I2S Transmitter and I2S Receiver v1.0

LogiCORE IP Product Guide

Vivado Design Suite

PG308 (v1.0) September 8, 2020

Table of Contents

IP Facts

The Xilinx® LogiCORE™ IP I2S Transmitter and LogiCORE™ Receiver cores are soft Xilinx IP cores for use with the Xilinx Vivado® Design Suite, which makes it easy to implement the inter-IC-sound (I2S) interface used to connect audio devices for transmitting and receiving PCM audio.

Features

- AXI4-Stream compliant
- Supports up to four I2S channels (up to eight audio channels)
- 16/24-bit datawidth support
- Supports master I2S mode
- Configurable FIFO depth
- Supports the AES channel status extraction/insertion
- Supports left and right justified I2S
- Optional 32-bit LRCLK support

IP Facts

LogiCORE IP Facts Table	
Core Specifics	
Supported Device Family[1]	UltraScale+™, UltraScale™, Zynq®-7000 SoC, 7 series, Zynq® UltraScale+™ MPSoC.
Supported User Interfaces	AXI4-Lite, AXI4-Stream, AXI4
Resources	Performance and Resource Use web page for transmitter and Performance and Resource Use web page for receiver.
Provided with Core	
Design Files	SystemVerilog
Example Design	SystemVerilog
Test Bench	SystemVerilog
Constraints File	Delivered at the time of IP generation
Simulation Model	Source HDL
Supported S/W Driver[2]	Standalone
Tested Design Flows[3]	
Design Entry	Vivado® Design Suite, Vivado IP Integrator
Simulation	For supported simulators, see the Xilinx Design Tools: Release Notes Guide.
Synthesis	Vivado Synthesis
Support	
Release Notes and Known Issues	Master Answer Records: 70288 (RX), 70699 (TX)
All Vivado IP Change Logs	Master Vivado IP Change Logs: 72775
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 Vitis™ software platform directory (`<install_directory>/vitis/<release>/data/embeddedsw/doc/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

The I2S Transmitter and I2S Receiver cores provide an easy way to interface the I2S based audio DAC/ADC. These IPs require minimal register programming and also support any audio sampling rates. These IPs can be used alongside HDMI, DisplayPort, and SDI for a complete audio video solution.

Navigating Content by Design Process

Xilinx® documentation is organized around a set of standard design processes to help you find relevant content for your current development task. This document covers the following design processes:

- **Hardware, IP, and Platform Development:** Creating the PL IP blocks for the hardware platform, creating PL kernels, subsystem functional simulation, and evaluating the Vivado® timing, resource use, and power closure. Also involves developing the hardware platform for system integration. Topics in this document that apply to this design process include:
  - Port Descriptions
  - I2S Transmitter Register Space
  - I2S Receiver Register Space
  - Clocking
  - Resets
  - Chapter 6: Example Design

Applications

Typical applications for I2S interfaces could be audio and video conferencing equipment, consumer multi-media devices, professional audio sources, and sinks. The I2S Transmitter and I2S Receiver IPs can be used to develop audio solution using I2S ADC/DACs. These IPs are typically used with video connectivity IPs such as HDMI and Display Port to play or insert the audio.

Unsupported Features

The following features of the standard are not supported in the core:

- Data width of 20-bit
- Slave mode
- Decode/encode user information bits

Licensing and Ordering

This Xilinx® LogiCORE™ IP module is provided at no additional cost with the Xilinx Vivado® Design Suite under the terms of the Xilinx End User License.

For more information about this core, visit the I2S Transmitter and I2S Receiver product web page

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

Product Specification

The I2S Transmitter and I2S Receiver IPs can be used to develop audio solution using I2S ADC/DACs. These IPs support any sampling rate and are very easy to configure with minimal register programming.

Figure 1: **TX Audio Sampling**

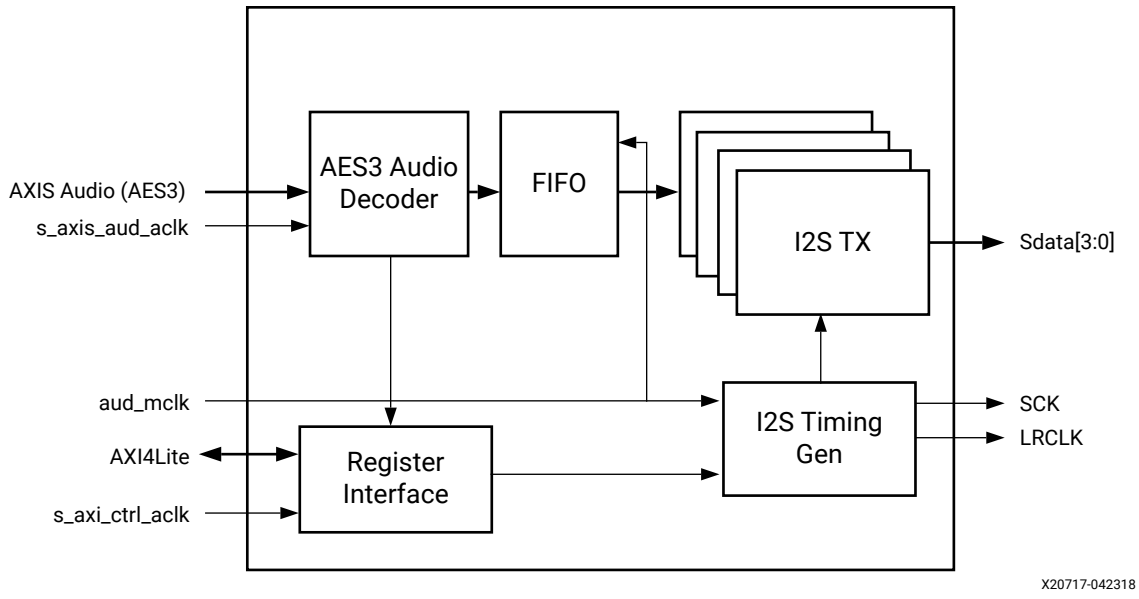
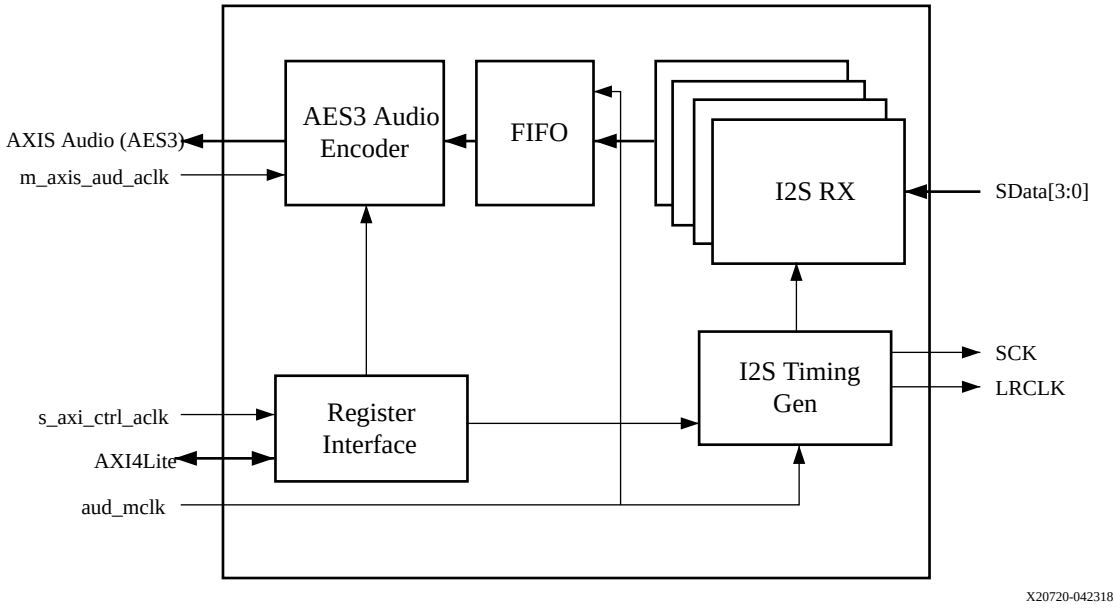


Figure 2: **RX Audio Sampling**



Performance and Resource Use

For full details about performance and resource use, visit the Performance and Resource Use web page for transmitter and Performance and Resource Use web page for receiver.

Port Descriptions

Port Names

Table 1: **Port Names**

Port Name	I/O	Clock	Description
Transmitter Ports			
s_axi_ctrl_aclk	I	Clock	Input clock for AXI4-Lite Interface
s_axi_ctrl_aresetn	I	Reset	Active-Low reset for AXI4-Lite Interface
s_axi_ctrl_*		s_axi_ctrl	AXI4-Lite Interface
aud_mclk	I	Clock	Input audio clock. Typically a multiple of Fs
aud_mrst	I	Reset	Active-High reset for audio interface
s_axis_aud_aclk	I	Clock	AXIS Audio streaming clock

Table 1: **Port Names** *(cont'd)*

Port Name	I/O	Clock	Description
s_axis_aud_resetn	I	Reset	Active-Low AXIS audio reset
s_axis_aud_*		Audio AXIS Interface	AXIS audio interface[1]
Irq	O	Interrupt	Active-High interrupt
lrclk_out	O	LRClk	Output LR Clock. Available when core is configured as Master
sclk_out	O	SCLK	Output SCK Clock. Available when core is configured as Master
lrclk_in	I	LRClk	Input LR Clock. Available when core is configured as Slave
Sclk_in	I	SCLK	Input SCK Clock. Available when core is configured as Slave
sdata_0_out	O	SDATA0	I2S Serial Data out
sdata_1_out	O	SDATA1	I2S Serial Data out. Available when number of audio channels is > 2
sdata_2_out	O	SDATA2	I2S Serial Data out. Available when number of audio channels is > 4
sdata_3_out	O	SDATA3	I2S Serial Data out. Available when number of audio channels is > 6
Receiver Ports			
s_axi_ctrl_aclk	I	Clock	Input clock for AXI4-Lite Interface
s_axi_ctrl_aresetn	I	Reset	Active-Low reset for AXI4-Lite Interface
s_axi_ctrl_*		s_axi_ctrl	AXI4-Lite Interface
aud_mclk	I	Clock	Input audio clock. Typically a multiple of Fs
aud_mrst	I	Reset	Active-High reset for audio interface
m_axis_aud_aclk	I	Clock	AXIS Audio streaming clock
m_axis_aud_resetn	I	Reset	Active-Low AXIS audio reset
m_axis_aud_*		Audio AXIS Interface	AXIS Audio Interface[1]
Irq	O	Interrupt	Active-High interrupt
lrclk_out	O	LRClk	Output LR Clock. Available when core is configured as master
sclk_out	O	SCLK	Output SCK Clock. Available when core is configured as master
lrclk_in	I	LRClk	Input LR Clock. Available when core is configured as slave
Sclk_in	I	SCLK	Input SCK Clock. Available when core is configured as slave
sdata_0_in	I	SDATA0	I2S Serial Data In
sdata_1_in	I	SDATA1	I2S Serial Data In. Available when number of audio channels is > 2
sdata_2_in	I	SDATA2	I2S Serial Data In. Available when number of audio channels is > 4

Table 1: **Port Names** *(cont'd)*

Port Name	I/O	Clock	Description
sdata_3_in	I	SDATA3	I2S Serial Data In. Available when number of audio channels is > 6

Notes:

1. For more details on Audio AXIS interface, see Audio AXIS Interface.

I2S Transmitter Register Space

***Note*:** The AXI4-Lite write access register is updated by the 32-bit AXI Write Data (`*_wdata`) signal, and is not impacted by the AXI Write Data Strobe (`*_wstrb`) signal. For a write, both the AXI Write Address Valid (`*_awvalid`) and AXI Write Data Valid (`*_wvalid`) signals should be asserted together.

Table 2: **Register Address Space**

Address (hex)	Register Name
0x00	**Core Version**: Returns the core major and minor versions
0x04	**Core Configuration**: Returns the core configuration details
0x08	**Core Control**: Register to enable/disable the core
0x0C	**Validity Register**: Validates the incoming sample word
0x10	**Interrupt Control**: Interrupts the enable/disable register
0x14	**Interrupt Status**: Interrupts the Status register
0x20	**I2S Timing Control**: Register to program the SCK divider value
0x30	**Channel 0/1 Control**: Channel 0/1 control register
0x34	**Channel 2/3 Control**: Channel 2/3 control register
0x38	**Channel 4/5 Control**: Channel 4/5 control register
0x3C	**Channel 6/7 Control**: Channel 6/7 control register
0x50	**AES Channel Status 0**: Register that returns the LSB 32-bit of the AES Channel Status
0x54	**AES Channel Status 1**: Register that returns the next LSB 32-bit of the AES Channel Status
0x58	**AES Channel Status 2**: Register that returns the 32-bit of the AES Channel Status
0x5C	**AES Channel Status 3**: Register that returns the 32-bit of the AES Channel Status
0x60	**AES Channel Status 4**: Register that returns the 32-bit of the AES Channel Status
0x64	**AES Channel Status 5**: Register that returns the MSB 32-bit of the AES Channel Status

Core Version (0x00)

This register returns the major and minor versions of the IP core.

Table 3: **Transmitter Core Version (0x00)**

Bit	Default Value	Access Type	Description
31:16	0x1	RO	**Major Revision**: This is the IP major revision value. For example, if the IP version is 1.2, then this will return a value of 1.
15:0	0x0	RO	**Minor Revision**: This is the IP minor revision value. For example, if the IP version is 1.2, then this will return a value of 2.

Core Configuration (0x04)

This register returns the IP Configuration.

Table 4: **Transmitter Core Configuration (0x04)**

Bit	Default Value	Access Type	Description
31:17	0x1		Reserved
16		RO	I2S Data Width: Indicates the I2S data width of the core 1 = 24-bit 0 = 16-bit
15:12			Reserved
11:8		RO	Number of audio channels: Indicates the number of audio channels supported. Valid values are 2, 4, 6, and 8.
7:1			Reserved
0		RO	Is I2S master: Indicates if the core has been generated as an I2S master or slave. 1 = I2S master

Control Register (0x08)

This register provides capability to enable/disable the core.

Table 5: **Transmitter Control Register (0×08)**

Bit	Default Value	Access Type	Description
31:4	0	RO	Reserved
3	0x0	RO	Selected 32-bit LRCLK mode
2	0x0	R/W	Valid when bit 1 is set. Selects left/right justification: • 0: Left justification • 1: Right justification
1	0x0	R/W	Enable left/right justification
0	0x9	R/W	Enable core operations. Setting this bit to '1' will enable the core operations. Setting this bit to '0' disables the core operations

Validity Register (0x0C)

This register can update the validity of the incoming Audio sample. Writing '1' will always make the input data Valid irrespective of the Validity bit (bit 28) on AXI4-Stream input. Else, the validity bit decides the validity of the sample data.

Table 6: **Validity Register #90x0C)**

Bit	Default Value	Access Type	Description
31:1	0	RO	Reserved
0	0x0	R/W	Validity Bit: • 1: The audio input sample is always valid • 0: The Validity bit in the incoming stream decides the validity of the sample

Interrupt Control Register (0x10)

This register determines the interrupt sources in the Interrupt Status Register that are allowed to generate an interrupt. Writing a '1' to a bit will enable the corresponding interrupt.

Table 7: **Transmitter Interrupt Control Register (0×10)**

Bit	Default Value	Access Type	Description
31	0	R/W	**Global Interrupt Enable**: Enables the global interrupt
30:4			Reserved
3	0	R/W	**Underflow Interrupt Enable**: Enables the underflow interrupt
2	0	R/W	**AES Channel Status Updated Interrupt enable**: Enables the AES channel status updated interrupt
1	0	R/W	**AES Block Sync Error Interrupt enable**: Enables the AES block sync interrupt
0	0	R/W	**AES Block Completed Interrupt enable**: Enables the AES block completed interrupt

Interrupt Status (0x14)

This register returns the status of the interrupt bits.

Table 8: **Transmitter Interrupt Status (0×14)**

Bit	Default Value	Access Type	Description
31:4			Reserved
3	0	R/W	**Underflow Interrupt**: This bit is set when the core did not receive the samples for all channels in time. This scenario can lead to distortions in the audio that is being played. Write a '1' to clear this bit.

Table 8: **Transmitter Interrupt Status (0×14)** *(cont'd)*

Bit	Default Value	Access Type	Description
2	0	R/W	**AES Channel Status Updated**: This bit is set when a change in the captured AES channel status has been detected. Write a '1' to clear this flag.
1	0	R/W	**AES Block Sync Error**: This bit is set when synchronization with the start of an AES block has been lost. This occurs if the incoming audio our AXIS does violates the guidelines. Write a '1' to clear this flag.
0	0	R/W	**AES Block Completed**: This bit is set when a complete AES block has been received (192 AES frames). This bit is set every time the IP receives one block of audio. Write a '1' to clear this flag.

I2S Timing Control (0x20)

This register is used to set the divider value to generate the SCLK. Typically SCLK = 2*24*Fs, where 24 (this value can also be 16) is the I2S data width and Fs is the audio sampling rate.

Table 9: **Transmitter I2S Timing Control (0x20)**

Bit	Default Value	Access Type	Description
31:8			Reserved
7:0	0	R/W	**SCLK Out Divider value**: Set a divider value for a generation of SCLK. The value of the divider should be such that MCLK/SCLK = Divider_value *2.

Channel 0/1 Control (0x30)

The IP provides a mechanism to route the audio channels onto any I2S output. For example, audio received on channels 2/3 can be routed to the output on any of the four I2S ports. Similarly, audio received on channels 0/1 can be routed to all of the four I2S ports.

Table 10: **Transmitter Channel 0/1 Control (0x30)**

Bit	Default Value	Access Type	Description
31:3			Reserved
2:0	0x1	RW	**Channel Mux value**: Specify a value to multiplex the audio channel output. 0x0: Output on I2S channel 0 is disabled 0x1: I2S channel 0 outputs the audio received on channel 0 /1 0x2: I2S channel 0 outputs the audio received on channel 2 /3 0x3: I2S channel 0 outputs the audio received on channel 4 /5 0x4: I2S channel 0 outputs the audio received on channel 6 /7 All other values are reserved.

Channel 2/3 Control (0x34)

The IP provides a mechanism to route the audio channels onto any I2S output. For example, audio received on channels 2/3 can be routed to the output on any of the four I2S ports. Similarly, audio received on channels 0/1 can be routed to all of the four I2S ports.

Table 11: **Transmitter Channel 2/3 Control (0x34)**

Bit	Default Value	Access Type	Description
31:3			Reserved
2:0	0x2	R/W	**Channel Mux Value**: Specify a value to multiplex the audio channel output. 0x0: Output on I2S channel 1 is disabled 0x1: I2S channel 1 outputs the audio received on channel 0 /1 0x2: I2S channel 1 outputs the audio received on channel 2 /3 0x3: I2S channel 1 outputs the audio received on channel 4 /5 0x4: I2S channel 1 outputs the audio received on channel 6 /7 All other values are reserved.

Channel 4/5 Control (0x38)

The IP provides a mechanism to route the audio channels onto any I2S output. For example, audio received on channels 2/3 can be routed to the output on any of the four I2S ports. Similarly, audio received on channels 0/1 can be routed to all of the four I2S ports.

Table 12: **Transmitter Channel 4/5 Control (0x38)**

Bit	Default Value	Access Type	Description
31:3			Reserved
2:0	0x3	R/W	**Channel Mux Value**: Specify a value to multiplex the audio channel output. 0x0: Output on I2S channel 2 is disabled 0x1: I2S channel 2 outputs the audio received on channel 0 /1 0x2: I2S channel 2 outputs the audio received on channel 2 /3 0x3: I2S channel 2 outputs the audio received on channel 4 /5 0x4: I2S channel 2 outputs the audio received on channel 6 /7 All other values are reserved.

Channel 6/7 Control (0x3C)

The IP provides a mechanism to route the audio channels onto any I2S output. For example, audio received on channels 2/3 can be routed to the output on any of the four I2S ports. Similarly, audio received on channels 0/1 can be routed to any of the four I2S ports.

Table 13: **Transmitter Channel 6/7 Control (0x3C)**

Bit	Default Value	Access Type	Description
31:3			Reserved
2:0	0x4	R/W	**Channel Mux Value**: Specify a value to multiplex the audio channel output. 0x0: Output on I2S channel 3 is disabled 0x1: I2S channel 3 outputs the audio received on channel 0 /1 0x2: I2S channel 3 outputs the audio received on channel 2 /3 0x3: I2S channel 3 outputs the audio received on channel 4 /5 0x4: I2S channel 3 outputs the audio received on channel 6 /7 All other values are reserved.

Notes:

1. Ensure that the value programmed in the four registers mentioned above are unique and different. The IP may not behave as expected if the same value is programmed in all the registers.

AES Channel Status (0x50-0x64)

These 6 registers together give the 192-bit channel status information that is received over the audio block. A write to any of the six registers would restart the process of accumulating the channel status and would result in the AES channel status updated interrupt. The 6 registers give the value in order of LSB to MSB. The register 0x50 returns bits [31:0] of 192-bit channel status, while the register 0x64 returns bits [191:160].

Table 14: **Transmitter AES Channel Status (0x50-0x64)**

Bit	Default Value	Access Type	Description
31:0	0	R/WC	**32-bit AES value**: 32-bit AES Channel Status value.

I2S Receiver Register Space

Table 15: **Register Address Space**

Address (hex)	Register Name
0x00	**Core Version**: Returns the core major and minor versions
0x04	**Core Configuration**: Returns the core configuration details
0x08	**Core Control**: Register to enable/disable the core
0x0C	**Validity Register**: Sets the Validity bit on output Stream data
0x10	**Interrupt Control**: Interrupts the enable/disable register
0x14	**Interrupt Status**: Interrupts the Status register
0x20	**I2S Timing Control**: Register to program the SCK divider value
0x30	**Channel 0/1 Control**: Channel 0/1 control register

Table 15: **Register Address Space** *(cont'd)*

Address (hex)	Register Name
0x34	**Channel 2/3 Control**: Channel 2/3 control register
0x38	**Channel 4/5 Control**: Channel 4/5 control register
0x3C	**Channel 6/7 Control**: Channel 6/7 control register
0x50	**AES Channel Status 0**: Register to specify the LSB 32-bit of the AES Channel Status
0x54	**AES Channel Status 1**: Register to specify the next LSB 32-bit of the AES Channel Status
0x58	**AES Channel Status 2**: Register to specify the 32-bit of the AES Channel Status
0x5C	**AES Channel Status 3**: Register to specify the 32-bit of the AES Channel Status
0x60	**AES Channel Status 4**: Register to specify the 32-bit of the AES Channel Status
0x64	**AES Channel Status 5**: Register to specify the MSB 32-bit of the AES Channel Status

Core Version (0x00)

This register returns the major and minor versions of the IP core.

Table 16: **Receiver Core Version (0x00)**

Bit	Default Value	Access Type	Description
31:16	0x1	RO	**Major Revision**: This is the IP major revision value. For example, if the IP version is 1.2, then this will return a value of 1.
15:0	0x0	RO	**Minor Revision**: This is the IP minor revision value. For example, if the IP version is 1.2, then this will return a value of 2.

Core Configuration (0x04)

This register returns the IP Configuration.

Table 17: **Receiver Core Configuration (0x04)**

Bit	Default Value	Access Type	Description
31:17			RSVD
16		RO	**I2S Data Width**: Indicates the I2S data width of the core. 1 = 24-bit 0 = 16-bit
15:12			RSVD
11:8		RO	Number of audio channels: Indicates the number of audio channels supported. Valid values are 2, 4, 6, and 8.
7:1			RSVD
0		RO	Is I2S Master: Indicates if the core has been generated as an I2S master or slave. 1 = I2S Master

Control Register (0x08)

This register lets you enable/disable the core.

Table 18: **Receiver Control Register (0×08)**

Bit	Default Value	Access Type	Description
31:17	0	R	Reserved
16	0	WO	**Latch AES Channel Status**: Program this bit to latch the AES channel status bits from the registers. This latched value is then put onto the AXIS interface. This register is auto cleared.
15:4			Reserved
3	0x0	RO	Selected 32-bit LR clock mode
2	0x0	R/W	Valid when bit 1 is set. Selects left/right justification: • 0: Left justification • 1: Right justification
1	0x0	R/W	Enable Left/Right Justification
0	0x0	R/W	**Enable**: Setting this bit to '1' enables the core operations. Setting this bit to '0' disables the core operations.

Validity Set Register (0x0C)

This register sets the Validity bit on the output AXI4-Stream.

Table 19: **Validity Set Register (0x0C)**

Bit	Default Value	Access Type	Description
31:1	0	RO	Reserved
0	0x0	R/W	Validity Bit on the Output Audio AXI4-Stream.

Interrupt Control Register (0x10)

This register determines the interrupts sources in the Interrupt Status register that are allowed to generate an interrupt. Writing a '1' to a bit enables the corresponding interrupt.

Table 20: **Receiver Interrupt Control Register (0×10)**

Bit	Default Value	Access Type	Description
31	0	R/W	**Global Interrupt Enable**: Enables the global interrupt.
30:2			Reserved
1	0	R/W	**Overflow Interrupt Enable**: Enables the overflow interrupt.

Table 20: **Receiver Interrupt Control Register (0×10)** *(cont'd)*

Bit	Default Value	Access Type	Description
0	0	R/W	**AES Block Completed Interrupt enable**: Enables the AES block completed interrupt.

Interrupt Status (0x14)

This register returns the status of the interrupt bits.

Table 21: **Receiver Interrupt Status (0×14)**

Bit	Default Value	Access Type	Description
31:2			Reserved
1	0	R/W1C	**Overflow Interrupt**: This bit is set when the IP is not able to send all enabled audio channels in time. This interrupt would indicate loss of samples. Write a '1' to clear this flag.
0	0	R/W1C	**AES Block Completed**: This bit is set when a complete AES block has been received (192 AES frames). This bit is set every time the IP receives one block of audio. Write a '1' to clear this flag.

I2S Timing Control (0x20)

This register is used to set the divider value to generate the SCLK. Typically SCLK = 2*24*Fs, where 24 is the I2S data width (this value can also be 16) and Fs is the audio sampling rate.

Table 22: **Receiver I2S Timing Control (0x20)**

Bit	Default Value	Access Type	Description
31:8			Reserved
7:0	0	R/W	**SCLK Out Divider value**: Set a divider value for generation of SCLK. The value of the divider should be such that MCLK/SCLK = Divider_value *2. This register has to be programmed when the core is configured as I2S master.

Channel 0/1 Control (0x30)

The IP provides a mechanism to route the audio from any I2S input. For example, audio received on I2S Channel 0 can be routed to any of the eight audio channels. Similarly, audio received on one I2S channel can be routed to all of the eight audio channels.

Table 23: **Receiver Channel 0/1 Control (0x30)**

Bit	Default Value	Access Type	Description
31:3			Reserved
2:0	0x1	R/W	**Channel Mux value**: Specify a value to multiplex the audio channel output. 0x0: disabled 0x1: Audio received on I2S channel 0 is routed as audio channel 0 /1 0x2: Audio received on I2S channel 0 is routed as audio channel 2 /3 0x3: Audio received on I2S channel 0 is routed as audio channel 4 /5 0x4: Audio received on I2S channel 0 is routed as audio channel 6 /7 All other values are reserved.

Channel 2/3 Control (0x34)

The IP provides a mechanism to route the audio from any I2S input. For example, audio received on I2S Channel 0 can be routed to any of the eight audio channels. Similarly, audio received on one I2S channel can be routed to all of the eight audio channels.

Table 24: **Receiver Channel 2/3 Control (0x34)**

Bit	Default Value	Access Type	Description
31:3			Reserved
2:0	0x2	R/W	**Channel Mux Value**: Specify a value to multiplex the audio channel output. 0x0: disabled 0x1: Audio received on I2S channel 1 is routed as audio channel 0 /1 0x2: Audio received on I2S channel 1 is routed as audio channel 2 /3 0x3: Audio received on I2S channel 1 is routed as audio channel 4 /5 0x4: Audio received on I2S channel 1 is routed as audio channel 6 /7 All other values are reserved.

Channel 4/5 Control (0x38)

The IP provides a mechanism to route the audio from any I2S input. For example, audio received on I2S Channel 0 can be routed to any of the eight audio channels. Similarly, audio received on one I2S channel can be routed to all of the 8 audio channels.

Table 25: **Receiver Channel 4/5 Control (0x38)**

Bit	Default Value	Access Type	Description
31:3			Reserved

Table 25: **Receiver Channel 4/5 Control (0x38)** *(cont'd)*

Bit	Default Value	Access Type	Description
2:0	0x3	R/W	**Channel Mux Value**: Specify a value to multiplex the audio channel output. 0x0: disabled 0x1: Audio received on I2S channel 2 is routed as audio channel 0 /1 0x2: Audio received on I2S channel 2 is routed as audio channel 2 /3 0x3: Audio received on I2S channel 2 is routed as audio channel 4 /5 0x4: Audio received on I2S channel 2 is routed as audio channel 6 /7 All other values are reserved.

Channel 6/7 Control (0x3C)

The IP provides a mechanism to route the audio from any I2S input. For example, audio received on I2S Channel 0 can be routed to any of the eight audio channels. Similarly, audio received on one I2S channel can be routed to all of the eight audio channels.

Table 26: **Receiver Channel 6/7 Control (0x3C)**

Bit	Default Value	Access Type	Description
31:3			Reserved
2:0	0x4	R/W	**Channel Mux Value**: Specify a value to multiplex the audio channel output. 0x0: disabled 0x1: Audio received on I2S channel 3 is routed as audio channel 0 /1 0x2: Audio received on I2S channel 3 is routed as audio channel 2 /3 0x3: Audio received on I2S channel 3 is routed as audio channel 4 /5 0x4: Audio received on I2S channel 3 is routed as audio channel 6 /7 All other values are reserved.

Notes:

1. Ensure that the value programmed in the four registers mentioned above are unique and different. The IP may not behave as expected if the same value is programmed in all the registers.

AES Channel Status (0x50-0x64)

These six registers together allow the user to specify the 192-bit channel status information that is inserted over the audio block. These registers give the value in order of LSB to MSB. The register 0x50 should have the bits [31:0] of 192-bit channel status, while register 0x64 should have the bits [191:160].

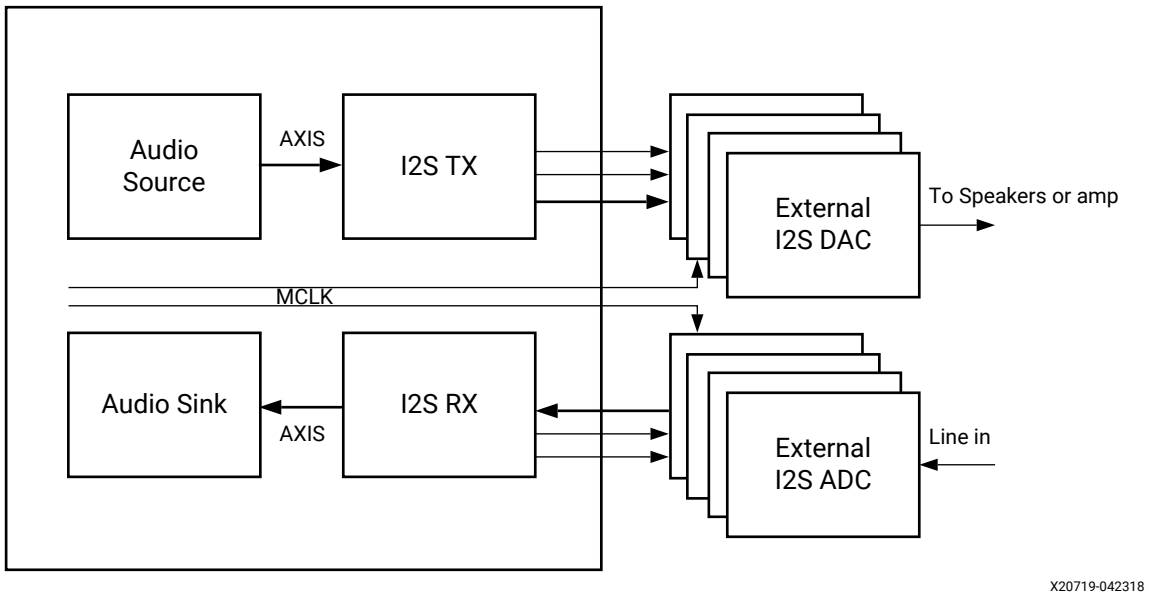
Table 27: **Receiver AES Channel Status (0x50-0x64)**

Bit	Default Value	Access Type	Description
31:0	0	R/W	**32-bit AES value**: 32-bit AES channel status value.

Designing with the Core

The I2S TX and RX IPs can be used in systems to send and receive I2S audio. A typical use case is as shown below.

Figure 3: **System Using TX RX**



The I2S IPs typically interface with the external ADC/DAC which facilitates the playback of audio.

General Design Guidelines

Use the Example Design

Each instance of the I2S Transmitter and I2S Receiver core created by the Vivado design tool is delivered with an example design that can be implemented in a device and then simulated. This design can be used as a starting point for your own design or can be used to sanity-check your application in the event of difficulty. See the Example Design content for information about using and customizing the example designs for the core.

Related Information
Xilinx Resources

Registering Signals

To simplify timing and increase system performance in a programmable device design, keep all inputs and outputs registered between the user application and the core. This means that all inputs and outputs from the user application should come from, or connect to, a flip-flop. While registering signals might not be possible for all paths, it simplifies timing analysis and makes it easier for the Xilinx® tools to place and route the design.

Recognize Timing Critical Signals

The constraints provided with the example design identify the critical signals and timing constraints that should be applied.

Related Information
Xilinx Resources

Make Only Allowed Modifications

You should not modify the core. Any modifications can have adverse effects on system timing and protocol compliance. Supported user configurations of the core can only be made by selecting the options in the customization IP dialog box when the core is generated.

Clocking

There are three possible clock inputs available. Ensure that a proper `aud_clk` is supplied so that the correct SCLK can be generated by the IP. The audio clock is typically an integer multiple of 128×Fs and is decided by the DAC/ADC being used. To minimize jitter, use a very stable clock source to generate the audio clock.

LRCLK is also generated by the IP in master mode. LRCLK edges coincide with the falling edge of SCLK following I2S protocol. Typically LR clock frequency is SCLK frequency divided by (2 * I2S data width). In cases where Left/Right justification is selected or when 32-bit LRCLK is selected, LR clock frequency is SCLK frequency divided by (2*32).

Table 28: **Clocks**

Clock	Description
s_axi_ctrl_aclk	Control interface clock
s_axis_aud_aclk	AXIS streaming clock
m_axis_aud_aclk	AXIS streaming clock
aud_aclk	A reference audio clock which is an integer multiple of Fs (typically 128×Fs, 384×Fs etc.)

Resets

The `s_axi_ctrl_aresetn` resets the register interface and puts all the registers in their default states.

The `aud_mrst` (an active-High reset) resets the audio domain, while the `s_axis_aud_aresetn` resets the AXIS domain. After a reset, it is advisable to disable and enable the IP for a clean recovery.

Programmimg Sequence

The I2S Transmitter can be setup using the following programming sequence:

1. Setup the Channel Mux registers, if required.

   Note: It is not recommended to change this value at runtime.

2. Program the SCLK Divider.

   Note: It is not recommended to change this value at runtime.

3. Enable the core.

The I2S Receiver can be setup using the following programming sequence:

1. Setup the Channel Mux registers, if required.

   Note: It is not recommended to change this value at runtime.

2. Program the SCLK Divider.

   Note: It is not recommended to change this value at runtime.

3. Program the AES registers to specify the 192 bits of Channel Status value
4. Enable the core and latch the AES Channel bit.

Note: After asserting either `aud_mrst` or `m_axis_aresetn`, the core has to disabled and enabled again.

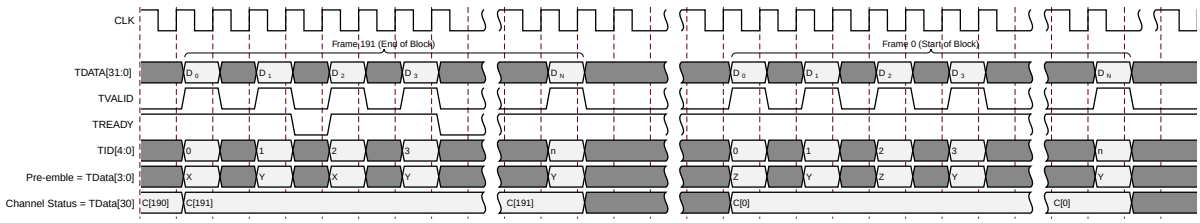
Interrupts

Each core has one interrupt output. The Interrupt output is level triggered and stays asserted until the interrupt status bits are cleared.

Audio AXIS Interface

An AXI4-Stream audio cycle is illustrated in the following figure. The data is valid when both the valid (TVLD) and ready (TRDY) signals are asserted. The I2S Receiver sends out adjacent channels in sequential order (CH0, CH1, etc.). Usually, the I2S Transmitter also expects the channels in sequential order. If the channel data is not in order, then the I2S Transmitter asserts an underflow or block sync error.

Figure 4: **Audio AXIS Interface**



You must ensure proper pre-emble and TIDs while sending more than two channels of audio data over AXIS. The data width over the AXI4-Stream interface is fixed at 32-bits. All bit positions are as per the IEC60958-3 standard except for the preamble bit format. The preamble provides the start of the audio block and audio channel information. The preamble patterns for the start of block, channelA audio data, and channelB audio data are listed as follows:

Table 29: **Audio Axis Interface Patterns**

Bits [3:0]	Description
0001	Start of Audio Block/Channel 0 audio sample
0010	Channel 0/2/4/6 audio data - Left Audio Data
0011	Channel 1/3/5/7 audio data - Right Audio Data

Table 30: **Audio Input Stream Interface for I2S Transmitter**

Ports	Direction	Width	Description
s_axis_aud_aclk	Input	1	Clock (the audio streaming clock must be greater than or equal to 128 times the audio sample frequency)
s_axis_aud_aresetn	Input	1	Reset (Active-Low)
s_axis_aud__tdata	Input	32	Data: • [31] P (Parity) • [30] C (Channel Status) • [29] U (user bit) • [28] V (Validity bit) • [27:4] Audio Sample word • [3:0] Preamble code • 4'b0001 Subframe 1/start of audio block • 4'b0010 Subframe 1 • 4'b0010 Subframe 2
s_axis_aud__tid	Input	3	Channel ID: 0/2/4/6 audio data - Left Audio Data 1/3/5/7 audio data - Right Audio Data
s_axis_aud__tready	Output	1	Ready
s_axis_aud__tvalid	Input	1	Valid

Table 31: **Audio Output Stream Interface for I2S Receiver**

Name	Direction	Width	Description
m_axis_aud_aclk	Input	1	Clock (the audio streaming clock must be greater than or equal to 128 times the audio sample frequency)
m_axis_aud_aresetn	Input	1	Reset (Active-Low)

Table 31: **Audio Output Stream Interface for I2S Receiver** *(cont'd)*

Name	Direction	Width	Description
m_axis_aud_tdata	Output	32	Data: • [31] P (Parity) • [30] C (Channel Status) • [29] U (user bit) • [28] V (Validity bit) • [27:4] Audio Sample word • [3:0] Preamble code • 4'b0001 Subframe 1/start of audio block • 4'b0010 Subframe 1 • 4'b0010 Subframe 2
m_axis_aud_tid	Output	3	Channel ID
m_axis_aud_tready	Input	1	Ready
m_axis_aud_tvaild	Output	1	Valid

***Note*:**

- The Audio sample word is sent on TDATA of AXI4-Stream using bits from 27 : 4
- When the I2S Datawidth is 24, all the reserved bits from 27:4 are used to send the data
- When the I2S Datawidth is 16, the sample data is sent on TDATA[27:12] bits. LSB 8 bits are padded with 0's

Design Flow Steps

This section 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)
- *Vivado Design Suite User Guide: Designing with IP* (UG896)
- *Vivado Design Suite User Guide: Getting Started* (UG910)
- *Vivado Design Suite User Guide: Logic Simulation* (UG900)

Customizing and Generating the Core

The I2S Transmitter and Receiver can be found under the following Audio Connectivity and Processing Vivado® IP catalog.

To access the I2S IPs, do the following:

1. Open an existing project or create a new project using the Vivado design tools.
2. Open the IP catalog and navigate to the taxonomies.
3. Double-click on either I2S Receiver or Transmitter to bring up the customize IP window.

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

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

For more information on generating the core in the Vivado IP integrator, see the *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994) for detailed information. Vivado IDE might auto-compute certain configuration values when validating or generating the design, as noted in this section. You can view the parameter value after successful completion of the `validate_bd_design` command.

I2S Receiver Customize IP

Figure 5: **I2S Receiver Configuration Tab**

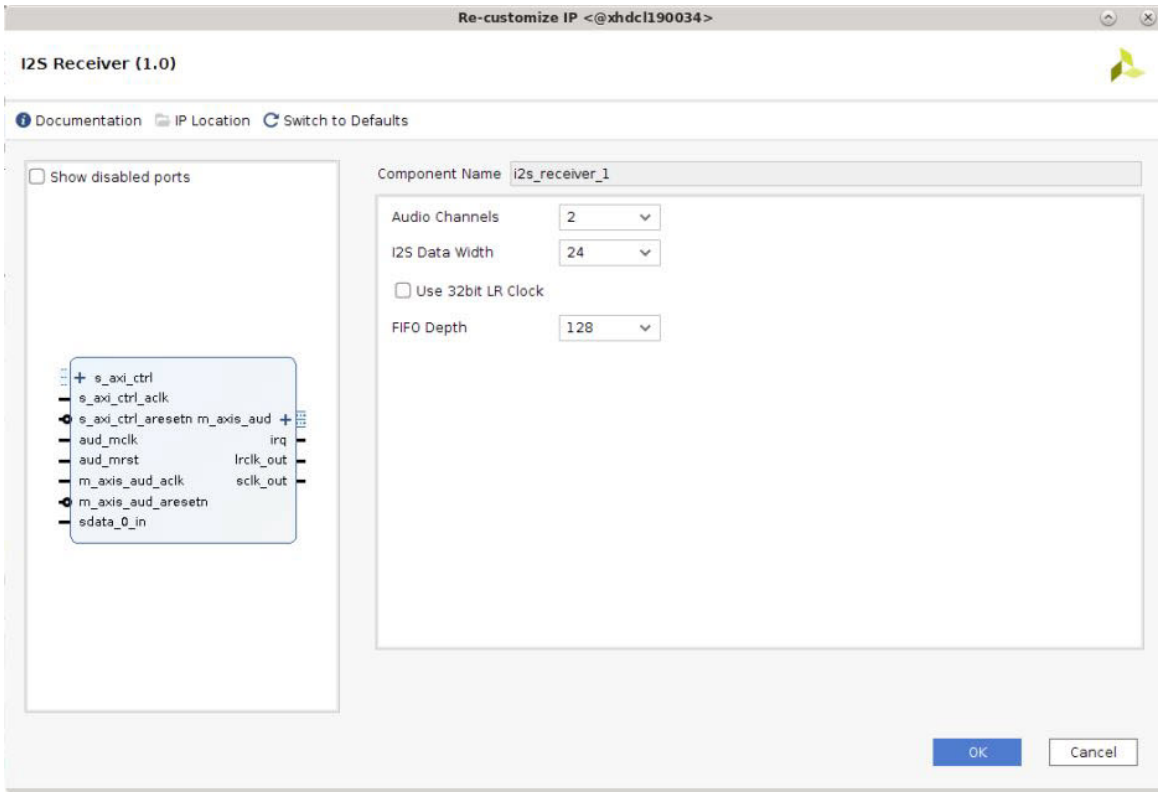
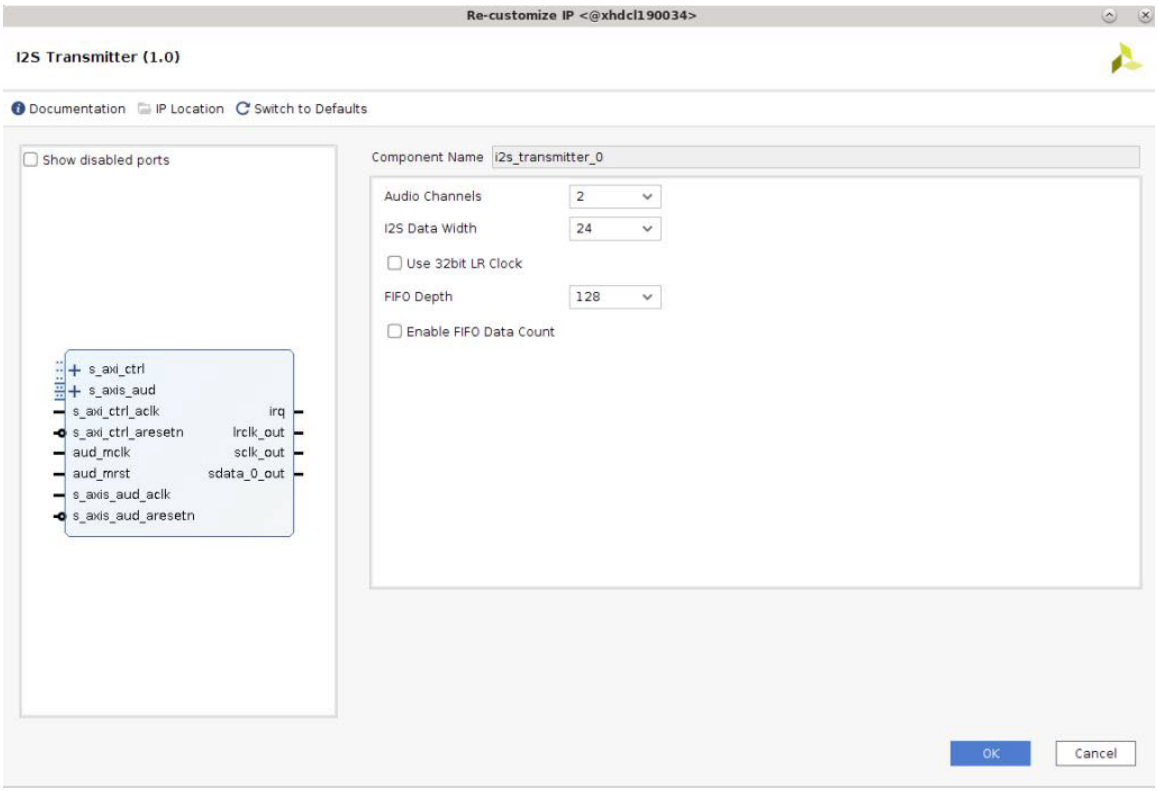


Figure 6: **I2S Transmitter Configuration Tab**



Field Descriptions

- **Component Name :** The base name of the output files generated for the core. Names must begin with a letter and can be composed of any of the following characters: a- z, 0 to 9, and "_".
- **Audio Channels :** Specify the number of audio channels. Allowed values are 2, 4, 6, and 8.
- **I2S Data Width :** Specify the I2S data width. Allowed values are 16 and 24.
- **Use 32 bit LR Clock:** Enables the transmission of data on the I2S channel with 32-bit SCLK. Valid sample bits are determined by the I2S data width.
- **FIFO Depth :** Specify the depth of the FIFO. Allowed values are 64, 128, 256, 512, and 1024. In case of I2S Transmitter, the data is output on I2S interface only after the FIFO is half-filled.
- **Enable FIFO Data Count:** Select this option to enable the IP to output the FIFO read data count.

User Parameters

The following table shows the relationship between the fields in the Vivado IDE and the User Parameters (which can be viewed in the tool command language (Tcl) Console).

Table 32: **User Parameters**

Vivado IDE Parameters	Parameter Name	Default Value	Allowed Value
I2S Receiver			
Audio Channels	C_NUM_CHANNELS	2	2, 4, 6, 8
I2S Data width	C_DWIDTH	24	16, 24
32bit LRCLK	C_32BIT_LR	0	0,1
FIFO Depth	C_DEPTH	128	64, 128, 256, 512, 1024
I2S Transmitter			
Audio Channels	C_NUM_CHANNELS	2	2, 4, 6, 8
I2S Data width	C_DWIDTH	24	16, 24
32bit LRCLK	C_32BIT_LR	0	0,1
FIFO Depth	C_DEPTH	128	64, 128, 256, 512, 1024
Enable FIFO Count	C_ENABLE_FIFO_COUNT	False	True, False

Output Generation

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

Constraining the Core

Required Constraints

This section is not applicable for this IP core.

Device, Package, and Speed Grade Selections

This section is not applicable for this IP core.

Clock Frequencies

For more information, see Clocking.

Clock Management

It is advisable to have the audio clock generated from a stable source for minimal jitter. If the jitter is of low importance, a MMCM can be used to generate the audio clock.

Clock Placement

Audio clock, if supplied from an external source, should be connected to a clock capable I/O so that it can be used by the FPGA fabric.

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).

Synthesis and Implementation

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

Example Design

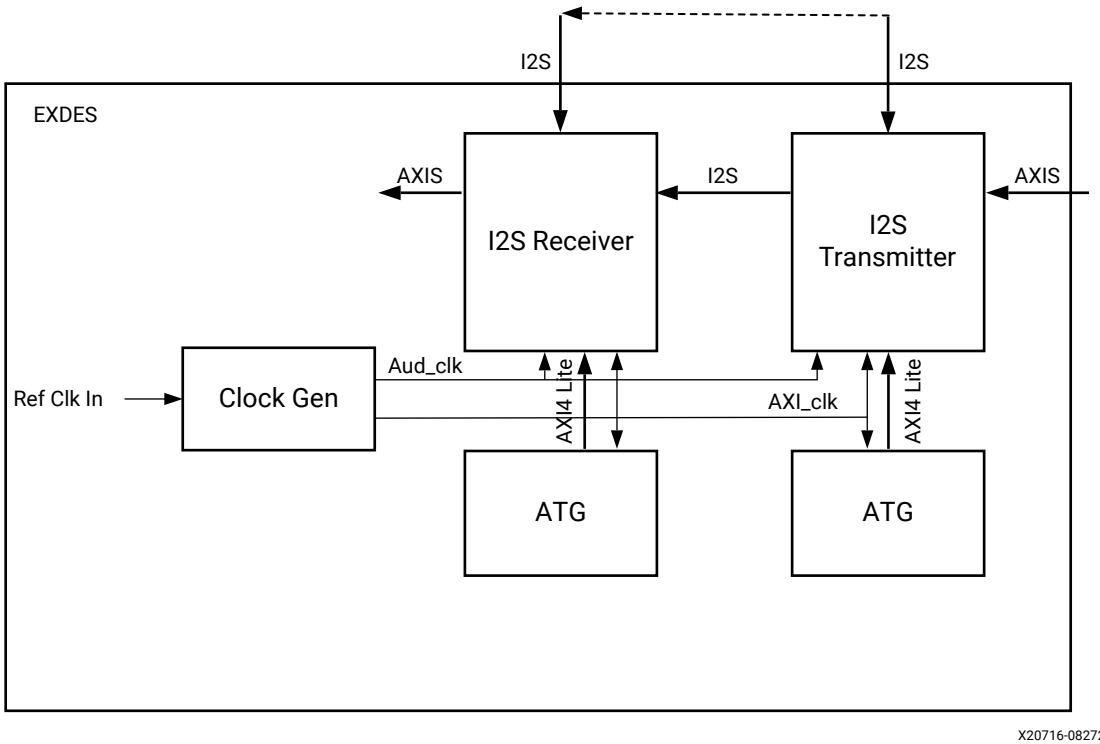
This chapter contains information about the example design provided in the Vivado® Design Suite. The top module instantiates all components of the core and example design that are needed to implement the design in hardware, as shown below. This includes the Clocking Wizard and the Register configuration modules. The available Example Design is shown in the following table.

Table 33: **Example Design**

Topology	Hardware	Processor
Loopback TX-RX	N/A - Simulation only	ATG

Note: Behavior of this IP is also shown in the HDMI Pass-Through +I2S Audio Example Design documented in the *HDMI 1.4/2.0 Transmitter Subsystem Product Guide* (PG235).

Figure 7: **Core Example Design**



***Note*:** The I2S Connection from Transmitter to Receiver is an external connection which is implemented in the test-bench of the design for simulation purposes. Practically, the connection has to be made outside the board.

The core example design is a simulation-only design; it cannot be validated on the board. This example design demonstrates transactions on the AXI4-Lite and AXI4-Stream interfaces of the DUT.

- **Clock generator:** A clocking wizard is used to generate the clocks for the example design. It generates the `aud_clk`, AXI4-Lite clock, and the AXI4-Stream clock. The example design is held in reset until the MMCM is locked.
- **Axi Traffic Generator (ATG):** The ATGs are used to program the I2S IPs. The ATGs start the configuration process as soon as the MMCM is locked.
- **I2S Transmitter:** This module receives the audio data and sends it over to the I2S bus that is connected to the I2S receiver.
- **I2S Receiver:** This module receives the I2S data and outputs it on the AXIS interface.

Implementing the Example Design

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

After following the steps described in Chapter 5: Design Flow Steps, implement the example design as follows:

1. Right-click the core in the Hierarchy window, and select **Open IP Example Design**.
2. A new window pops up, asking you to specify a directory for the example design. Select a new directory, or keep the default directory. A new project is automatically created in the selected directory and opened in a new Vivado IDE window.
3. In the Flow Navigator (left-side pane), click **Run Implementation** and follow the directions. In the current project directory, a new project with the name_0_ex is created and the files are delivered in that directory. This directory and its sub-directories contain all the source files that are required to create the I2S core example design.

Simulating the Example Design

Using the I2S core example designs delivered as part of each I2S core, the behavior of the core can be quickly simulated and observed. The simulation script compiles the core example design and the supporting simulation files. It then runs the simulation and checks if it completed successfully.

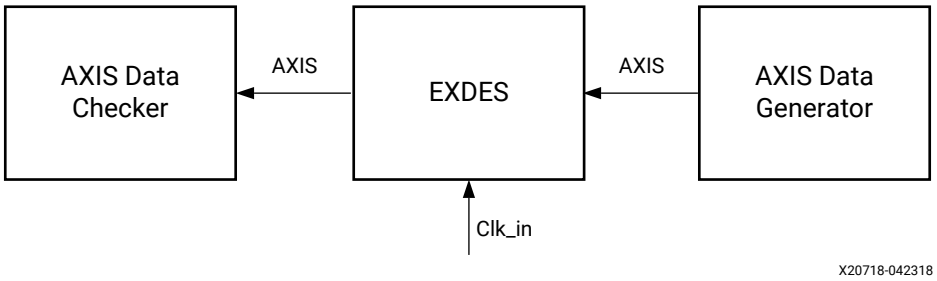
If the test fails, the following message displays: Test Failed!!!

If the test passes, the following message displays: Test Completed Successfully

Test Bench for Example Design

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

Figure 8: **Test Bench**



The above figure shows the test bench for example design. The top-level test bench feeds a clock input, AXIS data to the EXDES. The test bench also checks the received AXIS data.

- **AXIS Data Generator:** This module generates the AXIS audio traffic and feeds the I2S Transmitter.
- **AXIS Data Checker:** This modules reads the AXIS data and checks for data integrity.

Appendix A

Debugging

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

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.*

Finding Help on Xilinx.com

To help in the design and debug process when using the core, the Xilinx Support web page contains key resources such as product documentation, release notes, answer records, information about known issues, and links for obtaining further product support. The Xilinx Community Forums are also available where members can learn, participate, share, and ask questions about Xilinx solutions.

Documentation

This product guide is the main document associated with the core. This guide, along with documentation related to all products that aid in the design process, can be found on the Xilinx Support web page or by using the Xilinx® Documentation Navigator. Download the Xilinx Documentation Navigator from the Downloads page. For more information about this tool and the features available, open the online help after installation.

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 keywords such as:

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

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

Master Answer Record for the Core

For I2S Receiver, see Xilinx Answer 70288

For I2S Transmitter, see Xilinx Answer 70699

Technical Support

Xilinx provides technical support on the Xilinx Community Forums 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 ask questions, navigate to the Xilinx Community Forums.

Hardware Debug

Hardware issues can range from no audio to audio with noise. This section provides debug steps for common issues.

Following are some of the common problems encountered and possible solutions:

1. No audio received/played: Ensure that the ADC/DAC/CODEC is in slave mode. The I2S IPs operate as masters. The I2S IPs only support 16 or 24-bit I2S mode only.

2. Audio has a lot of noise: Ensure that DAC/ADC/CODEC are configured for the same data width as the I2S IPs. Also ensure that the MCLK supplied to the DAC/ADC/CODEC is same as the one supplied to I2S IPs.

Appendix B

Additional Resources and Legal Notices

Xilinx Resources

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

Documentation Navigator and Design Hubs

Xilinx® Documentation Navigator (DocNav) provides access to Xilinx documents, videos, and support resources, which you can filter and search to find information. To open DocNav:

- From the Vivado® IDE, select **Help → Documentation and Tutorials**.
- On Windows, select **Start → All Programs → Xilinx Design Tools → DocNav**.
- At the Linux command prompt, enter `docnav`.

Xilinx Design Hubs provide links to documentation organized by design tasks and other topics, which you can use to learn key concepts and address frequently asked questions. To access the Design Hubs:

- In DocNav, click the **Design Hubs View** tab.
- On the Xilinx website, see the Design Hubs page.

Note: For more information on DocNav, see the Documentation Navigator page on the Xilinx website.

References

These documents provide supplemental material useful with this product guide:

1. *Vivado Design Suite User Guide: Designing IP Subsystems using IP Integrator* (UG994)
2. *Vivado Design Suite User Guide: Designing with IP* (UG896)
3. *Vivado Design Suite User Guide: Getting Started* (UG910)
4. *Vivado Design Suite User Guide: Logic Simulation* (UG900)
5. *Vivado Design Suite User Guide: Programming and Debugging* (UG908)
6. *ISE to Vivado Design Suite Migration Guide* (UG911)
7. *Vivado Design Suite User Guide: Implementation* (UG904)

Revision History

The following table shows the revision history for this document.

Section	Revision Summary
09/08/2020 v1.0	
Audio AXIS Interface	Added AXI descriptions
Chapter 6: Example Design	Updated core example design
06/07/2019 v1.0	
Implementing the Example Design	Updated the example design section
Simulating the Example Design	Updated the example design section
04/04/2018 v1.0	
Initial release.	N/A

Please Read: Important Legal Notices

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 https://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 https://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.

Copyright