer 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 VITA 49 Packetizer 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 VITA 49 Packetizer 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 VITA 49 Packetizer 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 VITA 49 Packetizer 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 VITA 49 Packetizer 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	7	**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 VITA 49 Packetizer Core.
s_axi_csr_arprot	Input	3	**Protection:** These bits are ignored by the VITA 49 Packetizer 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 VITA 49 Packetizer 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 VITA 49 Packetizer 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 VITA 49 Packetizer 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 VITA 49 Packetizer 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 AXI4–Stream Core Interface

The Packetizer Core has the following AXI4–Stream Interface, used to receive and transfer data streams. The Packetized Sample Data/ Timestamp/ Information Stream (PPKT) Interface is an AXI4–Stream Master Interface of the core used to transfer packed AXI4–Streams in the PPKT format.

3.2.1 Stream Data (PPKT) Interface

The Pentek Jade series board products have AXI4–Streams that follow a combined Sample Data/ Timestamp/ Information Stream (PPKT) format. This type of data stream combines sample data with its time–aligned timestamp and data information. Table 3–2 defines the ports in the AXI4–Stream Slave Combined Sample Data/ Timestamp/ Information Stream Interface. See the *AMBA AXI4–Stream Specification* for more details on the operation of the AXI4–Stream Interface.

Table 3-2: Stream Data (PPKT) Interface Interface Port Descriptions

Port	Direction	Width	Description
axis_aclk	Input	1	**Clock for the core**
axis_aresetn	Input	1	**Reset for the core:** Active low
s_axis_tvalid	Input	1	**Input Data Valid**
s_axis_tready	Output	1	**‘1’ when core is ready to accept data**
s_axis_tdata	Input	input_ bytes*8	**Data to be packed into VITA 49 packet**
s_axis_tkeep	Input	input_ bytes*2	**tkeep for tdata:** Must be FF till tlast=’1’
s_axis_tlast	Input	1	**tlast for tdata**
s_axis_tuser	Input	80	**tuser bits:** tuser [63:0] = Timestamp [63:0] of the first sample in tdata tuser [64] = SOF, Start of Frame tuser [66:65] =Data Format: 0=8bit, 1=16bit, 2=24bit 3=32bit tuser [67] = Data Type: 0=Real, 1=I/Q Complex tuser [75:68] = Channel [7:0] tuser [79:76] = User [3:0]
m_axis_tvalid	Output	1	**tvalid**
m_axis_tready	Input	1	**tready**
m_axis_tdata	Output	256	**tdata**
m_axis_tkeep	Output	16	**tkeep**

Table 3-2: Stream Data (PPKT) Interface Interface Port Descriptions (Continued)

Port	Direction	Width	Description
m_axis_tlast	Output	1	**tlast**
m_axis_tuser	Output	80	**tuser bits:** tuser [63:0] = Timestamp [63:0] of the first sample in tdata tuser [64] = SOF, Start of Frame tuser [66:65] =Data Format: 0=8bit, 1=16bit, 2=24bit 3=32bit tuser [67] = Data Type: 0=Real, 1=I/Q Complex tuser [75:68] = Channel [7:0] tuser [79:76] = User [3:0]

This page is intentionally blank

Chapter 4: Register Space

This chapter provides the memory map and register descriptions for the register space of the AXI4–Stream VITA 49 Packetizer Core. The memory map is shown in Table 4–1.

Table 4–1: Register Space Memory Map

Register Name	Address (Base Address +)	Access	Description
Control Registers			
User Control	0x0000	R/W	User Controls for: Bit 0: User Reset Bit 1: User Enable Bit 2: Time Mode
Packet Header	0x0004	R/W	Sets the maximum packet size: [15:0] data_max (See Section 4.2.) [19:16] Reserved [21:20] TSF [23:22] TSI [25:24] Reserved [26] T [27] C [31:28] Pkt Type
Stream ID	0x0008	R/W	Stream ID Bits: 31:0
Trailer	0x000C	R/W	Trailer Bits: 31:0
Class Identifier 1	0x0010	R/W	Class Identifier 1 bits: 31:0
Class Identifier 2	0x0014	R/W	Class Identifier 2 bits: 31:0
Status Registers			
Process Status Register	0x18	RO	Bit 0: Header creation error Bit 1: Packet SM error
Payload FIFO Count	0x1C	RO	Payload FIFO Count: Bits 12 downto 0
Header FIFO Count	0x20	RO	Header FIFO Count: Bits 7 downto 0

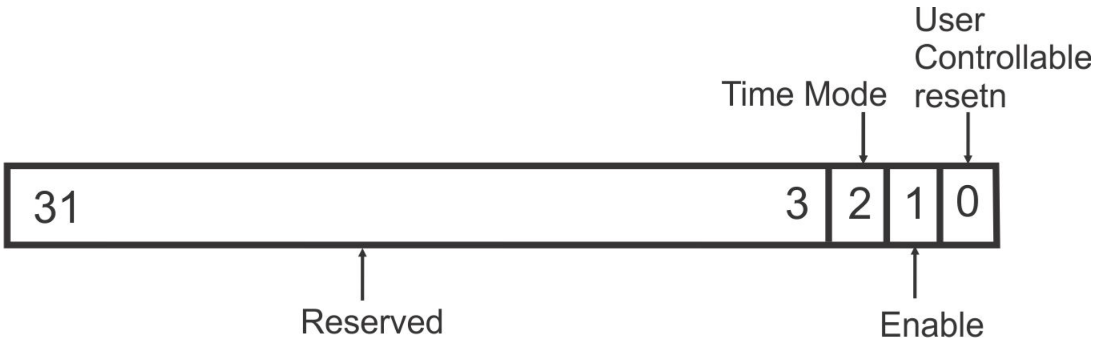
Table 4–1: Register Space Memory Map (Continued)

Register Name	Address (Base Address +)	Access	Description
Interrupt Enable/Status/Flag Registers			
Interrupt Enable Register	0x24	R/W	Interrupt Enable Bits: Bit 0: Header constructor error Bit 1: Packet SM error Bit 2: Payload FIFO empty Bit 3: Payload FIFO full Bit 4: Header FIFO empty Bit 5: Header FIFO full
Interrupt Status Register	0x28	RO	Interrupt Status Bits: Bit 0: Header constructor error Bit 1: Packet SM error Bit 2: Payload FIFO empty Bit 3: Payload FIFO full Bit 4: Header FIFO empty Bit 5: Header FIFO full
Interrupt Flag Register	0x2C	R/Clr	Interrupt flag bits: Bit 0: Header constructor error Bit 1: Packet SM error Bit 2: Payload FIFO empty Bit 3: Payload FIFO full Bit 4: Header FIFO empty Bit 5: Header FIFO full

4.1 User Control Register

This register is used to control main core controls. The User Control Register is illustrated in Figure 4–1 and described in Table 4–2.

Figure 4–1: User Control Register



.

Table 4–2: User Control Register (Base Address + 0x0000)				
Bits	**Field Name**	**Default Value**	**Access Type**	**Description**
0	user_resetn	0	R/W	**User controllable resetn:** Active low. Resets entire core. Clears all internal FIFOs and registers.
1	enable	0	R/W	**Enable:** The core will stop all processes when this bit is set to '0.' The core will function normally when this bit is set to '1.' **0 = not enabled** **1 = enabled**
2	time_mode	0	R/W	**Time Mode:** Set this bit to agree with the format of your input tdata timestamp. **0 = tuser is clock count only** **1 = tuser is seconds and clock count**
3:31	Reserved	–	–	**Reserved**

4.2 Packet Header Register

This register is used to control the Packet Header values. This register is illustrated in Figure 4–2 and described in Table 4–3.

Figure 4–2: Packet Header Register

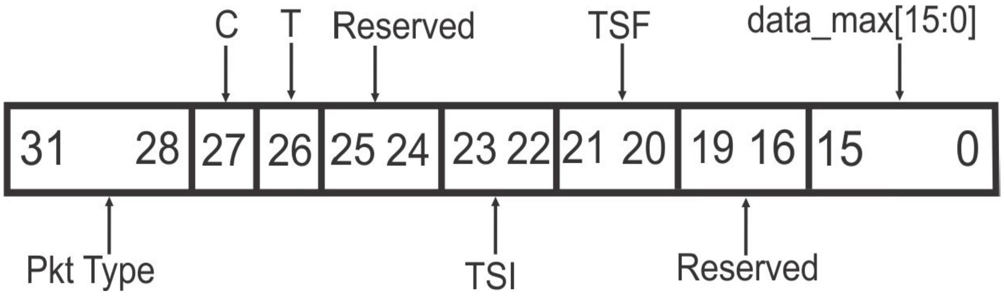


Table 4–3: Packet Header Register (Base Address + 0x0004)				
Bits	**Field Name**	**Default Value**	**Access Type**	**Description**
15:0	data_max	0x1000	R/W	**Maximum Packet Length:** The number of data words is (n). This value must be a multiple of 8. **This is not packet size length. This value is data payload only. It does not include header or trailer.** **Note:** The programmable maximum data length is 0x1000.
19:16	Reserved	–	–	**Reserved**
21:20	TSF	0	R/W	**Time Stamp Fractional:** 00 = No Fractional Secconds Timestamp field included 01 = Sample Count Timestamp 10 = Real Time Picoseconds Timestamp 11 = Free Running Count Timestamp
23:22	TSI	0	R/W	**Time Stamp Integer:** 00 = No Integer Seconds Timestamp field included 01 = Coordinated Universal Time (UTC) 10 = GPS Time 11 = Other
25:24	Reserved	–	–	**Reserved**
26	T	1	R/W	**Trailer Enable**
27	C	1	R/W	**Class ID Enable**
31:28	Pkt Type	0001	R/W	**Packet Type: Those not supported are underlined.** 0000 (0) IF Data packet without Stream Identifier 0001 (1) IF Data packet with Stream Identifier 0010 (2) Extension Data packet without Stream Identifier 0011 (3) Extension Data packet with Stream Identifier 0100 (4) IF Context packet 0101 (5) Extension Context packet The others are reserved for future VRT packet types.

4.3 Stream ID Register

This register is used to control Stream ID. The Stream ID shall be a 32–bit number assigned to a VRT Packet Stream. When used, the same Stream ID shall be carried in every packet in the Packet Stream. The Stream ID Field shall be either included in, or omitted from, every IF Data packet in an IF Data Packet Stream according to packet type. When Data–Context pairing is used to associate Context with an IF Data Packet Stream, the IF Data packet format used shall include a Stream ID. This register is illustrated in Figure 4–3 and described in Table 4–4.

Figure 4–3: Stream ID Register

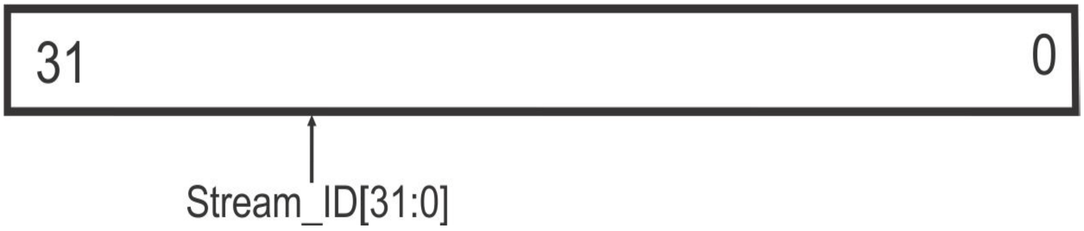


Table 4–4: Stream ID Register (Base Address + 0x0008)				
Bits	**Field Name**	**Default Value**	**Access Type**	**Description**
31:0	Stream_ID	0x00000000	R/W	**Stream ID:** See description above.

4.4 Trailer Register

This register is used to control the trailer. The trailer in this version of the core serves no real purpose other than to make the packet a multiple of 8 32–bit words in length.

Together, the Enables field and the State and Event Indicators field provide the capability to mark an IF Data packet with one or more Data events or state updates to be communicated from a VRT emitter to a VRT receiver. An Event Indicator might indicate a system synchronization signal or other event that affects some portion, or all, of the IF Data packet payload. A state update might be an indication of tuner phase–lock. When these fields are used, no provision is made for indicating the precise time of the events or state changes. They are only understood to be associated with the Data in that packet. The Enables field contains an enable bit for each Indicator bit in the State and Event Indicators field. Some of the Indicators (and their enable bits) are predefined and some are user–defined.

When the "E" bit is set to one the “Associated Context Packet Count” shall provide a count of all of transmitted Context packets that are directly or indirectly associated with the IF Data packet, OR a count of some special subset of these. When the "E" bit is cleared, the “Associated Context Packet Count” is undefined.

This register is illustrated in Figure 4–4 and described in Table 4–5.

Figure 4–4: Trailer Register

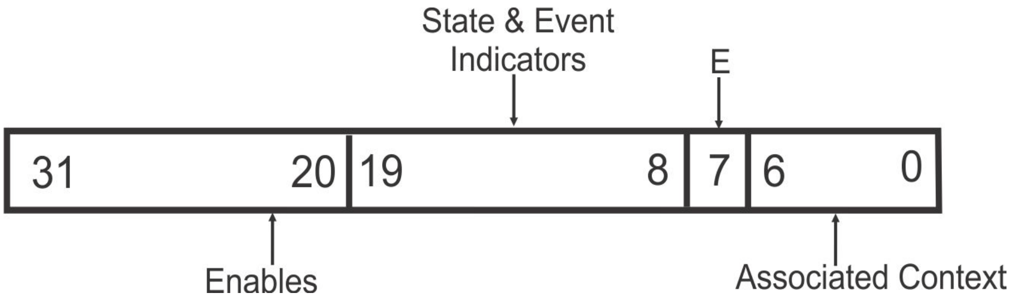


Table 4–5: Trailer Register (Base Address + 0x000C)				
Bits	**Field Name**	**Default Value**	**Access Type**	**Description**
31:20	Enables	0	R/W	**Enables for State and Event Indicators**
19:8	State_Event_ Indicators	0	R/W	**State and Event Indicators:** Refer to VITA 49 spec for details.
7	E	0	R/W	**Associated Context Enable:** See description above.
6:0	Associated_ Context	0	R/W	**Associated Context:** Refer to VITA 49 spec for details.

4.5 Class Identifier 1 Register

This register is used to control the Class Identifier 1 Register. This register is illustrated in Figure 4–5 and described in Table 4–6.

Figure 4–5: Class Identifier 1 Register

Table 4–6: Class Identifier 1 Register (Base Address + 0x0010)				
Bits	**Field Name**	**Default Value**	**Access Type**	**Description**
31:0	Class_ID1	0x0	R/W	**Class Identifier 1**

4.6 Class Identifier 2 Register

This register is used to control the Class Identifier 2 Register. This register is illustrated in Figure 4–6 and described in Table 4–7.

Figure 4–6: Class Identifier 2 Register

Table 4–7: Class Identifier 2 Register (Base Address + 0x0014)

Bits	Field Name	Default Value	Access Type	Description
31:0	Class_ID1	0x0	R/W	**Class Identifier 2**

4.7 Process Status Register

The Process Status Register reads core error statuses. This register is illustrated in Figure 4–7 and described in Table 4–8.

Figure 4–7: Process Status Register

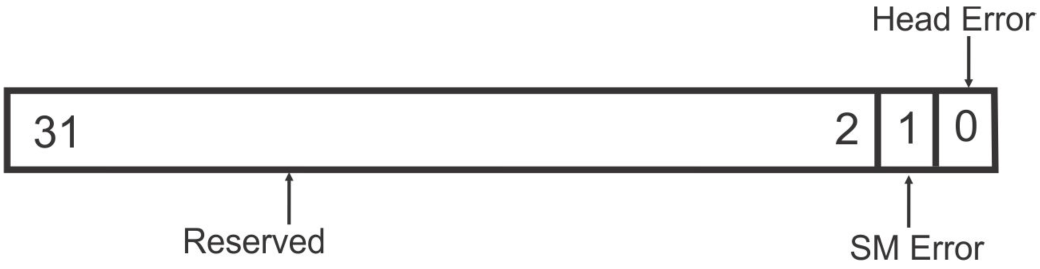


Table 4–8: Process Status Register (Base Address + 0x18)				
Bits	**Field Name**	**Default Value**	**Access Type**	**Description**
0	head_error	0	RO	**Header Creation Error** **0 = No error** **1 = Error (FIFO full when trying to write)**
1	sm_error	0	RO	**Packet State Machine Error** **0 = No Error** **1 = Error**
2:31	Reserved	–	–	**Reserved**

4.8 Payload FIFO Status Register

The Payload FIFO Status Register gives a count of the current amount of tdata in the payload FIFO. This register is illustrated in Figure 4–8 and described in Table 4–9.

Figure 4–8: Payload FIFO Status Register

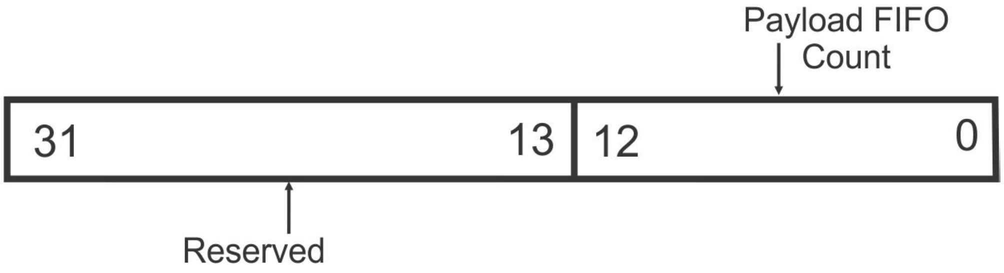


Table 4–9: Payload FIFO Status Register (Base Address + 0x1C)

Bits	Field Name	Default Value	Access Type	Description
12:0	axis_payload_ data_count	0	RO	**Payload FIFO Count:** Count of 32–bit payload words in FIFO
31:13	Reserved	0	–	**Reserved**

4.9 Header FIFO Status Register

The Header FIFO Status Register gives a count of the current amount of data in the header FIFO. This register is illustrated in Figure 4–9 and described in Table 4–10.

Figure 4–9: Header FIFO Status Register

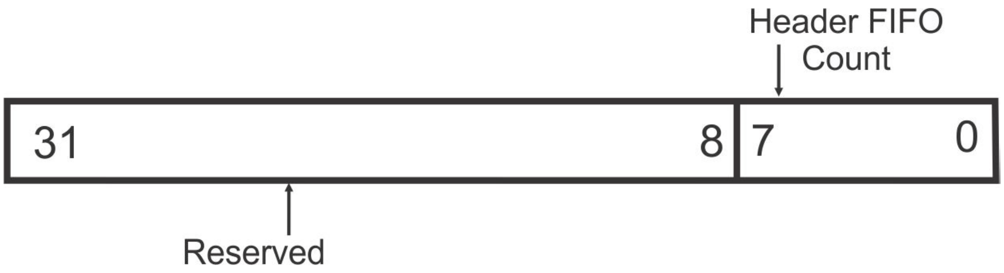


Table 4–10: Header FIFO Status Register (Base Address + 0x20)

Bits	Field Name	Default Value	Access Type	Description
7:0	axis_header_ data_count	0	RO	**Header FIFO Count:** Count number of headers awaiting insertion into the data stream
31:8	Reserved	0	RO	**Reserved**

4.10 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 4.5). This register is illustrated in Figure 4–10 and described in Table 4–11.

Figure 4–10: Interrupt Enable Register

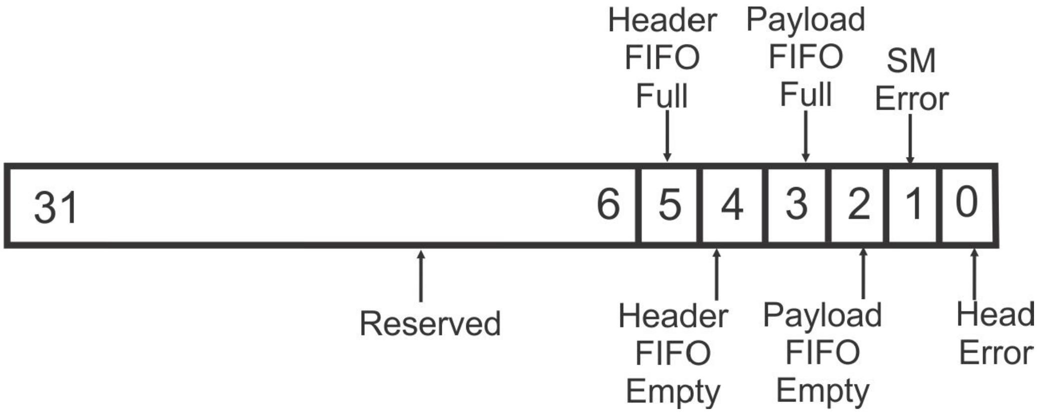


Table 4–11: Interrupt Enable Register (Base Address + 0x24)

Bits	Field Name	Default Value	Access Type	Description
0	head_error	0	R/W	**Header Constructor Error**
1	sm_error	0	R/W	**Packet SM Error**
2	payload_fifo_empty	0	R/W	**Payload FIFO empty**
3	payload_fifo_full	0	R/W	**Payload FIFO full**
4	header_fifo_empty	0	R/W	**Header FIFO empty**
5	header_fifo_full	0	R/W	**Header FIFO full**
31:6	Reserved	–	–	**Reserved**

4.11 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 register is illustrated in Figure 4–11 and described in Table 4–12.

Figure 4–11: Interrupt Status Register

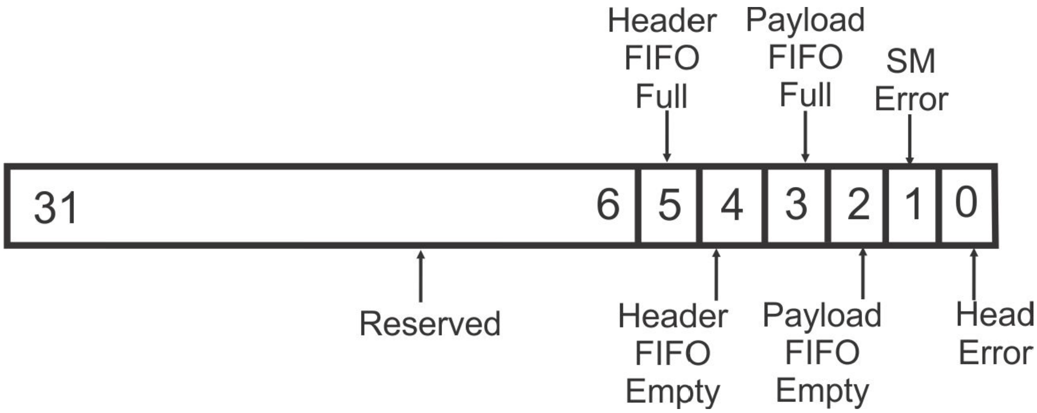


Table 4–12: Interrupt Status Register (Base Address + 0x28				
Bits	**Field Name**	**Default Value**	**Access Type**	**Description**
0	head_error	0	RO	**Header Constructor Error**
1	sm_error	0	RO	**Packet SM Error**
2	payload_fifo_empty	0	RO	**Payload FIFO empty**
3	payload_fifo_full	0	RO	**Payload FIFO full**
4	header_fifo_empty	0	RO	**Header FIFO empty**
5	header_fifo_full	0	RO	**Payload FIFO full**
31:6	Reserved	–	–	**Reserved**

4.12 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 Interrupt Enable Register.This register is illustrated in Figure 4–12 and described in Table 4–13.

Figure 4–12: Interrupt Flag Register

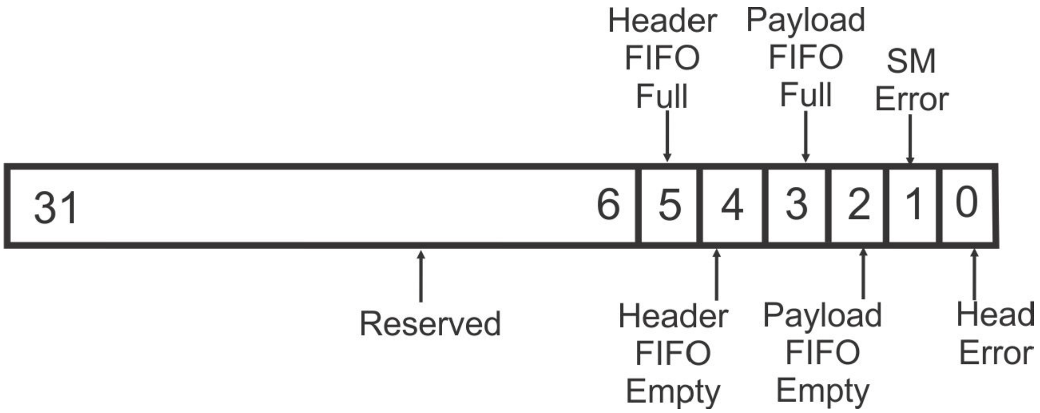


Table 4–13: Interrupt Flag Register (Base Address + 0x2C)				
Bits	**Field Name**	**Default Value**	**Access Type**	**Description**
0	head_error	0	R/CLR	**Header Constructor Error**
1	sm_error	0	R/CLR	**Packet SM Error**
2	payload_fifo_empty	0	R/CLR	**Payload FIFO empty**
3	payload_fifo_full	0	R/CLR	**Payload FIFO full**
4	header_fifo_empty	0	R/CLR	**Header FIFO empty**
5	header_fifo_full	0	R/CLR	**Header FIFO full**
31:6	Reserved	–	–	**Reserved**

Chapter 5: Designing with the Core

This chapter includes guidelines and additional information to facilitate designing with the AXI4–Stream VITA 49 Packetizer Core.

5.1 General Design Guidelines

The AXI4–Stream VITA 49 Packetizer Core takes a packetized data stream and adds a VITA 49 header and trailer with options based on programmable registers. This core outputs a packetized data stream. Programmable length must be divisible by 8. The packet stream will end when **tlast** is asserted. The core will then wait for **tvalid** = 1. Ensure that your selected time mode corresponds with the setup of your board's time–stamp in **tuser**.

5.2 Clocking

AXI4–Lite Clock: **s_axi_csr_aclk**

The **s_axi_csr_aclk** is used to clock the AXI4–Lite Control/Status Register (**s_axi_csr**) interface of the core.

AXI4–Stream Interface Clock: **axis_aclk** This clock is used for both input and output streams.

This clock is used to clock the AXI4–Stream inputs and outputs of the core.

5.3 Resets

Main reset: **axis_aresetn**

This is an active low synchronous reset associated with the **axis_aclk.** When asserted, all state machines in the core are reset, all FIFOs are flushed and all the control registers are cleared back to their initial default states.

CSR Reset: **s_axi_csr_resetn**

This is an active low sync reset associated with **s_axi_csr_aclk**. When asserted all control registers, initial values, and interrupts are reset.

5.4 Interrupts

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

5.4 **Interrupts** (continued)

This core has an edge type (rising edge–triggered) interrupt output. It is synchronous with the **`s_axi_csr_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. 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_axi_csr_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.

Stream Data Slave (PPKT) Interface: This interface is used to take data into the core for processing.

Stream Data Master (PPKT) Interface: This interface attaches a header and trailer to the input data.

5.6 Programming Sequence

This section briefly describes the programming sequence of registers in the VITA 49 Packetizer Core.

1) Use hardwired **`aresetn`**. Set to ‘1.’

2) Data Stream should not be active

3) Set user enable registers. The Control Register Address is 0. Set it to 0x00000003. This sets the user reset and user enable to ‘1.’ The core is now ready to use.

4) Program VITA 49 options (Data Length Stream ID, etc.) before streaming data.

5.7 Timing Diagrams

For more details about the test bench, refer to Section 6.5.

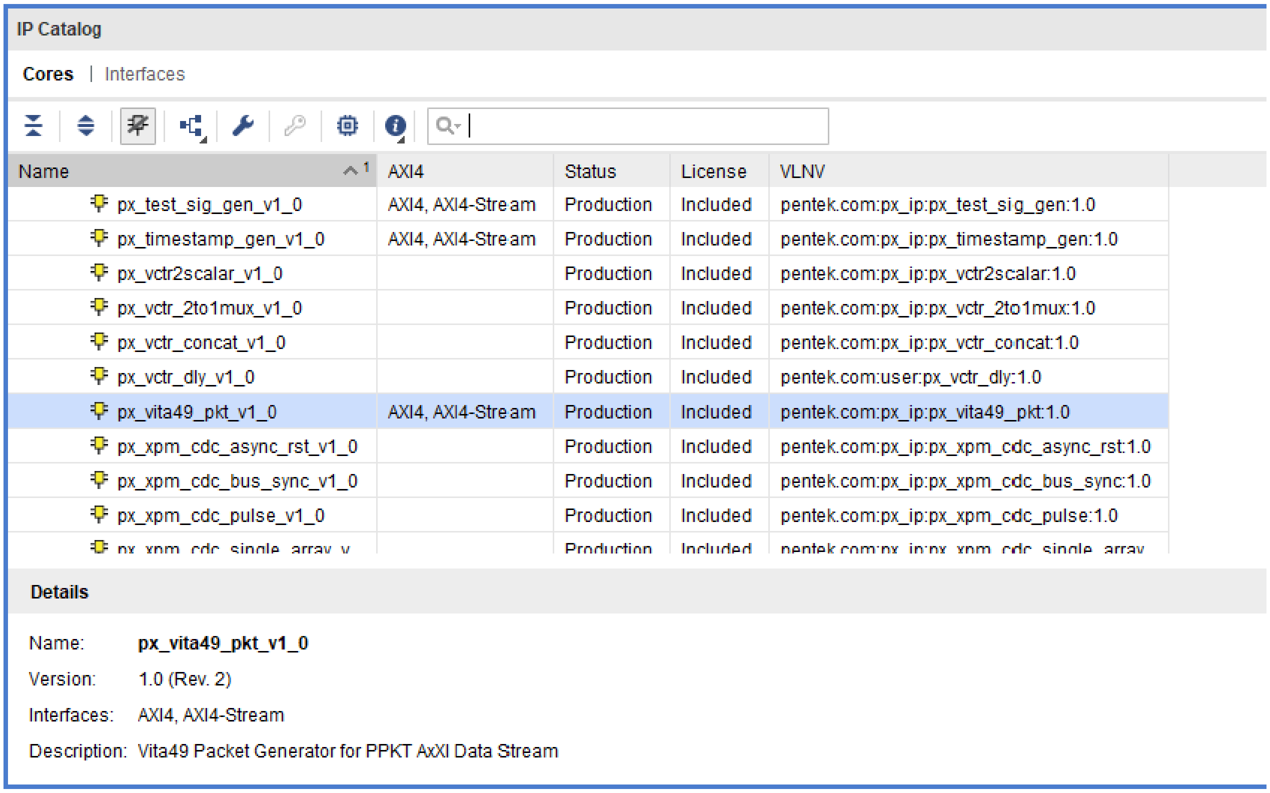
This page is intentionally blank

Chapter 6: Design Flow Steps

6.1 Pentek IP Catalog

This chapter describes customization and generation of the Pentek AXI4–Stream VITA 49 Packetizer 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_vita49_pkt_1_v1_0** as shown in Figure 6–1.

Figure 6–1: AXI4–Stream VITA 49 Packetizer Core in Pentek IP Catalog



Name	AXI4	Status	License	VLNV
px_test_sig_gen_v1_0	AXI4, AXI4-Stream	Production	Included	pentek.com:px_ip:px_test_sig_gen:1.0
px_timestamp_gen_v1_0	AXI4, AXI4-Stream	Production	Included	pentek.com:px_ip:px_timestamp_gen:1.0
px_vctr2scalar_v1_0		Production	Included	pentek.com:px_ip:px_vctr2scalar:1.0
px_vctr_2to1mux_v1_0		Production	Included	pentek.com:px_ip:px_vctr_2to1mux:1.0
px_vctr_concat_v1_0		Production	Included	pentek.com:px_ip:px_vctr_concat:1.0
px_vctr_dly_v1_0		Production	Included	pentek.com:user:px_vctr_dly:1.0
px_vita49_pkt_v1_0	AXI4, AXI4-Stream	Production	Included	pentek.com:px_ip:px_vita49_pkt:1.0
px_xpm_cdc_async_rst_v1_0		Production	Included	pentek.com:px_ip:px_xpm_cdc_async_rst:1.0
px_xpm_cdc_bus_sync_v1_0		Production	Included	pentek.com:px_ip:px_xpm_cdc_bus_sync:1.0
px_xpm_cdc_pulse_v1_0		Production	Included	pentek.com:px_ip:px_xpm_cdc_pulse:1.0
px_xpm_cdc_single_array_v		Production	Included	pentek.com:px_ip:px_xpm_cdc_single_array

Details

Name: **px_vita49_pkt_v1_0**

Version: 1.0 (Rev. 2)

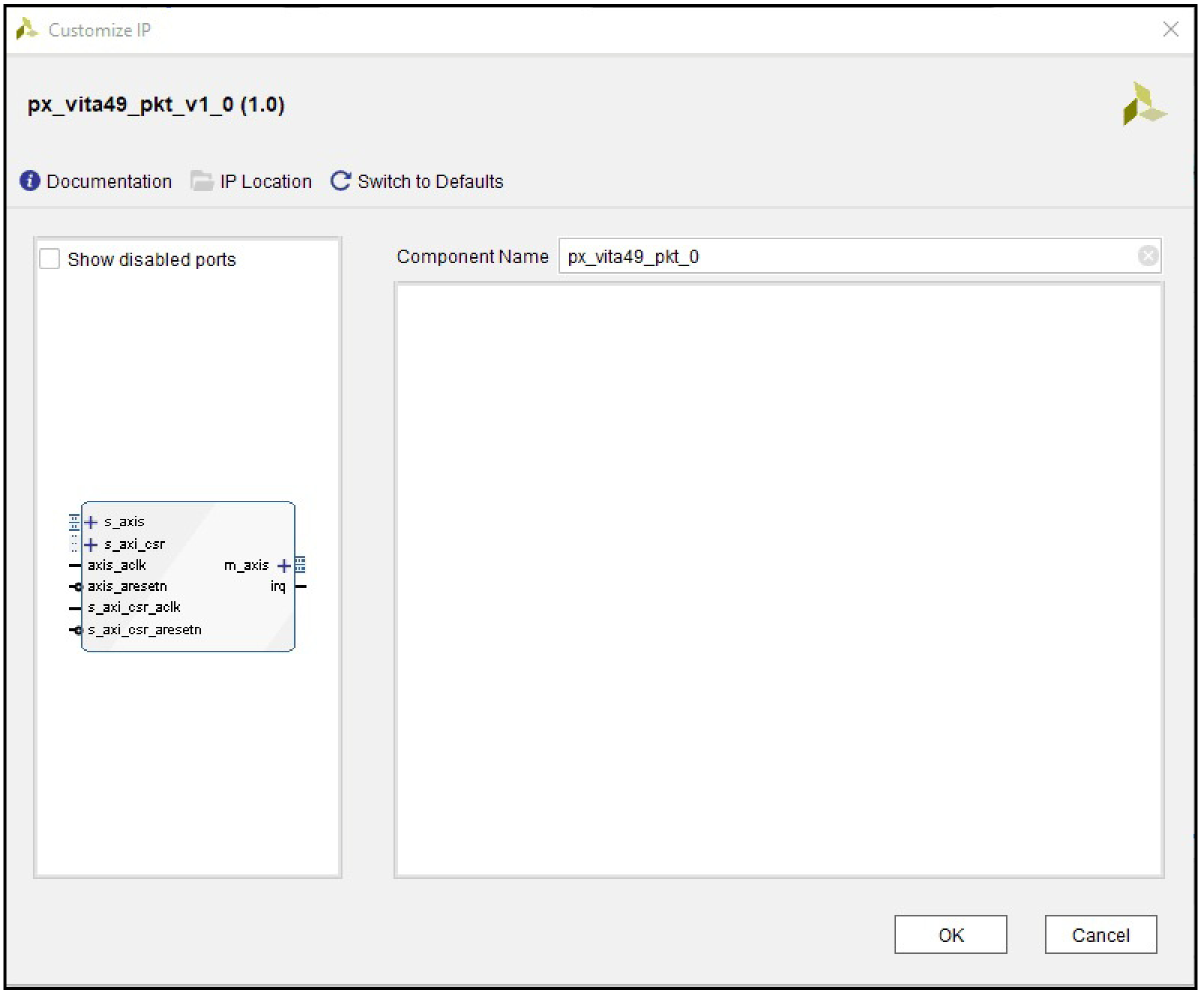
Interfaces: AXI4, AXI4-Stream

Description: Vita49 Packet Generator for PPKT AxXI Data Stream

6.1 Pentek IP Catalog (continued)

When you select the **px_vita49_pkt_1_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: AXI4–Stream Data Flow Control and Packetizer Core IP Symbol



☐ Show disabled ports

Component Name: px_vita49_pkt_0

6.2 User Parameters

This section is not applicable to this IP core.

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 Packetizer Core in Vivado Design Suite.

Required Constraints

The XDC constraints are not provided with the VITA 49 Packetizer Core. The 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

This section is not applicable for this IP core.

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

Figure 6−3: AXI4−Stream VITA 49 Packetizer Core Test Bench Simulation Output with SM Running at Maximum Data Size of 0x00000005

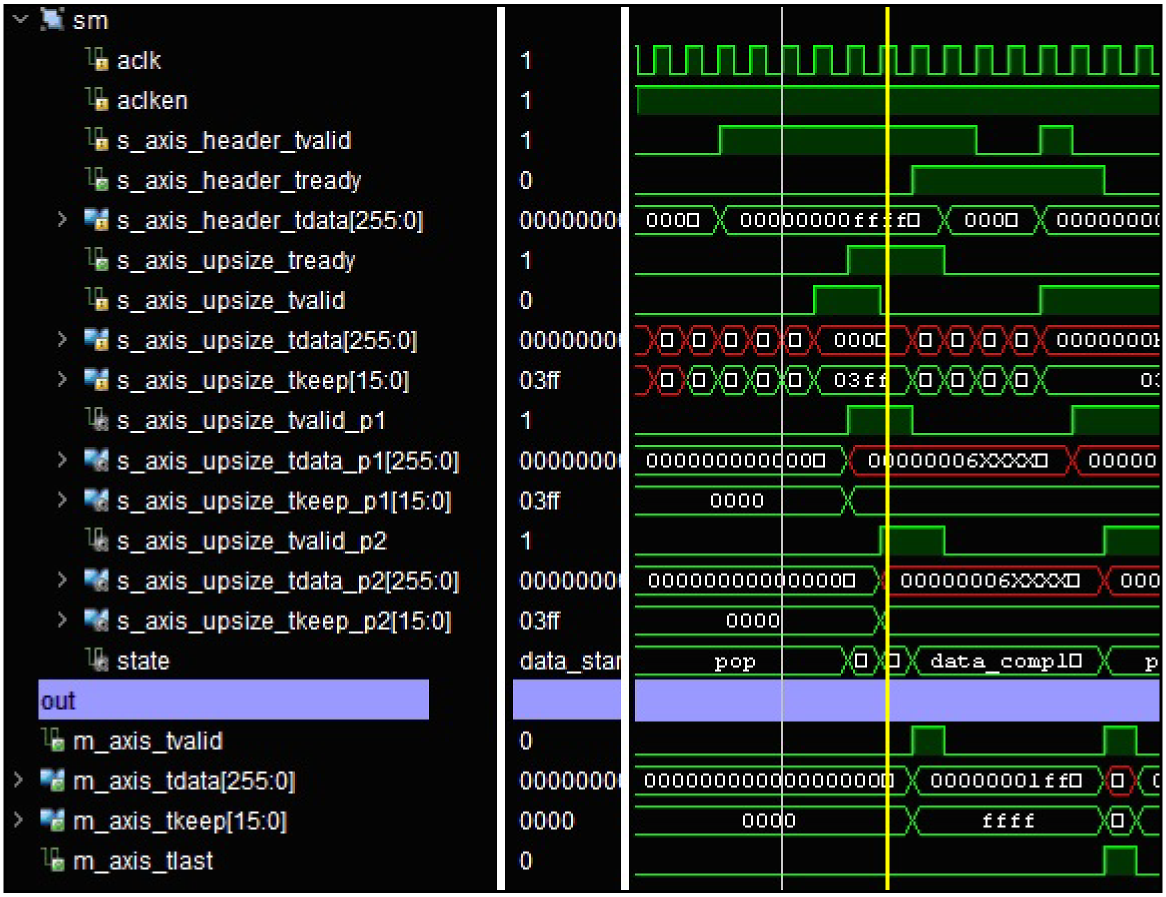


Figure 6–4: AXI4–Stream VITA 49 Packetizer Core Test Bench Simulation Input with Maximum data set to 0x00001000

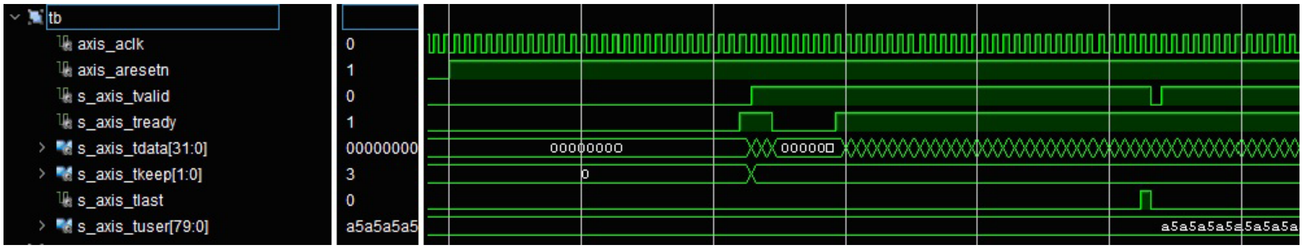
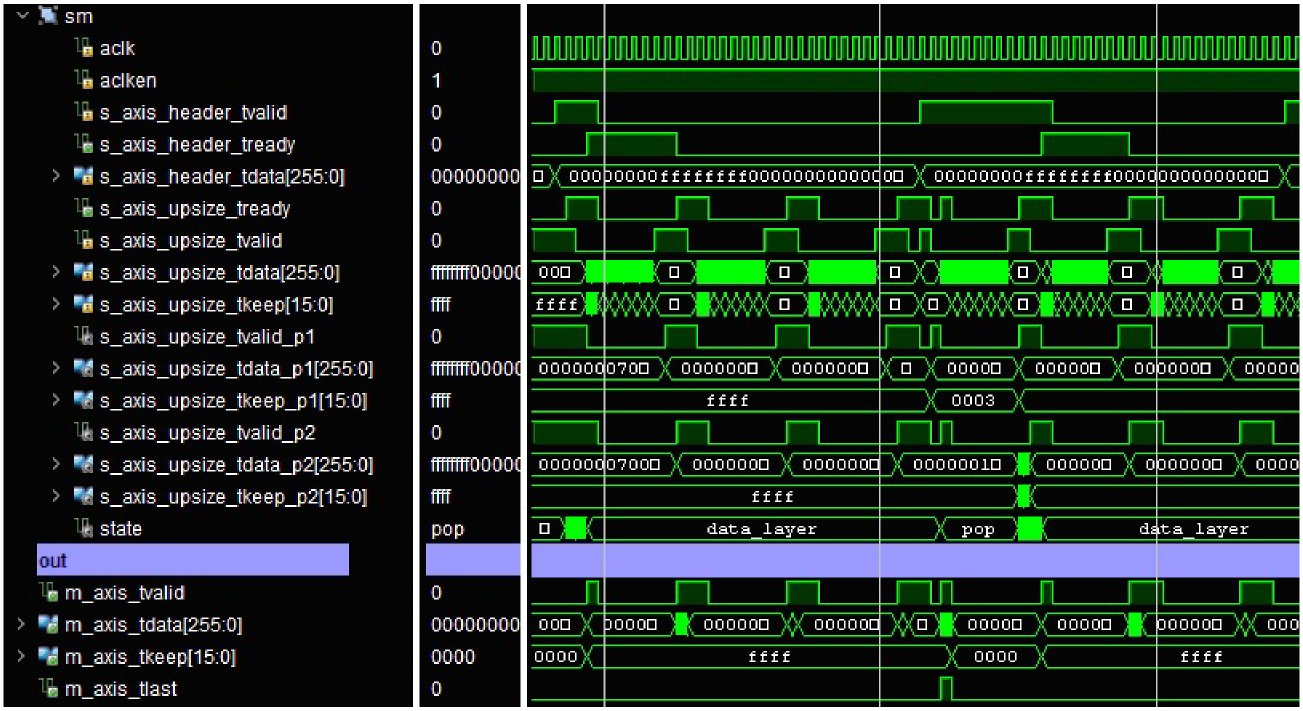


Figure 6–5: AXI4–Stream VITA 49 Packetizer Core Test Bench Simulation Output


sm
aclk 0
aclken 1
s_axis_header_tvalid 0
s_axis_header_tready 0
s_axis_header_tdata[255:0] 00000000
s_axis_upsize_tready 0
s_axis_upsize_tvalid 0
s_axis_upsize_tdata[255:0] ffffffff00000
s_axis_upsize_tkeep[15:0] ffff
s_axis_upsize_tvalid_p1 0
s_axis_upsize_tdata_p1[255:0] ffffffff00000
s_axis_upsize_tkeep_p1[15:0] ffff
s_axis_upsize_tvalid_p2 0
s_axis_upsize_tdata_p2[255:0] ffffffff00000
s_axis_upsize_tkeep_p2[15:0] ffff
state pop
out
m_axis_tvalid 0
m_axis_tdata[255:0] 00000000
m_axis_tkeep[15:0] 0000
m_axis_tlast 0
00000000ffffffff0000000000000
00000000ffffffff0000000000000000
ffff
0003
data_layer
pop
data_layer
00000007000
00000010
0000
ffff

6.6 Synthesis and Implementation

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

This page is intentionally blank