Video over IP FEC Receiver v2.0

LogiCORE IP Product Guide

Vivado Design Suite

PG207 October 5, 2016

Table of Contents

Introduction

The Xilinx® LogiCORE™ IP Video over IP FEC Receiver is a broadcast application module that recovers lost packets from RTP encapsulated payloads using SMPTE ST 2022-1, -5 Forward Error Correction methods. It is capable of handling a large number of media streams in 1 Gb/s networks (ST 2022-1,2) or limited number of media streams in 10 Gb/s networks (ST 2022-5,6). This core is used for developing Internet Protocol-based systems that reduce the overall cost of distribution and routing of audio and video data.

Features

- SMPTE ST 2022-1 or SMPTE ST 2022-5 based FEC recovery
- 256 or 512 channels supported in ST 2022-1/2 mode
- 4 or 8 channels supported in ST 2022-5/6 mode
- FEC Recovery per channel
- Statistic counters per channel
    - Valid packets
    - Unrecoverable packets
    - Recovered packets
    - Duplicate packets
    - Out-of-Range packets

LogiCORE IP Facts Table	
Core Specifics	
Supported Device Family[1]	UltraScale+™, UltraScale, Zynq®-7000, Virtex®-7, Kintex®-7
Supported User Interfaces	AXI4-Lite, AXI4-Stream, AXI-4
Resources	See Resource Utilization.
Provided with Core	
Design Files	Encrypted HDL
Example Design	System Verilog
Test Bench	System Verilog
Constraints File	XDC
Simulation Model	Encrypted RTL
Supported S/W Driver[2]	N/A
Tested Design Flows[3]	
Design Entry	Vivado® Design Suite
Simulation	For supported simulators, see the Xilinx Design Tools: Release Notes Guide.
Synthesis	Vivado Synthesis
Support	
Provided by Xilinx at the Xilinx Support web page	

Notes:

1. For a complete list of supported devices, see the Vivado IP catalog.
2. Standalone driver details can be found in the SDK directory (<*install_directory*>/doc/usenglish/xilinx_drivers.htm). Linux OS and driver support information is available from the Xilinx Wiki page.
3. For the supported versions of the tools, see the Xilinx Design Tools: Release Notes Guide.

Overview

As broadcast and communications markets converge, and the use of IP networks for transport of video streams becomes more attractive to broadcasters and telecommunication companies alike, the adoption of Ethernet for the transmission of multiple compressed media streams is becoming a major customer requirement. The industry is primarily looking at the SMPTE ST 2022 set of standards to create an open and interoperable way of connecting video over Ethernet equipment together and ensuring that Quality of Service (QoS) is high and packet loss is kept to a minimum or recovered through FEC. As shown in Figure 1-1, Video over IP FEC cores currently aim at multiple transport streams carried over 1 GbE or 10 GbE networks.

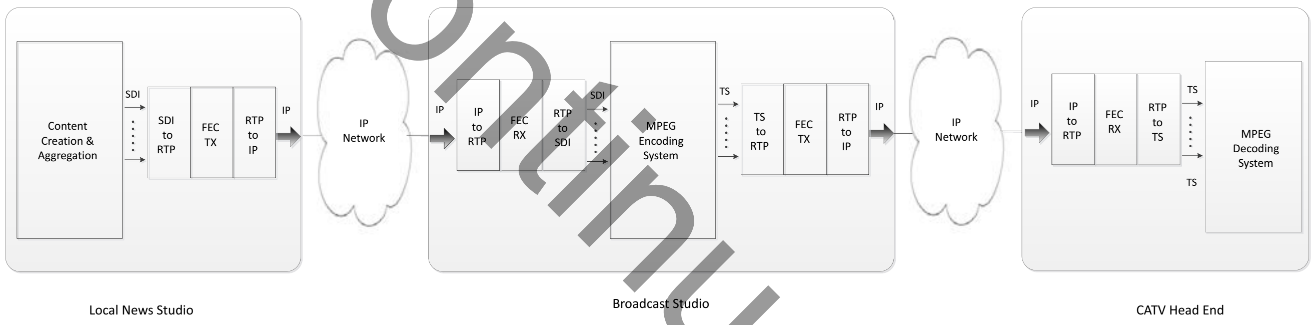


Figure 1-1: **Video over IP FEC Usage in Contribution and Distribution Networks**

The Video over IP FEC cores include Forward Error Correction (FEC). FEC protects the payload during the transport over IP networks. With FEC, the transmitter adds systematically generated redundant data. This carefully designed redundancy allows the receiver to detect and correct a limited number of packet errors occurring anywhere in the video without the need to ask the transmitter for additional video data. These errors, in the form of lost video packets, can be caused by many reasons, from thermal noise to storage system defects and transmission noise introduced by the environment. FEC gives the receiver the ability to correct these errors without needing a reverse channel to request retransmission of data. In real time systems, the latency is too great to request a retransmission. The ability of Xilinx® FPGAs to bridge the broadcast and the communications industries by providing a modular design with highly integrable interfaces helps broadcasters reduce costs as well as reduce the overall time it takes to acquire, edit and produce content. Now that video can be reliably delivered over Ethernet, broadcasters can replace some of the expensive mobile infrastructures supporting outside live broadcasts, as well as enable remote production from existing fixed studio set ups, which dramatically reduces both capital expenditure and operating expenses.

Feature Summary

The Video over IP FEC Receiver v2.0 core is a FEC recovery module that takes in SMPTE ST 2022 RTP encapsulated packets. It supports both SMPTE ST 2022-1, -2, -5, -6, and -7 standards. The core input and output are AXI4-Stream interfaces. Connection to the memory is via AXI4 while the core registers are accessed using AXI4-Lite interface. There are statistics counters that collect the status of the packets coming in and out of the core.

Applications

- Transport compressed constant bit rate MPEG-2 transport streams over IP networks.
- Transport uncompressed high bandwidth professional video streams over IP networks.
- Support real-time audio/video applications such as contribution, primary distribution, and digital cinema.

Licensing and Ordering Information

This Xilinx LogiCORE™ IP module is provided under the terms of the Xilinx Core License Agreement. The module is shipped as part of the Vivado® Design Suite. For full access to all core functionalities in simulation and in hardware, you must purchase a license for the core. Contact your local Xilinx sales representative for information about pricing and availability.

For more information, visit the **Video over IP FEC Receiver v2.0** product web page.

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

License Checkers

If the IP requires a license key, the key must be verified. The Vivado® design tools have several license checkpoints for gating licensed IP through the flow. If the license check succeeds, the IP can continue generation. Otherwise, generation halts with error. License checkpoints are enforced by the following tools:

- Vivado Synthesis
- Vivado Implementation
- write_bitstream (Tcl command)

IMPORTANT: *IP license level is ignored at checkpoints. The test confirms a valid license exists. It does not check IP license level.*

Product Specification

Architecture Overview

Figure 2-1 shows an overview of the Video over IP FEC Receiver core architecture.

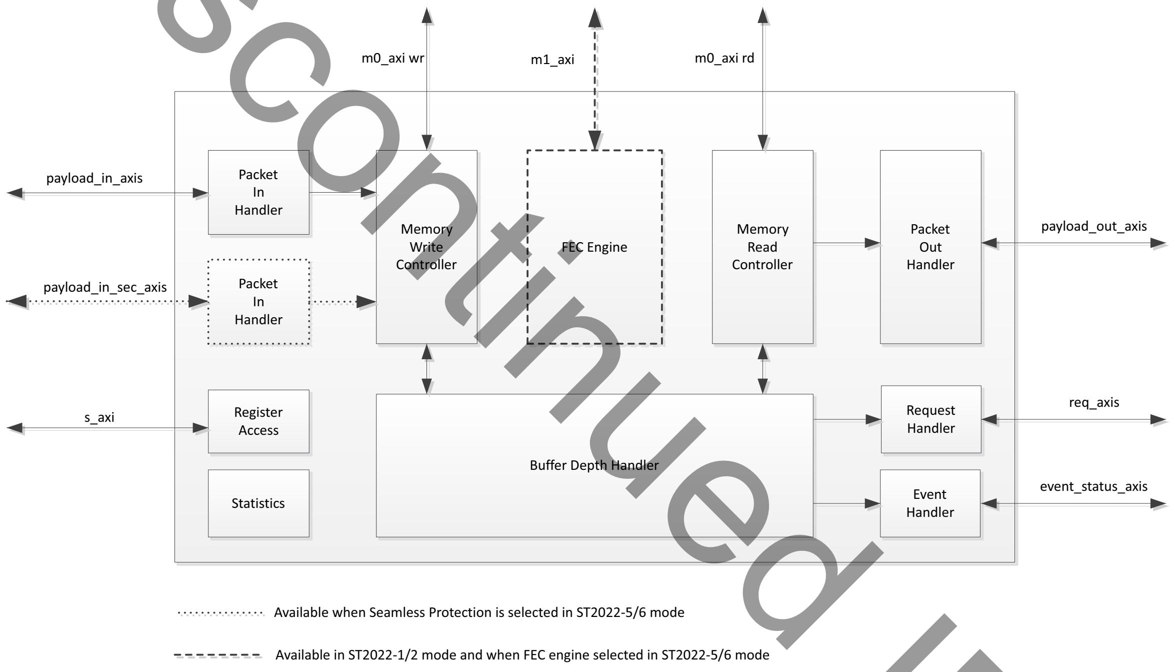


Figure 2-1: **Architecture Overview of Video over IP FEC Receiver**

The main functional blocks of the core are:

Packet In Handler: Demultiplex the RTP encapsulated packets into channel streams.

Memory Write Controller: Puts RTP encapsulated packets into the DDR memory buffer.

Memory Read Controller: Retrieves RTP encapsulated packets from the DDR memory buffer.

Packet Out Handler: Convert the payload of the packets into AXI-Stream format

Buffer Depth Handler: Update the buffer level of the packets in the DDR.

Request Handler: Takes in the request for packets or status from the user.

Event Handler: Sends out events of packets arrival or status to the user

FEC Engine: Performs FEC recovery on packet loss in the DDR memory buffer.

Register Access: Register configuration and status read-back on the core.

Statistics Counters: Tracks the status of the core.

Standards

The Video over IP Receiver core is compliant with the AXI4, AXI4-Stream and AXI4-Lite interconnect standards. See the Video IP: AXI Feature Adoption section of the *Vivado AXI Reference Guide* (UG1037) [Ref 6] for additional information.

The function of the core is compliant with SMPTE ST 2022-1, -2, -5, -6, and -7 standards.

Performance

Maximum Frequencies

The maximum achievable clock frequency and all resource counts can be affected by other tool options, additional logic in the FPGA device, different versions of Xilinx® tools, and other factors.

Resource Utilization

For details about resource utilization, visit Performance and Resource Utilization.

Resource utilization is calculated using `core_clk` and the frequency of other clock fixed at `s_axi_aclk`=100 MHz. The maximum clock frequency results were obtained by double-registering input and output ports to reduce dependence on I/O placement. The inner level of registers used a separate clock signal to measure the path from the input registers to the first output register through the core. The results are post-implementation, using tool default settings except for high effort.

The resource usage results do not include the "characterization" registers and represent the true logic used by the core. LUT counts include SRL16s or SRL32s.

Clock frequency does not take clock jitter into account and should be de rated by an amount appropriate to the clock source jitter specification. The maximum achievable clock frequency and the resource counts might also be affected by other tool options, additional logic in the FPGA, using a different version of Xilinx® tools, and other factors.

Port Descriptions

The Video over IP FEC Receiver core uses industry-standard control and data interfaces to connect to other system components. The following sections describe the various interfaces available with the core. Figure 2-2 provides an I/O diagram of the core.

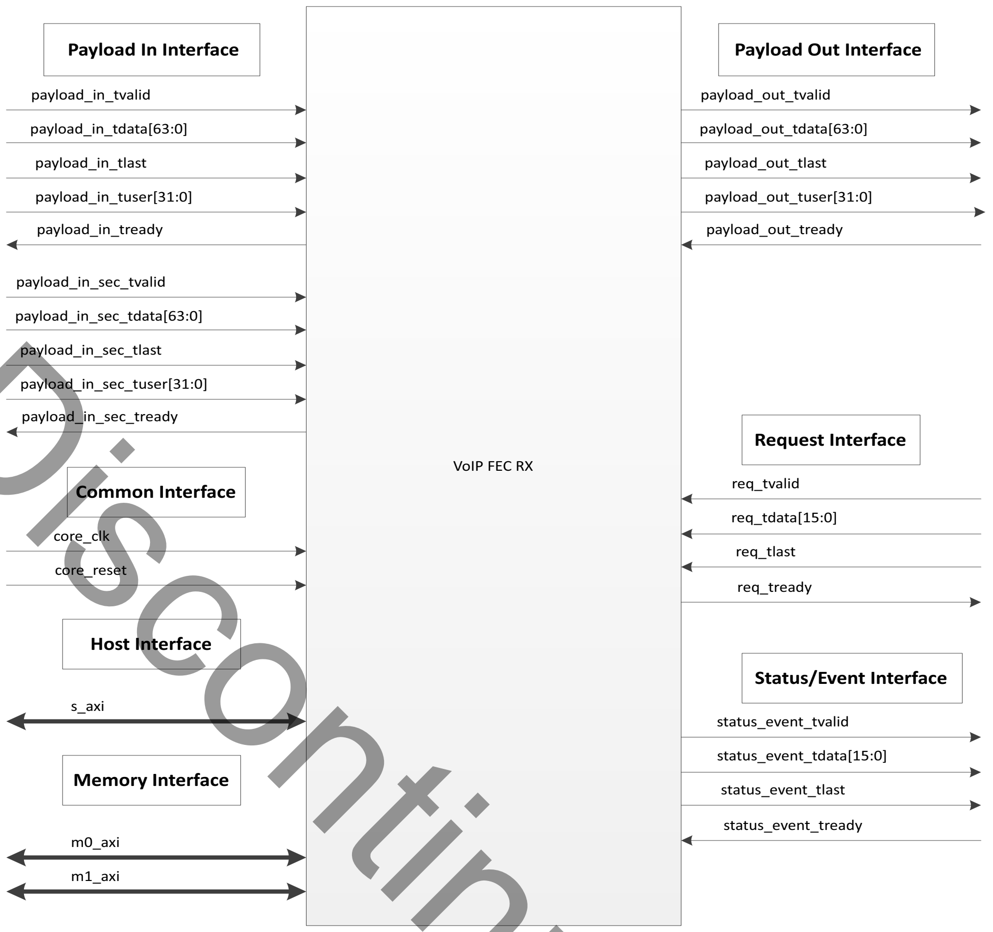


Figure 2-2: **Core I/O**

Notes:

1. payload_in_sec_* interface is available when Seamless Protection is selected in ST 2022-5/6 mode.
2. m1_axi interface is available in ST 2022-1/2 mode and when FEC engine selected in ST 2022-5/6 mode.

Common Interface

Table 2-1 describes the common signals.

Table 2-1: **Common Signals Description**

Signals	Direction	Width	Description
core_clock	In	1	The main operating clock for the core. It applies to all the interfaces except the host.
core_reset	In	1	The main reset for the core. It applies to payload_in, payload_out, request and status/event interfaces. It is synchronous to core_clock and is active high.

Payload In Interface

Table 2-2 describes the incoming stream to the core. payload_in_* is the primary AXI4-Stream input. payload_in_sec_* is the secondary AXI4-Stream input, only available in ST 2022-5/6 mode, when Seamless Protection is selected.

Table 2-2: **Receiver Payload In Signals**

Signals	Direction	Width	Description
payload_in_tvalid	In	1	High only from the start to the end of the packet transfer.
payload_in_tdata	In	64	Packet Data
payload_in_tlast	In	1	High only at the last word of the output packet

Table 2-2: **Receiver Payload In Signals *(Cont’d)***

Signals	Direction	Width	Description			
payload_in_tuser	In	32	Bit	Abbreviation	Description	
			0	Packet Start	High only at the first valid word of the input packet.	
			2:1	Protocol Version	Protocol version (set to “00”)	
			14:3	Channel Number	Shall be valid at packet start	
			15	Reserved		
			26:16	Packet Length	Shall be valid at packet start. It is the sum of packet length in bytes.	
			27	Reserved		
			31:28	Packet Type	Shall be valid at packet start	
					Value	Description
					0000	UDP encapsulated
					0001	RTP encapsulated ST2022-2 compliant media packet
					0010	RTP encapsulated ST2022-1 compliant Column FEC
					0011	RTP encapsulated ST2022-1 compliant Row FEC packet
					0101	RTP encapsulated ST2022-6 compliant media packet
					0110	RTP encapsulated ST2022-5 compliant Column packet
					0111	RTP encapsulated ST2022-5 compliant Row FEC packet
					1000	RTP encapsulated RFC4175 compliant media packet
					1001	RTP encapsulated RFC3190 compliant media packet
payload_in_tready	Out	1	Asserted low in between packet transfers.			

Notes:

1. The first packet byte is payload_in_tdata [7:0] of the first word transfer.

2. The maximum packet transfer length possible is approximately 2KB.

Table 2-3: **Receiver Secondary Payload In Signals**

ST2022-5/6 Mode: Include Seamless Protection Enabled			
Signals	**Direction**	**Width**	**Description**
payload_in_sec_tvalid	In	1	High only from the start to the end of the packet transfer.
payload_in_sec_tdata	In	64	Packet Data
payload_in_sec_tlast	In	1	High only at the last word of the output packet

Table 2-3: **Receiver Secondary Payload In Signals *(Cont'd)***

ST2022-5/6 Mode: Include Seamless Protection Enabled						
Signals	**Direction**	**Width**	**Description**			
payload_in_sec_tuser	In	32	Bit	Abbreviation	Description	
			0	Packet Start	High only at the first valid word of the input packet.	
			2:1	Protocol Version	Protocol version (set to "00")	
			14:3	Channel Number	Shall be valid at packet start	
			15	Reserved		
			26:16	Packet Length	Shall be valid at packet start. It is the sum of packet length in bytes.	
			27	Reserved		
			31:28	Packet Type	Shall be valid at packet start	
					Value	Description
					0000	UDP encapsulated
					0001	RTP encapsulated ST2022-2 compliant media packet
					0010	RTP encapsulated ST2022-1 compliant Column FEC
					0011	RTP encapsulated ST2022-1 compliant Row FEC packet
					0101	RTP encapsulated ST2022-6 compliant media packet
					0110	RTP encapsulated ST2022-5 compliant Column packet
					0111	RTP encapsulated ST2022-5 compliant Row FEC packet
					1000	RTP encapsulated RFC4175 compliant media packet
					1001	RTP encapsulated RFC3190 compliant media packet
payload_in_sec_tready	Out	1	Asserted low in between packet transfers.			

Payload Out Interface

Table 2-4 describes the outgoing stream from the core.

Table 2-4: **Receiver Stream Payload Out Signals**

Signals	Direction	Width	Description			
payload_out_tvalid	Out	1	High only from the start to the end of the packet transfer.			
payload_out_tdata	Out	64	Packet Data			
payload_out_tlast	Out	1	High only at the last word of the output packet			
payload_out_tuser	Out	32	Bit	Abbreviation	Description	
			0	Packet Start	High only at the first valid word of the input packet.	
			2:1	Protocol Version	Protocol version (set to “00”)	
			14:3	Channel Number	Shall be valid at packet start	
			15	Reserved		
			26:16	Packet Length	Shall be valid at packet start. It is the sum of packet length in bytes.	
			27	Reserved		
			31:28	Packet Type	Shall be valid at packet start	
					Value	Description
					0000	UDP encapsulated packet
					0001	RTP encapsulated ST2022-2 compliant media packet
					0101	RTP encapsulated ST2022-6 compliant media packet
					1000	RTP encapsulated RFC4175 compliant media packet
					1001	RTP encapsulated RFC3190 compliant media packet
payload_out_tready	In	1	Asserted low in between packet transfers.			

Request Interface

Table 2-5: **Request Signals**

Signals	Direction	Width	Description
req_tvalid	In	1	High only from the start to the end of the request.
req_tdata	In	16	Request data
req_tlast	In	1	High only at the last word of the request
req_tready	Out	1	May be asserted Low in between requests.

The `req_tdata` signal contains REQ_ID and channel number. Refer to Figure 3-10 for details.

Table 2-6: **Request ID**

Req ID	Request	Description
1	Payload	Request packet for a particular channel.
2	Buffer depth	Request buffer depth of a particular channel.

Status/Event Interface

Table 2-7: **Status/Event Signals Description**

Signals	Direction	Width	Description
status_event_tvalid	Out	1	High only from the start to the end of the status/event transfer.
status_event_tdata	Out	16	Status/Event data.
status_event_tlast	Out	1	High only at the last word of the status/event.
status_event_tready	In	1	May be asserted low in between events.

The `status_event_tdata` signal contains EVENT ID, channel number, buffer depth, FEC L and FEC D values. Refer to Figure 3-9 for details.

Table 2-8: **Status/Event ID**

Event ID	Event	Description
1	Received Packet	Generated when a packet is received for a particular channel.
2	Empty Buffer	Generated when a request is received for particular channel, but no packets in the buffer for that channel.
3	Channel Status	Generated when a request is received on enquiring buffer depth of a particular channel (Req ID 2).

Memory Interface (m0_axi and m1_axi)

The memory interface uses an AXI4 interface to connect to an AXI4 interconnect which provides the access to the external memory through an AXI4 DDR controller. The AXI4 interface data width is 256. See the *LogiCORE IP AXI Interconnect Product Guide* (PG059) for more information.

Table 2-9: **AXI4 Memory Interface Signals**

Signal Name	Direction	Width	Description
m0_axi_awid	Out	1	Write Address Channel Transaction ID.
m0_axi_awaddr	Out	32	Write Address Channel Address.
m0_axi_awlen	Out	8	Write Address Channel Burst Length code.
m0_axi_awsize	Out	3	Write Address Channel Transfer Size code.
m0_axi_awburst	Out	2	Write Address Channel Burst Type.
m0_axi_awlock	Out	2	Write Address Channel Atomic Access Type.
m0_axi_awcache	Out	4	Write Address Channel Cache Characteristics.
m0_axi_awprot	Out	3	Write Address Channel Protection Bits.
m0_axi_awqos	Out	4	Write Address Channel Quality of Service.
m0_axi_awvalid	Out	1	Write Address Channel Valid.
m0_axi_awready	In	1	Write Address Channel Ready.
m0_axi_wdata	Out	256	Write Data Channel Data.
m0_axi_wstrb	Out	32	Write Data Channel Data Byte Strobes.
m0_axi_wlast	Out	1	Write Data Channel Last Data Beat.
m0_axi_wvalid	Out	1	Write Data Channel Valid.
m0_axi_wready	In	1	Write Data Channel Ready.
m0_axi_bid	In	1	Write Response Channel Transaction ID.
m0_axi_bresp	In	2	Write Response Channel Response Code.
m0_axi_bvalid	In	1	Write Response Channel Valid.
m0_axis_bready	Out	1	Write Response Channel Ready.
m0_axi_arid	Out	1	Read Address Channel Transaction ID.
m0_axi_araddr	Out	32	Read Address Channel Address
m0_axi_arlen	Out	8	Read Address Channel Burst Length code.
m0_axi_arsize	Out	3	Read Address Channel Transfer Size code.
m0_axi_arburst	Out	2	Read Address Channel Burst Type.
m0_axi_arlock	Out	2	Read Address Channel Atomic Access Type.
m0_axi_arcache	Out	4	Read Address Channel Cache Characteristics.
m0_axi_arprot	Out	3	Read Address Channel Protection Bits.

Table 2-9: **AXI4 Memory Interface Signals *(Cont'd)***

Signal Name	Direction	Width	Description
m0_axi_arqos	Out	4	AXI4 Read Address Channel Quality of Service.
m0_axi_arvalid	Out	1	Read Address Channel Valid.
m0_axi_arready	In	1	Read Address Channel Ready.
m0_axi_rid	In	1	Read Data Channel Data Transaction ID.
m0_axi_rdata	In	256	Read Data Channel Data.
m0_axi_rresp	In	2	Read Data Channel Response Code.
m0_axi_rlast	In	1	Read Data Channel Last Data Beat.
m0_axi_rvalid	In	1	Read Data Channel Valid.
m0_axi_rready	Out	1	Read Data Channel Ready.
m1_axi_awid	Out	1	Write Address Channel Transaction ID.
m1_axi_awaddr	Out	32	Write Address Channel Address.
m1_axi_awlen	Out	8	Write Address Channel Burst Length code.
m1_axi_awsize	Out	3	Write Address Channel Transfer Size code.
m1_axi_awburst	Out	2	Write Address Channel Burst Type.
m1_axi_awlock	Out	2	Write Address Channel Atomic Access Type.
m1_axi_awcache	Out	4	Write Address Channel Cache Characteristics.
m1_axi_awprot	Out	3	Write Address Channel Protection Bits.
m1_axi_awqos	Out	4	Write Address Channel Quality of Service.
m1_axi_awvalid	Out	1	Write Address Channel Valid.
m1_axi_awready	In	1	Write Address Channel Ready.
m1_axi_wdata	Out	256	Write Data Channel Data.
m1_axi_wstrb	Out	32	Write Data Channel Data Byte Strobes.
m1_axi_wlast	Out	1	Write Data Channel Last Data Beat.
m1_axi_wvalid	Out	1	Write Data Channel Valid.
m1_axi_wready	In	1	Write Data Channel Ready.
m1_axi_bid	In	1	Write Response Channel Transaction ID.
m1_axi_bresp	In	2	Write Response Channel Response Code.
m1_axi_bvalid	In	1	Write Response Channel Valid.
m1_axis_bready	Out	1	Write Response Channel Ready.
m1_axi_arid	Out	1	Read Address Channel Transaction ID.

Table 2-9: **AXI4 Memory Interface Signals *(Cont'd)***

Signal Name	Direction	Width	Description
m1_axi_araddr	Out	32	Read Address Channel Address
m1_axi_arlen	Out	8	Read Address Channel Burst Length code.
m1_axi_arsize	Out	3	Read Address Channel Transfer Size code.
m1_axi_arburst	Out	2	Read Address Channel Burst Type.
m1_axi_arlock	Out	2	Read Address Channel Atomic Access Type.
m1_axi_arcache	Out	4	Read Address Channel Cache Characteristics.
m1_axi_arprot	Out	3	Read Address Channel Protection Bits.
m1_axi_arqos	Out	4	AXI4 Read Address Channel Quality of Service.
m1_axi_arvalid	In	1	Read Address Channel Valid.
m1_axi_arready	In	1	Read Address Channel Ready.
m1_axi_rid	In	1	Read Data Channel Data Transaction ID.
m1_axi_rdata	In	256	Read Data Channel Data.
m1_axi_rresp	In	2	Read Data Channel Response Code.
m1_axi_rlast	In	1	Read Data Channel Last Data Beat.
m1_axi_rvalid	In	1	Read Data Channel Valid.
m1_axi_rready	Out	1	Read Data Channel Ready.

Host Interface (S_AXI)

The host interface allows user to control core parameters. Core configuration can be done using an embedded ARM or soft system processor such as MicroBlaze™. This AXI4-Lite slave interface facilitates integrating the core into a processor system, or along with other video or AXI4-Lite compliant IP, connected via AXI4-Lite interface to an AXI4-Lite master. See the *LogiCORE IP AXI Interconnect Product Guide* (PG059) for more information.

Table 2-10: **Host Interface Signals**

Signal Name	Direction	Width	Description
s_axi_clk	In	1	AXI4-Lite Clock.
s_axi_aresetn	In	1	AXI4-Lite Active-Low Reset.
s_axi_awaddr	In	9	AXI4-Lite Write Address Bus
s_axi_awvalid	In	1	AXI4-Lite Write Address Channel Write Address Valid.
s_axi_wdata	In	32	AXI4-Lite Write Data Bus
s_axi_wstrb	In	4	AXI4-Lite Write Data Channel Data Byte Strobes.

Table 2-10: **Host Interface Signals *(Cont'd)***

Signal Name	Direction	Width	Description
s_axi_wvalid	In	1	AXI4-Lite Write Data Channel Write Data Valid.
s_axi_awready	Out	1	AXI4-Lite Write Address Channel Write Address Ready. Indicates DMA ready to accept the write address.
s_axi_wready	Out	1	AXI4-Lite Write Data Channel Write Data Ready. Indicates DMA is ready to accept the write data.
s_axi_bresp	Out	2	AXI4-Lite Write Response Channel. Indicates results of the write transfer.
s_axi_bvalid	Out	1	AXI4-Lite Write Response Channel Response Valid. Indicates response is valid.
s_axi_bready	In	1	AXI4-Lite Write Response Channel Ready. Indicates target is ready to receive response.
s_axi_arvalid	In	1	AXI4-Lite Read Address Channel Read Address Valid
s_axi_arready	Out	1	Ready. Indicates DMA is ready to accept the read address.
s_axi_araddr	In	9	AXI4-Lite Read Address Bus
s_axi_rready	In	1	AXI4-Lite Read Data Channel Read Data Ready. Indicates target is ready to accept the read data.
s_axi_rdata	Out	32	AXI4-Lite Read Data Bus
s_axi_rresp	Out	2	AXI4-Lite Read Response Channel Response. Indicates results of the read transfer.
s_axi_rvalid	Out	1	AXI4-Lite Read Data Channel Read Data Valid

Register Space

The registers are categorized into two sections, the general space and channel space. The registers in the general space apply to all the channels while the channel space registers apply only to the specific channel set by register offset 0x08.

The general registers can be access through normal address read and write.

To configure a channel, observe the following steps.

1. Before configuring a channel, check the busy bit (bit 0, register offset 0x04) to be low.
2. Set the channel to be configured at register offset, 0x08.
3. Configure the channel specific registers.
4. Pulse channel update bit (bit 1, register offset 0x00) to commit the channel parameters.
5. Repeat steps 1-4 for another channel. See Figure 2-3.

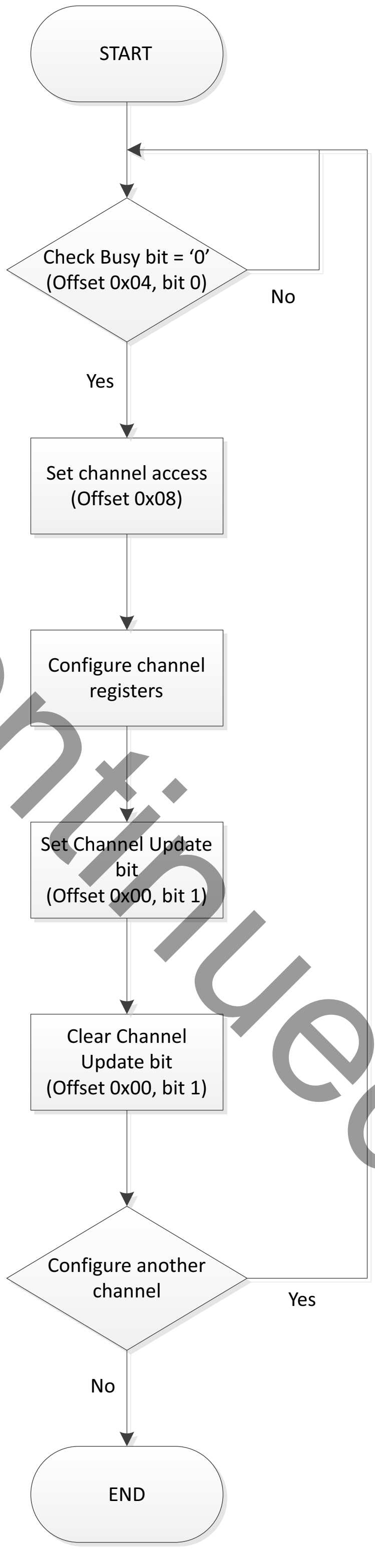


Figure 2-3: **Configuration of Channel Registers Flowchart**

To read off a channel register, set the channel access at offset 0x08 before proceeding.

Table 2-11 describes the core registers.

Table 2-11: **Registers**

Address Offset (HEX)	Register Name	Access Type	Default Value (HEX)	Register Description	
				Bit Range	**Value**
GENERAL					
0x00	control	R/W	0x00000000	Control	
				31:3	Reserved
				2	**Clear Channel** Reset the current channel which is configured on channel_access (0x08).
				1	**Channel Update** Allows configured channel registers to take effect for the current channel which is configured on channel_access (0x08).
				0	Reserved. Set at 0.
0x04	status	R	0x00000000	Status	
				31:1	Reserved
				0	**Update Busy** 1 - Core busy updating configured parameters or clearing internal buffer 0 - Core able to accept new configuration
0x08	channel_access	R/W	0x00000000	Channel access	
				31:12	Reserved
				11:0	The channel number to access registers
0x0C	sys_config	R	Value based on GUI selection	System configuration	
				31:18	Reserved
				17	Seamless switching supported
				16	FEC recovery supported
				15:0	Number of channels supported

Table 2-11: **Registers *(Cont'd)***

Address Offset (HEX)	Register Name	Access Type	Default Value (HEX)	Register Description	
				Bit Range	Value
0x10	version	R	0x02000000	Hardware version	
				31:24	Version major
				23:16	Version minor
				15:12	Version revision
				11:8	Patch ID
				7:0	Revision number
0x20	fec_processing_delay	R/W	0x00000000	FEC processing delay	
				31:0	Time delay for the incoming FEC packets to be processed. (Based on core_clock ticks)
0x24	fec_pkt_drop_cnt	R	0x00000000	FEC packet drop count	
				31:0	Number of unprocessed FEC packets discarded due to FEC payload buffer full.
CHANNEL					
0x80	chan_conf	R/W	0x00000000	Channel configuration	
				31:3	Reserved
				2	FEC recovery disable 1 - FEC recovery off 0 - FEC recovery on
				1	Media packet bypass 1 - Media packet is First In First Out 0 - Media packet is being reordered
				0	Channel enable 1 - Enable channel 0 - Disable channel.
0x90	valid_pkt_cnt	R	0x00000000	Valid Packet Count	
				31:0	Number of valid packets received in the channel
0x94	unrecv_ pkt_cnt	R	0x00000000	Unrecovered Packet Count	
				31:0	Number of missing packets in the channel

Table 2-11: **Registers *(Cont'd)***

Address Offset (HEX)	Register Name	Access Type	Default Value (HEX)	Register Description	
				Bit Range	Value
0x98	corr_ pkt_cnt	R	0x00000000	Corrected Packet Count	
				31:0	Number of corrected packets in the channel
0x9C	dup_ pkt_cnt	R	0x00000000	Duplicated Packet Count	
				31:0	Number of duplicated packets that are discarded
0xA8	channel_status	R	0x00000000	Channel Status	
				31:22	Reserved
				21	Row FEC detected
				20	Column FEC detected
				19:10	FEC D value detected
				9:0	FEC L value detected
0xAC	curr_buffer_depth	R	0x00000000	Current Buffer Depth	
				31:17	Reserved
				16:0	Expected number of media packets buffered in the DDR
0xB0	oor_ pkt_cnt	R	0x00000000	Out-of-range Packet Count	
				31:0	Number of discarded packets not falling the range supported by the media buffer size
0xB4	oor_ts_offset	R/W	0x00000000	Out-of-range Timestamp Offset	
				31	Enable OOR checking using RTP timestamp. It is important to enable this check when the media buffer size is configured to 65536 packets.
				30:0	Set time difference between outgoing packet and incoming packet for the OOR checking range (Based on RTP timestamp ticks). Any incoming packet beyond this range is discarded.
0xB8	link_ts_diff	R	0x00000000	Link Timestamp Difference	
				31:0	RTP timestamp difference between incoming media packets currently received in primary and secondary links (Based on RTP timestamp ticks)

Register Description

control (0x000) Register

Channel_update (bit 1) is a write-done semaphore for the host processor, which facilitates committing all user register updates in the channel space simultaneously. One set of registers (the processor registers) is directly accessed by the processor interface, while the other set (the active set) is actively used by the core. New values written to the processor registers are copied over to the active set if and only if the register update bit is set. Setting the bit to 0 before updating multiple registers and then setting the bit to 1 when updates are completed ensures all channel space registers are updated simultaneously.

An active high pulse to Clear_channel (bit 2) clear the channel to accept a new payload stream. The channel is set in Channel Access at offset 0x08.

status (0x004) Register

The update_busy bit asserts when the core is updating the new configuration or clearing internal buffer. Do not configure the core if this bit is high.

channel_access (0x008) Register

This register is used when accessing channel space registers. Always set the channel first as shown in Figure 2-3.

sys_config (0x00C) Register

Readback register for user to know if the core has being implemented to support FEC recovery, seamless protection, and the number of channels

version (0x010) Register

Bit fields of the register facilitate software identification of the exact version of the hardware peripheral incorporated into a system. The core driver can take advantage of this read-only value to verify that the software is matched to the correct version of the hardware.

fec_processing_delay (0x020) Register

Set time delay for incoming FEC packets before processing for recovery in order to cater scenario such as packets arriving out of order. Value is count based on core clock tick.

fec_pkt_drop_cnt (0x024) Register

Due to some unforeseen circumstances such as high DDR latency, FEC packets processing may get delay which result in the FEC payload buffer getting full. In this case, new FEC packets will be dropped and the counter increases. Counter clears itself upon read.

chan_conf (0x080) Register

If media_packet_bypass bit is set, FEC_recovery_disable bit is ignored. Packet recovery is not performed and incoming FEC packets are discarded. Media packets are pulled out in a First In, First Out fashion.

If media_packet_bypass bit is low, FEC_recovery_disable bit can be cleared for the core to perform FEC recovery. Media packet are reordered depending on the buffer depth maintained for the channel.

valid_pkt_cnt (0x090) Register

Valid packet count increments when a packet belonging to the channel is received. Channel register clears itself when read.

unrecv_pkt_cnt (0x094) Register

Unrecoverable packet count increments when a media packet is missing in the channel stream. Channel register clears itself when read.

corr_pkt_cnt (0x098) Register

Corrected packet count increments when a media packet is being recovered in the channel stream. Channel register clears itself when read.

dup_pkt_cnt (0x09C) Register

Duplicated packet count increments when a channel received a media packet that already exists in the media buffer. This duplicated media packet is discarded. Channel register clears itself when read.

channel_status (0x0A8) Register

This register reflects the FEC matrix size detected for the channel. Only detected FEC column packet update the FEC L and D value.

- Bit [9:0] - L value of the FEC matrix detected
- Bit [19:10] - D value of the FEC matrix detected

- Bit [20] - High indicates column FEC received
- Bit [21] - High indicates row FEC received

curr_buffer_depth (0x0AC) Register

Current number of media packets being buffered for channel.

oor_pkts_cnt (0x0B0) Register

Out-of-range packet count increments when an incoming media packet is discarded due to it being an earlier packet compared to the outgoing media packet from the core. Channel register clears itself when read.

oor_ts_offset (0x0B4) Register

Bit [30:0] - sets the timestamp range to discard packets that comes into the core, not having its RTP timestamp falling between the boundaries.

Incoming packets with RTP timestamp within the range below are accepted.

Outgoing packet RTP timestamp < Incoming packet RTP timestamp < Outgoing packet RTP timestamp + OOR TS OFFSET

Bit [31] - enables timestamp range checking

Note: It is important to enable this check when the media buffer size is set to 65536. The oor_ts_offset is 65535 * RTP timestamp interval of the incoming stream.

link_ts_diff (0x0B8) Register

Useful in determining the timestamp difference between two links when seamless protection is enabled in ST 2022-5/6. Not relevant for other configurations.

Designing with the Core

Figure 3-1 shows an example of an application design using Video over IP FEC RX core with other Xilinx IP.

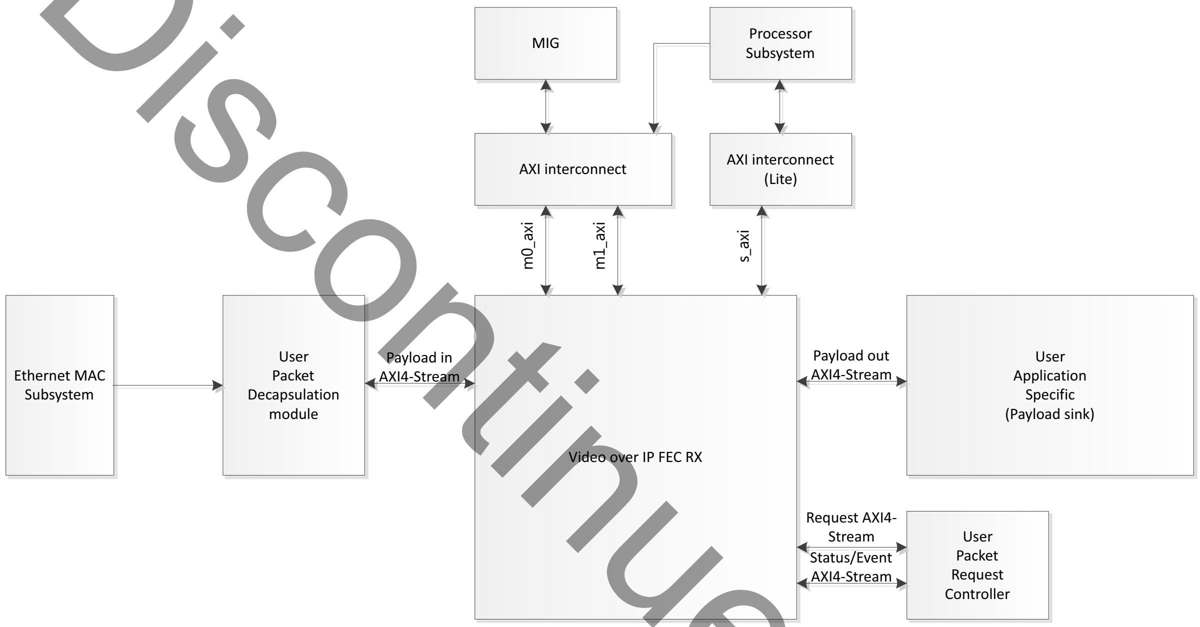


Figure 3-1: **Example of an Application Design with Video over IP FEC RX Core**

This section describes how Video over IP FEC Receiver core can be design in to build a fully functional design with user application logic.

The core takes in SMPTE ST 2022 encapsulated payloads for FEC recovery. Figure 3-2 to Figure 3-6 show the types of incoming payload data format acceptable to RX. When a packet is received from the Ethernet, a user packet decapsulation module is required to decode which stream the packet belongs to and also to strip down the packet to an RTP encapsulated payload. This data is then fed to the Video over IP FEC Receiver core through an AXI4-Stream transaction. Figure 3-7 shows the waveform of the transaction together with the sideband information.

WORD\BYTE POSITION	7 downto 0	15 downto 8	23 downto 16	31 downto 24	39 downto 32	47 downto 40	55 downto 48	63 downto 56
1	Media payload							
2								
...								
...								
...								

Figure 3-2: **UDP Encapsulated Payload**

WORD\BYTE POSITION	7 downto 0	15 downto 8	23 downto 16	31 downto 24	39 downto 32	47 downto 40	55 downto 48	63 downto 56
1	V(2)/P(1)/X(1)/CC(4)	M(1)/PT(7)	sequence number		timestamp			
2	SSRC identifier				Media payload			
...								
...								
...								

Figure 3-3: **RTP Encapsulated ST 2022-2/RFC4175/RFC3190 Compliant Media Packet**

TE POSITION	7 downto 0	15 downto 8	23 downto 16	31 downto 24	39 downto 32	47 downto 40	55 downto 48	63 downto 56
1	V(2)/P(1)/X(1)/CC(4)	M(1)/PT(7)	sequence number		timestamp			
2	SSRC identifier				SNBase low bits		Length Recovery	
3	E/PT recovery	Mask			TS recovery			
4	N/D/type/index	offset	NA	SNBase ext bits	FEC payload			
...								
...								
...								

Figure 3-4: **RTP Encapsulated ST 2022-1 Compliant FEC Packet**

WORD\BYTE POSITION	7 downto 0	15 downto 8	23 downto 16	31 downto 24	39 downto 32	47 downto 40	55 downto 48	63 downto 56
1	V(2)/P(1)/X(1)/CC(4)	M(1)/PT(7)	sequence number		timestamp			
2	SSRC identifier				Ext(4)/F(1)/VSID(3)	FR Count	R(2)/S(2)/FEC(3)/CF	CF(3)/Reserved(5)
3	Video source format				Video timestamp (CF > 0)			
...	Media payload							
...								
...								

Figure 3-5: **RTP Encapsulated ST 2022-6 Compliant Media Packet**

WORD\BYTE POSITION	7 downto 0	15 downto 8	23 downto 16	31 downto 24	39 downto 32	47 downto 40	55 downto 48	63 downto 56
1	V(2)/P(1)/X(1)/CC(4)	M(1)/PT(7)	sequence number		timestamp			
2	SSRC identifier				E/R/P/X/CC(4)/M	PT Recovery	SN Base	
3	TS Recovery				Length Recovery		Reserved	
4	Offset	Offset(2)/Reserved(6)	NA	Reserved	FEC payload			
...								
...								
...								

Figure 3-6: **RTP Encapsulated ST 2022-5 Compliant FEC Packet**

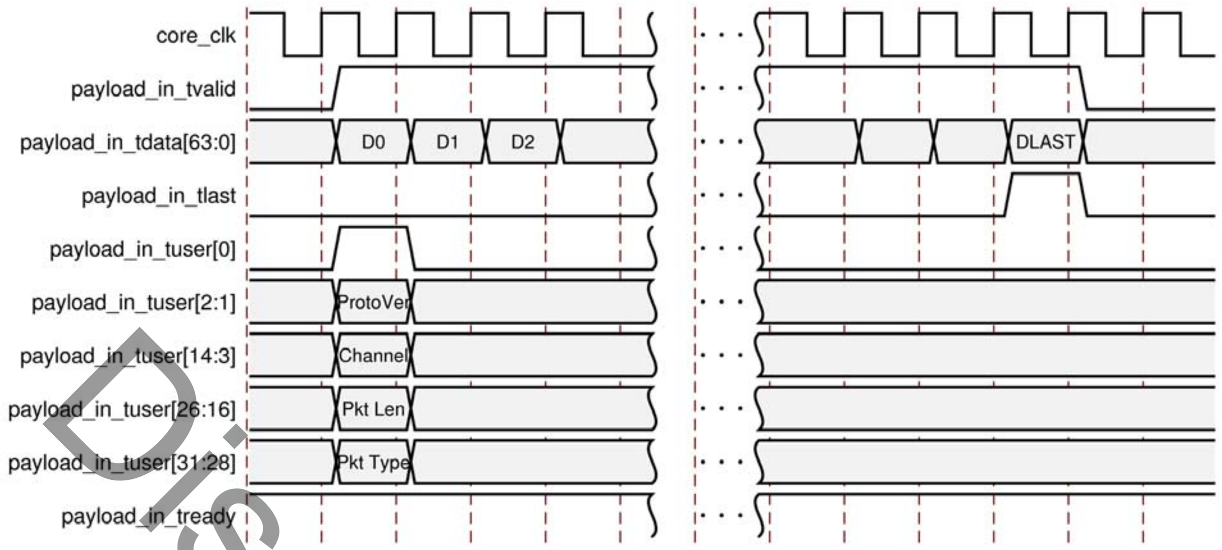


Figure 3-7: **Payload in AXI4-Stream Waveform**

The Video over IP FEC Receiver core requires access to DDR memory for normal operation and to perform FEC recovery. The core stores incoming packets into DDR and puts media and FEC payloads at different sections of the memory. The user has to set 2 base addresses in the GUI, one for the media payload buffer and the other for FEC payload buffer. Note that the memory space of media payload buffer should not overlap FEC payload buffer. The total size of the media payload buffer is based on the number of channels selected in the GUI. The FEC payload buffer is fixed at 4MB. The memory requirement and bandwidth consumption are given below.

Memory Requirements

Each valid incoming payload packet is allocated a fixed size of 2 kB when stored in the DDR memory. Based on 512 channels and media buffer size of 256 packets, DDR memory requirement for media packets is 256 MB. (512 channels * 256 packets * 2 kB).

In addition, the Video over IP FEC Receiver core requires 2048 packets storage for FEC payloads and the memory buffer requirement is 4 MB. (2048 packets * 2 kB).

Memory Bandwidth Consumption

The maximum bandwidth consumption of each AXI4 memory map port to the DDR is given in Table 3-1 and Table 3-2.

Table 3-1: **ST2022-1/2 Mode Bandwidth Utilization in 1Gbps Ethernet**

Bandwidth Utilization (Gbps)			
m0_axi WRITE	**m0_axi READ**	**m1_axi WRITE**	**m1_axi READ**
1	1	0.5	1

Table 3-2: **ST2022-5/6 Mode Bandwidth Utilization in 10Gbps Ethernet**

Bandwidth Utilization (Gbps)				
	m0_axi WRITE	**m0_axi READ**	**m1_axi WRITE**	**m1_axi READ**
FEC with Seamless	20	10	5	20
FEC only	10	10	5	20
Seamless only	20	10	0	0
No FEC No Seamless	10	10	0	0

Other than the input AXI4-Stream, the FEC RX core has three other AXI4-Stream interfaces connecting the user. The first is `payload_out` interface bus. It is symmetrical to the `payload_in` interface. Only media and recovered payloads come out of the `payload_out` AXI-Stream interface. There is 1 signal different at the `payload_out` interface, that is, unrecoverable packet indicator, `payload_out_tuser[15]`. This signal is High when the outgoing packet is non-existent in the core and the payload data is all zeros. Figure 3-8 shows the waveform of an outgoing packet transaction. The output data is in the form of RTP encapsulated SMPTE 2022 compliant media payload.

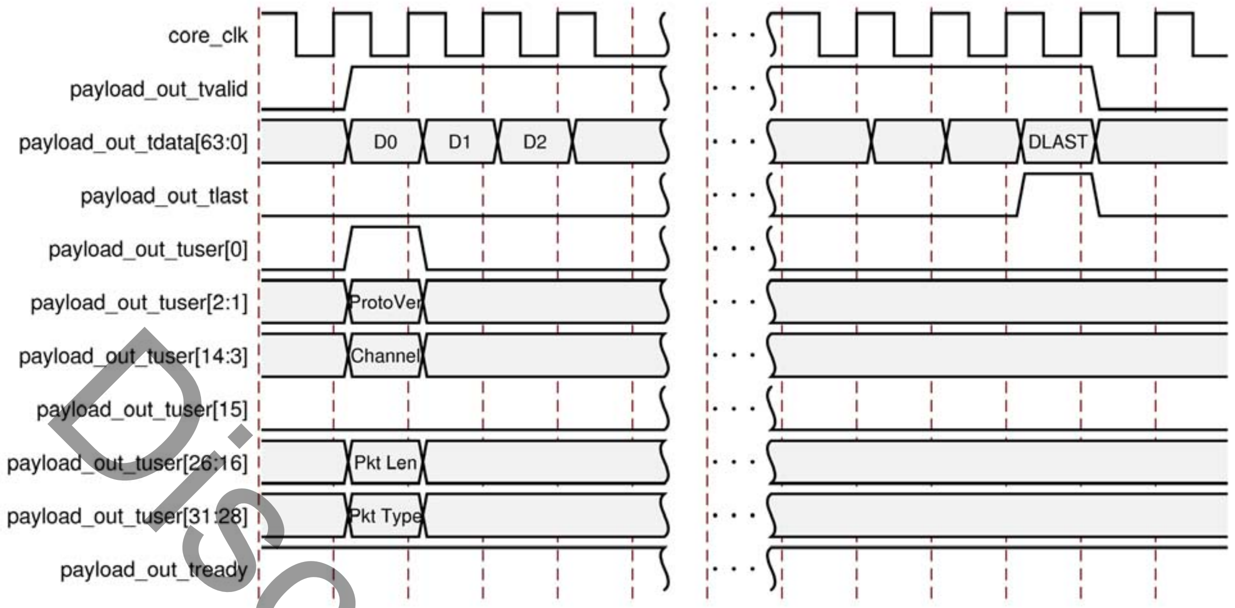


Figure 3-8: **Payload Out AXI4-Stream Waveform**

Whenever there is an incoming valid packet into the FEC RX, the core processes the packet and sends out an event through the `status_event` AXI4-Stream interface. This event informs the user which channel has newly arrived payload, the current accumulated packets in the core for this channel, and if there is any FEC packet detected on a certain matrix size.

Other than a received packet event as mention above, there are two more types of events generated by the core as described in Table 2-7. Figure 3-9 illustrates an event transaction.

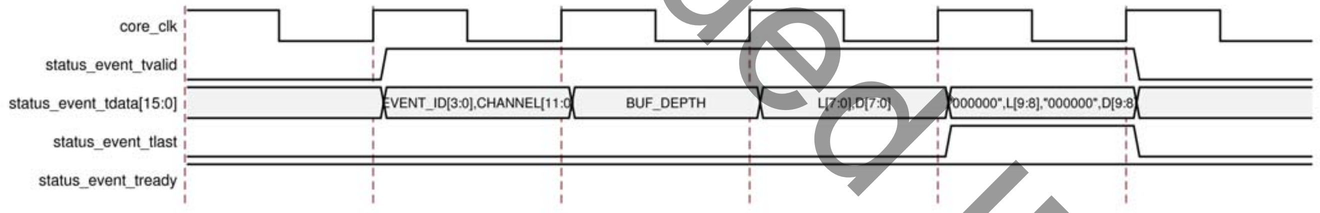


Figure 3-9: **Status/Event AXI4-Stream Waveform**

EVENT_ID[3:0]: Explained in Table 2-7.

CHANNEL[11:0]: Channel number of this event.

BUF_DEPTH[15:0]: Current accumulated packets for the channel.

L[9:0]: L value for the FEC matrix size detected. Extracted from the 3rd and 4th cycles of the AXI4-stream transaction.

D[9:0]: D value for the FEC matrix size detected. Extracted from the 3rd and 4th cycles of the AXI4-stream transaction.

At a suitable level of packets accumulated per channel, user can pull the packets out from the core via the request AXI4-Stream interface. Figure 3-10 shows how this can be done and Table 2-5 describes the types of request user can send to the core. Requests are served in a first in, first out basis by the core.

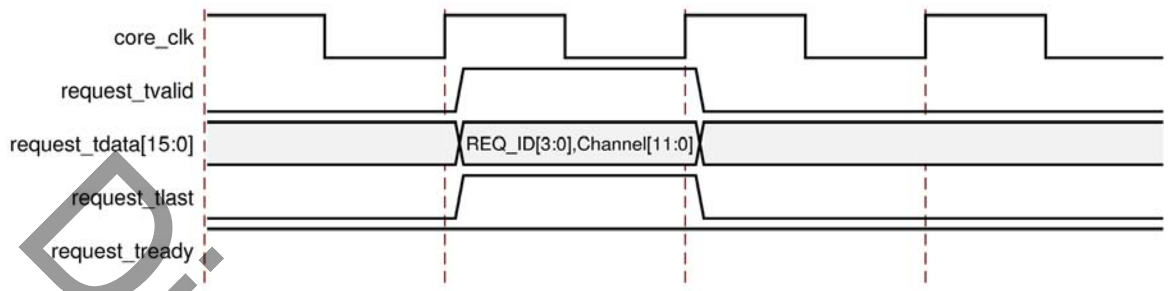


Figure 3-10: **Request AXI4-Stream Waveform**

The amount of packets buffered per channel depends on user requirement. However, to allow proper FEC recovery to be performed on loss packets, a minimum size of L*D*2 must be maintain at the buffer depth.

To allow seamless switching to initiate properly, it is important to start the core having input streams for both `payload_in` and `sec_payload_in` links. A media buffer depth larger than the links difference plus the network jitter has to be taken into account. If the two links are not maintained in the beginning, you must ensure that the media buffer size planned for the channel is larger than the initial maintained buffer depth plus the links difference and the network jitter. If you need to perform FEC recovery on top of seamless switching, the media buffer depth maintained must be larger than the links difference, plus the network jitter, plus L*D*2. In Figure 3-1, a user packet request controller may be built to monitor the events coming from the core and at the same time control the buffer depth level for each active channel.

User Packet Request Controller is a module to request packet out from the Video over IP FEC Receiver v2.0 core. It is recommended that you observe the following points when designing this module.

1. The controller should consistently monitor the Status/Event AXI4-Stream interface for Event ID 1 that contains information on the current media buffer depth, FEC L and FEC D per channel.
2. Request packet can start when the media buffer depth reach certain level based intended core operation. Refer to Figure 3-11.
   a. Neither FEC Recovery nor Seamless Switching operation are expected:

      current media buffer depth > user specified packet reordering depth

   b. Only FEC Recovery operation is expected:

current media buffer depth > (FEC L*FEC D*2) + FEC Processing Delay $_{\text{media packets}}$ + Network Jitter $_{\text{media packets}}$

c. Only Seamless Switching is expected:

current media buffer depth > Link Difference $_{\text{media packets}}$ + Network Jitter $_{\text{media packets}}$

d. Seamless switching and FEC Recovery operation are expected

current media buffer depth > Link Difference $_{\text{media packets}}$ + (FEC L*FEC D*2) + FEC Processing Delay $_{\text{media packets}}$ + Network Jitter $_{\text{media packets}}$

Notes:

1. Time based parameters are to be converted into media packet-based value using the expected channel packet rate.
   Time based parameters:
   - FEC Processing Delay
   - Link Difference
   - Network Jitter
2. Link Difference is the RTP time difference between identical incoming streams arriving at the payload_in (Primary Link) interface and payload_in_sec (Secondary Link) interface of the core.
3. FEC Processing Delay is value (in core clock ticks) set in register offset 0x20. Default value is 0.
4. FEC L * FEC D * 2, is expected number of media packets need to be stored for FEC recovery.

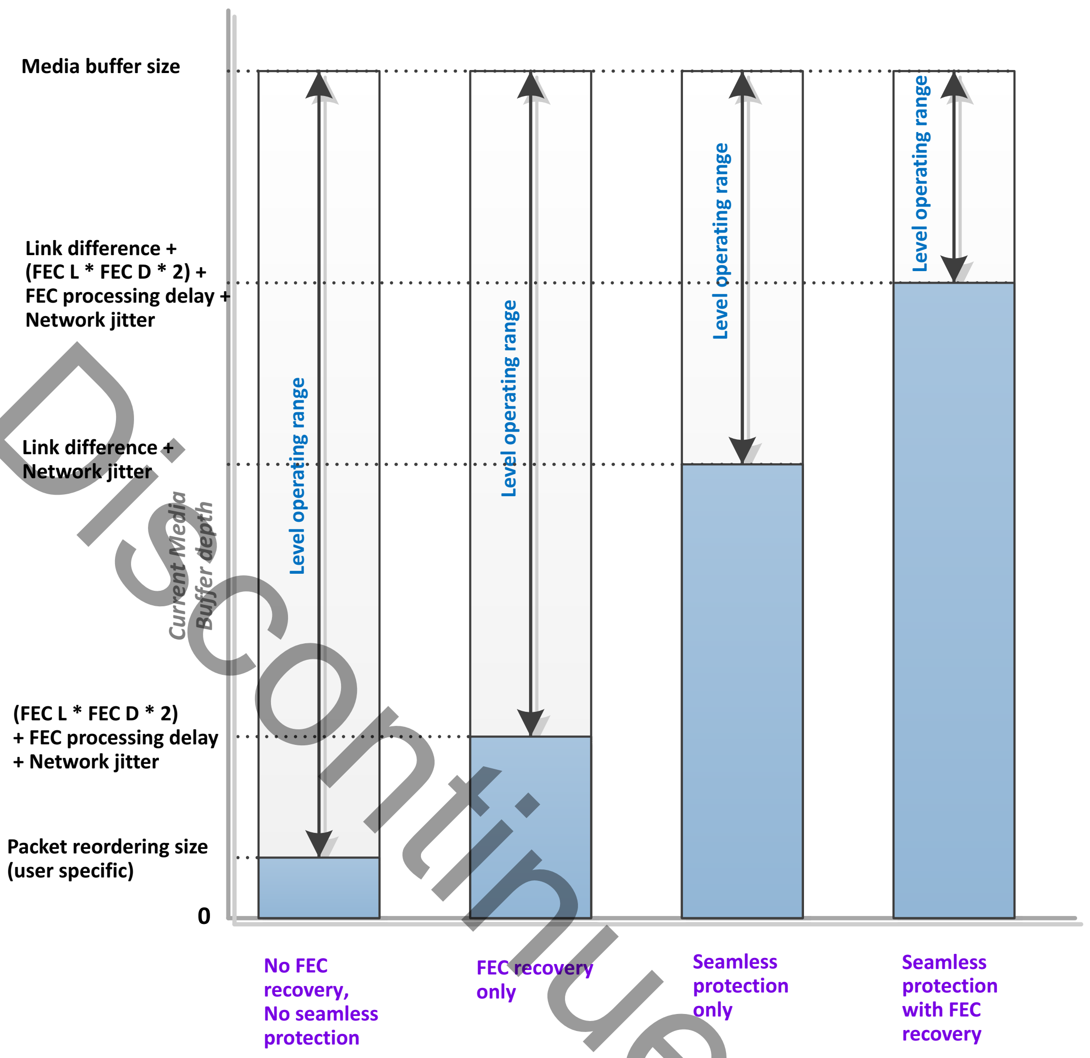
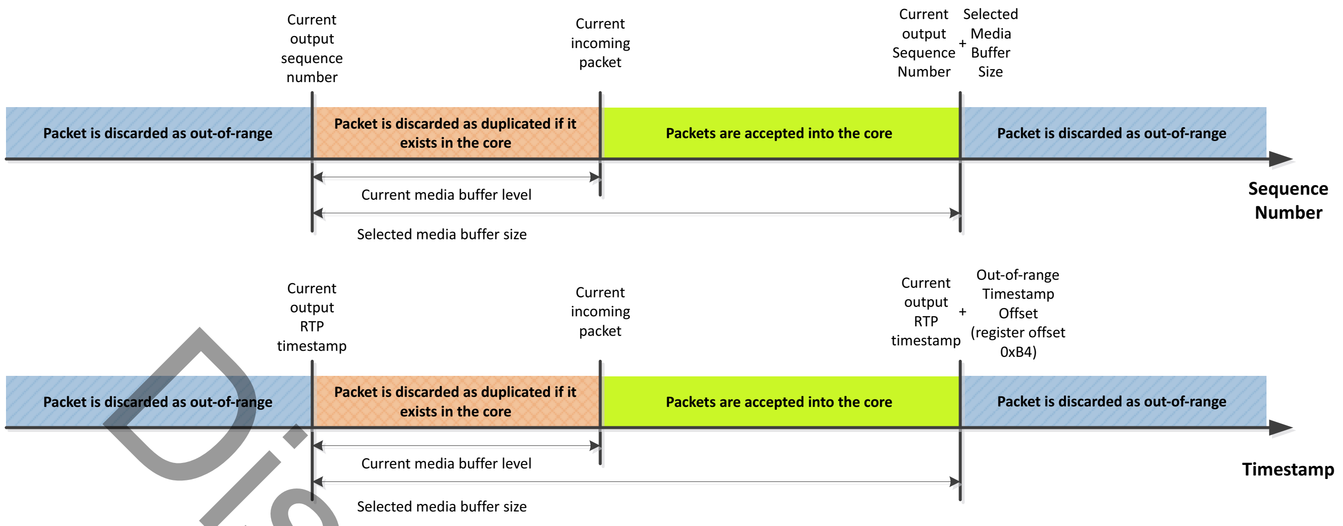


Figure 3-11: **Current Media Buffer Depth Maintained by the Controller**

Incoming Packet Handling

The Video over IP FEC Receiver core has a way of accepting incoming stream packets to ensure data integrity at the core output. Two types of checking are involved in the process, using RTP sequence number and RTP timestamp. RTP sequence number check is in placed all the time during the core operation. RTP timestamp check is user controlled. It is mandatory to set register offset 0xB4 when the media buffer size selection is 65536, to prevent data corruption happening within the core. Figure 3-12 illustrates the checking mechanisms.



Current output sequence number
Current incoming packet
Current output Sequence Number + Selected Media Buffer Size
Packet is discarded as out-of-range
Packet is discarded as duplicated if it exists in the core
Packets are accepted into the core
Packet is discarded as out-of-range
Current media buffer level
Selected media buffer size
Sequence Number
Current output RTP timestamp
Current incoming packet
Current output RTP timestamp + Out-of-range Timestamp Offset (register offset 0xB4)
Packet is discarded as out-of-range
Packet is discarded as duplicated if it exists in the core
Packets are accepted into the core
Packet is discarded as out-of-range
Current media buffer level
Selected media buffer size
Timestamp

Figure 3-12: **Incoming Packet Handling Mechanism**

FEC Packet Handling

The FEC recovery function of the Video over IP FEC Receiver core is a first-in, first-out, one pass process for the incoming FEC packets. This means the core considers for FEC recovery when it receives a FEC packet. It then checks the media packets stored within the DDR if there is 1 only and only 1 missing media packet in the row or column which the FEC packet protects. If that is the case, FEC correction starts. If there more missing media packets or all the media packets are available in the row or column, the FEC packet is discarded and not be used again. The effective recovery of the core against burst loss is L or less media packets of the received FEC matrix. For a burst loss greater than L, the amount of media packets that can be recovered may be less than L due to the way FEC packets are handled.

Channel Recovery

Any stream changes, disruption, or stoppage from the source in a particular channel warrants a channel recovery to be performed. In addition, any software configuration change on the fly to a particular channel (`chan_conf`, register offset 0x080) requires a channel recovery to also be performed.

During the channel recovery process, it is important to disable the channel first. This ensures that there is no activity at the input of the core for the channel. Channel recovery clears the channel's internal buffers and cease the channel operation. Status/event and request interface activity for the channel will react accordingly to the internal buffers' status. It is advisable that no request is being sent for the channel during channel recovery.

Note that other channels operation are unaffected. Figure 3-13 shows the channel recovery flow.

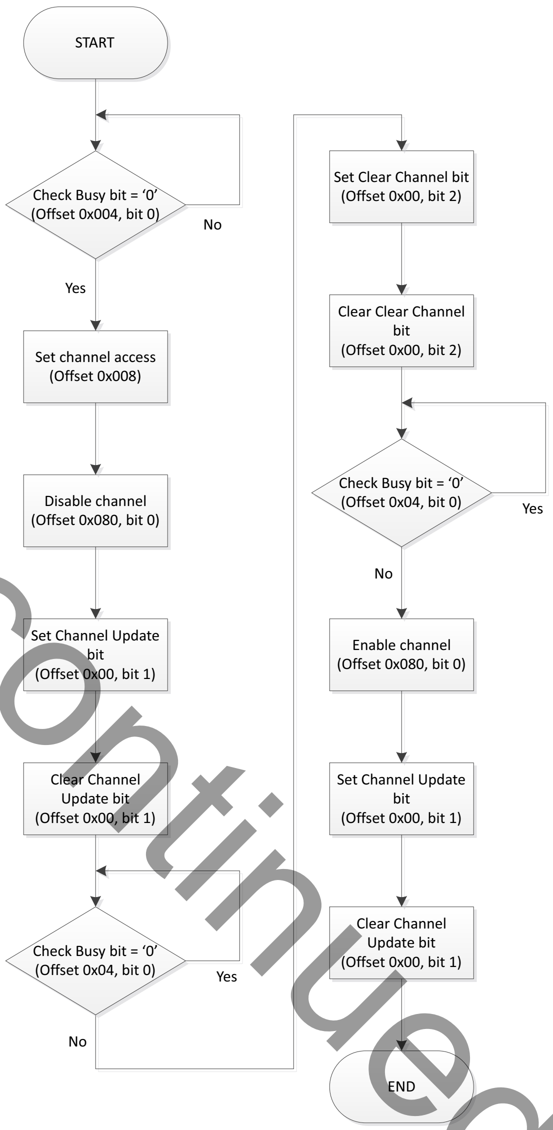

START
Check Busy bit = '0' (Offset 0x004, bit 0)
No
Yes
Set channel access (Offset 0x008)
Disable channel (Offset 0x080, bit 0)
Set Channel Update bit (Offset 0x00, bit 1)
Clear Channel Update bit (Offset 0x00, bit 1)
Check Busy bit = '0' (Offset 0x04, bit 0)
Yes
No
Set Clear Channel bit (Offset 0x00, bit 2)
Clear Clear Channel bit (Offset 0x00, bit 2)
Check Busy bit = '0' (Offset 0x04, bit 0)
Yes
No
Enable channel (Offset 0x080, bit 0)
Set Channel Update bit (Offset 0x00, bit 1)
Clear Channel Update bit (Offset 0x00, bit 1)
END

Figure 3-13: **Channel Recovery Flowchart**

Clocking

The VoIP FEC Receiver core has two clock domains:

- Core clock domain, `core_clk`.
  All the AXI4-Stream and AXI4 interfaces, along with the main core activities are under this clock. (Fmin for `core_clk` is arbitrarily set at 100 Mhz).

  For the core using in a 10G link environment, it is recommended to run the core clock at 200 Mhz.

- AXI4-Lite clock domain, `s_axi_aclk`.
  Core register access works with this clock. (Fmin for `s_axi_aclk` is set at 50 MHz.)

Resets

The VoIP FEC Receiver core has 2 resets:

- Core domain reset, `core_reset`.
- AXI4-Lite domain reset, `s_axi_aresetn`

The resets must be synchronous to their individual clock domains. A minimum of 16 clock cycles is recommended for the reset assertion. Reset signal `s_axi_aresetn` is to be de-asserted last.

Design Flow Steps

This chapter describes customizing and generating the core, constraining the core, and the simulation, synthesis and implementation steps that are specific to this IP core. More detailed information about the standard Vivado® design flows and the IP integrator can be found in the following Vivado Design Suite user guides:

- *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 4]
- *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 5]
- *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 7]
- *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 8]

Customizing and Generating the Core

This section includes information about using Xilinx tools to customize and generate the core in the Vivado® Design Suite.

If you are customizing and generating the core in the Vivado IP Integrator, see the *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) [Ref 4] for detailed information. IP Integrator might auto-compute certain configuration values when validating or generating the design. To check whether the values do change, see the description of the parameter in this chapter. To view the parameter value, run the `validate_bd_design` command in the Tcl console.

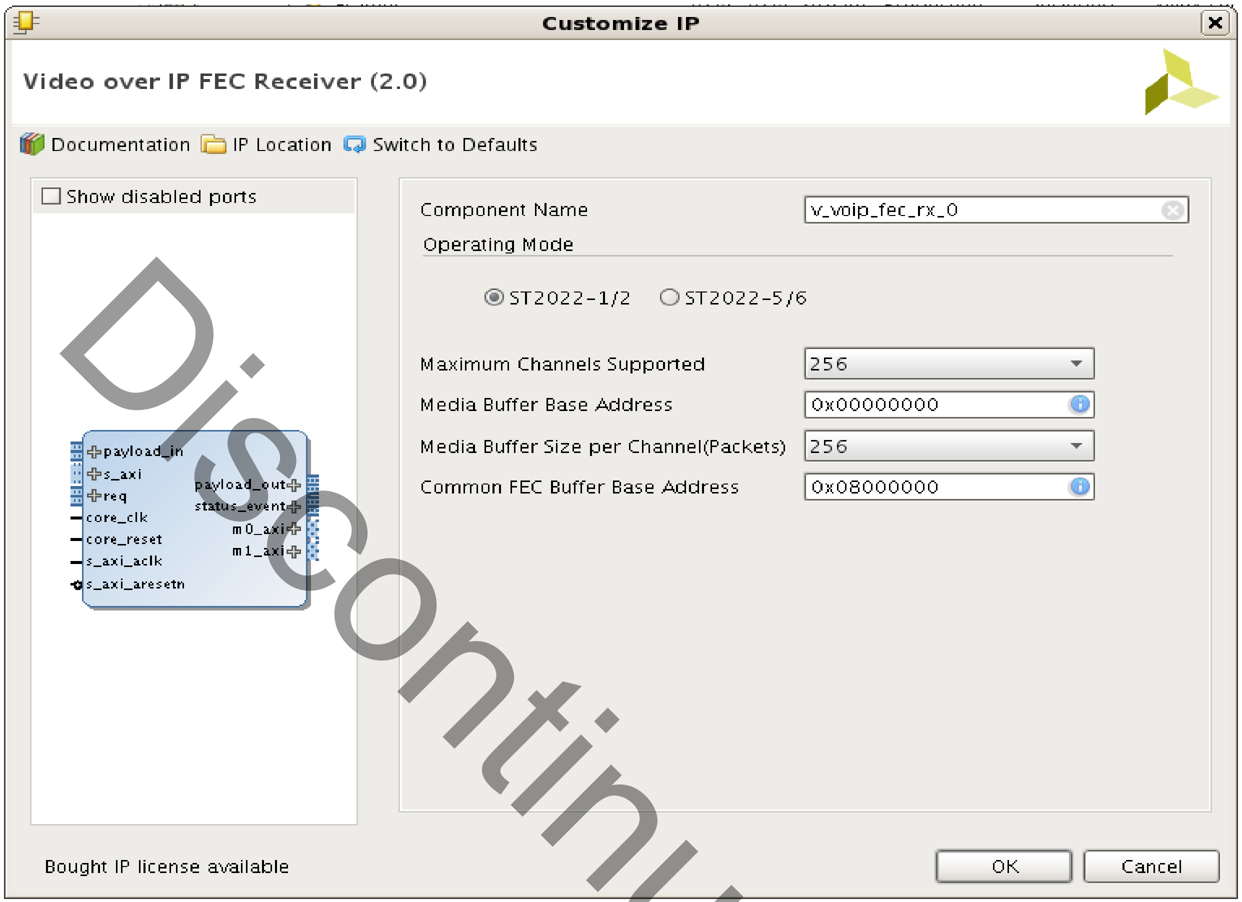
Vivado Integrated Design Environment

You can customize the IP for use in your design by specifying values for the various parameters associated with the IP core using the following steps:

1. Select the IP from the IP catalog.
2. Double-click the selected IP or select the Customize IP command from the toolbar or right-click menu.

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 5] and the *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 7].

Note: Figures in this chapter are illustrations of the Vivado IDE. The layout depicted here might vary from the current version.



Customize IP

Video over IP FEC Receiver (2.0)

Component Name: v_voip_fec_rx_0

Operating Mode: [✓ ST2022-1/2] ST2022-5/6

Maximum Channels Supported: 256

Media Buffer Base Address: 0x00000000

Media Buffer Size per Channel(Packets): 256

Common FEC Buffer Base Address: 0x08000000

Bought IP license available

Figure 4-1: **Video over IP FEC Receiver GUI**

The Vivado IDE displays a representation of the IP symbol on the left side and the parameter assignments on the right, as follows:

- **Component Name**: The base name of output files generated for the module. Names must begin with a letter and must be composed of characters a to z, 0 to 9 and "_". The name v_voip_fec_rx_v2_0 cannot be used as a component name.
- **Operating Mode**: Select the core operating mode. ST 2022-1/2, which core accepts SMPTE 2022-1/2 compliant packets. ST 2022-5/6, which core accepts SMPTE 2022-5/6 compliant packet.
- **Maximum Channels Supported**: Specifies the number of channels supported. The valid options for ST 2022-1/2 are 256 and 512 and ST 2022-5/6 operating modes are 4 and 8. (Note: Changing Maximum Channel Supported requires proper resetting of Media Buffer Base Address and Common FEC Buffer Base Address.
- **Media Buffer Base Address**: Specifies base address for media payload buffer. (Note: Changing Media Buffer Base Address requires proper re setting of Common FEC Buffer Base Address).

- **Include FEC Engine**: When checked, ST 2022-5 FEC Engine is generated in ST 2022-5/6 operating mode. The core is capable of recovering IP packets lost to network transmission errors. (Note: Selecting Include FEC Engine requires proper setting of Common FEC Buffer Base Address)
- **Include Seamless Protection**: When checked, the core is generated with secondary link in ST 2022-5/6 operating mode to support seamless operation.
- **Common FEC Buffer Base Address**: Specifies Common FEC Buffer Base Address for "Maximum Channels Supported", when FEC engine is included. In ST 2022-1/2 operating mode, FEC engine is always included. (Note: Changing Common FEC Buffer Base Address requires proper resetting of Media Buffer Base Address).

Configuring the GUI Parameters

Setting the FEC Buffer Base Address

$$Packet\ Buffer\ Size_{supp.\ channels} = Supported\ Channel * Packet\ Size_{Bytes} * Media\ Buffer\ Depth_{channel}$$

While setting the **FEC buffer base address**, should satisfy either of below rules:

1. $\boldsymbol{FEC\ buffer\ base\ address} \leq Media\ Buffer\ Base\ Address - FEC\ Buffer\ Pool\ Size$
2. $\boldsymbol{FEC\ buffer\ base\ address} \geq Media\ Buffer\ Base\ Address + Packet\ Buffer\ Size_{supp.\ channels}$

FEC Buffer Pool Size is (2048 Packets * 2048 Bytes)

Use Case

Table 4-1 shows the examples value while configuring Media buffer base address and FEC buffer base address which is based from the formula shown in Configuring the GUI Parameters.

Table 4-1: **Media Buffer Base Address, FEC Buffer Base Address, and FEC Buffer Pool Size Use Case**

Maximum Channels Supported	Media Buffer Base Address	Media Buffer Size per Channel (Packet)	Common FEC Buffer Base Address	Upper Ceiling of FEC Buffer Base Address (Non-Configurable)
Mode: ST2022-1/2				
256	0x0000_0000	1024	0x2000_0000	0x2040_0000
Mode: ST2022-5/6				
4	0x0000_0000	65536	0x2000_0000	0x2040_0000

Note: Assuming Memory Base Address starts at 0x0000_0000

User Parameters

Table 4-2 shows the relationship between the GUI fields in the Vivado IDE and the User Parameters (which can be viewed in the Tcl console).

Table 4-2: **User Parameters**

GUI Parameter/Value	User Parameter/Value	Default Value	
		ST2022-1/2	**ST2022-5/6**
Maximum Channels Supported	C_NUM_OF_CHANNELS	256	4
Common FEC Buffer Base Address	C_FEC_BASE_ADDR	0x08000000	0x00200000 (if FEC Engine included)
Media Buffer Base Address	C_MEDIA_BASE_ADDR	0x00000000	0x00000000

1. Parameter values are listed in the table where the GUI parameter value differs from the user parameter value. Such values are shown in this table as indented below the associated parameter.
2. The FEC and Media buffer base addresses should not overlap.

Output Generation

The Vivado design tools generate the files necessary to build the core and place those files in the `<project>/<project>.srcs/sources_1/ip/<core>` directory. For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 5].

Constraining the Core

This section contains information about constraining the core in the Vivado Design Suite.

Required Constraints

Constraints required for the core are clock frequency constraints for the clock domains described in Clocking in Chapter 3. Paths between the clock domains are constrained with a `max_delay` constraint and use the data path only flag, causing setup and hold checks to be ignored for signals that cross clock domains. These constraints are provided in the XDC constraints file included with the core.

Device, Package, and Speed Grade Selections

There are no device, package or speed grade requirements for this core. This core has not been characterized for use in low-power devices.

Clock Frequencies

See Maximum Frequencies in Chapter 2.

Clock Management

There are no clock management constraints for this core.

Clock Placement

There are no clock placement constraints.

Banking

There is no specific banking rule for this core.

Transceiver Placement

There is no transceiver placement constraints for this core.

I/O Standard and Placement

There are no specific I/O standards and placement requirements for this core.

Simulation

This section contains information about simulating IP in the Vivado Design Suite. For comprehensive information about Vivado simulation components, as well as information about using supported third-party tools, see the *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 8].

Synthesis and Implementation

This section contains information about synthesis and implementation in the Vivado Design Suite. For details about synthesis and implementation, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 5].

Test Bench

This chapter contains information about the test bench provided in the Vivado® Design Suite.

As shown in Figure 5-1, the demonstration test bench is a simple System Verilog module which configures and tests the VoIP Forward Error Correction Receiver core. The test bench consists of several modules like RTP packet generator for generating and driving RTP packets over transport stream, APIs and Drivers for configuring the core, Checker for integrity check of packet coming out of core. The test bench is supplied as part of the Example Simulation output product group.

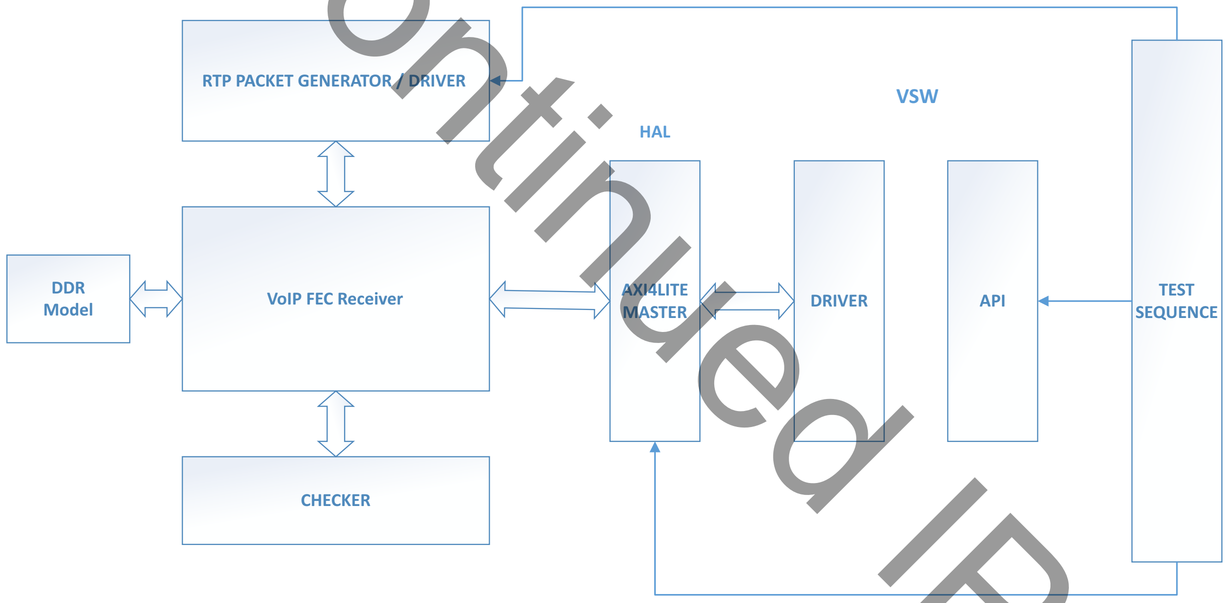


Figure 5-1: **Video over IP FEC Receiver Test Bench**

The main components of Demonstration test bench are described below:

RTP Packet Generaton/Driver: This module generates Real Time protocol packet over Transport stream and drives it to VoIP Forward Error correction Receiver core across all the enabled channels.

Checker: Packet checker module receives the packet from VoIP FEC receiver core and does packet pay load data integrity check, channel number mismatch check and packet size check

HAL: Hardware Access Layer is the register configuration layer. This layer has register read and write process.

VSW: Virtual Software layer. This layer consists of Driver and API. They control the Core configuration and are driven to Core by HAL. This layer is controlled using test case.

DDR model: This is Dummy DDR model used to store the RTP and FEC packets from core.

Verification, Compliance, and Interoperability

The Video over IP FEC Receiver core has been validated using the Xilinx® Kintex®-7 FPGA Broadcast Connectivity Kit.

Migrating and Upgrading

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

Upgrading in the Vivado Design Suite

This section provides information about any changes to the user logic or port designations that take place when you upgrade to a more current version of this IP core in the Vivado Design Suite.

Changes from v1.0 to v2.0

Video over IP FEC Receiver core version v2.0 had the following changes implemented, and may not be compatible with previous core version, v1.0.

Parameter Changes

Table B-1 shows the parameter changes.

Table B-1: **Parameter Changes**

Version		Note
V1.0	V2.0	
Component Name	Component Name	Unchanged
	Operating Mode	Newly added for Operating Mode selection: "ST2022-1/2" or "ST2022-5/6"
Maximum channels Supported	Maximum Channels Supported	Changed
Media buffer base address	Media Buffer Base Address	Changed
FEC buffer base address	Common FEC Buffer Base Address	Changed
FEC buffer pool size		Removed

Table B-1: **Parameter Changes *(Cont'd)***

Version		Note
	Include FEC Engine	New to version v2.0. Refer Customizing and Generating the Core for description
	Include Seamless Protection	New to version v2.0. Refer Customizing and Generating the Core for description

Port Changes

There are port changes between older version v1.0 compared to v2.0. Table B-2 shows the port changes.

Table B-2: **Port Changes**

Version		Note
V1.0	V2.0	
Operating Mode: ST2022-5/6 Mode and Include Seamless Protection Checked		
	Payload_in_sec_tdata	Newly added
	Payload_in_sec_tvalid	Newly added
	Payload_in_sec_tready	Newly added
	Payload_in_sec_tlast	Newly added
	Payload_in_sec_tuser	Newly added

Functionality Changes

Added Seamless Switching support, which can be enabled through XGUI option. Refer to Customizing and Generating the Core in Chapter 4 for more information.

Instructions for Minimum Change Migration

v_voip_fec_rx Migration from v1.0 to v2.0

Port Changes

In v_voip_fec_rx_v2_0, new feature was added which supports SMPTE ST 2022-7, where the receiver core receives two identical/seamless AXI4-Streams (Primary and Secondary) as configured by the user. The port changes related to the new features are shown in Table B-2.

The Secondary AXI4-Stream Slave interface signal is visible only when Seamless Protection feature is enabled in the core setting. By disabling the Seamless Protection, only primary AXI-Stream is visible and is similar to v_voip_fec_rx_v1_0 core.

Status/Event AXI4-Stream Protocol Changes

Status/Event AXI4-Stream protocol updated to include one more cycle of status/event data generation which helps to support new functionality and features as shown in Figure 3-9.

Register Setting

Change in existing registers and new register addition to support new functionality and features we added in the register map as shown in Table B-3.

These are crucial registers that need to take note during migration.

Table B-3: **Addition/Change in Register Map**

Address offset	General/Channel	Register Name		Note
		Bit	Name	
0x24	General	[31:0]	fec_pkt_drop_cnt	Added
0x0C	General	[17]	seamless switching	Added
0xA0	Channel	[31:0]	reordered_pkt_cnt	Removed
0xA4	Channel	[31:0]	oob_pkts_cnt	Removed
0xB0	Channel	[31:0]	oor_pkt_cnt	Functionality Update
0xB4	Channel	[31:0]	oor_ts_offset	Added
0xB8	Channel	[31:0]	link_ts_diff	Added

Note: Refer Table 2-11 for register descriptions.

Debugging

This appendix includes details about resources available on the Xilinx Support website and debugging tools.

TIP: *If the IP generation halts with an error, there might be a license issue. See License Checkers in Chapter 1 for more details.*

Finding Help on Xilinx.com

To help in the design and debug process when using the core, the Xilinx Support web page contains key resources such as product documentation, release notes, answer records, information about known issues, and links for obtaining further product support.

Documentation

This product guide is the main document associated with the core. This guide, along with documentation related to all products that aid in the design process, can be found on the Xilinx Support web page or by using the Xilinx Documentation Navigator.

Download the Xilinx Documentation Navigator from the Downloads page. For more information about this tool and the features available, open the online help after installation.

Solution Centers

See the Xilinx Solution Centers for support on devices, software tools, and intellectual property at all stages of the design cycle. Topics include design assistance, advisories, and troubleshooting tips.

The Solution Center specific to the Video over IP FEC Receiver core is listed below.

- Xilinx Ethernet IP Solution Center
- Xilinx MIG Solution Center
- Xilinx Solution Center for PCI Express

Answer Records

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

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

- Product name
- Tool message(s)
- Summary of the issue encountered

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

Master Answer Record for the Video over IP FEC Receiver

AR: 54548

Technical Support

Xilinx provides technical support in the Xilinx Support web page for this LogiCORE™ IP product when used as described in the product documentation. Xilinx cannot guarantee timing, functionality, or support if you do any of the following:

- Implement the solution in devices that are not defined in the documentation.
- Customize the solution beyond that allowed in the product documentation.
- Change any section of the design labeled DO NOT MODIFY.

To contact Xilinx Technical Support, navigate to the Xilinx Support web page.

Debug Tools

There are many tools available to address Video over IP FEC Receiver design issues. It is important to know which tools are useful for debugging various situations.

Vivado Design Suite Debug Feature

The Vivado® Design Suite debug feature inserts logic analyzer and virtual I/O cores directly into your design. The debug feature also allows you to set trigger conditions to capture application and integrated block port signals in hardware. Captured signals can then be analyzed. This feature in the Vivado IDE is used for logic debugging and validation of a design running in Xilinx.

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

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

See the *Vivado Design Suite User Guide: Programming and Debugging* (UG908) [Ref 10].

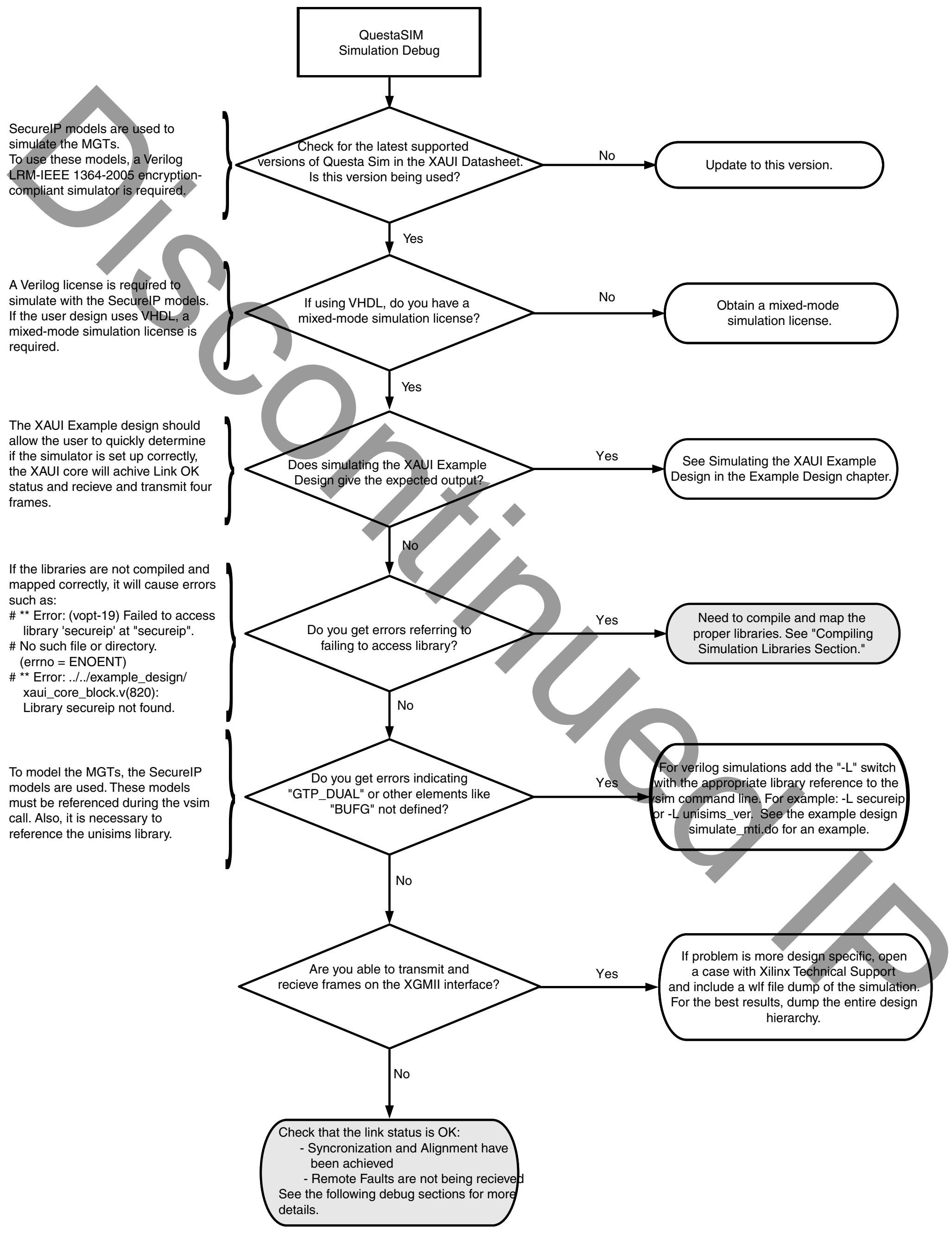
Reference Boards

Various Xilinx development boards support the Video over IP FEC Receiver. These boards can be used to prototype designs and establish that the core can communicate with the system.

- 7 series FPGA evaluation board KC705

Simulation Debug

The simulation debug flow for Questa® SIM is illustrated in Figure C-1. A similar approach can be used with other simulators.



QuestaSIM Simulation Debug
SecureIP models are used to simulate the MGTs. To use these models, a Verilog LRM-IEEE 1364-2005 encryption-compliant simulator is required.
Check for the latest supported versions of Questa Sim in the XAUI Datasheet. Is this version being used?
No
Update to this version.
Yes
A Verilog license is required to simulate with the SecureIP models. If the user design uses VHDL, a mixed-mode simulation license is required.
If using VHDL, do you have a mixed-mode simulation license?
No
Obtain a mixed-mode simulation license.
Yes
The XAUI Example design should allow the user to quickly determine if the simulator is set up correctly, the XAUI core will achive Link OK status and recieve and transmit four frames.
Does simulating the XAUI Example Design give the expected output?
Yes
See Simulating the XAUI Example Design in the Example Design chapter.
No
If the libraries are not compiled and mapped correctly, it will cause errors such as:
# ** Error: (vopt-19) Failed to access library 'secureip' at "secureip".
# No such file or directory. (errno = ENOENT)
# ** Error: ../../example_design/xaui_core_block.v(820): Library secureip not found.
Do you get errors referring to failing to access library?
Yes
Need to compile and map the proper libraries. See "Compiling Simulation Libraries Section."
No
To model the MGTs, the SecureIP models are used. These models must be referenced during the vsim call. Also, it is necessary to reference the unisims library.
Do you get errors indicating "GTP_DUAL" or other elements like "BUFG" not defined?
Yes
For verilog simulations add the "-L" switch with the appropriate library reference to the vsim command line. For example: -L secureip or -L unisims_ver. See the example design simulate_mti.do for an example.
No
Are you able to transmit and recieve frames on the XGMII interface?
Yes
If problem is more design specific, open a case with Xilinx Technical Support and include a wlf file dump of the simulation. For the best results, dump the entire design hierarchy.
No
Check that the link status is OK:
- Syncronization and Alignment have been achieved
- Remote Faults are not being recieved
See the following debug sections for more details.

Hardware Debug

Hardware issues can range from link bring-up to problems seen after hours of testing. This section provides debug steps for common issues. The Vivado lab tools are a valuable resource to use in hardware debug. The signal names mentioned in the following individual sections can be probed using the Vivado lab tools for debugging the specific problems.

General Checks

Ensure that all the timing constraints for the core were properly incorporated from the example design and that all constraints were met during implementation.

- Does it work in post-place and route timing simulation? If problems are seen in hardware but not in timing simulation, this could indicate a PCB issue. Ensure that all clock sources are active and clean.
- If using MMCMs in the design, ensure that all MMCMs have obtained lock by monitoring the `locked` port.
- If your outputs go to 0, check your licensing.

Interface Debug

AXI4-Lite Interfaces

Read from a register that does not have all 0s as a default to verify that the interface is functional. Output `s_axi_arready` asserts when the read address is valid, and output `s_axi_rvalid` asserts when the read data/response is valid. If the interface is unresponsive, ensure that the following conditions are met:

- The `s_axi_aclk` and `aclk` inputs are connected and toggling.
- The interface is not being held in reset, and `s_axi_areset` is an active-Low reset.
- The interface is enabled, and `s_axi_aclken` is active-High (if used).
- The main core clocks are toggling and that the enables are also asserted.
- If the simulation has been run, verify in simulation and/or a Vivado Lab tools capture that the waveform is correct for accessing the AXI4-Lite interface.

AXI4-Stream Interfaces

If data is not being transmitted or received, check the following conditions:

- If transmit `<interface_name>_tready` is stuck Low following the `<interface_name>_tvalid` input being asserted, the core cannot send data.
- If the receive `<interface_name>_tvalid` is stuck Low, the core is not receiving data.
- Check that the `aclk` inputs are connected and toggling.
- Check that the AXI4-Stream waveforms are being followed.
- Check core configuration.

Other Interfaces

IP Core Debug

Crucial Core and Register Setting

When generating core in Vivado, ensure that the base address for FEC Buffer Base Address & Media Buffer Base Address is set based on the recommended computations described in Memory Requirements in Chapter 3.

Debug Operation

1. If a non-zero value is observed while reading valid_pkts_cnt (0x090), it indicates the core is receiving valid packets.
2. If a non-zero value is observed while reading oor_pkts_cnt(0xB0), it indicates that the incoming packet is not within the protection range and the packet is dropped due to out of sync and a channel recovery need to be performed. Refer to oor_pkts_cnt (0x0B0) Register in Chapter 2 for more information.
3. Stream changes on a particular channel (or a time disruption or stoppage in the source) requires a channel recovery to be performed.
4. Software configuration change on a particular channel (reconfiguring chan_conf (0x080) on the fly) would also require a channel recovery to be performed.
5. If FEC Packet Drop count increases, it indicates that the FEC Buffer has overflowed due to AXI4-MM push back.

Additional Resources and Legal Notices and Legal Notices

Xilinx Resources

For support resources such as Answers, Documentation, Downloads, and Forums, see Xilinx Support.

References

These documents provide supplemental material useful with this product guide:

1. SMPTE ST 2022-5,6 Video Over IP Product Page
2. SMPTE ST 2022-1,2 Video Over IP Product Page
3. Modular Media over IP Product Page
4. *Vivado® Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
5. *Vivado Design Suite User Guide: Designing with IP* (UG896)
6. *Vivado AXI Reference Guide* (UG1037)
7. *Vivado Design Suite User Guide: Getting Started* (UG910)
8. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
9. *ISE® to Vivado Design Suite Migration Guide* (UG911)
10. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)
11. *Vivado Design Suite User Guide - Implementation* (UG904)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
10/05/2016	2.0	Added support for RFC3190 and RFC4175. Updated Xilinx automotive applications disclaimer.
04/06/2016	2.0	Added support for ST2022-5,6. Added support for Seamless Protection/ST2022-7 on ST2022-5,6 Mode.
11/18/2015	1.0	Added support for UltraScale+ families. Added a new register, oor_pkts_cnt (0x0B0).
04/01/2015	1.0	Initial Xilinx release.

Please Read: Important Legal Notices