ANKASYS Inter-IC Sound Bus (I2S) Verification IP

User Manual

VIP Version 0.1

January 2022

Table of Content

1. ANKASYS Verification IPs
    1.1 ANKASYS VIPs and Test Environment
    1.2.User Manual Text Formatting Convention
    1.3 Support
2. ANKASYS I2S VIP Overview
    2.1 I2S VIP Features
    2.2 Test Environment
    2.3 Directory Structure
3. ANKASYS I2S VIP Implementation
    3.1 Compiling ANKASYS I2S VIP
    3.2 Connecting I2S VIP with I2S DUT
4. ANKASYS I2S VIP Configuration
5. Simulate I2S VIP’s Sequence
6. Debug Results of VIP
    6.1 Error Logs
    6.2 Transaction Recording

1. ANKASYS Verification IPs

As digital designs are getting more complex, and production & fabrication and re-design costs are getting higher, industry needs more reliable/reusable verification approaches to verify the design.

UVM is one of the leading verification approaches, a standardized methodology to verify digital designs for FPGA/ASIC/SOC. UVM hierarchy has many components and the relations between these components are complex.

ANKASYS offers Verification IPs, so called ANKASYS VIPs. ANKASYS VIPs benefit from UVM features and they reduce the complexity of UVM hierarchy.

ANKASYS Verification IPs include both the universal verification component and corresponding integration with training services in a single pre-packaged library. Its ease and parametric structure enable verification engineers use customized VIPs and speed up verification process.

ANKASYS Verification IP Product Family includes

• ANKASYS BASE VIP

Protocol specific VIPs:

• ANKASYS SPI VIP
• ANKASYS I2C VIP
• ANKASYS UART VIP

ANKASYS Verification IPs are implemented in pure SystemVerilog language. No third-party language or script is needed for integration. The complete component hierarchy inside the verification IP and corresponding test codes (Sequences and tests) are all driven from UVM
standard library.

1.1 ANKASYS VIPs and Test Environment

UVM Test Environment is shown at Figure 1 and ANKASYS VIPs and Test Environment are shown at Figure 2.

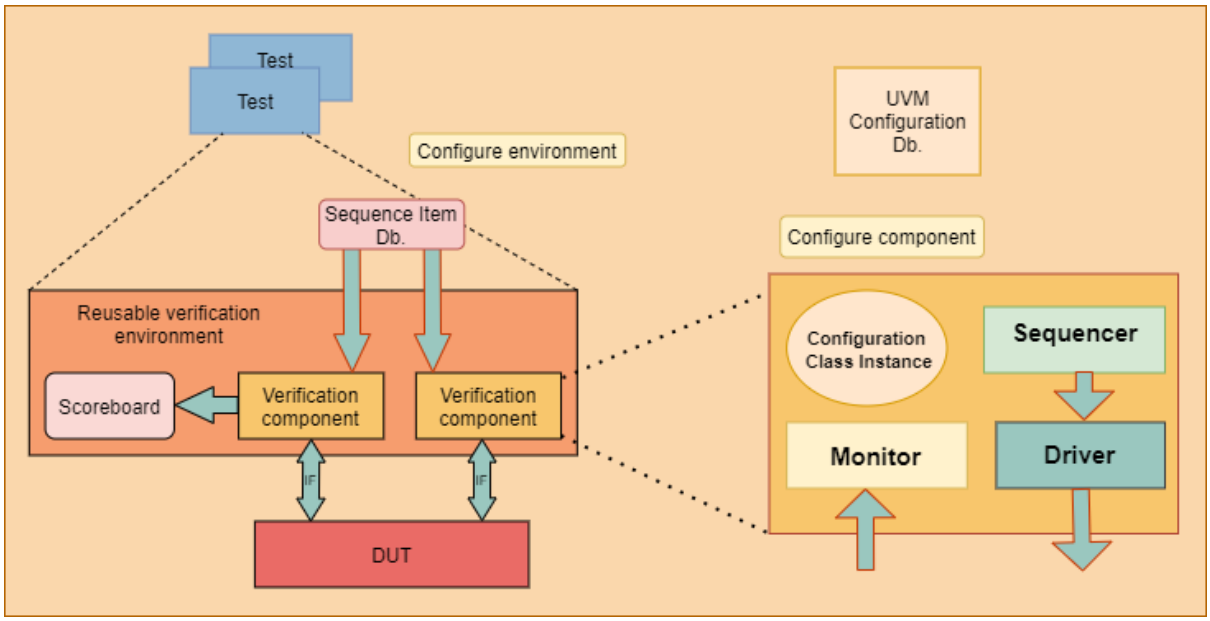

Figure 1

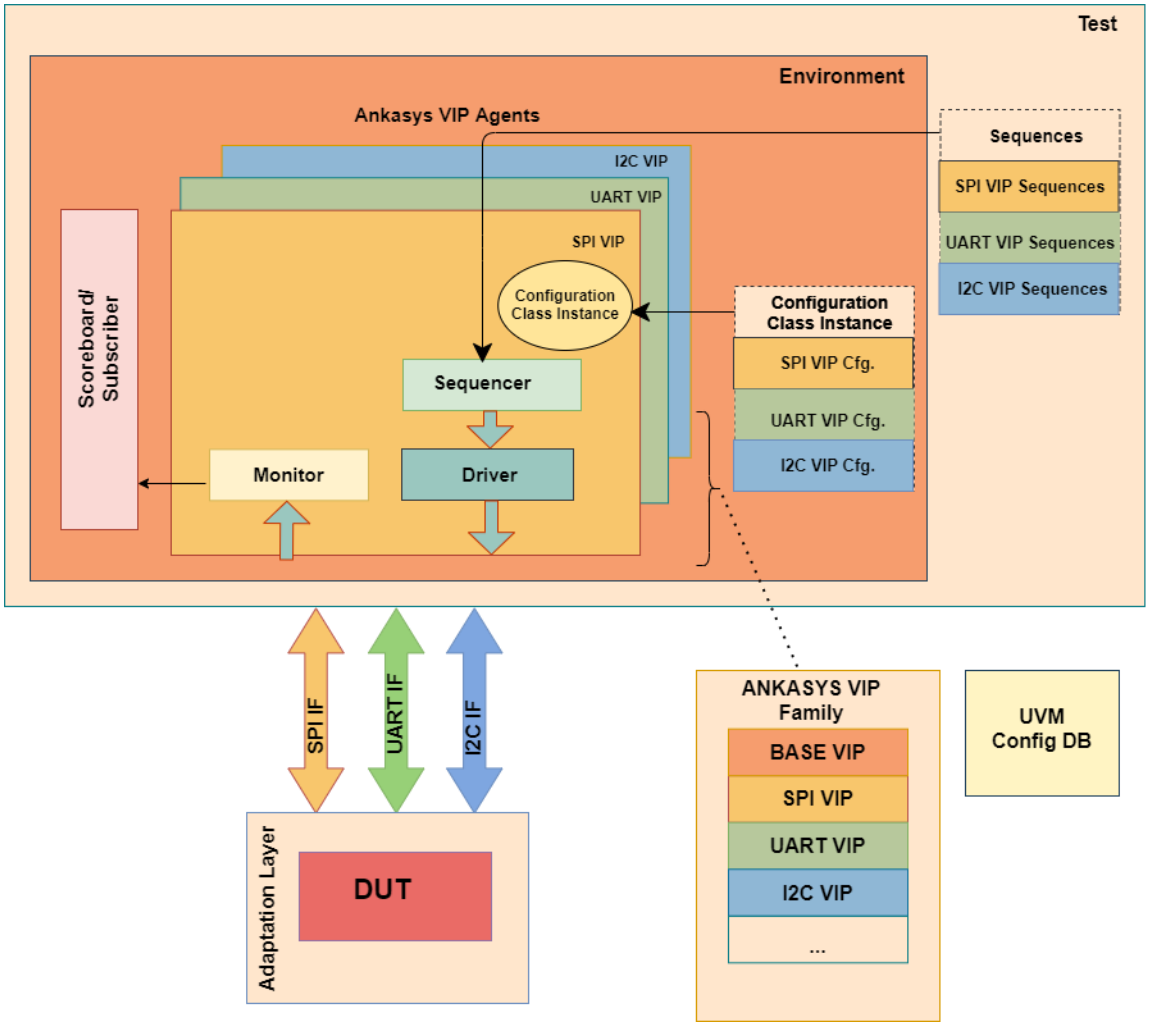

Figure 2

Each protocol specific Ankasys VIP is developed compatible to relevant protocol specification. Also, sequencer, sequence item, interface and analysis port are created for each VIP and example sequences are given. In addition, checking mechanism is supported by creating assertion and listeners for each of them.

1.2.User Manual Text Formatting Convention

Expression	Convention	Example
Code & Shell commands	Code samples and shell commands could be differentiated by their smaller font size than the size of general text.	logic spi_mosi; logic spi_miso

Emphasized Class or Module names	Emphasized Class and Module names are written in blue and bold style	**I2s_seq_item_rec**
Important notes	Important notes are written in	**Note: If an external clock ...**

1.3 Support

If you have support agreement with ANKASYS, all questions can be directed to vip@ankasys.com. Upon receiving the support request, a support specialist will contact you and tries to address your needs.

2.ANKASYS I2S VIP Overview

The Inter-IC Sound Bus (I2S) Verification IP allows you to verify synchronous and serial I2S protocol which is implemented to your design.

Using Ankasys I2S VIP which is derived from Ankasys Base VIP, you can easily construct a UVM testbench environment and thanks to configurable agent features, your I2S protocol will be verified with industry standard specifications.

2.1 I2S VIP Features

Ankasys I2S VIP has many industry compatible features with user friendly interface.

- 4 different modes of operation (half-duplex) :
    1. Slave Receiver
    2. Slave Transmitter
    3. Master Receiver
    4. Master Transmitter

- 5 different transfer formats below :
    5. Left-Justified

6. Right-Justified
7. Short Frame Sync, DSP or PCM
8. Long Frame Sync, PCM
9. The Division Multiplexed (TDM)

- Configurable data delay in Left-Justified, Short Frame Sync and TDM formats

- Configurable clock and frame polarity for Left-Justified and Right-Justified formats

- Support mono and stereo transferring modes for Short Frame and Long Frame Sync formats
- Configurable data length and LRCK length
- Configurable channel number for TDM mode

2.2. Test Environment

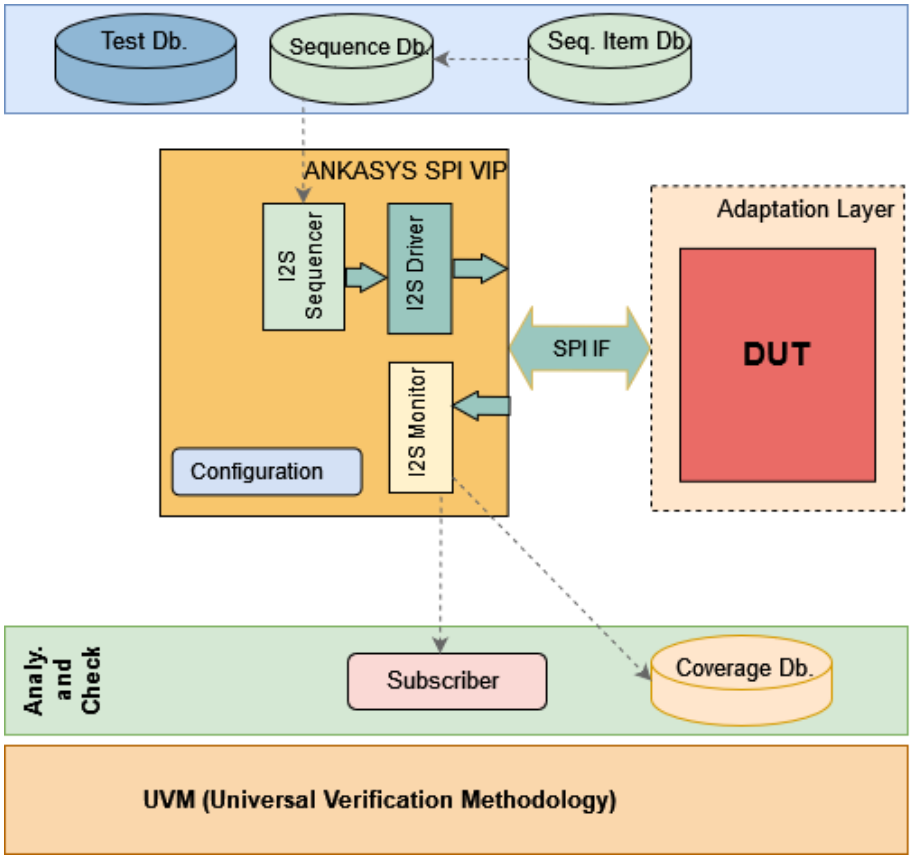

Figure 3

2.3. I2S VIP Directory Structure

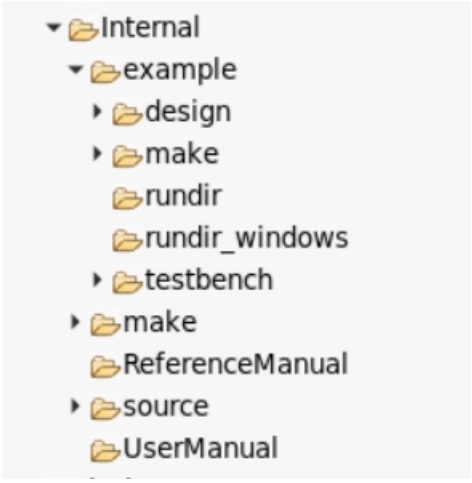

Figure 4

3.ANKASYS I2S VIP Implementation

3.1.Compiling ANKASYS I2S VIP

First of all, I2S VIP source files are compiled using given make file, makefile.mk

3.2. Connecting I2S VIP with I2S DUT

The interface signals are shown below :

Ports	Definition
SCLK	Clock, synchronizes transmitter and receiver
LRCK	Word Select or Left-Right Clock indicating the current channel (left or right)
SDIN	Data channel
reset	I2S system reset
m_clk	I2S system clock which set SCLK and LRCK

In order to eliminate confusion due to signal direction interface is not declared and used directly. Instead, wrapper modules are utilized to wrap interface instance, so that signals are assigned in a suitable manner. There are four wrapper modules defined as shown below,

i. **i2s_slave_receiver_wrapper**, VIP configured as a slave receiver connected to the master transmitter part of the DUT

ii. **i2s_master_receiver_wrapper**, VIP configured as a master receiver connected to the slave transmitter part of the DUT

iii. **i2s_slave_transmitter_wrapper**, VIP configured as a slave transmitter connected to the master receiver part of the DUT

iv. **i2s_master_transmitter_wrapper**, VIP configured as a master transmitter connected to the slave receiver part of the DUT

```
'define NUM_I2S_RX_SLAVE 1
'define NUM_I2S_TX_SLAVE 0
'define NUM_I2S_TX_MASTER 0
'define NUM_I2S_RX_MASTER 0
'define NUM_I2S ('NUM_I2S_RX_SLAVE + 'NUM_I2S_RX_MASTER + 'NUM_I2S_TX_SLAVE +
'NUM_I2S_TX_MASTER)
```

After defining the number of agent types (as shown above) which will be used, an example of the connection of interface to DUT in testbench is given below.

```
generate
      for(genvar i = 0; i < `NUM_In_RX_SLAVE; ++i) begin : gen_i2s_ifs_s_rx
            i2s_slave_receiver_wrapper m_i2s _slave receiver wrapper (
                  i2s_m_clk(),
                  .reset(resetnn),
                  .SCK(i2s_sck[i]),
                  .LRCK(i2s_ws[i]),
                  .SD(i2s_sd[i])
                  );
      initial begin
            string t_if_name;
            l_if_name = $psprintf("i2s_rx_slave_if_%0d", i);
            uvm_config_db#(virtual i2s_if)::set(null, "*", l_if_name,
      gen_i2s_ifs_s_rx[i].m_i2s_slave_receiver_wrapper.m_i2s_if);
     end
  end
endgenerate
AL AL_0(
            .RESET(resetn),
            .m_tx_sclk(i2s_sck[0]),
            .m_tx_lrclk(i2s_ws[0]),
            .m_tx_mclk(MCLK),
            .m_tx_sdata(i2s_sd[0]),
            .wave_left_in(wave_left),
             .wave_right_in(wave_right)
       );
```

An example run_phase of a test is given below:

```
task sample_i2s_test::run_phase(uvm_phase phase); super.run_phase(phase);
	phase.raise_objection(this);
	fork
		repeat(5) begin
			m_i2s _sample_sequence.randomize();
			m_i2s_sample_sequence.start(m_sample_env.m_i2s_agent.m_sequence
		r);
		end
	join_none
	#500ms;
phase.drop_objection(this);
endtask
```

4.ANKASYS I2S VIP Configuration

Ankasys I2S VIP configuration class provides a mechanism to control the behaviour of the VIP. The necessary assignments are made in **test_i2s** as the example below :

```
function void test_i2s::build_phase(uvm_phase phase);
	super. build_phase(phase);
	uvm config_db#(bit)::set(null,"*","enable", 1'bl) ;
	m_env_config.m_i2s_config[0].m_agent_type = I2S_S_RX  ;
	m_env_config.m_i2s_config[0].mode = timeDivMux ;
	m_env_config.m_i2s_config[0].mDelay = twoBitDelay ;
	m_env_config.m_i2s_config[0].mClockPol = inverseCP ;
	m_env_config.m_i2s_config[0].mFramePol = inverseFP ;
	m_env_config.m_i2s_config[0].mOperation = mono;
	m_env_config.m_i2s_config[0].dataLength = 24;
	m_env_config.m_i2s_config[0].lrckLength = 24;
	m_env_config.m_i2s_config[0].longFrameL = 23;
	m_env_config.m_i2s_config[0].mMonoChannel = left;
	m_env_config.m_i2s_config[0].tdmNumChannels_0 = 2;
	m_env_config.m_i2s_config[0].tdmNumChannels_1 = 1;
```

Configurable Variable	Description	Default Value	Possible values

mMode	Mode, Transfer format	leftJustified	leftJustified rightJustified shortFrame longFrame timeDivMux
mClockPol	Clock Polarity	inverseCP	inverseCP normalCP
mFramePol	Frame Polarity	inverseFP	inverseFP normalFP
mDelay	Number of Delay Bits	oneBitDelay	zeroBitDelay oneBitDelay twoBitDelay
dataLength	Length of the Data sent/ received	24	-
lrckLength	Length of the LRCK period	24	-
mOperation	Operation type	stereo	stereo mono
mMonoChannel	The channel in mono operation type	left	Left right
longFrameL	Length of theperiod where LRCK = 1, for longFrame mode	23	-
tdmNumChannels_0	Number of channels for LRCK = 0, in TDM mode	2	
tdmNumChannels_1	Number of channels for LRCK = 1, in TDM mode	1	
m_agent_type	Agent Type, Slave/ Master Receiver/ Transmitter	I2S_S_RX	I2S_S_RX I2S_S_TX I2S_M_RX I2S_M_TX

- mMode: Specifies the mode/ transfer format. The monitor and driver function depends on the mode selected. There are 5 different modes; left justified, right justified, short frame sync, long frame sync, the division multiplexed.
- mClockPol: The clock polarity. If the data is sent at the falling edge of SCLK, it indicates inverse clock polarity; if the data is sent at the rising edge of SCLK, it indicates normal clock polarity.
- mFramePol: The frame polarity. to represent the frame polarity (the channel (right or left) depends on LRCK

  Normal Frame Polarity -- when LRCK is 0, we are at the left channel ; when LRCK is 1, we are at the right channel

  Inverse Frame Polarity -- when LRCK is 0, we are at the right channel; when LRCK is 1, we are at the left channel
- mDelay: Enumeration to represent if data is ready at the start. (For adjusting delay)

  Value Range:

  zeroBitDelay - 0

  oneBitDelay - 1

  twoBitDelay - 2
- dataLength: Indicates the number of bits sent to one channel, the length of the data.
- lrckLength: Indicates the length of the LRCK's period; in other words, how many SLK cycles it takes for a LRCK cycle. When lrckLength is greater then the dataLength, '0' is sent for (lrckLength - dataLength) SCLK cycles.
- mOperation: This enumeration is for the short frame and long frame sync formats. It indicates the operation mode, if stereo both left and right channels are included; if mono only one of the channels are working.

- mMonoChannel: This enumeration is for the mono operation mode. It indicates which channel is operating, left or right.

- longFrameL: This integer is only for the Long Frame sync mode. It indicates how many SCLK cycles (bits) are in between LRCK posedge and LRCK negedge (in other words, LRCK = 1). Hence, between the negedge and posedge of LRCK (LRCK = 0) is (lrckLength-longFrameL) SCLK cycles long.

- tdmNumChannels_0: This integer specifies how many channels are used in TDM mode for when LRCK is 0.

- tdmNumChannels_1: This integer specifies how many channels are used in TDM mode for when LRCK is 1.
- m_agent_type: The agent type. For the selection of the agent specified for slave receiver, slave transmitter, master receiver or master transmitter types.

5.Simulate I2S VIP's Sequence

5.1.Sequence Items for I2S VIP

Table below shows Ankasys I2S Sequence items that are used to build sequences.

Sequence Item Name	Description
i2s_seq_item_base	Forms a basic structure for all other sequence items

i2s_seq_item_rec	This sequence item is for the monitor and the driver tasks for receiving data. It has two dynamic arrays (wave_left and wave_right) to store the serial data received by the monitor or the driver.
i2s_seq_item_tr	This sequence item is for the driver tasks which transmit data. It has two dynamic arrays (wave_left_in and wave_right_in) to be randomized and sent serially to a
i2s_seq_item_tdm_packet	This sequence item is only used for TDM mode. It has one dynamic array named wave_tdm.
i2s_seq_item_tdm_rec	This sequence item is only used in monitor and the driver tasks functioning as a receiver for TDM mode. It has a dynamic instance of i2s_seq_item_tdm_packet.
i2s_seq_item_tdm_tr	This sequence item is only used in the driver tasks functioning as a transmitter for TDM mode. It has a dynamic instance of i2s_seq_item_tdm_packet.

5.2.Sequences for I2S VIP

Ankasys I2S VIP has variety of built -in sequences that covers all possible scenarios and cases. Table below shows Ankasys I2S VIP built in sequences.

Sequence Name	Description
i2s_sequence_base	Forms a basic structure for all other sequences.
i2s_sequence_rec	This sequence is for the i2s_seq_item_rec to be created according to the m_config.
i2s_sequence_tr	This sequence is for the i2s_seq_item_tr to be created and randomized according to the m_config.

i2s_sequence_tdm_rec	This sequence is for the i2s_seq_item_tdm_rec to be created according to the m_config.
i2s_sequence_tdm_tr	This sequence is for the i2s_seq_item_tdm_tr to be created and randomized according to the m_config.

1.The instances of selected sequences are created inside test_i2s as shown below.

```
i2s_sequence_tdm_rec m_sequence_0;
i2s_sequence_tdm_tr m_sequence_1;
i2s_sequence_tdm_tr m_sequence_2;
i2s_sequence_tdm_rec m_sequence_3;
```

2.These instances are created under the run_phase task of **test_i2s** after the configurations are set, an example is given below :

```
task test_i2s::run_phase(uvm_phase phase);
	super.run_phase(phase);
	phase.raise_objection(this);
	m_sequence_0 = i2s_sequence_tdm_rec::type_id::create("m_sequence_0", this) ;
	m_sequence_1 = i2s_sequence_tdm_tr::type_id::create("m_sequence_1", this) ;
	m_sequence_2 = i2s_sequence_tdm_tr::type_id::create("m_sequence_2", this) ;
	m_sequence_3 = i2s_sequence_tdm_rec::type_id::create("m_sequence_3", this) ;
	fork
		m_sequence_0.start(m_anka_env.m_env.m_dut_agents.m_i2s_agent[0].m_sequ
	encer);
		m_sequence_1.start(m_anka_env.m_env.m_dut_agents.m_i2s_agent[1].m_sequ
	encer);
		m_sequence_2.start(m_anka_env.m_env.m_dut_agents.m_i2s_agent[2].m_sequ
	encer);
		m_sequence_3.start(m_anka_env.m_env.m_dut_agents.m_i2s_agent[3].m_sequ
	encer);
	join_none
	#500us;
	phase.drop_objection(this);
endtask
```

6.Debug Results of VIP

6.1.Error Logs

Inside driver and monitor, there are error and fatal logs giving info about ongoing I2S transfer in case of any error occurred.

Fatal errors:

"Failed to cast the requested transaction" - Indicates the transaction recording done in monitor task doesn’t match the expected type.

I2S_VIP: Casting error for Transmitter or Receiver sequence item!" - Indicates the sequence item created for the driver task is not the true type.

Errors :

"I2S_VIP: There can't be a delay in Right Justified mode"
”I2S_VIP: Short Frame Mode works only in inverse clock polarity"
"I2S_VIP: Short Frame Mode works only in inverse frame polarity"
"I2S_VIP: Long Frame Mode works only in inverse clock polarity"
"I2S_VIP: Long Frame Mode works only in inverse frame polarity"
"I2S_VIP: There can't be a delay in Long Frame mode"
"I2S_VIP: There can't be a delay in Right Justified mode"
"I2S_VIP: Only inverse clock and lrck polarisation in TDM mode"
"I2S_VIP: Long Frame Mode, frame length error"

6.2.Transaction Recording

ANKASYS I2S VIP has transaction recording feature which enables user to see transaction attributes on the waveform easily. For now, only Mentor QUESTASIM simulator is supported. In order to add transaction attribute view on the waveform of the simulator, transaction record interface instance should be added into waveform as shown figures below.

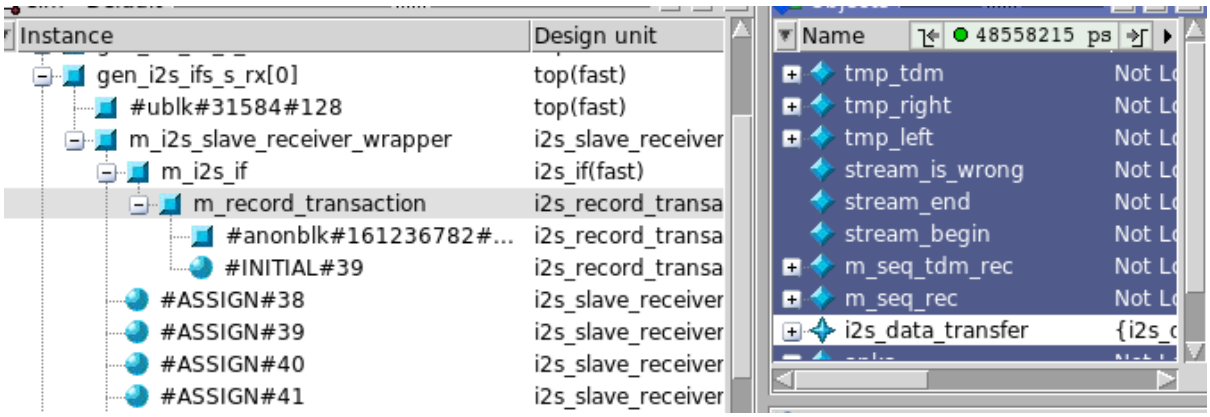

Figure 5

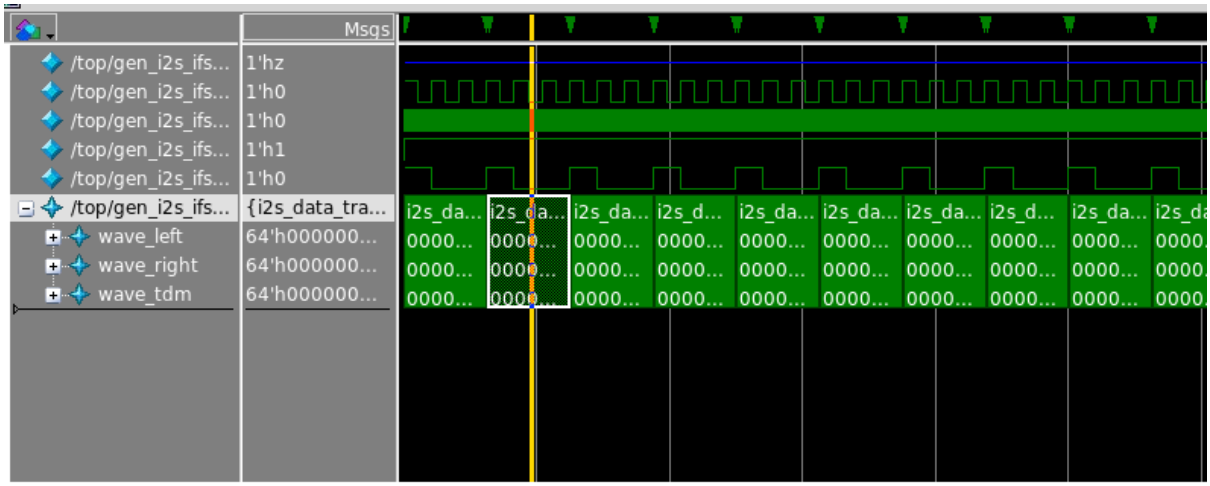

Figure 6