Modular Media over IP Infrastructure

LogiCORE IP Product Guide

Vivado Design Suite

PG241 June 7, 2017

XILINX
ALL PROGRAMMABLE™

Table of Contents

Chapter 1: Overview

Chapter 2: Product Specification

Chapter 3: Designing with the Core

Chapter 4: Design Flow Steps

Chapter 5: Example Design

Chapter 6: Test Bench

Appendix A: Verification, Compliance, and Interoperability

Appendix B: Migrating and Upgrading

Appendix C: Debugging

Appendix D: Additional Resources and Legal Notices

IP Facts

Introduction

The Modular Media over IP Infrastructure is a collection of cores that can be integrated together to support a variety of protocols for video, audio, and data over IP-based networks. Primarily developed to support the SMPTE ST 2022 [Ref 4] [Ref 5] [Ref 6] standards for video over IP initially, the modules and their associated reference designs can be used as a starting point framework for many types of media over IP implementations, both standardized and proprietary. Offering superior flexibility and scalability, modules can be added to or replaced by existing and future cores from Xilinx, our Alliance Program Members or customer's own IP. Access to intermediate signaling is inherently available.

Features

- Video stream conversion modules
  - ST 2022-6 Packetizer Module:
    Converts Serial Digital Interface (SDI) video stream (SDI over AXI4-Stream) into a media datagram stream in accordance with the SMPTE ST 2022-6 protocol.
  - ST 2022-6 Depacketizer Module:
    Converts received Real-time Transport Protocol (RTP) encapsulated SMPTE ST 2022-6 Media packets into SDI video stream (SDI over AXI4-Stream)
- Ethernet packet processing modules
  - Framer Module:
    Framing Ethernet Packets by adding the Ethernet, Internet Protocol (IP) and User Datagram Protocol (UDP) headers to incoming Real-time Transport Protocol (RTP) packets.
  - Decapsulator Module:
    Accepts Ethernet Packets and perform header stripping (output RTP packets), channel matching and filtering, video stream detection and video frame boundary alignment.

LogiCORE™ IP Facts Table	
Core Specifics	
Supported Device Family[1]	UltraScale+™ Families, Kintex® UltraScale™, Zynq®-7000, Virtex®-7, Kintex®-7
Supported User Interfaces	AXI4-Lite, AXI4-Stream, AXI4
Resources	See Resource Utilization in Chapter 2
Provided with Core	
Design Files	Encrypted RTL
Example Design	*Modular SMPTE ST 2022-567 on Kintex-7 Evaluation Board Application Note* (XAPP1272) [Ref 3]
Test Bench	Verilog and VHDL
Constraints File	XDC
Simulation Model	Encrypted RTL, VHDL Behavioral, VHDL or Verilog source HDL
Supported S/W Driver[2]	Standalone
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

In broadcast and professional audio/video markets, vendors and end users are pushing for the widespread adoption of IP-based networking to transport video, audio and data either between locations, or between equipment. This can reduce costs, enable seamless workflows, and offer expanded capabilities to handle all kinds of traffic in novel and more efficient ways. Up until now, the main way this has been done in broadcast is using standards such as SMPTE ST 2022, which can encapsulate multiple uncompressed SDI video channels, or multiple compressed transport streams into an Ethernet stream.

The modular media over IP infrastructure focuses initially on SMPTE ST 2022-6, with ST 2022-6 Packetizers and ST 2022-6 Depacketizers supporting the SMPTE ST 2022-6 standard and converting between SDI video and (Real Time Protocol) RTP packet streams (see IETF RFC 3550 [Ref 2]). Framer and Decapsulator modules are used to do conversion between RTP and Ethernet packets. The infrastructure can be configured to enable SMPTE ST 2022-7 which adds further robustness by adding a second redundant network link and being able to switch seamlessly between them should one link lose packets.

The SMPTE ST 2022 modules integrate easily with the Xilinx SMPTE SD/HD/3G-SDI or SMPTE UHD-SDI (up to 3G rate) LogiCORE IP cores, the 10 Gigabit Ethernet subsystem and other cores to provide a fully inter-operable system-level design for high-bit rate Video over IP.

Over time this modular approach provides a framework to support protocols and standards beyond SMPTE ST 2022 and to accelerate overall time-to-market.

Feature Summary

The Modular Media over IP Infrastructure supports the ST 2022 Media over IP system design. The core supports all SD/HD3G-SDI modes, except the following:

- HD: 1035i, 59.94/60
- 3G-levelA: 1080P, 47.95/48.
- 3G-levelB: 1080P, 47.95/48/50/59.94/60

The ST 2022-6 Packetizer and ST 2022-6 Depacketizer are based on SMPTE ST 2022-6 standards. Framer and Decapsulator can be used for SMPTE ST 2022-5/6 standards.

- AXI4-Stream compliant
  - Supports these AXI4-Stream defined signals: `TVALID`, `TREADY`, `TDATA`, `TKEEP`, `TLAST`, `TUSER`.
  - Supports SDI over AXI4-Stream (customized AXI4-Stream for video frame stream transmitting/receiving) (see Chapter 3, Designing with the Core).
  - Supports RTP over AXI4-Stream (customized AXI4-Stream for RTP packet stream transmitting/receiving) (see Chapter 3, Designing with the Core).
- ST 2022-6 Packetizer
  Converts an SDI video stream (SDI over AXI4-Stream) into a media datagram stream in accordance with the SMPTE ST 2022-6 protocol.
- ST 2022-6 Depacketizer
  Converts received RTP Encapsulated SMPTE ST 2022-6 Media Packets into SDI video stream (SDI over AXI4-Stream).
- Framer
  Framing Ethernet Packets by adding the Ethernet, Internet Protocol and User Datagram Protocol (UDP) headers to incoming RTP Encapsulated SMPTE ST 2022, RFC4175, or RFC3190 packets.
- Decapsulator
  Accepts Ethernet packets that encapsulate SMPTE ST 2022 5/6, RFC4175, or RFC3190 media data and performs header stripping (transmit RTP Encapsulated SMPTE ST 2022-5/6, RFC4175, or RFC3190 packets), channel matching and filtering, video stream detection and video frame boundary alignment.

Applications

- Transport high bandwidth RTP Encapsulated ST 2022, RFC4175, or RFC3190 packets over IP Network
- Support real-time ST 2022, RFC4175, or RFC3190 applications such as contribution, primary distribution and digital cinema.
- SMPTE ST 2022-56 Example Application: See *Modular SMPTE ST 2022-567 on Kintex-7 Evaluation Board Application Note* (XAPP1272) [Ref 3].

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 Modular Media over IP Infrastructure product 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 an 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.*

Hardware Evaluation License

If a Hardware Evaluation License is being used, the core will stop transmitting video after timeout.

Product Specification

Media over IP Infrastructure Modules

ST 2022-6 Packetizer

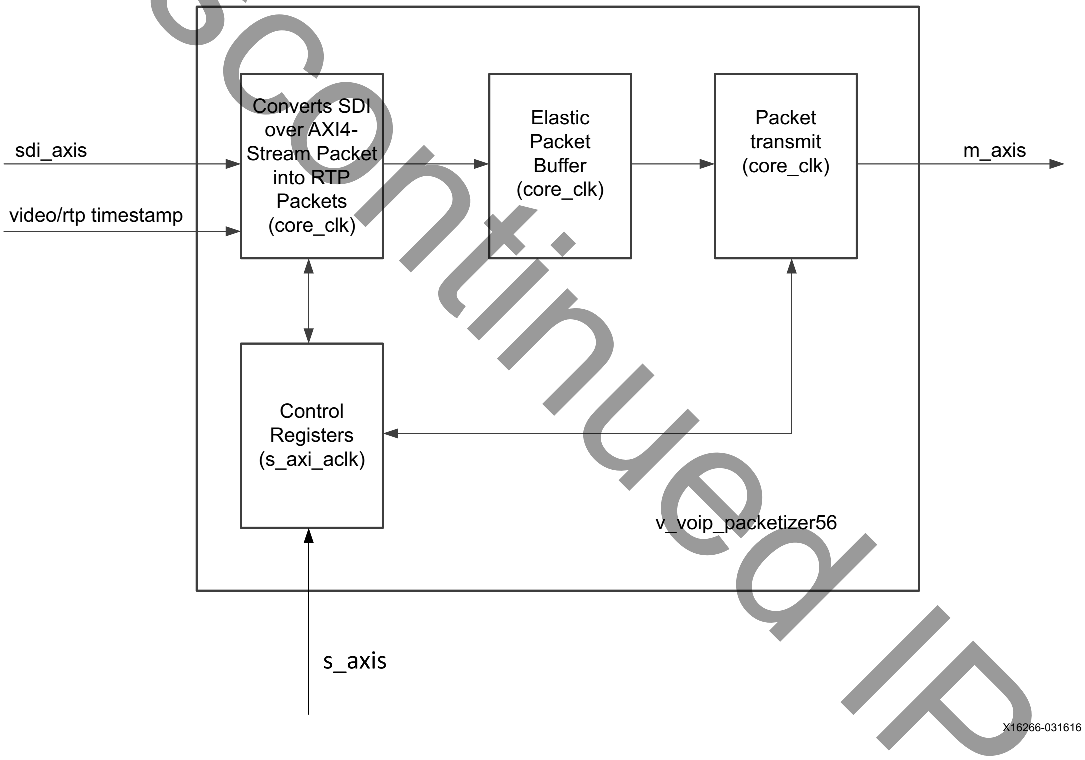

sdi_axis
video/rtp timestamp
Converts SDI over AXI4-Stream Packet into RTP Packets (core_clk)
Elastic Packet Buffer (core_clk)
Packet transmit (core_clk)
m_axis
Control Registers (s_axi_aclk)
v_voip_packetizer56
s_axis
X16266-031616

Figure 2-1: **ST 2022-6 Packetizer Block Diagram**

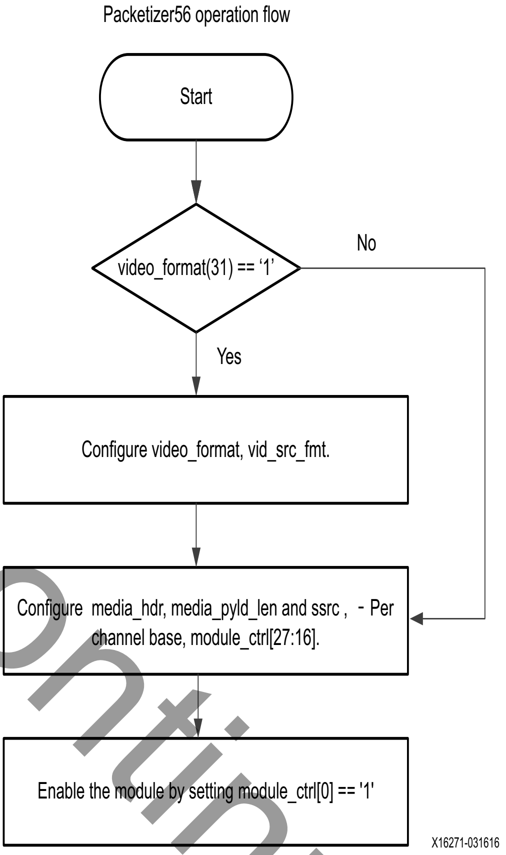

Packetizer56 operation flow
Start
video_format(31) == '1'
No
Yes
Configure video_format, vid_src_fmt.
Configure media_hdr, media_pyld_len and ssrc , - Per channel base, module_ctrl[27:16].
Enable the module by setting module_ctrl[0] == '1'
X16271-031616

Figure 2-2: **ST 2022-6 Packetizer Operation Flow**

The ST 2022-6 Packetizer module accepts SDI over AXI4-Streams and transmits out SMPTE ST 2022-6 RTP packets to a downstream module. The packet information is defined in the `TUSER` bus of the m_axis interface (see "RTP over AXI4-Stream Protocol" in Chapter 4, Design Flow Steps.)

The module has an AXI4-Lite interface that allows dynamic control of the register settings. See Table 2-3.

The ST 2022-6 media payload length by default is set to 1,376 bytes. You have the flexibility to configure the media payload length by configuring bit [10:0] of the media payload length register (offset 0x20).

The video format information is required by the module for correct operation. The video format information can be either from an upstream module by using the SDI over AXI4-Stream interface TUSER sideband signal or by direct register programming on the module. The decision is made through the video_format (offset 0x14) register bit 31.

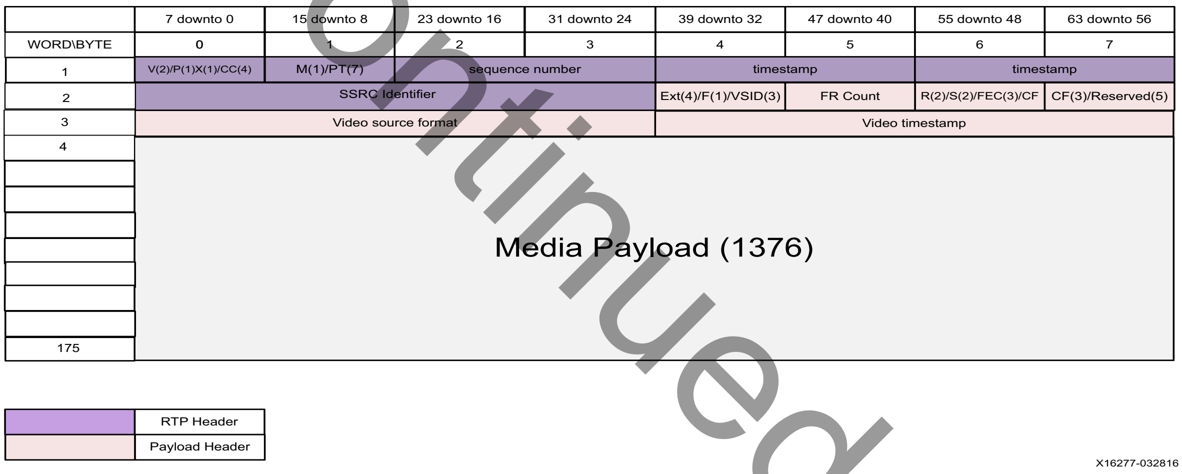
The module must be enabled before accepting SDI over AXI4-Stream data by setting bit 0 to 1 of the module control register (offset 0x10). The channel number must be configured before enabling the module and the configured channel number is reflected in the RTP over AXI4-Stream master interface `TUSER` bus.

The RTP time stamp and Video time stamp is provided via the RTP Time Stamp input port and the Video Time Stamp input port respectively.

The media payload length register (offset 0x20) and register bit[27:16] of the module control register (offset 0x10) should be configured first before enabling the module.

Whenever the video format changes, a reset or clear action must be done on the module, which is done by programing the module enable to 0.

Figure 2-3 shows the packet format that comes out from the RTP over AXI4-Stream master interface of the module. This packet format strictly follows the SMPTE ST 2022-6 standard. Note that the Ext[3:0] field in the media payload header are fixed to zero which means the header extension is not supported.



	7 downto 0	15 downto 8	23 downto 16	31 downto 24	39 downto 32	47 downto 40	55 downto 48	63 downto 56
WORD\BYTE	0	1	2	3	4	5	6	7
1	V(2)/P(1)X(1)/CC(4)	M(1)/PT(7)	sequence number		timestamp		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			
4	Media Payload (1376)							
175								

▨	RTP Header
░	Payload Header

X16277-032816

Figure 2-3: **ST 2022-6 RTP Packet Format**

ST 2022-6 Depacketizer

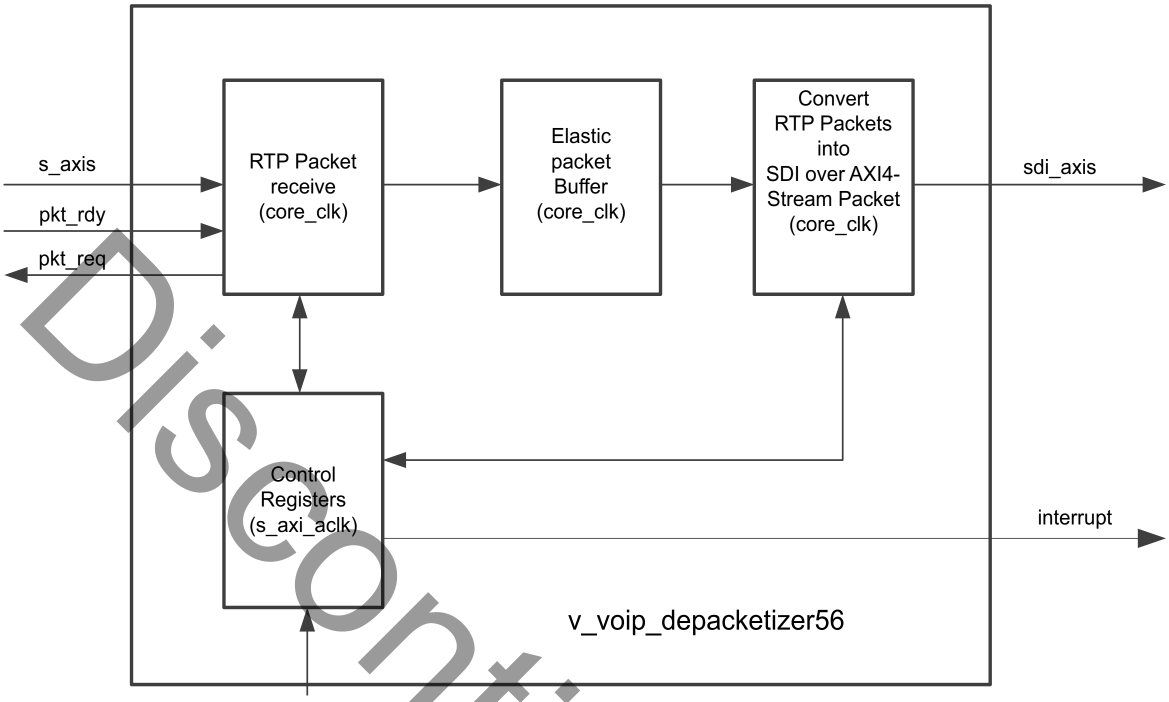

s_axis
pkt_rdy
pkt_req
RTP Packet receive (core_clk)
Elastic packet Buffer (core_clk)
Convert RTP Packets into SDI over AXI4-Stream Packet (core_clk)
sdi_axis
Control Registers (s_axi_aclk)
interrupt
v_voip_depacketizer56

Figure 2-4: **ST 2022-6 Depacketizer Block Diagram**

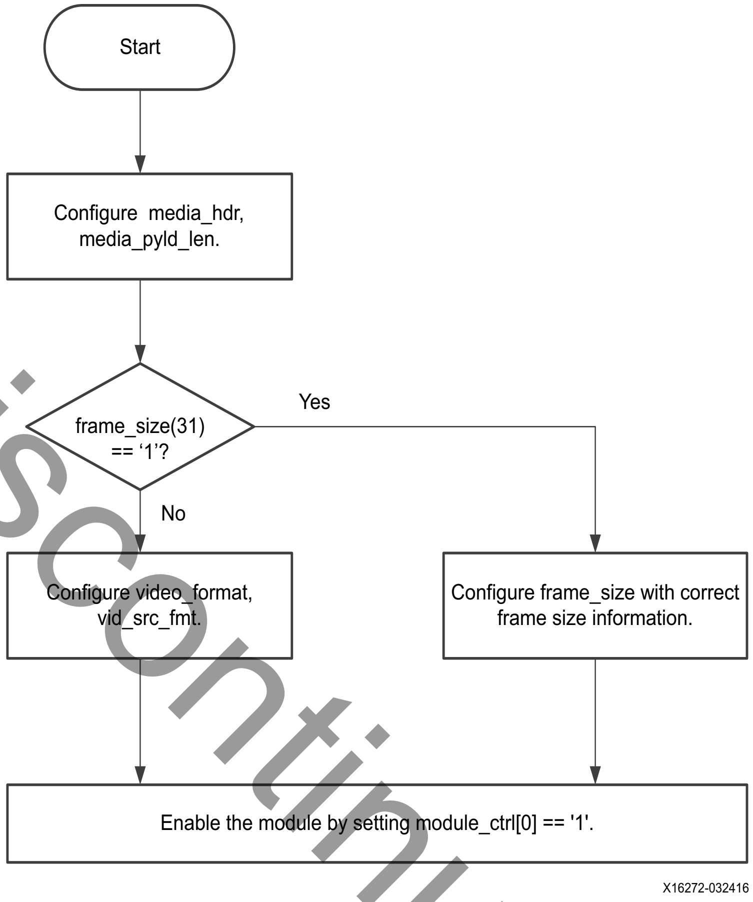

Start
Configure media_hdr, media_pyld_len.
frame_size(31) == ‘1’?
Yes
No
Configure video_format, vid_src_fmt.
Configure frame_size with correct frame size information.
Enable the module by setting module_ctrl[0] == '1'.
X16272-032416

Figure 2-5: **ST 2022-6 Depacketizer Operation Flow**

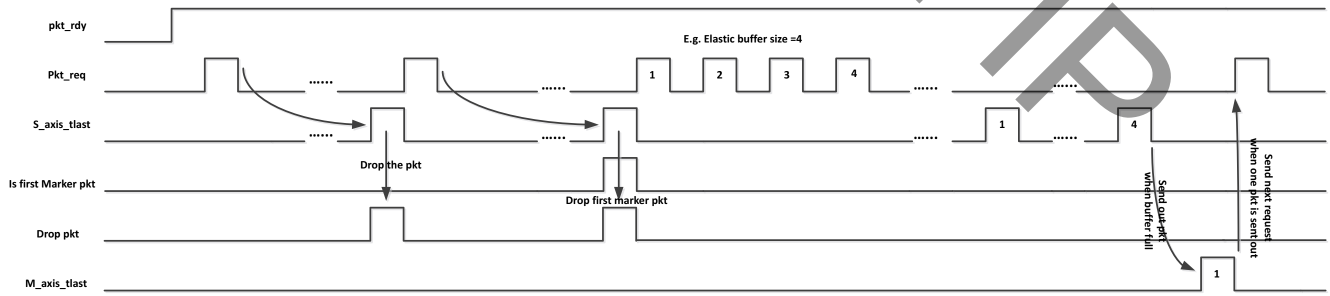


Figure 2-6: **ST 2022-6 Depacketizer Packet Request Protocol**

The ST 2022-6 Depacketizer module converts RTP packets back to SDI over the AXI4-Stream video stream. The incoming stream packet type must be an SMPTE ST 2022-6 RTP packet. The RTP header and media payload header are stripped off. The input interface to the ST 2022-6 Depacketizer module must comply with "RTP over AXI4-Stream" (customized AXI4-Stream for RTP packet stream transmitting/receiving.) See Chapter 3, Designing with the Core.

The frame size information is required by the module for correct operation. The frame size consists of two important pieces of information: datagram per frame and last datagram length. You are required to program the video_format (offset 0x14) and vid_src_fmt (offset 0x18). The module can decode the frame size from these two video format registers.

If you need to use a non-standard video frame size, you can program the frame_size (offset 0x24) register bit 31 to 1 and give the correct frame size information on the remaining bits of the register.

The media payload length is configured by the media_pyld_len (offset 0x20) register, which has a default value of 1,376 (decimal). It is the size in bytes for the video data carried in the incoming packet.

Because the incoming packet might not have a video timestamp inserted in the media payload header, you need to explicitly tell the module about video timestamp information by programming the media_hdr (offset 0x1C) register bit 0. This is important because it determines whether the module can succeed extracting the correct media payload data from the packet.

The `pkt_req` output port of the ST 2022-6 Depacketizer module is connected to the upstream module for the packet request process. If the upstream module does not have this handshaking protocol but only the AXI4-Stream master interface, you can tie the `pkt_rdy` to High and ignore the `pkt_req` output. This is illustrated in Figure 2-6. When the `pkt_rdy` input is High, the module sends out one request pulse and waits for the packet to arrive at the RTP over AXI4-Stream slave interface. When the packet arrives, it checks whether it is the last packet of a video frame or not by checking on the marker bit ("M" bit) in the packet RTP header. If it is not the marker packet, the packet is dropped at the input until a marker packet is detected. After the first marker packet is detected (will be dropped), the module will send out a number of packet request pulses in every other cycle (the number of pulses sent in this burst equal to the elastic buffer size). This is to fill the internal buffer until full. Packets will be processed and data will be sent out on the SDI over AXI4-Stream master interface only when the buffer becomes full. This feature can help to avoid video data starvation at the module output that can be caused by upstream modules.

Disabling the module causes the internal buffer and state machine to be cleared and reset, which is needed when you want to reset the module or during stream changes.

Currently, there is only one interrupt source from the module which is "buffer become empty interrupt." In normal operation, when the data stream is flowing through the module, the buffer level of the module should maintain approximately the same buffer size. Due to upstream module error/instability, the buffer level might go down.

If the buffer becomes empty because of this, it will cause video stream interruption. In this case this interrupt is fired to notify the system of the error condition.

Framer

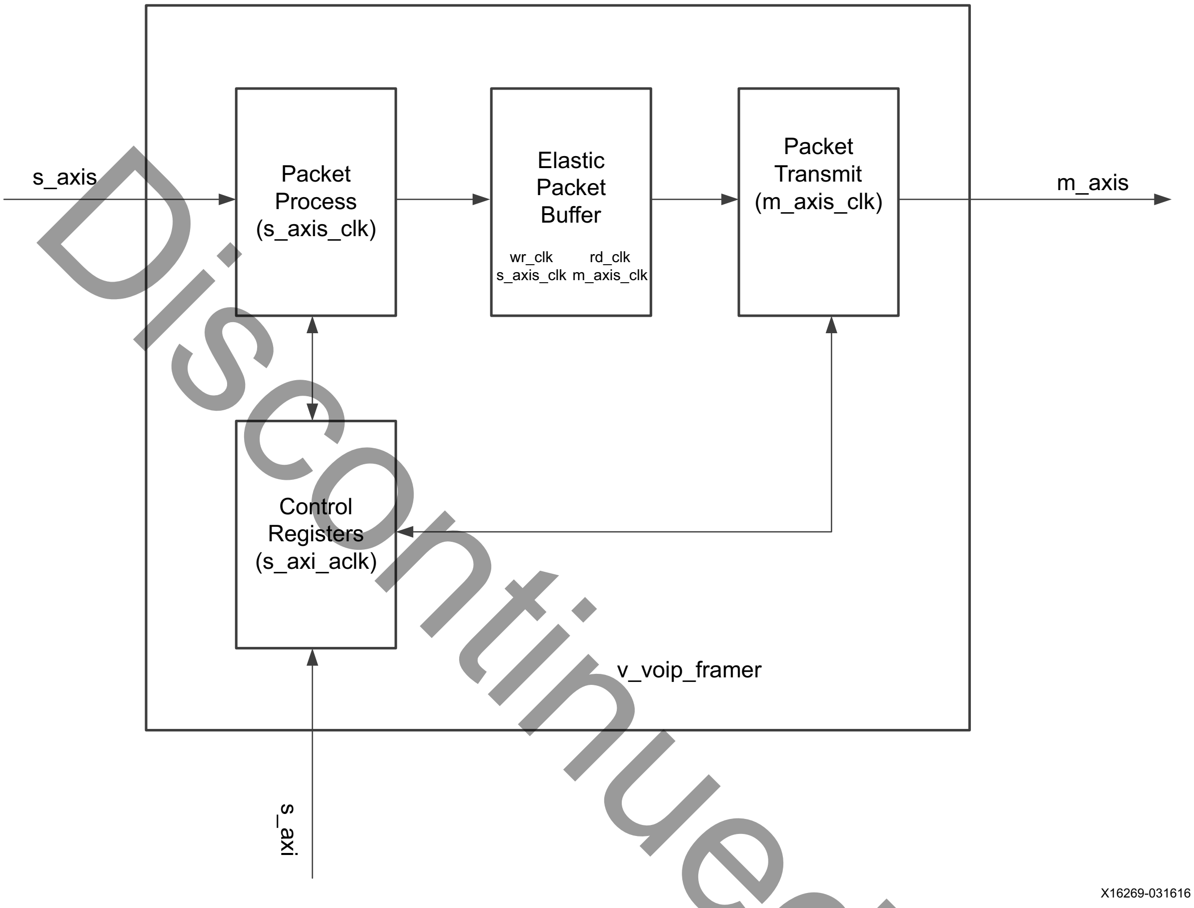


Figure 2-7: **Framer Block Diagram**

The Figure 2-8 left flow chart shows the operation of the configuration channel space registers; this operation is common for multi-channel register modules, for example, the Decapsulator and Framer modules.

The main steps to access a channel space register are:

1. Read the general space register “status” bit0 to check whether the hardware is busy or not.
2. If not busy, program the channel number into the general space register "channel_access".
3. Access the channel space registers.
4. If the channel registers get new programmed values, program the general space register "control" bit1 to 1.

5. Program the general space register "control" bit1 to 0. Note that the preceding steps are used to access one or more channel space registers for one channel.

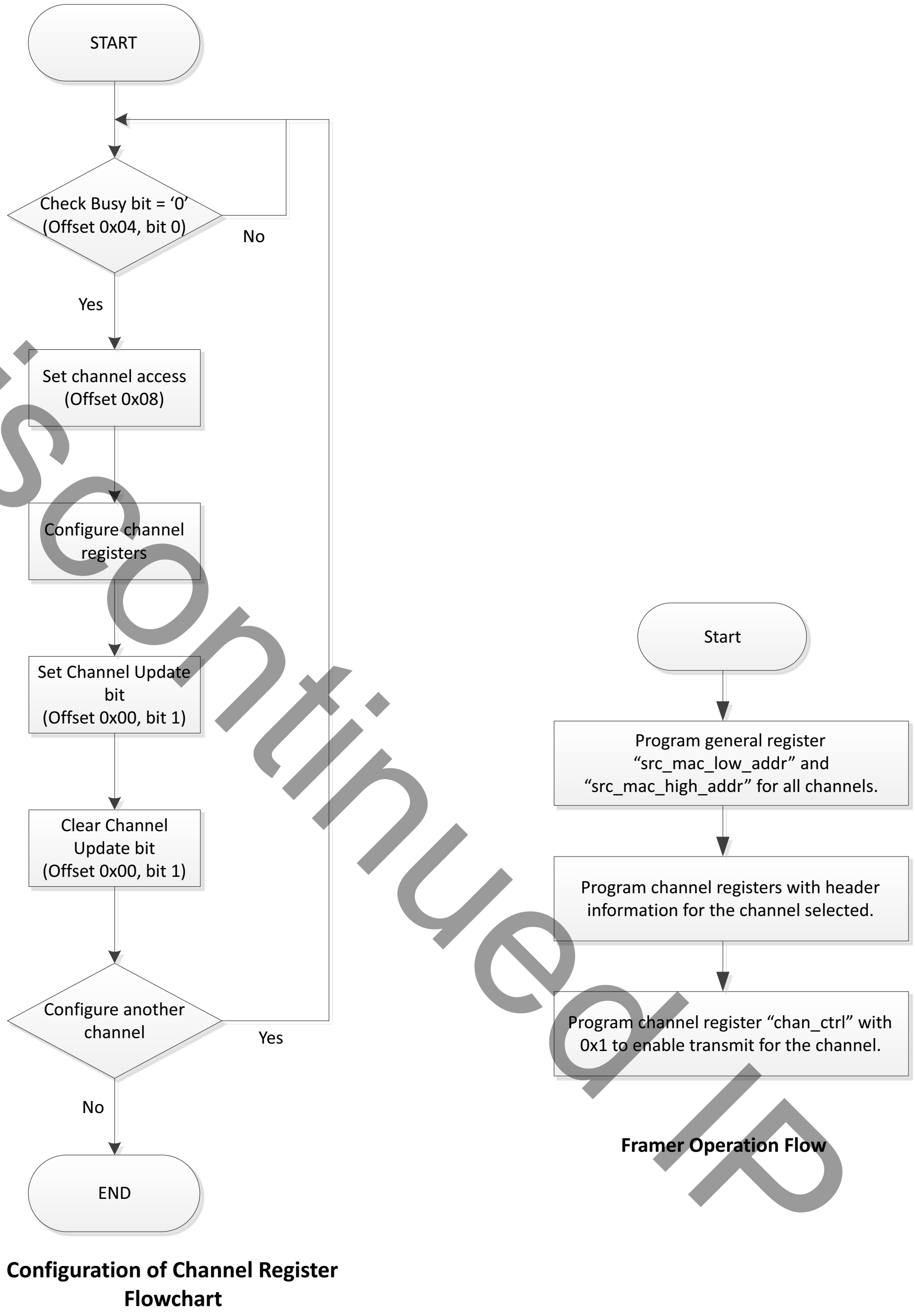

START
Check Busy bit = ‘0’
(Offset 0x04, bit 0)
No
Yes
Set channel access
(Offset 0x08)
Configure channel registers
Set Channel Update bit
(Offset 0x00, bit 1)
Clear Channel Update bit
(Offset 0x00, bit 1)
Configure another channel
Yes
No
END
Configuration of Channel Register Flowchart
Start
Program general register “src_mac_low_addr” and “src_mac_high_addr” for all channels.
Program channel registers with header information for the channel selected.
Program channel register “chan_ctrl” with 0x1 to enable transmit for the channel.
Framer Operation Flow

Figure 2-8: **Configuration of Channel Register Flowchart and Framer Operation Flow**

The module accepts RTP Encapsulated SMPTE ST 2022-5/6, RFC4175, or RFC3190 packets from the upstream module and adds user-programmed UDP/IP/Ethernet headers. The generated Ethernet packets can be sent to the Ethernet Media Access Control (MAC).

If the generic "Overflow Handling Strategy" is set to 1, when the internal elastic buffer becomes full, packets will be dropped at the RTP over AXI4-Stream slave interface and will not be pushed back to the upstream module. If this generic is set to 0, if the buffer becomes full, packets will be pushed back to the upstream module by deassertion of TREADY of the RTP over AXI4-Stream slave interface.

The module is configured through the s_axi (AXI4-Lite) interface which is usually connected to the processor subsystem.

The s_axis input interface comply with "RTP over AXI4-Stream". The m_axis output interface is the simple AXI4-Stream interface without tuser.

As shown in Figure 2-8, the module is a multi-channel module that can support multiple channels.

The generic "Max Channels" should be properly set because it determines how many channels that the module can support. If it is set too large, FPGA resource utilization can be high.

The input packet types are RTP Encapsulated SMPTE ST 2022-5/6, RFC4175, or RFC3190 packets driven on the RTP over AXI4-Stream slave interface of the module. The output packet types are Ethernet packet types driven on the AXI4-Stream master interface of the module. The Ethernet/IP/UDP headers are user configurable. Some header field values are shared among all channels (general register space) and most of the header field values are per channel programmable.

Figure 2-9 and Figure 2-10 show a typical Ethernet packet format without a VLAN field (for the RTP Encapsulated SMPTE ST 2022-6 Media packet and RTP Encapsulated SMPTE ST 2022-5 Media FEC packet). Framer will not change the content of the RTP header and Payload Data. Framer only adds the Ethernet header, IP header, and UDP header to the RTP packet according to user-configured parameters.

As for RTP Encapsulated SMPTE ST 2022-5 FEC packets, the UDP Destination Port will be added by 2 for column SMPTE ST 2022-5 FEC packets and by 4 for row SMPTE ST 2022-5 FEC packets from the user-configured UDP Destination Port.

The IP header checksum is calculated by the module but the UDP header checksum will be zeros.

Figure 2-9 and Figure 2-10 show packet format for the RTP and FEC packet.

As for the payload header, Figure 2-9 shows the case that the field **CF[3:0] /= 0** (including timestamp). The design also supports the packet format without timestamp (CF[3:0]=0). The Payload header field **Ext[3:0]** should always be zero; an extension is not supported. For the payload header, refer to the SMPTE ST 2022-6 standard.

As for the Internet IP header, the **IHL[3:0]** field is hard coded to zeros which means the Internet IP header length is fixed as shown in the figure. The **Identification[15:0]** field is set to zeros. The **Flags[3:0]** field is set to 010 to mean "no fragmentation" and "no more fragments". The **Fragment Offset** field is set to zero as well.

Note that for the Internet IP header field **TTL[7:0]**, it must be programmed with a big enough value that guarantees the packet can survive before reaching its destination. The default value from the design is zero.

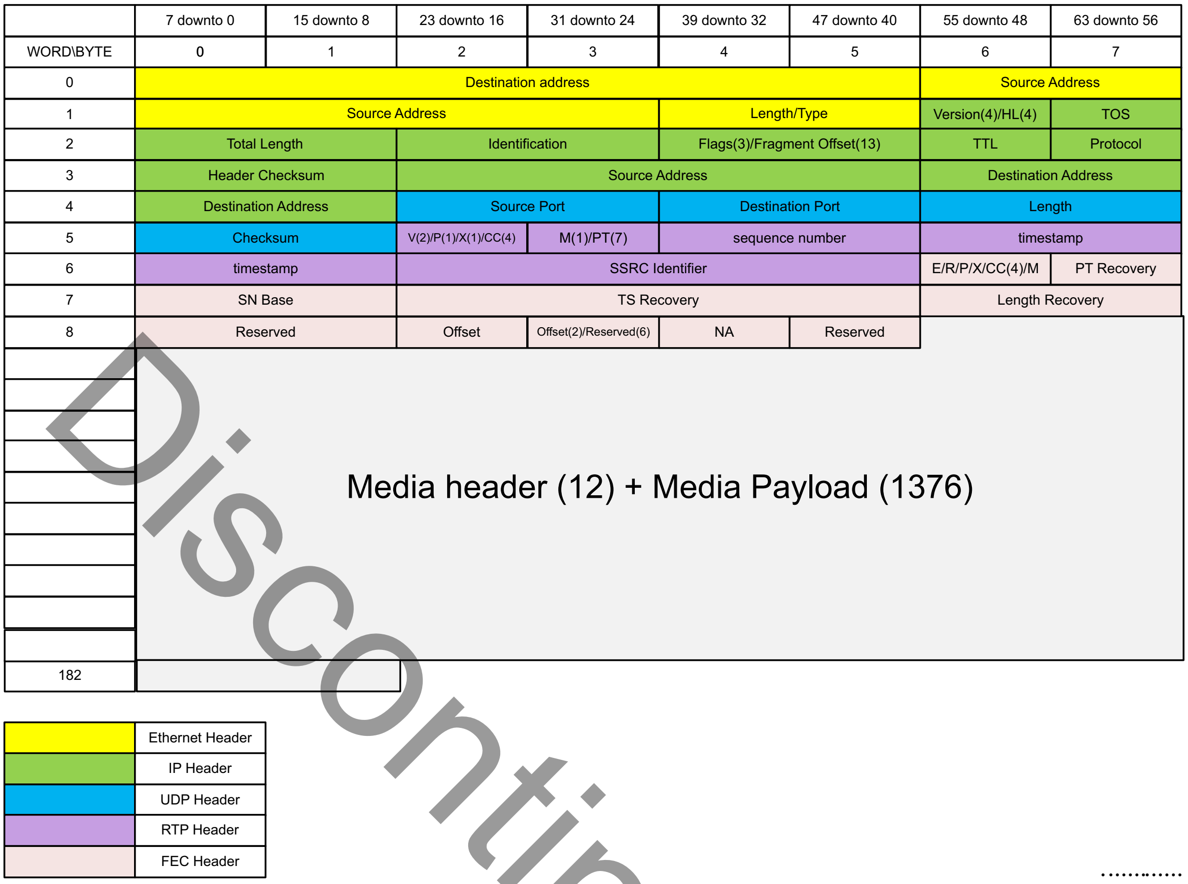
The Internet IP header field **Protocol[7:0]** is hard coded to be 0x11 to force the packet to be a User Datagram according to the Internet Protocol Numbers (refer to RFC762 [Ref 15]).

	7 downto 0	15 downto 8	23 downto 16	31 downto 24	39 downto 32	47 downto 40	55 downto 48	63 downto 56
WORD\BYTE	0	1	2	3	4	5	6	7
0	Destination address						Source Address	
1	Source Address				Length/Type		Version(4)/HL(4)	TOS
2	Total Length		Identification		Flags(3)/Fragment Offset(13)		TTL	Protocol
3	Header Checksum		Source Address				Destination Address	
4	Destination Address		Source Port		Destination Port		Length	
5	Checksum		V(2)/P(1)/X(1)/CC(4)	M(1)/PT(7)	sequence number		timestamp	
6	timestamp		SSRC Identifier				Ext(4)/F(1)VSID(3)	FR Count
7	R(2)/S(2)/FEC(3)/CF	CF(3)/Reserved(5)	Video source format				Video timestamp	
8	Video timestamp		Media Payload (1376)					
180								

🟨	Ethernet Header
🟩	IP Header
🟦	UDP Header
🟪	RTP Header
▨	Payload Header

X16276-032816

Figure 2-9: **Ethernet Packet Format for ST 2022-6 RTP Packet Type (without VLAN)**



	7 downto 0	15 downto 8	23 downto 16	31 downto 24	39 downto 32	47 downto 40	55 downto 48	63 downto 56
WORD\BYTE	0	1	2	3	4	5	6	7
0	Destination address						Source Address	
1	Source Address				Length/Type		Version(4)/HL(4)	TOS
2	Total Length		Identification		Flags(3)/Fragment Offset(13)		TTL	Protocol
3	Header Checksum		Source Address				Destination Address	
4	Destination Address		Source Port		Destination Port		Length	
5	Checksum		V(2)/P(1)/X(1)/CC(4)	M(1)/PT(7)	sequence number		timestamp	
6	timestamp		SSRC Identifier				E/R/P/X/CC(4)/M	PT Recovery
7	SN Base		TS Recovery				Length Recovery	
8	Reserved		Offset	Offset(2)/Reserved(6)	NA	Reserved		
	Media header (12) + Media Payload (1376)							
182								

🟨	Ethernet Header
🟩	IP Header
🟦	UDP Header
🟪	RTP Header
⬜	FEC Header

....

Figure 2-10: **Ethernet Packet Format for ST 2022-5 FEC Packet Type (without VLAN)**

Decapsulator

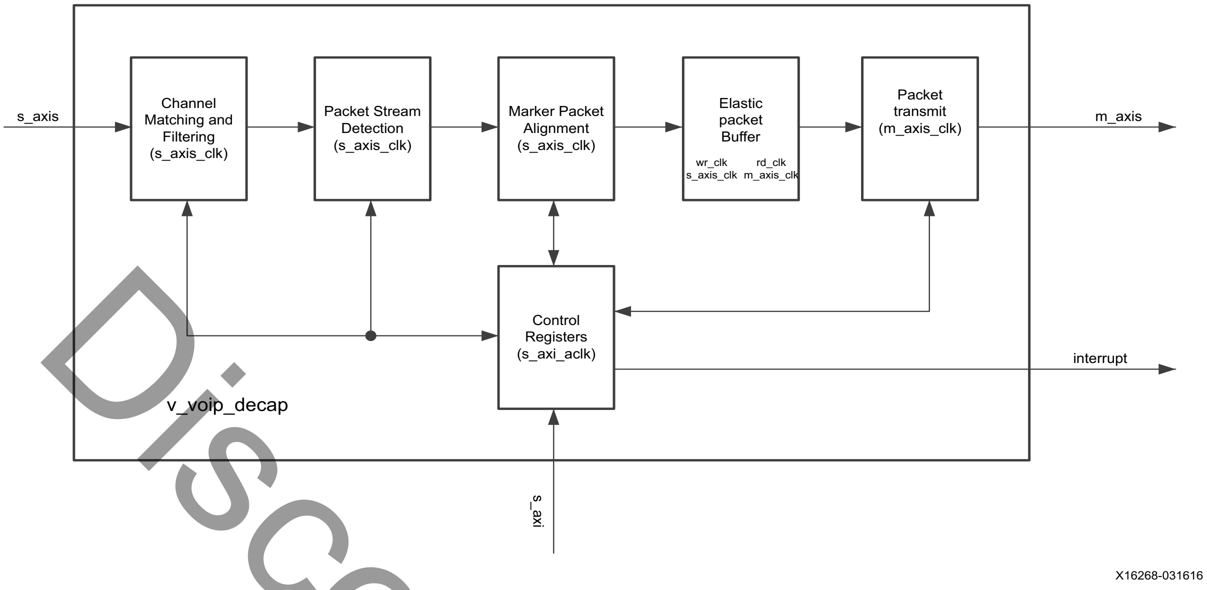

s_axis
Channel Matching and Filtering (s_axis_clk)
Packet Stream Detection (s_axis_clk)
Marker Packet Alignment (s_axis_clk)
Elastic packet Buffer
wr_clk
s_axis_clk
rd_clk
m_axis_clk
Packet transmit (m_axis_clk)
m_axis
Control Registers (s_axi_aclk)
interrupt
v_voip_decap
s_axi
X16268-031616

Figure 2-11: **Decapsulator Block Diagram**

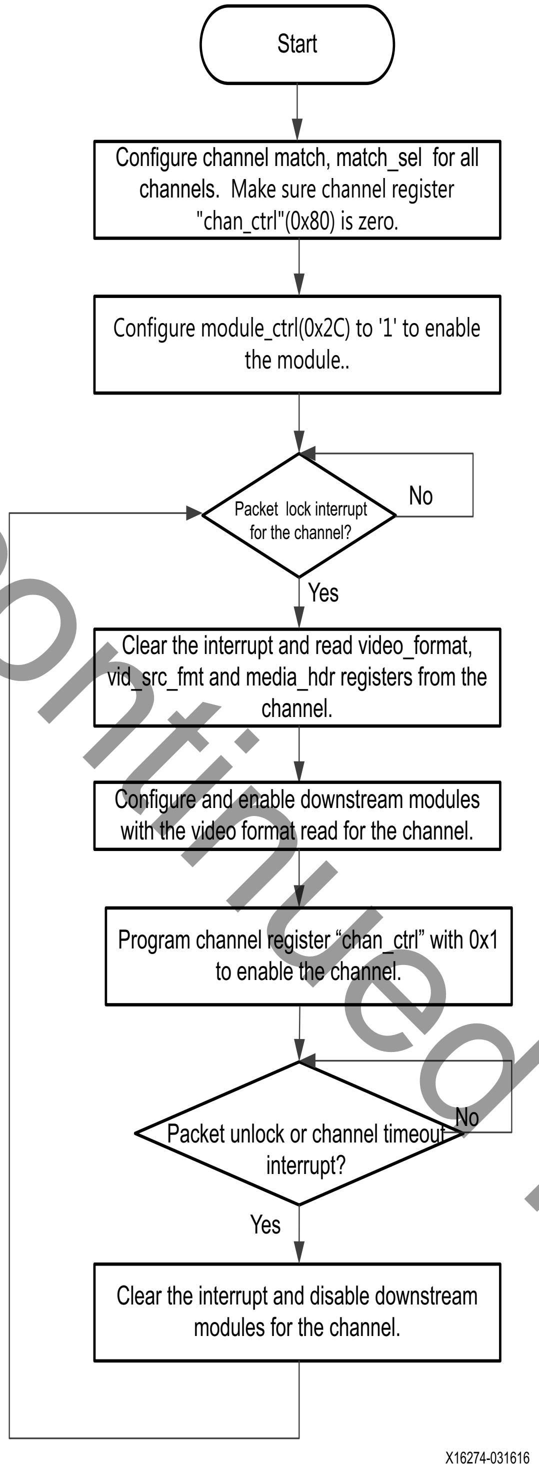
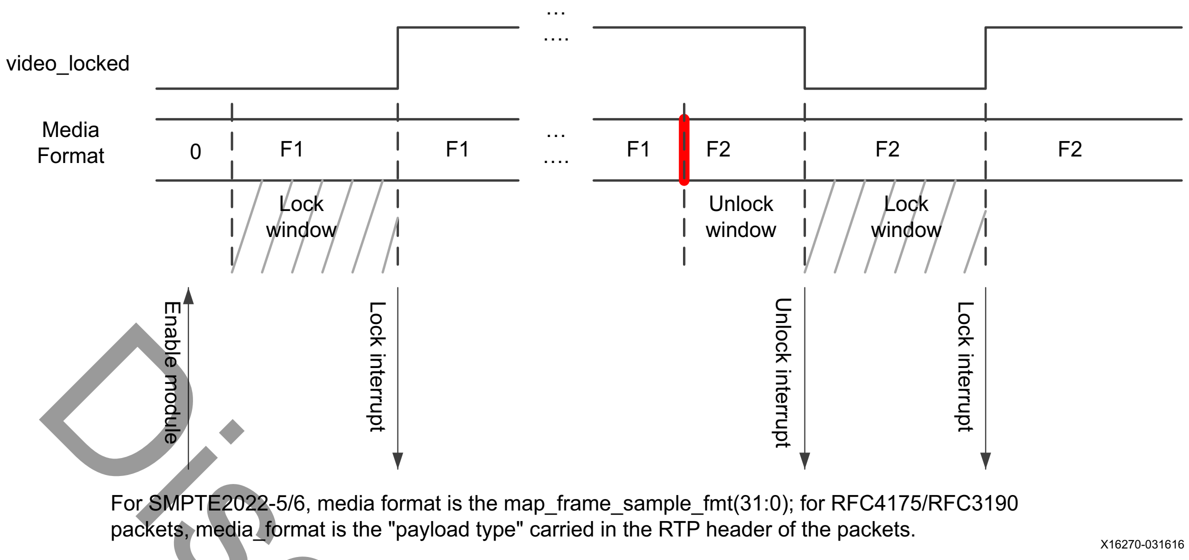

Start
Configure channel match, match_sel for all channels. Make sure channel register "chan_ctrl"(0x80) is zero.
Configure module_ctrl(0x2C) to '1' to enable the module..
Packet lock interrupt for the channel?
No
Yes
Clear the interrupt and read video_format, vid_src_fmt and media_hdr registers from the channel.
Configure and enable downstream modules with the video format read for the channel.
Program channel register "chan_ctrl" with 0x1 to enable the channel.
Packet unlock or channel timeout interrupt?
No
Yes
Clear the interrupt and disable downstream modules for the channel.
X16274-031616

Figure 2-12: **Decapsulator Operation Flow**

Note: To see how to access the channel register, refer to the "Configuration of Channel Registers Flowchart" in Figure 2-8 because the concept is the same.



For SMPTE2022-5/6, media format is the map_frame_sample_fmt(31:0); for RFC4175/RFC3190 packets, media_format is the "payload type" carried in the RTP header of the packets.

Figure 2-13: **Decapsulator Video Format Lock/Unlock Interrupt**

The Decapsulator module receives Ethernet datagrams and transmits RTP datagrams to the downstream module via the RTP over AXI4-Stream master interface. The Decapsulator module performs these operations:

- Channel mapping or filtering based on per channel filtering settings.
- Stripping off the Ethernet, IP and UDP headers to form an RTP packet.
- Media stream detection based on the media payload header from incoming packets. A few video stream detection interrupts are generated to notify the system about the packet stream change.

Note that the `s_axis_tuser` input port is a single-bit signal to indicate that the current packet is a good (1) or bad (0) packet. It is sampled when `s_axis_tlast` is valid. If you do not want to connect this signal, tie it to High.

The module has a `s_axi` (AXI4-Lite) interface that allows dynamic control of register settings. See Decapsulator.

The module is activated by programming the module enable to 1; otherwise, it will be in sleep or mute mode in which the module will not accept packets.

When the module is enabled, the incoming packets pass through these processing stages before it can be accepted by the module and sent out to downstream modules:

- **Channel Matching**: The Ethernet header (or sometimes the RTP header's payload type) information is used to match incoming packets into a different channel based on per channel programmed matching header information. If packets get mismatched, the packet will be dropped. Only the matched packets on the channel will be passed down to the next processing stage: video detection.
- **Stream detection (only for SMPTE ST 2022-6 and RFC4175)**: For SMPTE2022-5/6, the media format is the map_frame_sample_fmt(31:0); for RFC4175/RFC3190 packets, media_format is the "payload type" carried in the RTP header of the packets. By default, the media stream is unlocked in a channel and the channel is disabled. When a continuous number of received packets (lock window) are matched on the channel and have the same ST 2022-6 media format in the packet header, the channel is locked. A packet lock interrupt is fired. When the channel is in the media stream locked state, if a continuous number of received packets (unlock window) are matched on the channel and they have a different media format from the locked media format, the channel is unlocked. The channel is disabled by hardware and a packet unlock interrupt is fired. When the channel is locked and suddenly no packets get matched on the channel for a certain amount of time (user configured channel timeout: in number of core clocks), the channel is disabled by hardware and a channel timeout interrupt is fired. Note that all packets will be dropped on the channel if the channel is disabled or the channel is enabled but the first marker packet is not detected excepting RFC3190 which doesn't have Marker packet detection. Also note that media stream detection is activated only when the module is enabled.
- **Marker packet alignment**: For RFC3190, there is not marker packet concept and then there is not marker packet alignment for RFC3190 packet stream. For SMPTE2022-5/6 and RFC4175, the marker bit of RTP header is set for last packet of the video frame. When the channel gets enabled, all packets will be dropped until the very first marker packet is detected. All packets after the first marker packet are processed by the module and sent out to the downstream module. Note that marker pkt alignment is activated only when the channel is enabled.
  There are two register fields in the `channel_control` register that can be programmed to decide whether marker packet detection should be enabled or not. Another register bit is defined in the `channel_control` register to let you decide whether the first marker packet should be sent downstream or dropped inside the Decapsulator module.

Note: Packet process means stripping off the Ethernet/IP/UDP headers.

Register configuration is required for normal Decapsulator module operation.

Decapsulator decodes the current packet type based on the PT field extracted from the RTP header. (Note: for RFC4175/RFC3190, the PT field is also used as media format and channel matching parameters). Note that the module can only accept RTP version 2. The decoded packet type and other packet information is output on the `m_axis_tuser(31:0)` bus (see

Port Descriptions). The Protocol field of the IP header of the packet must be 0x11; otherwise, the packet cannot be matched.

For SMPTE2022-5/6 packets processing, the RTP header “PT(6:0)” payload type field are fixed. However, the RFC4175/RFC3190 encapsulated packets won’t have fixed payload type. The payload type is dynamic and programmed at the sender and passed to the receiver through SDP. At the receiver, for RFC4175/RFC3190, you must program the payload type for channel matching. After an incoming packet is matched on the channel, the packet processing continues.

Table 2-1: **Decapsulator: Decode Incoming Packet Type**

PT	m_axis_tuser (31:28)	Packet type	Comments
N.A	"0000"	Non-RTP	
"0100001" (33)	"0001"	RTP Encapsulated SMPTE ST 2022-2 Media Packet	
"1100000" (96)	"0001" or "0010"	RTP Encapsulated SMPTE ST 2022-1 FEC Packet	Further decoded as Column or Row FEC packet by FEC header.
"1100010" (98)	"0101"	RTP Encapsulated SMPTE ST 2022-6 Media Packet	
"1100011" (99)	"0110" or "0111"	RTP Encapsulated SMPTE ST 2022-5 FEC Packet	Further decoded as Column or Row FEC packet by FEC header.
(100~127)	“1000”	RTP Encapsulated RFC4175 Media Packet	Dynamic payload type.
(100~127)	“1001”	RTP Encapsulated RFC3190 Media Packet	Dynamic payload type.

Performance (Maximum Frequencies)

The performance of the Modular Media over IP Infrastructure is limited only by the FPGA logic speed. Each core utilizes only block RAMs, LUTs, and registers and contains no I/O elements.

The maximum achievable clock frequency can vary. The maximum achievable clock frequency and all resource counts can be affected by other tool options, additional logic in the FPGA, using a different version of Xilinx tools and other factors. See the resource utilization section for device family specific information.

Latency

For the ST 2022-6 Packetizer, different video formats (SD/HD/3G) causes different latencies on the module. The latency relies on the clock frequency and video format.

For the ST 2022-6 Depacketizer, SDI data is output only when the Elastic Buffer becomes full. The latency relies on both the clock frequency and Elastic Buffer size.

For the Framer, latency relies on the clock frequency.

For the Decapsulator, the latency is measured after marker packet alignment. In other words, the time spent performing video detection and frame boundary alignment do not contribute to the latency. Its latency relies on the clock frequency.

Table 2-2 shows the latency for all IP cores (based on simulation).

Table 2-2: **Latency**

IP Name	Clock Frequency	Latency (clock cycles)	Comments

ST 2022-6 Packetizer	core_clk=200 MHz
	s_axi_aclk=100 MHz
		SD: 8171@core_clk
		HD:1493@core_clk
		3G: 754@core_clk
			Media_payload_length=1376Bytes.
			Include video timestamp.

ST 2022-6 Depacketizer	core_clk=200 MHz
	s_axi_aclk=100 MHz
		910@core_clk	Elastic Buffer = 4.
			Media_payload_length=1376Bytes.
			Include video timestamp.

Framer	s_axis_clk=200 MHz
	m_axis_clk=156.25 MHz
	s_axi_aclk=100 MHz
		192@s_axis_clk	SMPTE ST 2022 5/6 packets.

Decapsulator	s_axis_clk=156.25 MHz
	m_axis_clk=200 MHz
	s_axi_aclk=100 MHz
		245@m_axis_clk	SMPTE ST 2022 5/6 packets.

Resource Utilization

For resource utilization for the ST 2022-6 Packetizer, ST 2022-6 Depacketizer, Decapsulator. and Framer, see the following links.

- ST 2022-Packetizer
- ST 2022-6 Depacketizer
- Framer
- Decapsulator

Port Descriptions

AXI4-Lite Interface

Table 2-3: **AXI-Lite Interface**

Signal Name	Direction	Width	Description
s_axi_aclk	In	1	AXI4-Lite clock.
s_axi_aresetn	In	1	AXI4-Lite active-Low reset.
s_axi_awaddr	In	32	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.
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 that DMA is 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 a 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	32	AXI4-Lite Read Address Bus.

Table 2-3: **AXI-Lite Interface *(Cont'd)***

Signal Name	Direction	Width	Description
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.

Notes:

1. Refer to the *Vivado Design Suite: AXI Reference Guide* (UG1037) [Ref 14] on the AXI4-Lite interface and its protocol.

RTP over AXI4-Stream Interface Protocol

Table 2-4: **RTP over AXI4-Stream Interface Protocol**

Signals (Master/Slave)	Direction (Master/ Slave)	Description
m/s_axis_tvalid	Out/In	Valid indicator for m/s_axis_tdata, m/s_axis_tlast, m/s_axis_tuser signals.
m/s_axis_tdata[63:0]	Out/In	Data
m/s_axis_tlast	Out/In	High at the last word of the output packet

Table 2-4: **RTP over AXI4-Stream Interface Protocol *(Cont'd)***

Signals (Master/Slave)	Direction (Master/ Slave)	Description			
m/s_axis_tuser[31:0]	Out/In	Bit	Abbreviation	Description	
		0	Packet Start	High only at the first valid word of the output packet.	
		2:1	Protocol Version	Protocol version ("00")	
		14:3	Channel Number	Shall be valid at packet start	
		15	Reserved		
		26:16	Packet Length	Shall be valid at payload start. It is the sum of packet length in bytes.	
		27	Reserved		
		31:28	Packet Type	0000	UDP encapsulated
				0001	RTP encapsulated SMPTE ST 2022-2 compliant media packet
				0010	RTP encapsulated SMPTE ST 2022-1 compliant Column FEC
				0011	RTP encapsulated SMPTE ST 2022-1 compliant Row FEC packet
				0101	RTP encapsulated SMPTE ST 2022-6 compliant media packet
				0110	RTP encapsulated SMPTE ST 2022-5 compliant Column packet
				0111	RTP encapsulated SMPTE ST 2022-5 compliant Row FEC packet
				1000	RTP encapsulated RFC4175 compliant media packet
				1001	RTP encapsulated RFC3190 compliant media packet
m/s_axis_tready	In/Out	TREADY indicates that the slave can accept a transfer in the current cycle.			

SDI over AXI4-Stream Interface Protocol (Master or Slave Interfaces)

Table 2-5: **SDI over AXI4-Stream Interface Protocol (Master or Slave Interfaces)**

<table>
<tr><th>Signals</th><th>Direction (Master/Slave)</th><th colspan="3">Description</th></tr>
<tr><td>sdi_axis_tready</td><td>In/Out</td><td colspan="3">TREADY indicates that the slave can accept a transfer in the current cycle.</td></tr>
<tr><td>sdi_axis_tvalid</td><td>Out/In</td><td colspan="3">Valid indicator for sdi_axis_tdata, sdi_axis_tlast, and sdi_axis_tuser signals.</td></tr>
<tr><td>sdi_axis_tlast</td><td>Out/In</td><td colspan="3">Together with sdi_axis_tdata to indicate the last word of the frame.</td></tr>
<tr><td rowspan="6">sdi_axis_tdata</td><td rowspan="6">Out/In</td><td><b>Bit</b></td><td><b>Native SDI Mapping</b></td><td><b>Description</b></td></tr>
<tr><td>9:0</td><td>ds1a</td><td>Video data stream 1 which is dependent on the SDI mode:
• SD-SDI: Multiplexed Y/C data stream
• HD-SDI: Y data stream
• 3G-SDI level A: Data stream 1
• 3G-SDI level B-DL: Data stream 1 of link A
• 3G-SDI level B-DS: Y data stream of HD-SDI signal 1</td></tr>
<tr><td>19:10</td><td>ds2a</td><td>Video data stream 2 which is dependent on the SDI mode:
• SD-SDI: Not used
• HD-SDI: C data stream
• 3G-SDI level A: Data stream 2
• 3G-SDI level B-DL: Data stream 2 of link A
• 3G-SDI level B-DS: C data stream of HD-SDI signal 1</td></tr>
<tr><td>29:20</td><td>ds1b</td><td>This is only used in 3G-SDI level B mode. The data stream on this port is:
• 3G-SDI level B-DL: Data stream 1 of link B
• 3G-SDI level B-DS: Y data stream of HD-SDI signal 2</td></tr>
<tr><td>39:30</td><td>ds2b</td><td>This is only used in 3G-SDI level B mode. The data stream on this port is:
• 3G-SDI level B-DL: Data stream 2 of link B
• 3G-SDI level B-DS: C data stream of HD-SDI signal 2</td></tr>
<tr><td>63:40</td><td></td><td>Reserved</td></tr>
</table>

Table 2-5: **SDI over AXI4-Stream Interface Protocol (Master or Slave Interfaces) *(Cont'd)***

Signals	**Direction (Master/Slave)**	**Description**			
		Bit	**Abbreviation**	**Description**	
sdi_axis_tuser	Out/In	0	sof	Indicates Start of Frame	
		1	Reserved		
		3:2	sdi_mode	00	HD-SDI
				01	SD-SDI
				10	3G-SDI
				11	Invalid Video Format
		4	level_b_3g	In 3G-SDI mode, this output is asserted High when the input signal is level B and Low when it is level A. This output is only valid when rx_mode_3g is High.	
		5	rx_bit_rate	This input port indicates which bit rate is being received in HD-SDI and 3G-SDI modes. When using the transceivers in Xilinx FPGAs, the device-specific transceiver control module contains a bit rate detector that generates the signal to be connected to the rx_bit_rate input port. HD-SDI mode: • rx_bit_rate = 0: Bit rate = 1.485 Gb/s • rx_bit_rate = 1: Bit rate = 1.485/1.001 Gb/s 3G-SDI mode: • rx_bit_rate = 0: Bit rate = 2.97 Gb/s • rx_bit_rate = 1: Bit rate = 2.97/1.001 Gb/s	
		29:6	vid_src_fmt	Bit	Description
				29:26	MAP
				25:18	FRAME
				17:10	FRATE
				9:6	SAMPLE
		31:30	Reserved		

ST 2022-6 Packetizer Port Descriptions

Table 2-6: **ST 2022-6 Packetizer Port Description**

Signal	Direction	Description
core_clk	input	Main clock for the core.
core_reset	input	Main reset for the core, high active.
rtp_timestamp[31:0]	input	Used inside RTP header.
video_timestamp[31:0]	input	Used inside SMPTE ST 2022-6 media payload header.

Notes:

1. For `sdi_axis` interface refer to SDI over AXI4-Stream Interface Protocol (Master or Slave Interfaces).
2. For `m_axis` interface refer to

ST 2022-6 Depacketizer Port Descriptions

Table 2-7: **ST 2022-6 Depacketizer Port Descriptions**

Signal	Direction	Description
core_clk	input	Main clock for the core.
core_reset	input	Main reset for the core, active-High.
pkt_rdy	input	Used to indicate ST 2022-6 Depacketizer to request packet.
pkt_req	output	Single cycle pulse to request for a packet, generated only when pkt_rdy is High.
interrupt	output	Active-High interrupt, which is asserted when any interrupt source in the module is triggered. It will go down when all interrupt source are cleared.

Notes:

1. For `sdi_axis` interface refer to SDI over AXI4-Stream Interface Protocol (Master or Slave Interfaces).
2. For `s_axis` interface refer to RTP over AXI4-Stream Interface Protocol

Framer Port Descriptions

Table 2-8: **Framer Port Descriptions**

Signal	Direction	Description
s_axis_clk	input	Clock for s_axis interface logic.
s_axis_rst	input	Active-High reset for s_axis interface logic.
m_axis_clk	input	Clock for m_axis interface logic.
m_axis_rst	input	Active-High reset for m_axis interface logic.
m_axis_tready	input	TREADY indicates that the slave can accept a transfer in the current cycle.
m_axis_tvalid	output	Valid indicator for m_axis_tdata, m_axis_tlast, and m_axis_tuser signals.

Table 2-8: **Framer Port Descriptions *(Cont'd)***

Signal	Direction	Description
m_axis_tlast	output	High at the last word of the output packet.
m_axis_tdata(63:0)	output	Data
m_axis_tkeep(7:0)	output	Byte enable for each byte in m_axis_tdata.

Notes:

1. For `s_axis` interface refer to RTP over AXI4-Stream Interface Protocol.

Decapsulator Port Descriptions

Table 2-9: **Decapsulator Port Descriptions**

Signal	Direction	Description
s_axis_clk	input	Clock for s_axis interface logic.
s_axis_rst	input	Active-High reset for s_axis interface logic.
m_axis_clk	input	Clock for m_axis interface logic.
m_axis_rst	input	Active-High reset for m_axis interface logic.
s_axis_tready	output	TREADY indicates that the slave can accept a transfer in the current cycle.
s_axis_tvalid	input	Valid indicator for m_axis_tdata, m_axis_tlast, and m_axis_tuser signals.
s_axis_tlast	input	High at the last word of the output packet
s_axis_tdata[63:0]	input	Data
s_axis_tkeep[7:0]	input	Byte enable for each byte in s_axis_tdata. ***Note:*** Currently this port is not used inside the design.
s_axis_tuser	input	Single bit signal to indicate that the current packet is a good or bad packet. And it is sampled when s_axis_tlast is valid. If you do not want to connect this signal, please tie it to High.
interrupt	output	Active-High interrupt, which is asserted when any interrupt source in the module is triggered. It will go down when all interrupt source are cleared.

Notes:

1. For `m_axis` interface refer to

Register Space

ST 2022-6 Packetizer Module Register Space

Table 2-10: **ST 2022-6 Packetizer Module Register Map**

Address Offset (HEX)	Register Name	Access Type (HW: hardware writable)	Cleared with SOFT reset	Default Value (HEX)	Description		
					Bit Range	Value	
0x0000	control	R/W	N	0x00000000	**Control**		
					31:1	Reserved	
					0	**Soft reset**	
						1	Reset all pcore registers
						0	Unreset the pcore registers
0x0008	peak_buf_level	R	Y	0x00000000	**Peak Packet Buffer Level**		
					31:6	Reserved	
					5:0	Peak packet buffer level, in number of packets.	
0x000C	version	R	N	0x01000000	**Hardware version**		
					31:24	Version major	
					23:16	Version minor	
					15:12	Version revision	
					11:8	Patch ID	
					7:0	Revision number	
0x0010	module_ctrl	R/W	Y	0x00000000	**Module Control**		
					31:28	Reserved	
					27:16	Channel number, which is reflected in the tuser output bit 14 to bit 3.	
					15:2	Reserved	
						Lossless mode configuration	
					1	0	Normal mode
						1	Lossless mode
					0	Module Enable	
						0	Module is disabled.
						1	Module is enabled.

Table 2-10: **ST 2022-6 Packetizer Module Register Map *(Cont'd)***

Address Offset (HEX)	Register Name	Access Type (HW: hardware writable)	Cleared with SOFT reset	Default Value (HEX)	Description		
					Bit Range	Value	
0x0014	video_format	R/W	Y	0x00000000	**Video Format**		
					31	Module Video Format and Video Source Format selection	
						0	Obtain Video Format and Video Source Format from SDI over AXI4-Stream TUSER bus.
						1	Obtain Video Format and Video Source Format from programmed register (Address 0x14 and 0x18) value.
					30:4	Reserved	
					3	bit_rate	
						0	Not divided by 1.001.
						1	Divided by 1.001.
					2	3G Level A/B	
						0	3G Level A
						1	3G Level B
					1:0	video_mode	
						00	HD
						01	SD
						10	3G
						11	None
0x0018	vid_src_fmt	R/W	Y	0x00000000	**Video Source Format [Ref 4]**		
					31:28	MAP	
					27:20	FRAME	
					19:12	FRATE	
					11:8	SAMPLE	
					7:0	Reserved	

Table 2-10: **ST 2022-6 Packetizer Module Register Map *(Cont'd)***

Address Offset (HEX)	Register Name	Access Type (HW: hardware writable)	Cleared with SOFT reset	Default Value (HEX)	Description		
					Bit Range	Value	
0x001C	media_hdr	R/W	Y	0x00000000	**Media Header**		
					31:20	Reserved	
					19:16	Clock frequency (CF) 4 bits for video time stamp:	
						0000	No time stamp
						0001	27 MHz
						0010	148.5 MHz
						0011	297 MHz
						0101	297/1.001 MHz
						0110-1111	Reserved
					15:3	Reserved	
					2:1	Reference for video time stamp:	
						00	Not locked
						01	Reserved
						10	Locked to Coordinated Universal Time (UTC) time/ frequency reference
						11	Locked to private time frequency reference
					0	Video time stamp include (set this bit according to CF value)	
						0	Do not include video time stamp (CF = 0000)
						1	Include video time stamp (CF /= 0000)
0x0020	media_pyld_len	R/W	Y	0x00000560	**media_payload_length (Default is 1,376; program only when different)**		
					31:11	Reserved	
					10:0	media payload length	
0x0024	ssrc	R/W	Y	0x00000000	**RTP Header SSRC Value**		
					31:0	32-bit Synchronization Source Identifier (SSRC) value for RTP header. It is used to create RTP header.	

Table 2-10: **ST 2022-6 Packetizer Module Register Map *(Cont'd)***

Address Offset (HEX)	Register Name	Access Type (HW: hardware writable)	Cleared with SOFT reset	Default Value (HEX)	Description		
					Bit Range	Value	
0x0028	output_pkt_cnt	R	Y		**Statistics: Output Packet Count**		
					31:0	RTP Encapsulated SMPTE ST 2022-6 Media Packet Count	
0x002C	frame_cnt	R	Y	0x00000000	**Statistics: Frame Count**		
					31:0	Incoming SDI Frame Count	
0x0030	stat_reset	W	Y	0x00000000	**Clear Statistics Registers (Self Clear)**		
					31:3	Reserved	
					2	Clear peak buffer level statistics registers	
						0	Do not clear.
						1	Clear.
					1	Clear frame count statistics registers	
						0	Do not clear
						1	Clear.
					0	Clear output pkt count statistics registers	
						0	Do not clear.
						1	Clear.
0x0040	interrupt_status	R	Y	0x00000000	**Interrupt Status**		
					31:2	Reserved	
					1	Buffer become empty interrupt during operation	
						0	Interrupt cleared
						1	Interrupt still valid.
					0	Reserved	
0x0044	interrupt_mask	R/W	Y	0x00000000	**Interrupt Masking (1 means masking)**		
					31:2	Reserved	
					1	Mask buffer become empty interrupt	
						0	Do not mask
						1	Mask
					0	Reserved	

Table 2-10: **ST 2022-6 Packetizer Module Register Map *(Cont'd)***

Address Offset (HEX)	Register Name	Access Type (HW: hardware writable)	Cleared with SOFT reset	Default Value (HEX)	Description		
					Bit Range	Value	
0x0048	interrupt_clear	R/W	Y	0x00000000	Interrupt Clear (clear upon write)		
					31:2	Reserved	
					1	Clear buffer become empty interrupt	
						0	Do not clear
						1	Clear
					0	Reserved	

control (0x0000)

Bit 0 is a soft reset that is used to reset all other registers.

peak_buf_level (0x0008)

This statistics register shows the highest buffer level hit during processing. When this value is equal to "Elastic Buffer (Packets)" generic setting, it means the buffer is full and push back occurs to the upstream module.

module_ctrl (0x0010)

The channel number in this register is used to configure the channel number assigned on this instance of ST 2022-6 Packetizer. The channel number information is needed at the master AXI4-Stream port `tuser` bus. The lossless mode bit is used to configure the module to be either in normal mode (0) or lossless mode (1). Normally the module is set to normal mode (bit1 set to 0). The module enable bit is used to enable or disable the module. When the module is enabled, it starts to accept and process incoming data. When it is disabled, it resets the internal packet buffer and state machine. (When disabled, the module will push back to the upstream module).

video_format (0x0014)

Bit 31 is the video format selection bit. When bit 31 is set to 0, the video format used by the module is taken from the AXI4-Stream interface `TUSER` bus. When bit 31 is set to 1, the video format used by the module is taken from the video_format register and the vid_src_fmt register.

vid_src_fmt (0x0018)

Video source format that is used by hardware only when the video_format register bit 31 is set to 1.

media_hdr (0x001C)

Configure this register carefully because it affects the final RTP packet size. The system level should decide whether the video time stamp should be inserted in the final RTP packet or not. The source of the video time stamp should also be set correspondingly in the register. Note that bit 0 is derived from bit 19:16 (CF). When CF(3:0)=0000, bit 0 should be set to 0 as well which indicates no video time stamp. Bit 0 is set to 1 when CF is not equal to 0000.

media_pyld_len (0x0020)

The default value of the media payload length from this register is 1,376 bytes. Normally, you are required not to touch this register because the media payload length is normally 1,376 bytes.

ssrc (0x0024)

This register is used to configure the ssrc field in the RTP header of the final RTP packet. Refer to the SMPTE ST 2022_6 standard [Ref 4].

output_pkt_cnt (0x0028)

This statistics register shows the number of packets sent out from the module.

frame_cnt (0x002C)

This statistics register shows the number of frames received at the SDI_AXIS slave port of the module.

stat_reset (0x0030)

Each bit of this register resets the corresponding statistics register to zero. This stat_reset register is self-clearing; you only need to program once for statistics reset.

ST 2022-6 Depacketizer Module

Table 2-11: **ST 2022-6 Depacketizer Module Register Map**

Address Offset	Register Name	Access Type	Cleared with SOFT reset	Default Value	Description		
					Bit Range	Register Description	
0x0000	control	R/W	N	0x00000000	**Control**		
					31:1	Reserved	
					0	Soft reset	
						1	Reset all pcore registers
						0	Unreset the pcore registers
0x0008	buf_level	R	Y	0x00000000	**Packet Buffer Level**		
					31:22	Reserved	
					21:16	Current buffer level in number of packets.	
					15:6	Reserved	
					5:0	Peak packet buffer level in number of pkts.	
0x000C	version	R	N		**Hardware version**		
					31:24	Version major	
					23:16	Version minor	
					15:12	Version revision	
					11:8	Patch ID	
					7:0	Revision number	
0x0010	module_ctrl	R/W	Y	0x00000000	**Module Control**		
					31:1	Reserved	
					0	Module Enable	
						1	Module Enabled
						0	Module Disabled

Table 2-11: **ST 2022-6 Depacketizer Module Register Map *(Cont'd)***

Address Offset	Register Name	Access Type	Cleared with SOFT reset	Default Value	Description		
					Bit Range	Register Description	
0x0014	video_format	R/W	Y	0x00000000	**Video Format (Refer to SMPTE ST 2022-6 specification)**		
					31:4	Reserved	
					3	bit_rate	
						0	Not divided by 1.001.
						1	Divided by 1.001.
					2	3G Level A/B, used to instruct the module to do depacketizing.	
						0	3G Level A
						1	3G Level B
					1:0	video_mode, used to instruct the module to do depacketizing.	
						00	HD
						01	SD
						10	3G
						11	None
0x0018	vid_src_fmt	R/W	Y	0x00000000	**Video Source Format (Refer to SMPTE ST 2022-6 specification. [Ref 4])**		
					31:28	MAP	
					27:20	FRAME	
					19:12	FRATE	
					11:8	SAMPLE	
					7:0	Reserved	
0x001C	media_hdr	R/W	Y	0x00000000	**Media Header (Software program with the same value read from Decap module)**		
					31:1	Reserved	
					0	Video time stamp includes:	
						0	Do not include video time stamp.
						1	Include video time stamp.

Table 2-11: **ST 2022-6 Depacketizer Module Register Map *(Cont'd)***

Address Offset	Register Name	Access Type	Cleared with SOFT reset	Default Value	Description		
					Bit Range	Register Description	
0x0020	media_pyld_ len	R/W	Y	0x00000560	**media_payload_length (Default is 1,376; program only when it is different)**		
					31:11	Reserved	
					10:0	media payload length.	
0x0024	frame_size	R/W	Y	0x00000000	**Frame Size (Program only when software overwrites frame size; otherwise frame size is decoded in hw from vid_src_fmt register value.)**		
					31	Frame size selection	
						0	Decoded from vid_src_fmt. You are required to program the vid_src_fmt register.
						1	User configured. Vid_src_fmt register is not used in hardware.
					30:27	Reserved	
					26:16	Last datagram length: number of bytes for last datagram of frame.	
					15:0	Datagram per frame: number of datagrams (packets) for one frame.	
0x0028	input_pkt_cnt	R	Y	0x00000000	**Statistics: Input Packet Count**		
					31:0	Number of packets received after the very first marker packet detected.	
0x002C	frame_cnt	R	Y	0x00000000	**Statistics: Frame Count**		
					31:0	Number of frames received.	

Table 2-11: **ST 2022-6 Depacketizer Module Register Map *(Cont'd)***

Address Offset	Register Name	Access Type	Cleared with SOFT reset	Default Value	Description			
					Bit Range	Register Description		
0x0030	stat_reset	W	Y	0x00000000	**Clear Statistics Registers (self clear)**			
					31:3	Reserved		
					2	Clear peak_buf_level register:		
					1	Clear frame_cnt statistics registers:		
					0	Clear input_pkt_cnt statistics registers:		
0x0040	interrupt_status	R/W	Y	0x00000000	**Interrupt status**			
					31:2	Reserved		
					1	Buffer become empty interrupt during operation		
							0	Interrupt cleared.
							1	Interrupt still valid.
					0	Reserved		
0x0044	interrupt_mask	R/W	Y	0x00000000	**Interrupt masking (1 means masking)**			
					31:2	Reserved		
					1	Mask buffer become empty interrupt		
							0	Do not mask.
							1	Mask.
					0	Reserved		

Table 2-11: **ST 2022-6 Depacketizer Module Register Map *(Cont'd)***

					Description		
Address Offset	**Register Name**	**Access Type**	**Cleared with SOFT reset**	**Default Value**	**Bit Range**	**Register Description**	
0x0048	interrupt_clear	R/W	Y	0x00000000	**Interrupt clear (clear upon write)**		
					31:2	Reserved	
					1	Clear buffer become empty interrupt	
						0	Do not clear.
						1	Clear.
					0	Reserved	

control (0x0000)

Bit 0 is used to reset all pcore registers.

buf_level (0x0008)

Bit 5 to bit 0 shows the highest buffer level hit during processing. When this value is equal to "Elastic Buffer (Packets)" generic setting, it means the buffer is full and push back occurred to the upstream module. Bit 21 to bit 16 shows the current buffer level in the number of packets stored in the buffer.

module_ctrl (0x0010)

The module enable bit is used to enable or disable the module. When the module is enabled, it starts searching for the first marker big packet and will accept and process incoming packets. When it is disabled, it resets the internal packet buffer and state machine. (Module will push back on input port).

video_format (0x0014)

This video format information is needed by the module to do proper processing. You are able to obtain this information from the Decapsulator registers.

vid_src_fmt (0x0018)

This video format information is needed by the module to do proper processing. You are able to obtain this information from the Decapsulator registers.

media_hdr (0x001C)

This media payload information is needed by the module to do proper processing. You are able to obtain this information from the Decapsulator registers.

media_pyld_len (0x0020)

This is the media payload length information. The default value of this register is 1376 (1376 Bytes).

frame_size (0x0024)

This register is optional. It is used only when you have a special video format that has a special frame size which is not in the frame size table hard coded in hardware (the frame size is industry standard).

input_pkt_cnt (0x0028)

This statistics register only counts the number of valid packets that are after the first marker packet when the module is enabled. Module disable or stat_reset both can reset the register to zero.

frame_cnt (0x002C)

This statistics register shows the number of frames processed by the module. It is reset by module disable or stat_reset.

stat_reset (0x0030)

Each bit of this register resets the corresponding statistics registers to zero. This stat_reset register is self-clearing.

interrupt_status (0x0040)

The module might have multiple interrupt sources. Each bit of this register shows the interrupt status from each source. After the interrupt source is served and cleared, the bit becomes zero.

Bit 1 is the buffer become empty interrupt. In normal operation, module buffer level will remain around "Elastic Buffer (Packets)" generic settings. If a system error happens or there is a packet stream interruption, the buffer level might decrease and finally become empty (an interrupt is fired in this condition).

interrupt_mask (0x0044)

You can decide to mask some interrupt sources for the channel. This is done before interrupt occurrences. After an interrupt has occurred, the interrupt can only be cleared; masking does not clear the interrupt.

interrupt_clear (0x0048)

When an interrupt occurs, software serves the interrupt and clears the interrupt by programming the corresponding bit of this register to 1. After clearing the interrupt, the corresponding bit in the interrupt_status register becomes zero.

Framer Module Register Space

Table 2-12: **Framer Module Register Map**

Address Offset (HEX)	Register Name	Access Type	Cleared with SOFT reset	Default Value (HEX)	Description		
					Bit Range	Value	
General							
0x0000	control	R/W	N	0x00000000	**Control**		
					31:2	Reserved	
					1	**Channel Update** Allows configured registers to take effect for the channel on Channel Access at offset 0x08.	
					0	**Soft reset**	
						1	Reset all pcore registers
						0	Unreset all pcore registers.
0x0004	status	R	N	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
0x0008	channel_access	R/W	Y	0x00000000	**Channel Access**		
					31:12	Reserved	
					11:0	The channel number to access registers	

Table 2-12: **Framer Module Register Map *(Cont'd)***

Address Offset (HEX)	Register Name	Access Type	Cleared with SOFT reset	Default Value (HEX)	Description		
					Bit Range	Value	
0x000C	sys_config	R	N	0x00000000	**System Configuration**		
					31:25	Reserved	
					24	OVERFLOW PKT DROP	
						0	No drop when buffer is full.
						1	Drop packets when buffer is full.
					23:12	Reserved	
					11:0	MAX CHANNELS: Number of channels supported	
0x0010	version	R	N	0x01000000	**Hardware Version**		
					31:24	Version major	
					23:16	Version minor	
					15:12	Version revision	
					11:8	Patch ID	
					7:0	Revision number	
0x0014	src_mac_low_ addr	R/W	Y	0x00000000	**Source MAC Low Address Register**		
					31:0	Lower 32-bit of the mac address	
0x0018	src_mac_high_ addr	R/W	Y	0x00000000	**Source MAC High Address Register**		
					31:16	Reserved	
					15:0	Upper 16-bit of the mac address	
0x001C	peak_buf_level	R	Y	0x00000000	**Peak Packet Buffer Level**		
					31:6	Reserved	
					5:0	Peak packet buffer level, in number of packets.	
0x0020	rx_pkt_cnt	R	Y	0x00000000	**Received Packet Count**		
					31:0	Received packets count.	
0x0024	drop_pkt_cnt	R	Y	0x00000000	**Drop Packet Count**		
					31:0	Dropped number of packets at framer input.	

Table 2-12: **Framer Module Register Map *(Cont'd)***

Address Offset (HEX)	Register Name	Access Type	Cleared with SOFT reset	Default Value (HEX)	Description		
					Bit Range	Value	
0x0030	stat_reset	W	Y	0x00000000	**Clear General Space Statistics Registers (Self Clear)**		
					31:3	Reserved	
					2	Clear drop_pkt_cnt statistics register	
						0	Do not clear
						1	Clear
					1	Clear rx_pkt_cnt statistics register:	
						0	Do not clear
						1	Clear
					0	Clear peak_buf_level statistics register:	
						0	Do not clear
						1	Clear
Channel							
0x0080	chan_ctrl	R/W	Y	0x00000000	**Channel Control**		
					31:1	Reserved	
					0	Output transmit enable	
						0	Transmit Disable
						1	Transmit Enable
0x0084	dest_mac_low_addr	R/W	Y	0x00000000	**Destination MAC Low Address Register**		
					31:0	Lower 32-bits of the mac address	
0x0088	dest_mac_high_addr	R/W	Y	0x00000000	**Destination MAC High Address Register**		
					31:16	Reserved	
					15:0	Upper 16-bit of the mac address	
0x008C	vlan_tag_info	R/W	Y	0x00000000	**Virtual Local Area Network (VLAN) Fields**		
					31	0	without VLAN
						1	with VLAN
					30:16	Reserved	
					15:0	vlan_pcp_cfi_vid	

Table 2-12: **Framer Module Register Map *(Cont'd)***

Address Offset (HEX)	Register Name	Access Type	Cleared with SOFT reset	Default Value (HEX)	Description		
					Bit Range	Value	
0x0090	ip_hdr_media	R/W	Y	0x00000000	**ip ver/ihl/tos/identification fields for media pkts**		
					31:16	Reserved	
					15:8	Type of service (TOS)	
					7:0	Time to live (TTL)	
0x0094	ip_hdr_fec	R/W	Y	0x00000000	**ip ver/ihl/tos/identification fields for fec pkts**		
					31:16	Reserved	
					15:8	Type of service (TOS)	
					7:0	Time to live (TTL)	
0x0098	src_ip_host_low_addr	R/W	Y	0x00000000	**Source IP Host Low Address**		
					31:0	32-bit source IP address	
0x00A8	dest_ip_host_low_addr	R/W	Y	0x00000000	**Destination IP host low address**		
					31:0	32-bit destination IP address	
0x00B8	udp_src_port	R/W	Y	0x00000000	**UDP Source Port**		
					31:16	Reserved	
					15:0	UDP source port.	
0x00BC	udp_dest_port	R/W	Y	0x00000000	**UDP Destination Port**		
					31:16	Reserved	
					15:0	UDP destination port.	
0x00C0	tx_pkt_cnt	R	Y	0x00000000	**Transmitted Packet Count**		
					31:0	Number of transmitted packets	
0x00C4	chan_stat_reset	W	Y	0x00000000	**Clear Channel Statistics Registers (Self Clear)**		
					31:1	Reserved	
						Clear tx_pkt_cnt statistics register:	
					0	0	Do not clear
						1	Clear

control (0x0000)

Bit 0 is used to reset all registers (not the core logic). 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 bit1 to 1 when updates are completed ensures all channel space registers are updated simultaneously.

status (0x0004)

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

channel_access (0x0008)

This register is used when accessing channel space registers. Always set the channel first.

sys_config (0x000C)

Read only register to show current generic settings for the module.

src_mac_low_addr (0x0014)

The source MAC address that is common for all channels. This register is the lower 32-bits of the source MAC address.

src_mac_high_addr (0x0018)

Higher 16-bits of the source MAC address.

peak_buf_level (0x001C)

This statistics register shows the highest buffer level hit during processing. When this value is equal to “Elastic Buffer (Packets)” generic setting, it means the buffer is full and incoming packets get dropped (Overflow Handling Strategy set to 1) or push back occurs to the upstream module (Overflow Handling Strategy is set to 0). Note that the packet buffer in this module is shared among all channels.

rx_pkt_cnt (0x0020)

General statistics to show the total number of packets received at the AXI4-Stream slave input interface from all channels.

drop_pkt_cnt (0x0024)

When the module generic “Overflow Handling Strategy” is set to 1, packets are dropped when the packet buffer becomes full. In this case, the drop_pkt_cnt increases.

stat_reset (0x0030)

Each bit of this register resets the corresponding general space statistics registers to zero. This stat_reset register is self-clearing; you only need to program once for statistics reset.

chan_ctrl (0x0080)

The channel control register that controls each channel. Bit 0 is used to enable transmitting at the AXI4-Stream master interface of the module. If it is set to zero, packets from the AXI4-Stream slave will not be pushed back but are dropped inside the module. The drop_pkt_cnt will not increase for these dropped packets since it is not caused by the buffer being full and the AXI4-Stream master interface will drive zeros and `Tvalid` will be Low. If it is set to 1, the packets are processed and sent onto the AXI4-Stream master interface.

dest_mac_low_addr (0x0084)

Destination MAC address lower 32 bits for the packets received on the channel.

dest_mac_high_addr (0x0088)

Destination MAC address higher 16 bits for the packets received on the channel.

vlan_tag_info (0x008C)

Bit 31 is used to instruct the module to insert (1) or not insert (0) in the VLAN field in the packet header for the channel. Bit 15:0 is the vlan_pcp_cfi_vid value that is inserted into the packet header when bit 31 is 1.

ip_hdr_media (0x0090)

Used to configure the TOS/TTL field for the non-FEC packets received on the channel.

ip_hdr_fec (0x0094)

Used to configure the TOS/TTL field for the FEC packets received on the channel.

src_ip_host_low_addr (0x0098)

Source IP address for the packets received on the channel.

dest_ip_host_low_addr (0x00A8)

Destination IP address for the packets received on the channel.

udp_src_port (0x00B8)

UDP source port for the packets received on the channel.

udp_dest_port (0x00BC)

UDP destination port for the non-FEC packets received on the channel. For the column FEC packets, the UDP destination port number will be the register value + 0x2. And for the row FEC packets, the UDP destination port number will be the register value + 0x4.

tx_pkt_cnt (0x00C0)

Number of packets that are transmitted out on the channel. When `transmit_enable` is set to zero for the channel, this statistics register remains unchanged.

chan_stat_reset (0x00C4)

Each bit of this register is used to reset the corresponding channel statistics registers.

Decapsulator Module Register Map

Table 2-13: **Decapsulator Module Register Map**

Address Offset (HEX)	Register Name	Access Type	Cleared with SOFT reset	Default Value (HEX)	Description		
					Bit Range	Value	
General							
0x0000	control	R/W	N	0x00000000	Control		
					31:2	Reserved	
					1	Channel update Allows configured registers to take effect for the channel on Channel Access	
					0	Soft reset	
						1	Reset all pcore
						0	Unreset all pcore registers.

Table 2-13: **Decapsulator Module Register Map** ***(Cont'd)***

Address Offset (HEX)	Register Name	Access Type	Cleared with SOFT reset	Default Value (HEX)	Description		
					Bit Range	Value	
0x0004	status	R	N	0x00000000	Status		
					31:1	Reserved	
					0	Update Busy	
						1	Core busy updating configured parameters or
						0	Core able to accept new
0x0008	channel_access	R/W	Y	0x00000000	Channel Access		
					31:12	Reserved	
					11:0	The channel number to access registers	
0x000C	sys_config	R	N	0x00000000	System Configuration: Generics		
					31:12	Reserved	
					11:0	MAX CHANNELS	
0x0010	version	R	N	0x01000000	Hardware Version		
					31:24	Version major	
					23:16	Version minor	
					15:12	Version revision	
					11:8	Patch ID	
					7:0	Revision number	
0x0014	pkt_lock_window	R/W	Y	0x04000400	Packet Lock/Unlock Window Size		
					31:16	Packet Unlock Window	
					15:0	Packet Lock Window	
0x0018	rx_pkt_cnt	R	Y	0x00000000	Number Of Packets Received At Input Of Decap Module		
					31:0	Total number of packets received at Decap module input before matching	

Table 2-13: **Decapsulator Module Register Map *(Cont'd)***

Address Offset (HEX)	Register Name	Access Type	Cleared with SOFT reset	Default Value (HEX)	Description		
					Bit Range	Value	
0x001C	mismatched_pkt_cnt	R	Y	0x00000000	Mismatched Packets		
					31:0	Number of packets mismatched	
0x0020	err_pkt_cnt	R	Y	0x00000000	Error Packets Received From MAC		
					31:0	Number of received packets that are error packets from the s_axis interface. (Bad packets indicated by s_axis_tuser=0 when s_axis_tlast is valid).	
0x0024	stat_reset	W	Y	0x00000000	Clear General Space Statistics Registers (Self Clear)		
					31:4	Reserved	
					3	Clear peak_buf_level general statistics registers	
					2	Clear err_pkt_cnt general statistics registers	
					1	Clear mismatched_pkt_cnt general statistics registers	
					0	Clear rx_pkt_cnt general statistics registers	
0x0028	peak_buf_level	R	Y	0x00000000	Peak Packet Buffer Level		
					31:6	Reserved	
					5:0	Peak packet buffer level, in number of packets.	
0x002C	module_ctrl	R/W	Y	0x00000000	Modular control		
					31:1	Reserved	
					0	Module Enable	
						0	Module
						1	Module

Table 2-13: **Decapsulator Module Register Map** ***(Cont'd)***

Address Offset (HEX)	Register Name	Access Type	Cleared with SOFT reset	Default Value (HEX)	Description		
					Bit Range	Value	
0x0030	ch_int_group_ored	R	Y	0x00000000	Channel Group Interrupt Status		
					31:16	Reserved	
					15:0	Interrupt status of each group, bit0~bit15 corresponding to channel ch_int_group0 ~ch_int_group15. The value of each bit means:	
						0	The group does not have an
						1	The group has interrupts.
0x0034	ch_int_group0	R	Y	0x00000000	Channel Group0 Interrupt Status		
					31:0	Interrupt status of each channel in this group. Bit0~bit31 corresponding to channel0~channel31. The valueof each bit means:	
						0	The channel does not have
						1	The channel has an interrupt.
Channel							

Table 2-13: **Decapsulator Module Register Map** ***(Cont'd)***

Address Offset (HEX)	Register Name	Access Type	Cleared with SOFT reset	Default Value (HEX)	Description		
					Bit Range	Value	
0x0080	chan_ctrl	R/W	Y	0x00000000	Channel Control		
					31:4	Reserved	
					3	Drop First Marker Packet (valid only when bit2 =1, default not drop)	
						0	Do not drop first marker packet
						1	Drop the first Marker packet
					2	Marker packet detection enable (default enabled)	
						0	Disable Marker packet detection
						1	Enable marker packet detection.
					1	Lossless mode configuration; valid only when in SMPTE ST 2022-5/6 mode	
						0	Normal mode
						1	Lossless mode
					0	Channel enable	
						0	channel
						1	channel enabled

Table 2-13: **Decapsulator Module Register Map** *(Cont'd)*

Address Offset (HEX)	Register Name	Access Type	Cleared with SOFT reset	Default Value (HEX)	Description		
					Bit Range	Value	
0x0084	chan_timeout	R/W	Y	0x09502F90	Channel Timeout		
					31:0	Timer expire value for packet stream stop detection which is the number of eth_clk cycles ***Note:*** Default value is 1 second at eth_clk=156.25MHz.	
0x0088	ip_hdr_param	R	Y	0x00000000	IP Header Parameter		
					31:16	Reserved	
					15:8	Type of service (TOS)	
					7:0	Time to live (TTL)	
0x0090	match_vlan	R/W	Y	0x00000000	Match VLAN Port		
					31	VLAN filtering	
						0	Filter stream without VLAN
						1	Filter stream with VLAN having tag info
					30:12	Reserved	
					11:0	VLAN Identifier	
0x0094	match_dest_ip0	R/W	Y	0x00000000	Match Destination IP Address 31:0		
					31:0	32-bit destination IP address	
0x00A4	match_src_ip0	R/W	Y	0x00000000	Match Source IP Address 31:0		
					31: 0	32-bit source IP address	
0x00B4	match_udp_src_port	R/W	Y	0x00000000	Match UDP Source Port		
					15:0	16-bit UDP source port address	
0x00B8	match_udp_dest_port	R/W	Y	0x00000000	Match UDP Destination Port		
					15:0	16-bit UDP source port address	
0x00BC	match_ssrc	R/W	Y	0x00000000	Match SSRC		
					31:0	32-bit SSRC used for matching	

Table 2-13: **Decapsulator Module Register Map *(Cont'd)***

Address Offset (HEX)	Register Name	Access Type	Cleared with SOFT reset	Default Value (HEX)	Description		
					Bit Range	Value	
0x00C0	match_sel	R/W	Y	0x00000000	Matching Selection		
					31:7	Reserved	
					6	To match RTP Payload Type programmed in dynamic_payload_type[6:0].	
					5	To match SSRC	
					4	To match UDP dest port	
					3	To match UDP src port	
					2	To match Destination IP	
					1	To match Source IP	
					0	To match VLAN	
0x00C4	video_format	R	Y	0x00000000	Video Format		
					31	Channel SDI Packet stream locked status:	
						0	SDI Packet stream
						1	SDI Packet stream locked
					30:4	Reserved	
					3	tx_bit_rate	
						0	Not divided by
						1	Divided by
					2	3G Level A/B	
						0	3G Level A
						1	3G Level B
					1:0	video_mode	
						00	HD
						01	SD
						10	3G
						11	None

Table 2-13: **Decapsulator Module Register Map** ***(Cont'd)***

Address Offset (HEX)	**Register Name**	**Access Type**	**Cleared with SOFT reset**	**Default Value (HEX)**	**Description**	
					Bit Range	Value
0x00C8	vid_src_fmt	R	Y	0x00000000	Video Source Format. Refer to the SMPTE ST 2022-6 specification [Ref 4]	
					31:28	MAP
					27:20	FRAME
					19:12	FRATE
					11:8	SAMPLE
					7	Reserved
					6:0	For RFC4175/RFC3190 packet, packet stream detection is done by observing the RTP header's PT field. Detected RTP header Payload Type "PT(6:0)" is latched in this register field.

Table 2-13: **Decapsulator Module Register Map *(Cont'd)***

Address Offset (HEX)	Register Name	Access Type	Cleared with SOFT reset	Default Value (HEX)	Description		
					Bit Range	Value	
0x00CC	media_hdr	R	Y	0x00000000	Media Header		
					31:28	Extension field	
					27:20	Frame counter	
					19:16	CF Value	
					15:9	Reserved	
					8	Video source format present (F).	
					7	Reserved	
					6:4	FEC Usage	
					3	Reserved	
					2:1	Specific reference to the source of the time stamp. It is used to create a media header.	
						00	Not locked
						01	Reserved
						10	Locked to UTC time/ frequency
						11	Locked to private time
					0	Video time stamp includes:	
						0	Video time stamp not
						1	Video time stamp included
0x00D0	valid_media_pkt_cnt	R	Y	0x00000000	Channel Valid Media Packet Count		
					31:0	Number of valid media packets received in the channel after being matched.	
0x00D4	valid_fec_pkt_ cnt	R	Y	0x00000000	Channel Valid FEC Packet Count		
					31:0	Number of valid FEC packets received in the channel after being matched.	
0x00D8	reordered_pkt_cnt	R	Y	0x00000000	Reordered Packet Count		
					31:0	Number of reordered packets	

Table 2-13: **Decapsulator Module Register Map** ***(Cont'd)***

Address Offset (HEX)	Register Name	Access Type	Cleared with SOFT reset	Default Value (HEX)	Description	
					Bit Range	Value
0x00DC	drop_pkt_cnt	R	Y	0x00000000	Number Of Pkts Dropped	
					31:0	Number of pkts dropped at s_axis interface.
0x00E0	chan_stat_reset	W	Y	0x00000000	Clear Channel Space Statistics Registers (Self Clear)	
					31:4	Reserved
					3	Clear drop_pkt_cnt statistics register for current channel
					2	Clear reordered_pkt_cnt statistics register for current channel
					1	Clear valid_fec_pkt_cnt statistics register for current channel
					0	Clear valid_media_pkt_cnt statistics register for current channel
0x00E4	pkt_interval	R	Y	0x00000000	Packet Interval From Source	
					31:0	Time stamp difference between two consecutive sequence number packets. The time stamp is from the RTP header.
0x00E8	pkt_interval_network	R	Y	0x00000000	Packet Interval From Network	
					31:0	Time difference between two consecutive sequence number packets. The time is measured by the local counter at the eth_clk domain.

Table 2-13: **Decapsulator Module Register Map** ***(Cont'd)***

Address Offset (HEX)	Register Name	Access Type	Cleared with SOFT reset	Default Value (HEX)	Description		
					Bit Range	Value	
0x00EC	Match_payload_type	R/W	Y	0x00000000	Match Payload Type		
					31:28	Payload type protocol selection used for pkt processing after matching: 0000 - Not dynamic payload type. 0001 - RFC4175 payload type. 0010 - RFC3190 payload type. 0011~1111 - Reserved.	
					29:7	Reserved	
					6:0	RTP payload type used for matching.	
0x00F0	interrupt_status	R	Y	0x00000000	Interrupt Status		
					31:3	Reserved	
					2	Channel timeout interrupt.	
					1	Packet unlock interrupt.	
					0	Packet lock interrupt	
0x00F4	interrupt_mask	R/W	Y	0x00000000	Interrupt Masking (1 Means Masking)		
					31:3	Reserved	
					2	Mask channel timeout interrupt.	
						0	Do not mask.
						1	Mask.
					1	Mask packet unlock interrupt.	
						0	Do not mask.
						1	Mask.
						Mask packet lock interrupt.	
						0	Do not mask.
					0	1	Mask.

Table 2-13: **Decapsulator Module Register Map** ***(Cont'd)***

Address Offset (HEX)	Register Name	Access Type	Cleared with SOFT reset	Default Value (HEX)	Description	
					Bit Range	Value
0x00F8	interrupt _clear	W	Y	0x00000000	interrupt Clear (Self-clear)	
					31:3	Reserved
					2	Clear channel timeout interrupt. 1 - Clear.
					1	Clear packet unlock interrupt. 1 - Clear.
					0	Clear packet lock interrupt. 1 - Clear.

control (0x0000)

Bit 0 is used to reset all registers (not core logic). 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 1 to 1 when updates are completed ensures all channel space registers are updated simultaneously.

status (0x0004)

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

channel_access (0x0008)

This register is used when accessing channel space registers. Always set the channel first.

sys_config (0x000C)

Read only register to show current generic settings for the module.

pkt_lock_window (0x0014)

Bits 15:0 define the packet lock window in number of packets, default is 1,024. Bits 31:16 define the packet unlock window, default is 1,024.

rx_pkt_cnt (0x0018)

Total number of packets received at the AXI4-Stream slave input interface of the module.

mismatched_pkt_cnt (0x001C)

Total number of packets that are mismatched and the mismatched packets will not be sent to the AXI4 Stream master interface.

err_pkt_cnt (0x0020)

The one-bit input port `s_axis_tuser` in the s_axis interface is used to indicate the current incoming packet is GOOD (s_axis_tuser=1) or BAD (s_axis_tuser=0) packet. The `s_axis_tuser` value is sampled when s_axis_tlast is High. For BAD packets, it will not be used for either stream detection or packet processing.

This statistics register shows total number of error packets received on the AXI4-Stream slave input interface. Note that error packets are dropped inside Decap module. Error packets will not be used for packet stream detection.

stat_reset (0x0024)

Each bit of this register resets the corresponding general space statistics register to zero. This stat_reset register is self-clearing; you only need to program once for statistics reset.

peak_buf_level (0x0028)

This statistics register shows the highest buffer level hit during processing. When this value is equal to “Elastic Buffer (Packets)” generic setting, it means the buffer is full and push back occurs to the upstream module. Note that the packet buffer in this module is shared among all channels.

module_ctrl (0x002C)

Bit 0 is used to deactivate (0) or activate (1) the module.

- 0 - Mutes the module and pushes back at the input port, no pkt stream detection
- 1 - Activates the module and lets it start detecting the video stream and processing incoming packets.

ch_int_group_ored (0x0030)

Channels are grouped into different groups. Group size is 32. Bit 0 of this register shows the channel group0 interrupt status. Group0 contains channel0 through channel31.

ch_int_group0 (0x0034)

Each bit of this register shows the interrupt status for each channel in the range of 0-31. For example, bit 0 shows the interrupt status of channel0.

chan_ctrl (0x0080)

The channel control register that controls each channel.

Bit 0 is used to enable the channel.

- 0 - Channel disabled. When disabled, the Decap module stops capturing incoming pkts and resets FSM to wait to be enabled again.
- 1 - channel enabled. Starts search for the first st marker pkt and then starts processing.

Bit 1 defines if the channel works in normal mode or lossless mode.

Bit 2 is used to enable/disable marker packet detection. Default is enabled.

- 0 - Disable Marker Packet Detection
- 1 - Enable Marker Packet Detection

Bit 3 is used to decide to drop first marker packet or not. Default is not dropping.

- 0 - Don’t drop First Marker Packet
- 1 - Drop the first Marker Packet

chan_timeout (0x0084)

This register defines channel timeout time. The default is one second if the Ethernet clock is 156.25 MHz.

ip_hdr_param (0x0088)

The TOS/TTL field extracted from the packet header is latched in this register for the channel.

match_vlan (0x0090)

Channel match register for VLAN matching.

match_dest_ip0 (0x0094)

Channel match register for destination IP address matching.

match_src_ip0 (0x00A4)

Channel match register for source IP address matching.

match_udp_src_port (0x00B4)

Channel match register for UDP source port matching.

match_udp_dest_port (0x00B8)

Channel match register for UDP destination port matching. For column FEC packets, matching logic uses this register value plus 0x2 for dest port matching. And for row FEC packets, matching logic uses this register value plus 0x4 for dest port matching.

match_ssrc (0x00BC)

Channel match register for the RTP header SSRC field matching.

match_sel (0x00C0)

Channel match strategy. You can choose which field of the packet header is used for matching (mismatched packets will be filtered out).

video_format (0x00C4)

Bit 31 is used to indicate the SDI packet locked status of the channel. And lower bits are used to capture media header information. The media payload header is analyzed by the module to extract video format information. This information is needed by downstream modules.

vid_src_fmt (0x00C8)

The media payload header is analyzed by the module to extract media format information. This information is needed by downstream modules.

media_hdr (0x00CC)

The media payload header is analyzed by the module to extract video format information. This information is needed by downstream modules.

valid_media_pkt_cnt (0x00D0)

The channel statistics register shows the number of valid media (non-FEC) packets received in the channel after being matched.

valid_fec_pkt_cnt (0x00D4)

The channel statistics register shows the number of valid FEC packets received in the channel after being matched.

reordered_pkt_cnt (0x00D8)

The channel statistics register shows the number of reordered packet counts by observing the sequence number from the RTP header. Whenever the current packet sequence number is smaller than the maximum sequence number already observed (excluding the sequence number roll over case) in this channel, the reordered packet count increases by one.

drop_pkt_cnt (0x00DC)

The channel statistics register shows the number of packets that get dropped on the channel when the shared packet buffer is full or the channel is disabled or first marker packet is not detected. These dropped packets are matched packets.

chan_stat_reset (0x00E0)

Each bit of this register is used to reset the corresponding channel statistics registers.

pkt_interval (0x00E4)

This statistics register measures the difference of the RTP time stamp value from the packet header. The register is updated only when the current packet sequence number is greater than the previous packet sequence number by one.

pkt_interval_network (0x00E8)

This statistics register measures the local timer value difference between the previous packet and the current packet. The local timer is running at the Ethernet clock domain. The register is updated only when the current packet sequence number is greater than the previous packet sequence number by one.

match_payload_type (0x00EC)

Bit 31 through bit 28 are used to determine the channel is working for SMPTE ST2022 (not dynamic payload). or RFC4175 or RFC3190..

Bit 6 through bit 0 are used for channel matching, which is the user programmed "payload type" value used to match incoming packet's PT field of RTP header.

interrupt_status (0x00F0)

The module has multiple interrupt sources. Each bit of this register shows the interrupt status from each source. After the interrupt source is served and cleared, the bit becomes zero.

Bit 0 is the packet lock interrupt. If a continuous number of packets (lock window) have the same media format in the packet header, the packet lock interrupt is fired on the channel. Note that packets are sent out only when packet lock interrupt happens.

Bit 1 is the packet unlock interrupt. If a continuous number of packets (unlock window) have a different media format in the packet header compared with the locked media format, the packet unlock interrupt is fired on the channel. Note that during the unlock window, packets whose media format are different from the locked media format are dropped inside the module.

Bit 2 is the channel timeout interrupt. If the channel is in a packet locked state and the packet does not come after some time (chan_timeout), the channel timeout interrupt is fired. It is an indication of a channel stop.

interrupt_mask (0x00F4)

You can decide to mask some interrupt sources for the channel. This is done before interrupt occurrences. After an interrupt has occurred, the interrupt can only be cleared; masking does not clear the interrupt.

interrupt_clear (0x00F8)

When an interrupt occurs, software serves the interrupt and clears the interrupt by programming the corresponding bit of this register to 1. After clearing the interrupt, the corresponding bit in the interrupt_status register becomes zero.

Chapter 3

Designing with the Core

This chapter includes guidelines and additional information to make designing with the core easier.

General Design Guidelines

When designing a system using the Media over IP Infrastructure IP core, the first step is to establish the topology of the system. This requires an understanding of the main interface characteristics of each customized AXI4-Stream master and slave that must be able to communicate together. Refer to Protocol Description for a definition of customized AXI4-Stream protocols.

Other than the connectivity, the generic settings for all IP cores must be set properly in order to function correctly and achieve optimum resource utilization. For example, the size of the elastic buffer should be determined based on the overall system data rate. Figure 3-1 shows a typical topology of a system that uses ST 2022-6 Packetizer, ST 2022-6 Depacketizer, Framer and Decapsulator. Note that this is just an example, you can choose a different topology based on a different usage as long as all the interface protocols are complied.

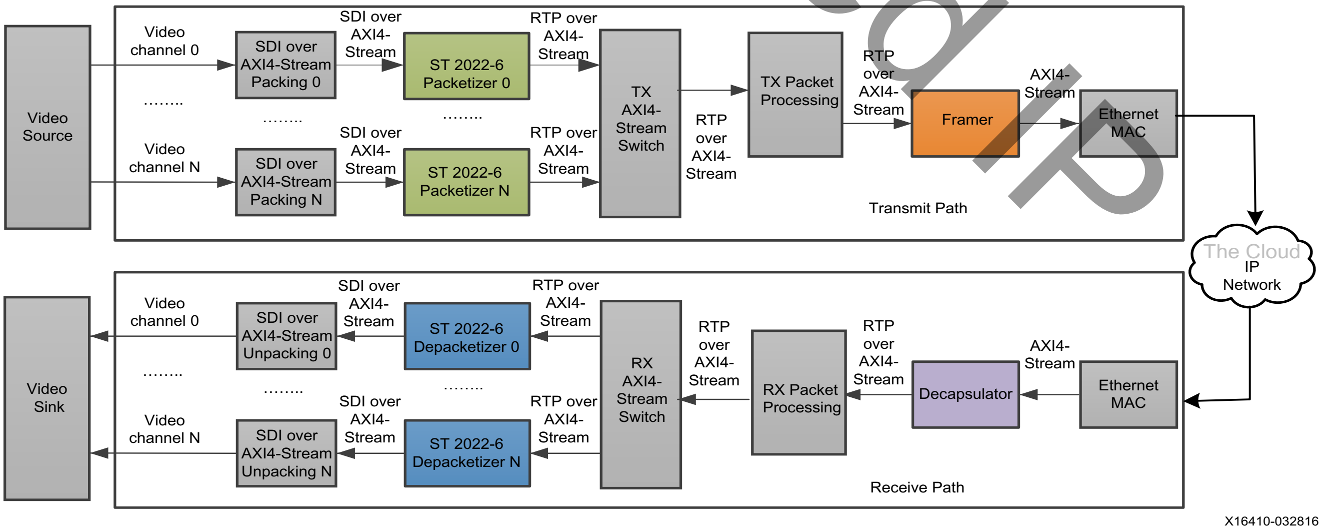


Figure 3-1: **Example Usage of ST 2022-5/6 Media Over IPs**

For further understanding of Media over IP usage, refer to the *Modular SMPTE ST 2022-567 on Kintex-7 Evaluation Board Application Note* (XAPP1272) [Ref 3], which is similar to the topology in Figure 3-1.

Clocking

All the Media over IPs have the AXI4-Lite interface which has its own clock. The AXI4-Lite clock is suggested to be 100 MHz. Besides the AXI4-Lite interface clock, each IP has one or more core clocks. For ST 2022-6 Packetizer and ST 2022-6 Depacketizer, it has one core clock that is used by both the slave and master AXI4-Stream interfaces and main processing logic in the core. For Framer and Decapsulator, they both have two core clocks which belong to the slave and master AXI4-Stream interfaces separately.

To support the 10G network, Xilinx suggests a 200 MHz clock frequency be applied on `core_clk` for the ST 2022-6 Packetizer/Depacketizer, `s_axis_clk` for the Framer and `m_axis_clk` for the Decapsulator. Xilinx suggests a 156.25 MHz clock frequency be applied on `m_axis_clk` for the Framer and `s_axis_clk` for the Decapsulator.

All clocks must be stable before reset release.

Resets

The reset for the AXI4-Lite interfaces are all active-Low. All the core resets that belong to corresponding core clock domain are all active-High and synchronous to the corresponding core clock.

Protocol Description

SDI over AXI4-Stream Protocol

The SDI over AXI4-Stream protocol is a customized AXI4-Stream protocol that is used to send video frames from one module to another module with the `TLAST` signal as the frame end indicator. Refer to Port Descriptions in Chapter 2 for all the ports related to this protocol.

Currently, ST 2022-6 Packetizer (slave interface) and ST 2022-6 Depacketizer (master interface) use this protocol to receive (slave) or transmit (master) video frames.

Figure 3-2 shows the timing for the protocol.

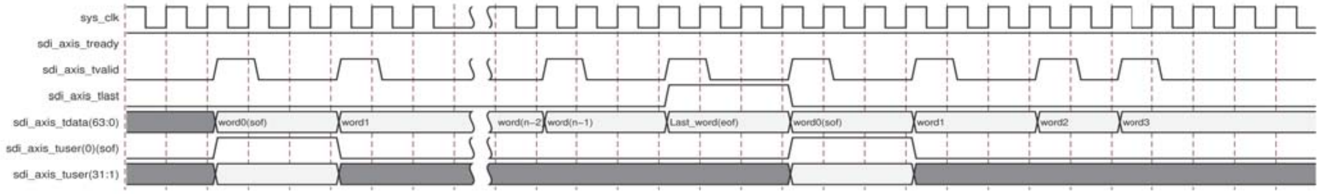


Figure 3-2: **SDI over AXI4-Stream Timing Diagram**

RTP over AX4-Stream Protocol

The RTP over AXI4-Stream protocol is a customized AXI4-Stream protocol that is used to send RTP packets along with packet information (carried in the pre-defined `axis_tuser(31:0)` bus) from one module to another module. The `axis_tuser(31:0)` is defined to carry packet information and the `axis_tuser(0)` is the indicator of the beginning of a packet and `axis_tlast` indicates the end of a packet. Refer to Port Descriptions in Chapter 2 for all the ports related to this protocol.

As for the timing diagram, refer to the Stream Payload Protocol Waveform diagram in the *Video over IP FEC Transmitter LogiCORE IP Product Guide* (PG206) [Ref 13] because it is the same protocol.

Currently, ST 2022-6 Packetizer (master interface), ST 2022-6 Depacketizer (slave interface), Framer (slave interface) and Decapsulator (master interface) use this protocol to receive (slave) or transmit (master) RTP packets.

Note: Refer to the Example Design chapter for core configuration guidelines.

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 7]
- *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 8]
- *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 9]
- *Vivado Design Suite User Guide: Logic Simulation* (UG900) [Ref 10]

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 3] 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.

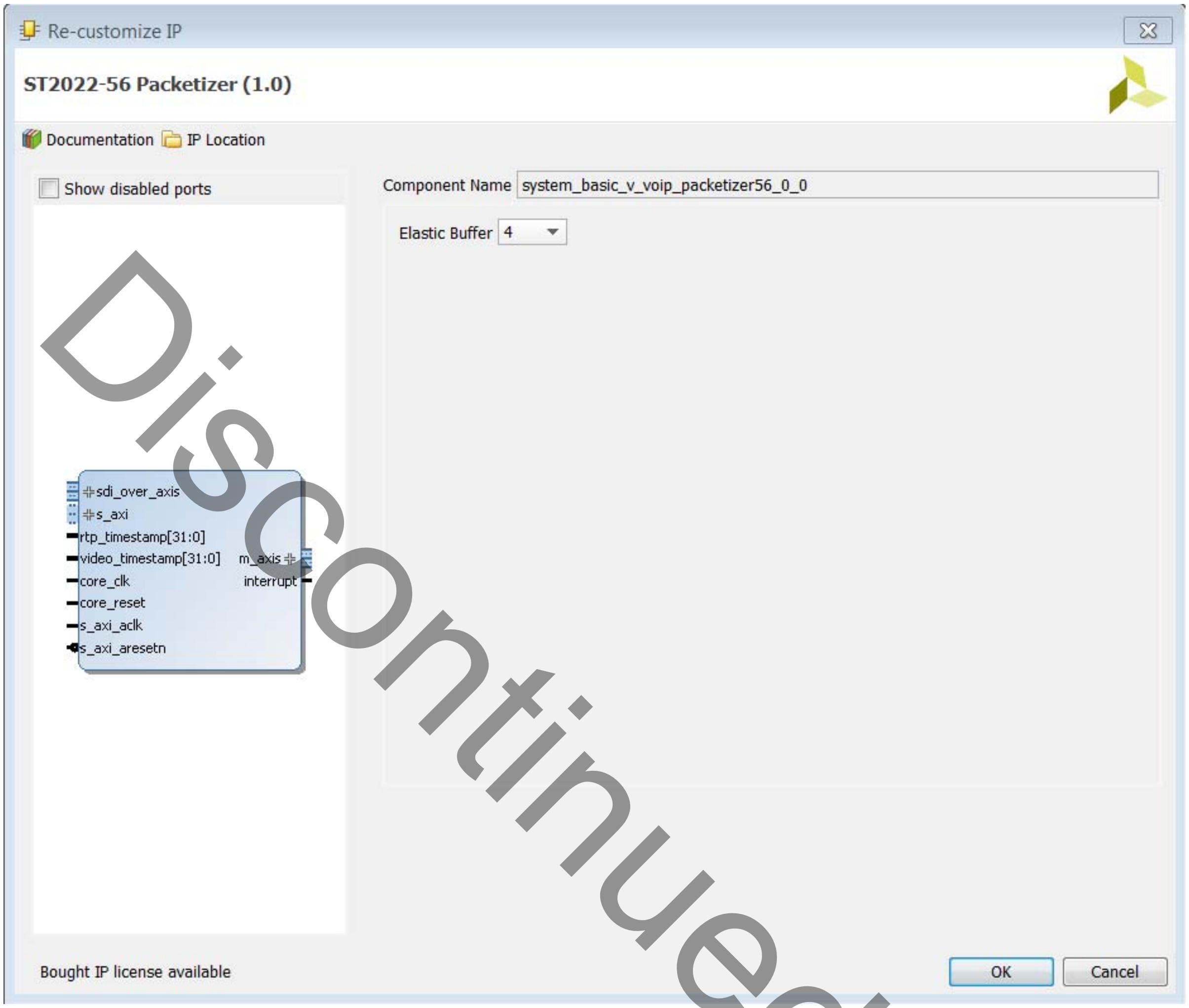
You can customize the IP for use in your design by specifying values for the various parameters associated with the IP core using the following steps:

1. Select the IP from the Vivado 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 8] and the *Vivado Design Suite User Guide: Getting Started* (UG910) [Ref 9].

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

ST 2022-6 Packetizer



Re-customize IP

ST2022-56 Packetizer (1.0)

Component Name: system_basic_v_voip_packetizer56_0_0

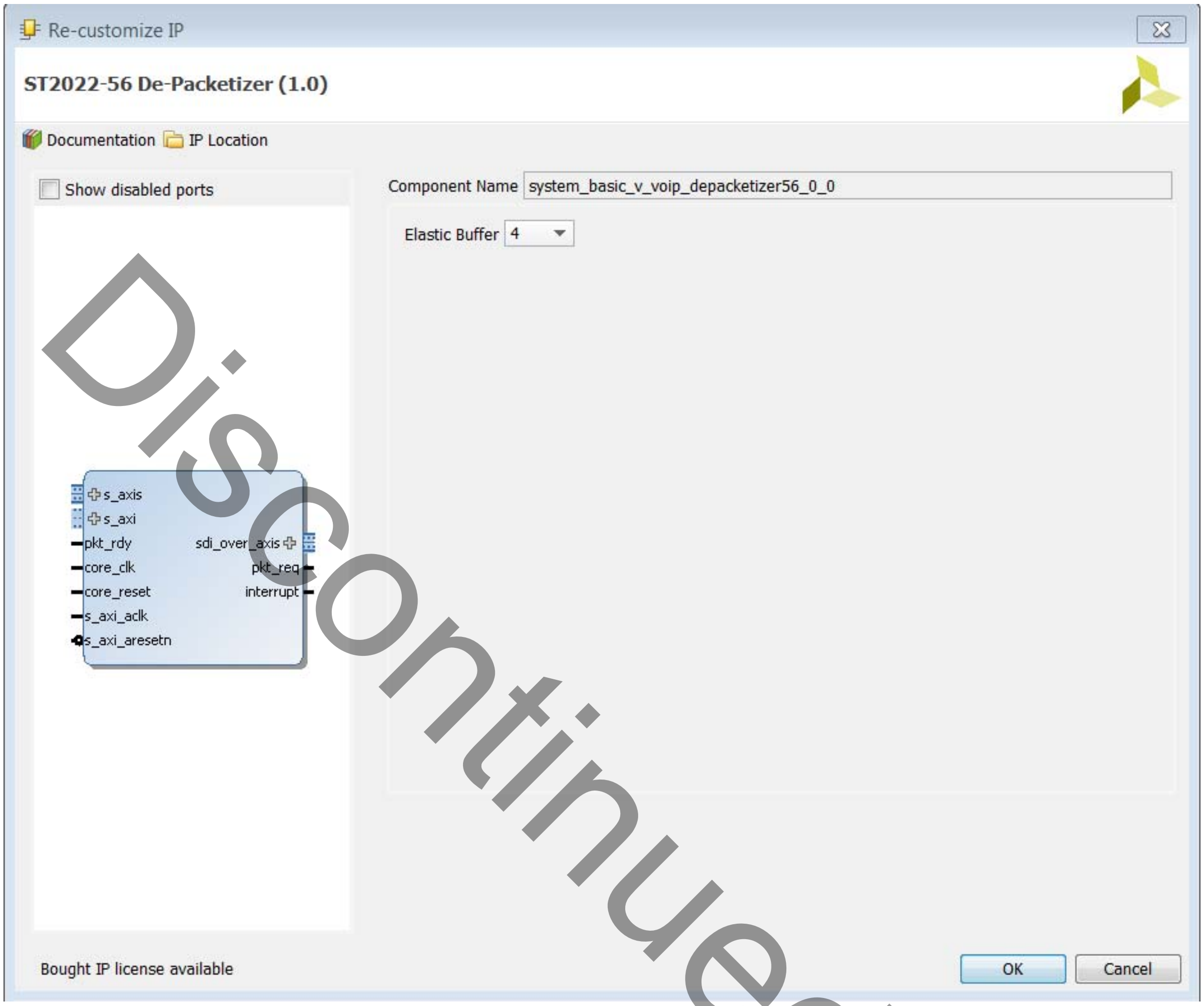
Elastic Buffer: 4

Bought IP license available

Figure 4-1: **ST 2022-6 Packetizer Customization Dialog Box**

- **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_packetizer56_v1_0 cannot be used as a component name.
- **Elastic Buffer**: Determines the number of packets that can be buffered inside a module. It is useful when the downstream modules have short pushback per packet processing. In different system, the value settings can be different. In order to reduce block memory usage, try to set the value as small as possible as long as the value is enough and it will not cause system overflow.

ST 2022-6 Depacketizer



Re-customize IP

ST2022-56 De-Packetizer (1.0)

Component Name: system_basic_v_voip_depacketizer56_0_0

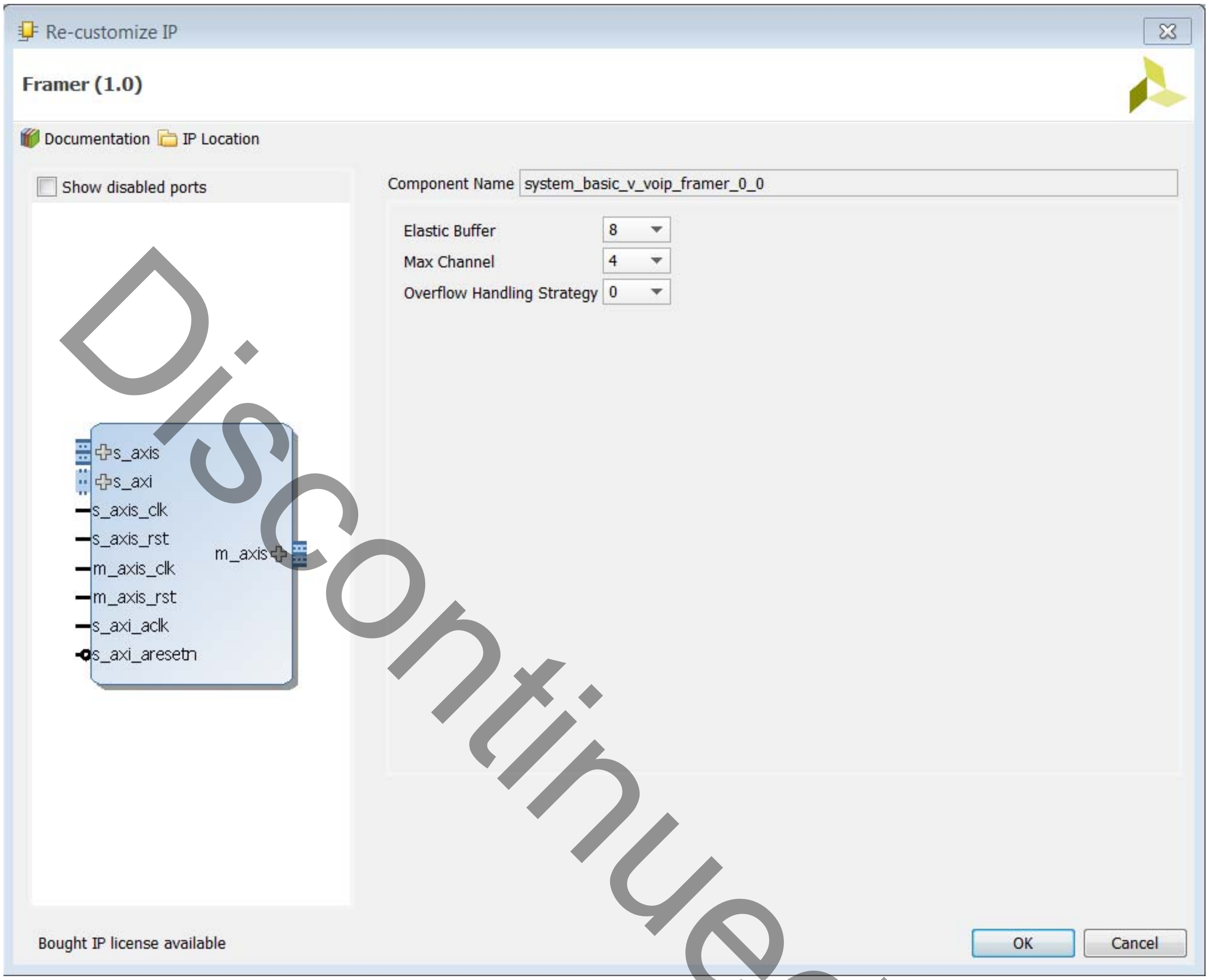
Elastic Buffer: 4

Bought IP license available

Figure 4-2: **ST 2022-6 Depacketizer Customization Dialog Box**

- **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_depacketizer56_v1_0 cannot be used as a component name.
- **Elastic Buffer**: Determines the number of packets that can be buffered inside a module. It is useful when downstream module have short pushback per packet processing. In different systems, the value settings can be different. In order to reduce block memory usage, you can try to set the value as small as possible as long as the value is enough and it will not cause system overflow.

Framer


Re-customize IP

Framer (1.0)

Component Name system_basic_v_voip_framer_0_0

Elastic Buffer 8

Max Channel 4

Overflow Handling Strategy 0

Bought IP license available

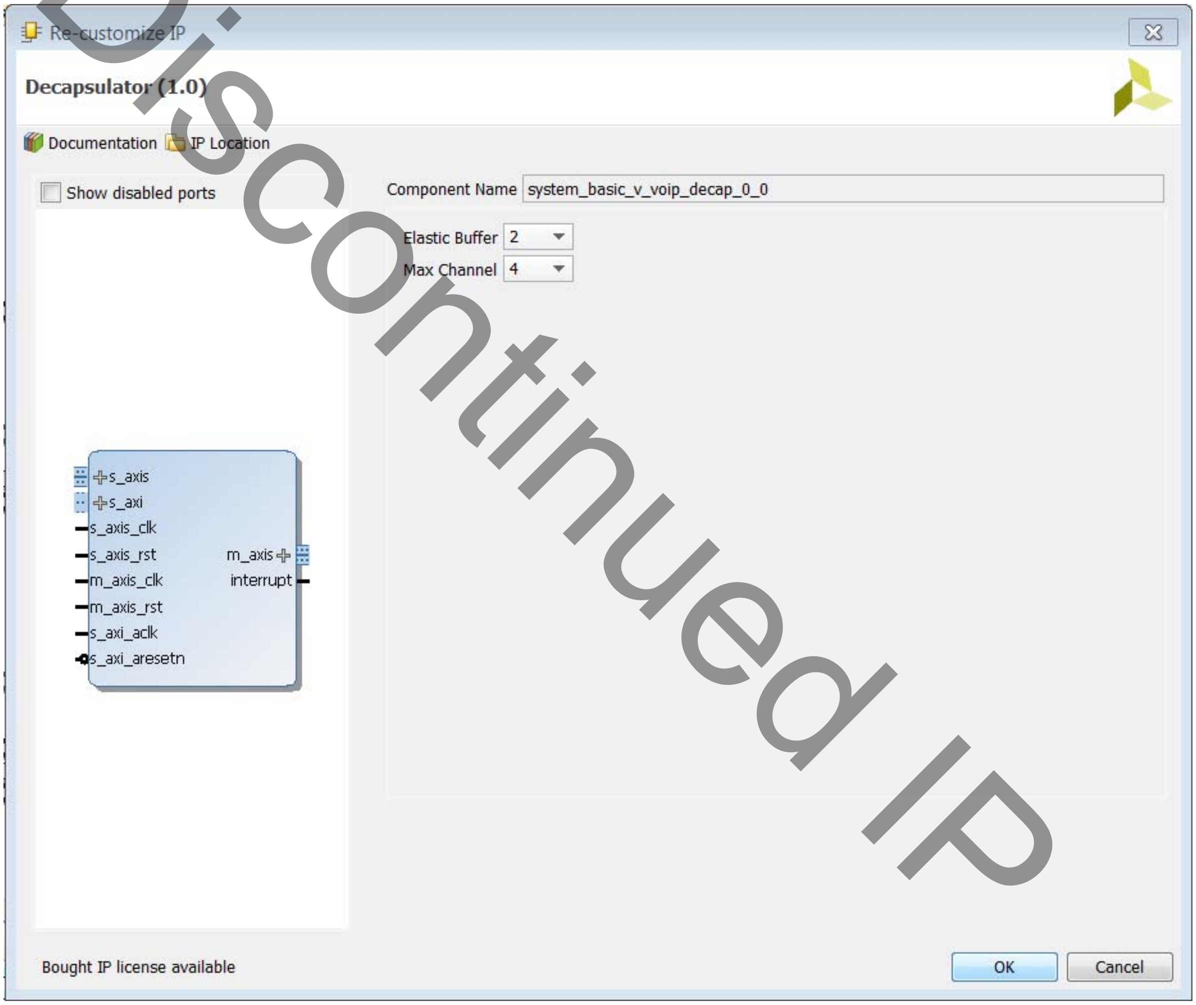
Figure 4-3: **Framer Customization Dialog Box**

- **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_framer_v1_0 cannot be used as a component name.
- **Elastic Buffer**: Determines the number of packets that can be buffered inside a module. It is useful when downstream modules have short pushback per packet processing. In different systems, the value settings can be different. In order to reduce block memory usage, you can try to set the value as small as possible as long as the value is enough and it will not cause system overflow.
- **Max Channel**: Determines the maximum number of channels that the module can support since Framer is multi-channel module. In order to reduce hardware resource utilization, you should try to set this generic as small as possible as long as it meets the system function requirements.

- **Overflow Handling Strategy**: When the internal elastic buffer becomes full due to the downstream module push back, you can decide to further push back to the upstream module.

  If it is set to 0 (default), the incoming packets will be pushed back to the upstream module if the internal buffer becomes full. If it is set to 1, the incoming packets will be dropped at the module input and no push back to the upstream module when buffer becomes full.

Decapsulator



Component Name: system_basic_v_voip_decap_0_0

Elastic Buffer: 2

Max Channel: 4

Figure 4-4: **Decapsulator Customization Window**

- **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_decap_v1_0 cannot be used as a component name.
- **Elastic Buffer**: Determines the number of packets that can be buffered inside a module. It is useful when a downstream module has a short pushback per packet processing. In different systems, the value settings can be different. In order to reduce block memory usage, you can try to set the value as small as possible as long as the value is enough and it will not cause system overflow.
- **Max Channel**: Determines the maximum number of channels that the module can support since Decapsulator is multi-channel module. In order to reduce hardware resource utilization, you should try to set this generic as small as possible as long as it meets the system function requirements.

User Parameters

Table 4-1 through Table 4-4 show the relationship between the fields in the Vivado IDE and the User Parameters (which can be viewed in the Tcl Console).

Table 4-1: **ST 2022-6 Packetizer GUI Parameters to User Parameter Relationship**

GUI Parameter/Value	User Parameter/Value	Default Value
Elastic Buffer	C_BUF_PAGE	2

Table 4-2: **ST 2022-6 Depacketizer GUI Parameters to User Parameter Relationship**

GUI Parameter/Value	User Parameter/Value	Default Value
Elastic Buffer	C_BUF_PAGE	4

Table 4-3: **Framer GUI Parameters to User Parameter Relationship**

GUI Parameter/Value	User Parameter/Value	Default Value
Elastic Buffer	C_BUF_PAGE	2
Max Channel	C_CHANNELS	4
Overflow Handling Strategy	C_HITLESS	0

Table 4-4: **Decapsulator GUI Parameters to User Parameter Relationship**

GUI Parameter/Value	User Parameter/Value	Default Value
Elastic Buffer	C_BUF_PAGE	2
Max Channel	C_CHANNELS	4

Output Generation

For details, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 3].

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 datapathonly 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

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

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.

Simulation

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 10].

IMPORTANT: *For cores targeting 7 series or Zynq-7000 devices, UNIFAST libraries are not supported. Xilinx IP is tested and qualified with UNISIM libraries only.*

Synthesis and Implementation

For details about synthesis and implementation, see the *Vivado Design Suite User Guide: Designing with IP* (UG896) [Ref 3].

Chapter 5

Example Design

See *Modular SMPTE ST 2022-567 on Kintex-7 Evaluation Board Application Note* (XAPP1272) [Ref 3].

Test Bench

When the core is generated using the Vivado® IP catalog, a demonstration test bench is optionally created. This is a simple SV test bench that exercises the core. The demonstration test bench source code is created from mixed Verilog/Vhdl and system Verilog files under the `demo_tb/` directory in the Vivado Design Suite output directory. The test bench top file is named as `tb_<component_name>.sv`.

Using the Demonstration Test Bench

The demonstration test bench instantiates the generated Modular Media over IP core (Framer, Decap, ST 2022-6 Packetizer, ST 2022-6 Depacketizer). Either the behavioral model or the netlist can be simulated within the demonstration test bench.

Run the demonstration test bench using the following steps:

1. Generate the core using the IP catalog and set it as the top level.
2. Go to **Simulation Setting** and append the prefix `tb_` at the component name stated in the **Simulation top module** name field. Then click **OK**.
3. Click **Run Simulation** to start the behavioral simulation.

The test bench generates stimulus (RTP packets or SDI data) and simulation stops when the checker detects output from the receiver side. Any mismatch data occurrence will be detected and display on the Vivado® IDE console.

Demonstration Test Bench Architecture for both Framer and Decap

The demonstration test bench in Figure 6-1 is a simple System Verilog module that configures and tests the VoIP Framer or Decapsulator cores. The test bench components consist of the RTP packet generator, APIs and drivers for configuring the core, simulation model, and checker for integrity check of packets exiting the core. The test bench is supplied as part of the example simulation output product group.

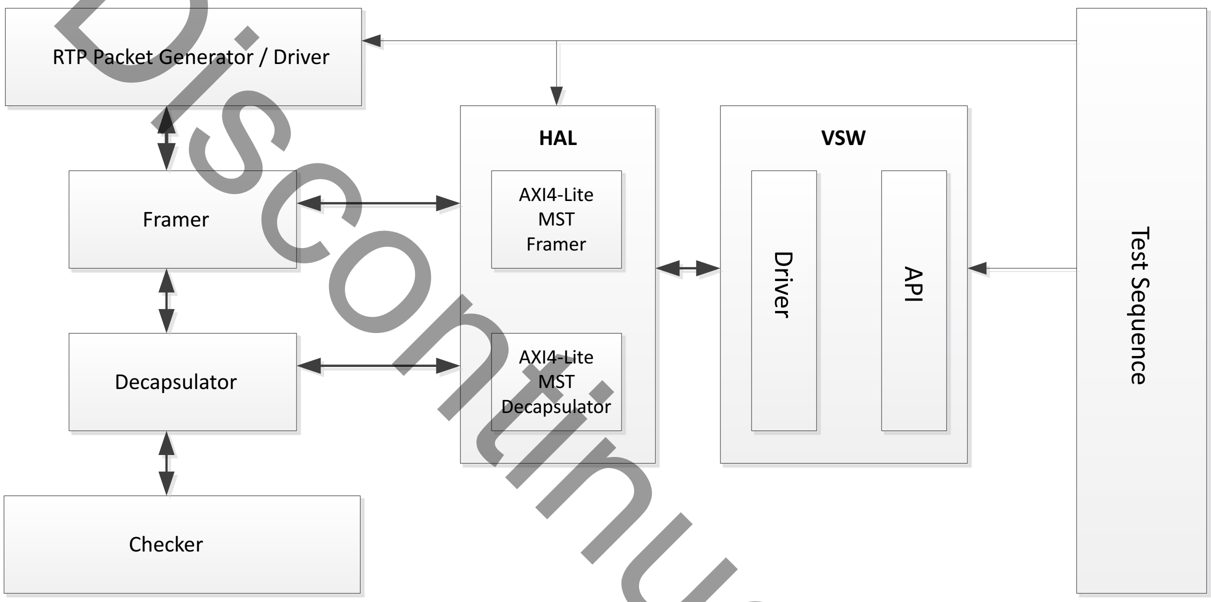


Figure 6-1: **Framer and Decapsulator Test Bench**

- **RTP Packet Generator/Driver**: This module generates Real-time Transport Protocol (RTP) Encapsulated SMPTE ST 2022-2/6 Media compliant Real Time Protocol packets over Transport Stream and drives it to the VoIP Framer and VoIP Decapsulator cores across all the enabled channels.
- **Checker**: Packet checker module receives the packet from the VoIP Decapsulator core and does packet payload 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 is a Verilog task file where all the core configuration is consolidated into tasks. This layer consists of a Driver and API. They control the core configuration and are driven to the core by HAL. This layer is controlled using a test case.
- **DDR model**: This is Dummy DDR model used to store the IP and FEC packets from the core.

- **Simulation Model Decap/Framer**: For the VoIP Decap demonstration test bench, an encrypted version of the Framer core will be used as a simulation model. Likewise for the VoIP Framer demonstration test bench, an encrypted version of the Decap core will be used as a simulation model.

Demonstration Test Bench Architecture for both ST 2022-6 Packetizer and ST 2022-6 Depacketizer

The demonstration test bench in Figure 6-2 is a simple System Verilog module that configures and tests the ST 2022-6 Packetizer and ST 2022-6 Depacketizer cores. The test bench components consist of the SDI stream generator, APIs and drivers for configuring the core, simulation models, and an SDI stream checker for integrity. The test bench is supplied as part of the example simulation output product group.

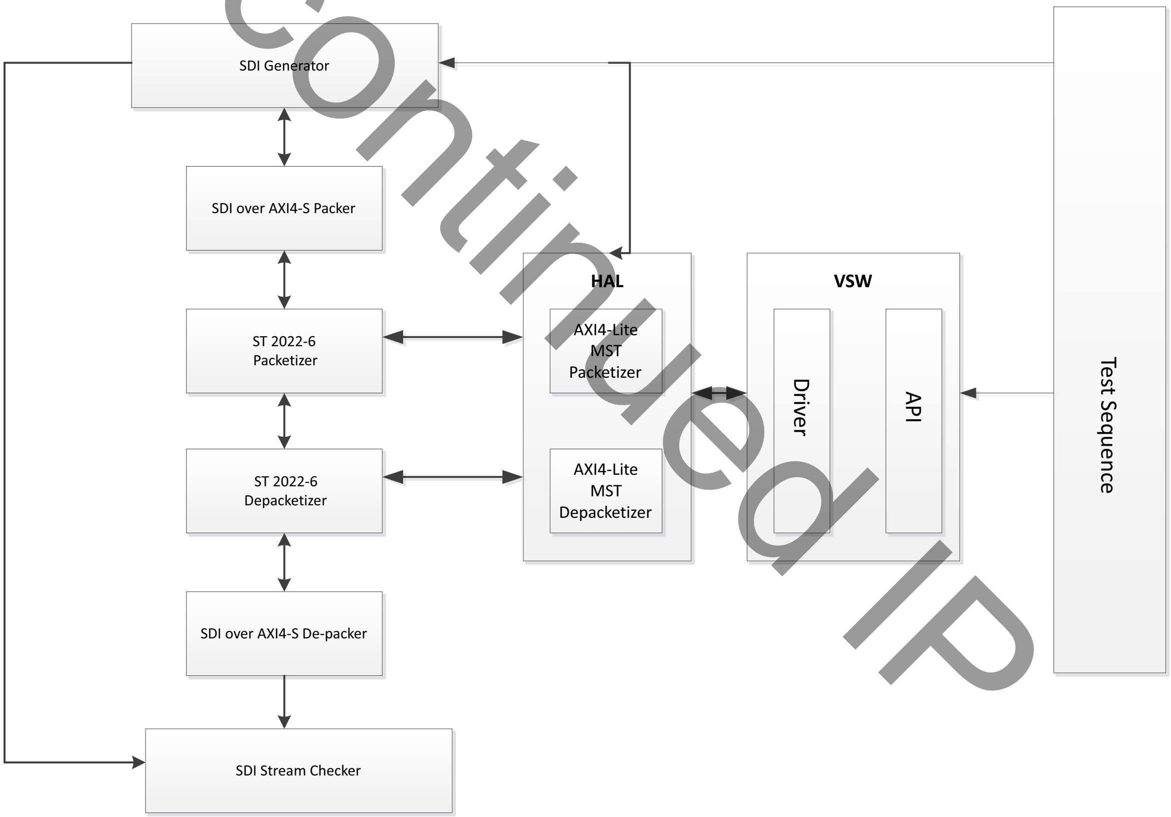


Figure 6-2: **ST 2022-6 Packetizer and SMPTE ST 2022-6 Depacketizer Test Bench**

- **SDI Stream Generator**: Provides the video inputs to the SDI2AXIS (v_voip_sdi2axis) simulation module based on the SDI format selection (3G/HD/SD) configured through the SDI Video Generator Configuration Module setup. This module generates data for the 3G-A mode by default.
- **SDI Stream Checker**: Compares the SDI data output received from the simulation model AXIS2SDI (v_voip_axis2sdi) with the original data from the SDI stream Generator. It asserts an error when there is any data mismatched.
- **HAL**: Hardware Access Layer is the register configuration layer. This layer has register read and write process.
- **VSW**: Virtual Software layer. This is a Verilog task file where all the core configuration is consolidated into tasks. This layer consists of a Driver and API. They control the core configuration and are driven to the core by HAL. This layer is controlled using a test case.
- **SDI over AXI4-Stream Packer**: An encrypted module that converts SDI stream into AXI4-Stream compliance signals that are supported by the ST 2022-6 Packetizer core.
- **SDI over AXI4-Stream De-packer**: An encrypted module that converts AXI4-Stream compliance signals from the ST 2022-6 Depacketizer core into SDI stream output.
- **Simulation Model ST 2022-6 Packetizer/Depacketizer**: For VoIP ST 2022-6 Packetizer demonstration test bench, an encrypted version of the ST 2022-6 Depacketizer core will be used as a simulation model. Likewise for VoIP ST 2022-6 Depacketizer demonstration test bench, an encrypted version of the ST 2022-6 packetizer core will be used as a simulation model.

Verification, Compliance, and Interoperability

The Modular Media Over IP Infrastructure that includes Decapsulator, Framer, ST2022-6 Packetizer and ST2022-6 Depacketizer has been validated using the Xilinx® Kintex®-7 FPGA KC705 Evaluation Kit. See *Modular SMPTE2022-567 on Kintex-7 Evaluation Board* (XAPP1272)[Ref 3] for more information.

Appendix B

Migrating and Upgrading

This appendix is not applicable for the first release of the core.

Appendix C

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 Modular Media over IP Infrastructure 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 Modular Media over IP Infrastructure 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.

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 Records for the Modular Media over IP Infrastructure

AR 66764: Decapsulator

AR 66765: Framer

AR 66766: ST 2022-56 De-Packetizer

AR 66767: ST2022-56 Packetizer

Technical Support

Xilinx provides technical support at 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 (Reference Boards)

The 7 series KC705 FPGA evaluation board supports ST 2022-6 Packetizer, ST 2022-6 Depacketizer, Framer and Decap. This board can be used to prototype designs and establish that the core can communicate with the system.

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. See Figure C-1 for a read timing diagram. 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 debug feature capture that the waveform is correct for accessing the AXI4-Lite interface.

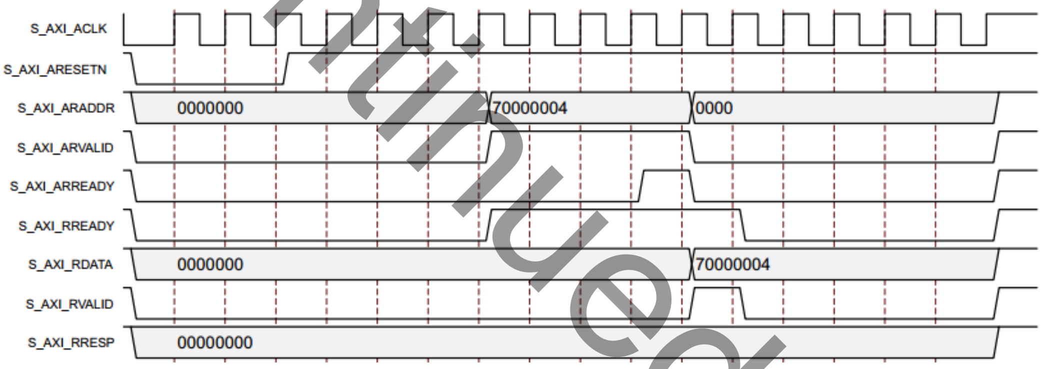


Figure C-1: **Timing Diagram**

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. See Figure 3-2.
- Check core configuration.

Core Debug

ST 2022-6 Packetizer

1. Ensure that the peak_buf_level (0x8) is not greater than the configured elastic buffer during the core generation. If this happens, the core is in pushback state.
2. Ensure the statistic register, output_pkt_cnt (0x28) and frame_cnt (0x2C), is incrementing during the active incoming stream. If `frame_cnt` is not incrementing, this means there are no incoming SDI over AXI4-Stream packets, and if `output_pkt_cnt` is not incrementing this means the core is not transmitting packets.

ST 2022-6 Depacketizer

Ensure the statistic register, input_pkt_cnt (0x28) and frame_cnt (0x2C), is incrementing during the active incoming stream. If `frame_cnt` is not incrementing, this means there are no transmissions of SDI over AXI4-Stream packets from the core, and if `input_pkt_cnt` is not incrementing this means the core is not receiving packets.

Framer

1. Ensure that the peak_buf_level (0x1C) is not greater than the configured elastic buffer during the core generation. If this happens, the core is in pushback state or incoming packet drop state based on the Overflow Handling setting.
2. Ensure the bit 0 (Transmit Enable) of chan_ctrl (0x80) is set to 1 to enable the transmission.
3. rx_pkt_cnt (0x20) is incrementing to indicate the core is receiving packets.
4. tx_pkt_cnt (0xC0) is incrementing to indicate the core is transmitting the packet.

Decapsulator

1. Ensure that the peak_buf_level (0x28) is not greater than the configured elastic buffer during the core generation. If this happens, the core is in pushback state.
2. Ensure the pkt_lock_window (0x14); for both lock and unlock window is configured properly (non-zero).
3. Ensure the Channel Packet Header Filter is set properly:
   a. match_sel (0xC0) (non-zero value for active channel)
   b. match_vlan (0x90) (Set if it is active filter in match_sel (0xC0))
   c. match_dest_ip0 (0x94) (Set if it is active filter in match_sel (0xC0))
   d. match_src_ip0 (0xA4) (Set if it is active filter in match_sel (0xC0))

e. match_udp_src_port (0xB4) (Set if it is active filter in match_sel (0xC0))

f. match_udp_dst_port (0xB8) (Set if it is active filter in match_sel (0xC0))

g. match_ssrc (0xBC) (Set if it is active filter in match_sel (0xC0))

4. rx_pkt_cnt (0x18) is incrementing to indicate the core is receiving packets.
5. mismatched_pkt_cnt (0x1C) incrementing indicates the incoming packet does not match with the Channel Packet Header Filter.
6. err_pkt_cnt (0x20) incrementing indicates the incoming packet has TUSER = 1.

Additional Resources 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. NUMERICAL INDEX OF SMPTE STANDARDS
2. IETF RFC 3550
3. *Modular SMPTE2022-567 on Kintex-7 Evaluation Board Application Note* (XAPP1272)
4. ST 2022-6:2012 - Transport of High Bit Rate Media Signals over IP Networks (HBRMT)

   Note: Only registered users can access.
5. ST 2022-5:2012 - Forward Error Correction for High Bit Rate Media Transport Over IP Networks
6. ST 2022-7:2013 - Seamless Protection Switching of SMPTE ST 2022 IP Datagrams
7. *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
8. *Vivado Design Suite User Guide: Designing with IP* (UG896)
9. *Vivado Design Suite User Guide: Getting Started* (UG910)
10. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
11. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)
12. *Vivado Design Suite User Guide: Implementation* (UG904)
13. *Video over IP FEC Transmitter LogiCORE IP Product Guide* (PG206)
14. *Vivado Design Suite: AXI Reference Guide* (UG1037)
15. RFC762 Assigned Numbers (RFC762)

Revision History

The following table shows the revision history for this document.

Date	Version	Revision
06/07/2017	1.0	Updated Table 2-5: SDI over AXI4-Stream Interface Protocol (Master or Slave Interfaces).
10/05/2016	1.0	Added support for RFC3190 and RFC4175. Updated Xilinx **automotive applications disclaimer**.
04/06/2016	1.0	Initial Xilinx release.

Please Read: Important Legal Notices

The information disclosed to you hereunder (the "Materials") is provided solely for the selection and use of Xilinx products. To the maximum extent permitted by applicable law: (1) Materials are made available "AS IS" and with all faults, Xilinx 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) Xilinx 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 connection with, the Materials (including your use of the 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 or loss was reasonably foreseeable or Xilinx had been advised of the possibility of the same. Xilinx assumes no obligation to correct any errors 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 Xilinx's limited warranty, please refer to Xilinx's Terms of Sale which can be viewed at http://www.xilinx.com/legal.htm#tos; IP cores may be subject to warranty and support terms contained in a license issued to you by Xilinx. Xilinx 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 use of Xilinx products in such critical applications, please refer to Xilinx's Terms of Sale which can be viewed at http://www.xilinx.com/legal.htm#tos.

AUTOMOTIVE APPLICATIONS DISCLAIMER

AUTOMOTIVE PRODUCTS (IDENTIFIED AS "XA" IN THE PART NUMBER) ARE NOT WARRANTED FOR USE IN THE DEPLOYMENT OF AIRBAGS OR FOR USE IN APPLICATIONS THAT AFFECT CONTROL OF A VEHICLE ("SAFETY APPLICATION") UNLESS THERE IS A SAFETY CONCEPT OR REDUNDANCY FEATURE CONSISTENT WITH THE ISO 26262 AUTOMOTIVE SAFETY STANDARD ("SAFETY DESIGN"). CUSTOMER SHALL, PRIOR TO USING OR DISTRIBUTING ANY SYSTEMS THAT INCORPORATE PRODUCTS, THOROUGHLY TEST SUCH SYSTEMS FOR SAFETY PURPOSES. USE OF PRODUCTS IN A SAFETY APPLICATION WITHOUT A SAFETY DESIGN IS FULLY AT THE RISK OF CUSTOMER, SUBJECT ONLY TO APPLICABLE LAWS AND REGULATIONS GOVERNING LIMITATIONS ON PRODUCT LIABILITY.