AXI4-Stream BFM – Quick Reference

axistream_transmit (data_array, [user_array], msg, clk, axistream_if, [scope, [msg_id_panel, [config]]])

Example (tdata'length = 16) : axistream_transmit ((x"D0", x"D1", x"D2", x"D3"), (x"00", x"0A"), "Send a 4 byte packet with tuser=A at the 2nd (last) word", clk, axistream_if);
Example (tdata'length = 8) : axistream_transmit ((x"D0", x"D1", x"D2", x"D3"), (x"00", x"00", x"00", x"0A"), "Send a 4 byte packet with tuser=A at the 4th (last) word", clk, axistream_if);

Example: axistream_transmit(v_data_array(0 to 1), "Send two bytes", clk, axistream_if_m, C_SCOPE, shared_msg_id_panel, axistream_bfm_config);
Example: axistream_transmit(v_data_array(0 to v_numBytes-1), v_user_array(0 to v_numWords-1), "Send two bytes", clk, axistream_if_m, C_SCOPE, shared_msg_id_panel, axistream_bfm_config);

***Suggested usage**: axistream_transmit((v_data_array(0 to 1), v_user_array(0 to v_numWords-1), "send two bytes"); -- Suggested usage requires local overload (see section 5)*

axistream_receive (data_array, data_length, user_array, msg, clk, axistream_if, [scope, [msg_id_panel, [config, [proc_name]]]])

Example: axistream_receive(v_received_data_array, v_received_length, v_received_user_array, "Receive packet", clk, axistream_if);

BFM

axistream_bfm_pkg.v

axistream_expect (exp_data_array, [exp_user_array], alert_level, msg, clk, axistream_if, [scope, [msg_id_panel, [config]]])

Example (tdata'length = 16) : axistream_expect((x"D0", x"D1", x"D2", x"D3"), (x"00", x"0A"), "Expect a 4 byte packet with tuser=A at the 2nd (last) word", clk, axistream_if);
Example (tdata'length = 8) : axistream_expect((x"D0", x"D1", x"D2", x"D3"), (x"00", x"00", x"00", x"0A"), "Expect a 4 byte packet with tuser=A at the 4th (last) word", clk, axistream_if);
Example: axistream_expect(v_data_array(0 to 1), "Expect a 2 byte packet, ignoring the tuser bits", clk, axistream_if);
Example: axistream_expect(v_data_array(0 to v_numBytes-1), v_user_array(0 to v_numWords-1), "Expect a packet, also check the tuser bits", clk, axistream_if);

***Suggested usage**: axistream_expect((v_data_array(0 to 1), v_user_array(0 to v_numWords-1), "Expect a packet"); -- Suggested usage requires local overload (see section 5)*

init_axistream_if_signals (is_master, data_width, user_width)

```
Example:  axistream_if <= init_axistream_if_signals(is_master  => true,          -- When the BFM is master
                                                    data_width => axistream_if.tdata'length,
                                                    user_width => axistream_if.tuser'length);
```

BFM Configuration record ´**t_axistream_bfm_config**´

Name	Type	C_AXISTREAM_BFM_CONFIG_DEFAULT
max_wait_cycles	natural	10
max_wait_cycles_severity	t_alert_level	ERROR
clock_period	time	0 ns
check_packet_length	boolean	false
protocol_error_severity	t_alert_level	ERROR
ready_low_at_word_num	integer	0
ready_low_duration	integer	0
ready_default_value	std_logic	'0'
id_for_bfm	t_msg_id	ID_BFM
id_for_bfm_wait	t_msg_id	ID_BFM_WAIT
id_for_bfm_poll	t_msg_id	ID_BFM_POLL

Signal record ´**t_axistream_if**´

Name	Type
tdata	std_logic_vector
tkeep	std_logic_vector
tuser	std_logic_vector
tvalid	std_logic
tlast	std_logic
tready	std_logic

VHDL 2008 only

BFM non-signal parameters

Name	Type	Example(s)	Description
data_array	t_slv8_array	x"D0" & x"D1"	A byte array containing the packet data to be sent or the data received. Regardless of the width of axistream_if.tdata, each data_array entry is 8-bit wide. data_array(0) is sent/received first. data_array(data_array'high) is sent/received last. For clarity, data_array is required to be ascending: For example defined by the test sequencer as follows : `variable v_data_array : t_slv8_array(0 to c_max_bytes-1);`
user_array	t_user_array	x"01" & x"02"	Sideband data to send or has been received via the tuser signal. The number of entries in user_array equals the number of data words to be transmitted (i.e. number of clock cycles). If 16 bytes shall be sent, and there are 8 bytes transmitted per word, the user_array has 2 entries. The number of bits actually used in each user_array entry depends on the width of axistream_if.tuser. Note: If axistream_if.tuser is wider than 8, increase the value of the constant c_max_tuser_bits in axistream_bfm_pkg. This is where the t_user_array is defined.
data_length	natural	2	The number of bytes received, i.e. the number of valid bytes in dataArray.
alert_level	t_alert_level	ERROR or TB_WARNING	Set the severity for the alert that may be asserted by the procedure.
msg	string	"Set state active on peripheral 1"	A custom message to be appended in the log/alert.
scope	string	"AXISTREAM BFM"	A string describing the scope from which the log/alert originates. In a simple single sequencer typically "AXISTREAM BFM". In a verification component typically "AXISTREAM_VVC ".
msg_id_panel	t_msg_id_panel	shared_msg_id_panel	Optional msg_id_panel, controlling verbosity within a specified scope. Defaults to a common message ID panel defined in the UVVM-Util adaptations package.
config	t_axistream_bfm_config	C_AXISTREAM_BFM_CONFIG_DEFAULT	Configuration of BFM behaviour and restrictions. See section 0 for details.

BFM signal parameters

Name	Type	Description
clk	std_logic	The clock signal used to read and write data in/out of the AXI4-Stream BFM.
axistream_if	t_axistream_if	See table "Signal record 't_axistream_if'". Note: the `tuser` element must be present even when it is not used (connected to DUT). If the record signal connects to a Slave BFM, where the tuser element is an input, assign a dummy value: `axistream_if.tuser <= (others => '0');` Regardless, the tuser check will be skipped when the test sequencer calls axistream_expect() without providing the user_array argument.

For more information on the AXI4-Stream signals, refer to "AMBA® 4 AXI4-Stream Protocol Specification", document number ARM IHI 0051A (ID030510), available from ARM

BFM features

This BFM supports the following subset of the AXI4-Stream protocol :
- Continuous aligned stream, as described in chapter 1.2.2 in AMBA 4 AXI4-Stream protocol Specification (ARM IHI 0051A)

The following subset of the AXI4-Stream interface is supported:

Signal	Source	Width	Supported by BFM	Description
ACLK	Clock	1	Yes	Sample on the rising edge
ARESETn	Reset	-	No	BFM doesn't control the reset.
TVALID	Master	1	Yes	A transfer takes place when both TVALID and TREADY are asserted
TREADY	Slave	1	Yes	A transfer takes place when both TVALID and TREADY are asserted
TDATA	Master	n*8	Yes	Data word. The width must be a multiple of bytes.
TSTRB	Master	-	No	Mark Position Byte. Not typically used. For marking packet remainders, TKEEP rather than TSTRB is used.
TKEEP	Master	TDATA'length/8	Partly	When TKEEP is '0', it indicates a null byte that can be removed from the stream. The same limitations apply for this BFM as in the *Xilinx ug761 AXI Reference Guide*: Null bytes are only used for signalling the number of valid bytes in the last data word. Leading or intermediate Null bytes are not supported.
TLAST	Master	1	Yes	When '1', it indicates that the tdata is the last word of the packet.
TID	Master	-	No	Indicates different streams of data. Usually used by routing infrastructures
TDEST	Master	-	No	Provides routing info. Usually used by routing infrastructures
TUSER	Master	1:c_max_tuser_bits	Yes	Sideband info transmitted alongside the data stream. Note 1 : The TUSER signal must exist in the axistream_if record even when it is not used by DUT. See the "BFM signal parameters" table above. Note 2: If axistream_if.tuser is wider than c_max_tuser_bits in axistream_bfm_pkg, increase the value of the constant c_max_tuser_bits.

BFM details

1 BFM procedure details

Procedure	Description
axistream_transmit()	**axistream_transmit (data_array, [user_array], msg, clk, axistream_if, [scope, [msg_id_panel, [config]]])** The axistream_transmit () procedure transmits a packet on the AXI interface. The packet length and data are defined by the “data_array” argument. data_array is a byte array. One byte is sent per data_array entry, but multiple bytes may be sent each word (clock cycle). data_array(0) is sent first. data_array(data_array’high) is sent last. Byte locations within the data word are defined in chapter 2.3 in “AMBA® 4 AXI4-Stream Protocol Specification”, document number ARM IHI 0051A (ID030510), available from ARM. The side band data signal TUSER is defined by the optional user_array argument. If user_array is omitted, all the TUSER bits are zero during every data word transmitted. At the last word, the BFM asserts the TLAST bit, and it asserts the TKEEP bits corresponding to the data bytes that are valid within the word. At all other words, all TKEEP bits are ‘1’, thus the BFM supports only “continuous aligned stream”, as described in chapter 1.2.2 in AMBA 4 AXI4-Stream protocol Specification (ARM IHI 0051A).
axistream_receive()	**axistream_receive(data_array, data_length, user_array, msg, clk, axistream_if, [scope, [msg_id_panel, [config]]])** The axistream_receive() procedure receives a packet on the AXI interface. The received packet data is stored in the data_array output, which is a byte array. data_array'length can be longer than the actual packet received, so that you can call receive() without knowing the length to be expected. The packet length (number of bytes received) is indicated in the packet_length output. The side band data signal *tuser* is stored in user_aray, which has one entry per data word (clock cycle). When TLAST = ‘1’ the TKEEP bits are used to determine the number of valid data bytes within the last word. At all other words, the BFM checks that all TKEEP bits are ‘1’, since the BFM supports only “continuous aligned stream” described in chapter 1.2.2 in AMBA 4 AXI4-Stream protocol Specification (ARM IHI 0051A)
axistream_expect()	**axistream_expect(exp_data_array, [exp_user_array], alert_level, msg, clk, axistream_if, [scope, [msg_id_panel, [config]]])** Calls the axistream_receive() procedure, then compares the received data with exp_data_array. If exp_user_array is provided as an argument, the exp_user_array is compared to the received user_array.
init_axistream_if_signals()	**init_axistream_if_signals(is_master, data_width, user_width)** This function initializes the AXI4-Stream interface. All the BFM outputs are set to zeros ('0')

2 BFM Configuration record

Type name: t_axistream_bfm_config

Name	Type	C_AXISTREAM_BFM_CONFIG_DEFAULT	Description
max_wait_cycles	natural	10	Used for setting the maximum cycles to wait before an alert is issued when waiting for ready or valid signals from the DUT.
max_wait_cycles_severity	t_alert_level	failure	The above timeout will have this severity
clock_period	time	10 ns	Period of the clock signal.
check_packet_length	boolean	false	When true, receive() will check that tlast is set at dataArray'high. Set to false when length of packet to be received is unknown.
protocol_error_severity	t_alert_level	ERROR	severity if protocol errors are detected
ready_low_at_word_num	integer	0	When the Slave BFM shall deassert ready while receiving the packet
ready_low_duration	integer	0	Number of clock cycles to deassert ready
ready_default_value	std_logic	'0'	Determines the ready output value while the Slave BFM is idle
id_for_bfm	t_msg_id	ID_BFM	The message ID used as a general message ID in the BFM
id_for_bfm_wait	t_msg_id	ID_BFM_WAIT	The message ID used for logging waits in the BFM
id_for_bfm_poll	t_msg_id	ID_BFM_POLL	The message ID used for logging polling in the BFM

3 Additional Documentation

For additional documentation on the AXI4-Stream standard, refer to "AMBA® 4 AXI4-Stream Protocol Specification", document number ARM IHI 0051A (ID030510), available from ARM.

4 Compilation

The AXI4-Stream BFM may only be compiled with VHDL 2008. It is dependent on the UVVM Utility Library (UVVM-Util), which is only compatible with VHDL 2008.
See the separate UVVM-Util documentation for more info. After UVVM-Util has been compiled, the axistream_bfm_pkg.vhd BFM can be compiled into any desired library.

4.1 Simulator compatibility and setup

This BFM has been compiled and tested with Modelsim version 10.3d and Riviera-PRO version 2015.10.85.

For required simulator setup see UVVM-Util Quick reference.

5 Local BFM overloads

A good approach for better readability and maintainability is to make simple, local overloads for the BFM procedures in the TB process.
This allows calling the BFM procedures with the key parameters only
e.g.

axistream_transmit(v_data_array(0 to 1), v_user_array(0), “Send two bytes”);

rather than

axistream_transmit(v_data_array(0 to 1), v_user_array(0), "Send two bytes", clk, axistream_if_m, C_SCOPE, shared_msg_id_panel, axistream_bfm_config);

By defining the local overload as e.g.:

```
procedure axistream_transmit (
  constant data_array   : in t_slv8_array;
  constant user_array   : in t_user_array;
  constant msg          : in string) is
begin
    axistream_transmit(data_array,                          -- keep as is
                       user_array,                          -- keep as is
                       msg,                                 -- keep as is
                       clk,                                 -- Clock signal
                       axistream_if,                        -- Signal must be visible in local process scope
                       C_SCOPE,                             -- Just use the default
                       shared_msg_id_panel,                 -- Use global, shared msg_id_panel
                       C_AXISTREAM_BFM_CONFIG_LOCAL);       -- Use locally defined configuration or C_AXISTREAM_BFM_CONFIG_DEFAULT
end;
```

Using a local overload like this also allows the following – if wanted:

- Set up defaults for constants. May be different for two overloads of the same BFM
- Apply dedicated message_id_panel to allow dedicated verbosity control

IMPORTANT
This is a simplified Bus Functional Model (BFM) for AXI4-Stream. The given BFM complies with the basic AXI4-Stream protocol and thus allows a normal access towards an AXI4-Stream interface. This BFM is not AXI4-Stream protocol checker. For a more advanced BFM please contact Bitvis AS at support@bitvis.no