AXI4-Stream BFM – Quick Reference

AXI4-Stream Master (see page 2 for AXI4-Stream Slave)

axistream_transmit[_bytes] (data_array, [user_array, [strb_array, id_array, dest_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 v_numBytes-1), "Send v_numBytes 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)(16 downto 0), "Send 2 x v_numBytes 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 v_numBytes 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), v_strb_array(0 to v_numWords-1), v_id_array(0 to v_numWords-1), v_id_array(0 to v_numWords-1), "Send" ….

Note! Use axistream_transmit_bytes ( ) when using t_byte_array.

BFM

axistream_bfm_pkg.vhd

init_axistream_if_signals (is_master, data_width, user_width, id_width, dest_width)

Example: axistream_if <= init_axistream_if_signals(true, axistream_if.tdata'length, axistream_if.tuser'length, axistream_if.tid'length, axistream_if.tdest'length);

UVVM™

VHDL 2008 only

AXI4-Stream BFM – Quick Reference

AXI4-Stream Slave (see page 1 for AXI4-Stream Master)

BFM

axistream_bfm_pkg.vhd

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

Example: axistream_receive(v_rx_data_array, v_rx_length, v_rx_user_array, v_rx_strb_array, v_rx_id_array, v_rx_dest_array, “Receive packet”, clk, axistream_if);

Note! Use axistream_receive_bytes ( ) when using t_byte_array.

axistream_expect[_bytes] (exp_data_array, [exp_user_array, [exp_strb_array, exp_id_array, exp_dest_array]], msg, clk, axistream_if, [alert_level, [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, check data and tuser, but ignore tstrb, tid, tdest”, clk, axistream_if);
Example: axistream_expect(v_data_array(0 to v_numBytes-1), v_user_array(0 to v_numWords-1), v_strb_array(0 to v_numWords-1), v_id_array(0 to v_numWords-1), v_id_array(0 to v_numWords-1), “Expect packet”, clk, axistream_if);

Note! Use axistream_expect_bytes ( ) when using t_byte_array.

init_axistream_if_signals (is_master, data_width, user_width, id_width, dest_width)

Example: axistream_if <= init_axistream_if_signals(false, axistream_if.tdata'length, axistream_if.tuser'length, axistream_if.tid'length, axistream_if.tdest'length);

UVVM™

BFM Configuration record ´**t_axistream_bfm_config**´

Record element	Type	C_AXISTREAM_BFM_CONFIG_DEFAULT
max_wait_cycles	natural	10
max_wait_cycles_severity	t_alert_level	ERROR
clock_period	time	0 ns
clock_period_margin	time	0 ns
clock_margin_severity	t_alert_level	TB_ERROR
setup_time	time	0 ns
hold_time	time	0 ns
byte_endianness	t_byte_endianness	FIRST_BYTE_LEFT
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

Record element	Type
tdata	std_logic_vector
tkeep	std_logic_vector
tuser, tstrb, tid, tdest	std_logic_vector
tvalid	std_logic
tlast	std_logic
tready	std_logic

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'" in page 1 and 2. Note: All supported signals, including `tuser, tstrb, tid, tdest` are included in the record type, even when not used or connected to DUT.

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 non-signal parameters

Name	Type	Example(s)	Description

data_array	t_byte_array,
	t_slv_array or
	std_logic_vector
		x"D0" & x"D1"
		(x"D0D1", x"D2D3")
		x"D0D1"
			An array of bytes, SLVs or a single SLV containing the packet data to be sent.
			Note the name change in procedure calls when using t_byte_array.
			Regardless of the width of axistream_if.tdata, each data_array entry is 8-bit wide, unless t_slv_array or slv is used. When data_array entry is a single SLV or an array, an overloading procedure will convert data_array into an array of bytes.
			
			data_array(0) is sent/received first, while data_array(data_array'high) is sent/received last. Note that for slv and t_slv_array, the 8 upper bits in the data word is sent/received first, and the 8 lower bits are 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_byte_array(0 to C_MAX_BYTES-1);`
			`variable v_slv_data_array : t_slv_array(0 to C_MAX_BYTES-1)(C_MAX_WORD_LENGTH-1 downto 0);`

exp_data_array	t_byte_array, t_slv_array or std_logic_vector	x"D0" & x"D1" (x"D0D1", x"D2D3") x"D0D1"	An array of bytes, SLVs or a single SLV containing the packet of data that is expected to be received. The data_array specifications listed above applies for exp_data_array as well.
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, i.e. transfers[1]. For example, if 16 bytes shall be sent, and there are 8 bytes transmitted per transfer, the user_array has 2 entries. The number of bits actually used in each user_array entry corresponds to 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.
strb_array	t_strb_array	"00" & "10"	Sideband data to send or has been received via the TSTRB signal. The BFM transmits/receives the values without affecting TDATA. The number of entries in this array equals the number of data words, i.e. transfers[1]. The number of bits actually used in each array entry corresponds to the width of axistream_if.TSTRB. Note: If axistream_if.TSTRB is wider than 32, increase the value of the constant C_MAX_TSTRB_BITS in axistream_bfm_pkg.
id_array	t_id_array	x"01" & x"02"	Sideband data to send or has been received via the TID signal. The number of entries in this array equals the number of data words, i.e. transfers[1]. The number of bits actually used in each array entry corresponds to the width of axistream_if.TID. Note: If axistream_if.TID is wider than 8, increase the value of the constant C_MAX_TID_BITS in axistream_bfm_pkg.
dest_array	t_dest_array	x"1" & x"2"	Sideband data to send or has been received via the TDEST signal. The number of entries in this array equals the number of data words, i.e. transfers[1]. The number of bits actually used in each array entry corresponds to the width of axistream_if.TDEST. Note: If axistream_if.TDEST is wider than 4, increase the value of the constant C_MAX_TDEST_BITS in axistream_bfm_pkg.
data_length	natural	2	The number of bytes received, i.e. the number of valid bytes in data_array.
alert_level	t_alert_level	ERROR or TB_WARNING	Set the severity for the alert that may be asserted by the procedure.
msg	string	"Send packet"	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 2 for details.

[1] In AXI4-Stream, a transfer is defined as a TVALID/TREADY handshake.

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 signals are 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[2]	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.
TUSER	Master	1:c_max_tuser_bits	Yes[2]	Sideband info transmitted alongside the data stream. If axistream_if.tuser is wider than c_max_tuser_bits in axistream_bfm_pkg, increase the value of the latter.
TSTRB	Master	1:c_max_tstrb_bits	Yes[2]	The protocol uses this signal for marking TDATA as position byte, but the BFM simply sends/receives/checks the values of TSTRB as specified by the sequencer without affecting TDATA: While transmitting, the test sequencer defines what TSTRB values to send. The BFM transmits TDATA regardless of the TSTRB value. While receiving, the received TSTRB values are presented to the test sequencer. The BFM presents TDATA regardless of the TSTRB value. If axistream_if.tstrb is wider than c_max_tstrb_bits in axistream_bfm_pkg, increase the value of the latter.
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	1:c_max_tid_bits	Yes[2]	Indicates different streams of data. Usually used by routing infrastructures. When BFM is transmitting, the test sequencer defines what TID values to send. When BFM is receiving, the received TID values are presented to the test sequencer. If axistream_if.tid is wider than c_max_tid_bits in axistream_bfm_pkg, increase the value of the latter
TDEST	Master	1:c_max_tdest_bits	Yes[2]	Provides routing info for the data stream. Usually used by routing infrastructures When BFM is transmitting, the test sequencer defines what TDEST values to send. When BFM is receiving, the received TDEST values are presented to the test sequencer. If axistream_if.tdest is wider than c_max_tdest_bits in axistream_bfm_pkg, increase the value of the latter

[2] Although defined as optional in the AXI4-Stream protocol, the signal must exist in the axistream_if record, even when not used / connected to DUT.

BFM details

1 BFM procedure details

Procedure	Description
axistream_transmit[_bytes]()	**axistream_transmit[_bytes] (data_array, [user_array, [strb_array, id_array, dest_array]], msg, clk, axistream_if, [scope, [msg_id_panel, [config]]])** The axistream_transmit () procedure transmits a packet on the AXI interface. Note that axistream_transmit_bytes ( ) has to be used for t_byte_array data_array. The packet length and data are defined by the "data_array" argument, and is either a byte array, a t_slv_array or a SLV. If a t_slv_array or a SLV is used an overloading procedure will convert data_array to an array of bytes. One byte is sent per data_array entry, but multiple bytes may be sent on each transfer (word). data_array(0) is sent first. data_array(data_array'high) is sent last. In a t_slv_array and a SLV, the upper 8 bits are sent first and the lower 8 bits are 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 values to be transmitted on the signal TUSER is defined by the optional user_array parameter. There is one user_array index per transfer (data word). If user_array is omitted in the BFM call, the BFM transmits all zeros on the TUSER signal. The values to be transmitted on the signals TSTRB, TID, TDEST are defined by the parameters strb_array, id_array and dest_array. There is one array index per transfer (data word). All or none of these three arrays may be omitted in the BFM call. If they are omitted, the BFM transmits all zeros on the TSTRB, TID, TDEST signals. 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[_bytes]()	**axistream_receive[_bytes] (data_array, data_length, user_array, strb_array, id_array, dest_array, msg, clk, axistream_if, [scope, [msg_id_panel, [config]]])** The axistream_receive() procedure receives a packet on the AXI interface. Note that axistream_receive_bytes ( ) has to be used for t_byte_array data_array. 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 number of bytes received is indicated in the packet_length output. The sampled values of the TUSER signal are stored in user_array, which has one entry per transfer (data word). The sampled values of the TSTRB signal are stored in strb_array, which has one entry per transfer (data word). The sampled values of the TID signal are stored in id_array, which has one entry per transfer (data word). The sampled values of the TDEST signal are stored in dest_array, which has one entry per transfer (data word). 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[_bytes]()	**axistream_expect[_bytes] (exp_data_array, [exp_user_array, [exp_strb_array, exp_id_array, exp_dest_array]], msg, clk, axistream_if, [alert_level, [scope, [msg_id_panel, [config]]]])** Calls the axistream_receive() procedure, then compares the received data with exp_data_array. Note that axistream_expect_bytes ( ) has to be used for t_byte_array exp_data_array. Note that if exp_data_array is a t_slv_array or slv, an overload will convert it to t_byte_array. The exp_user_array, exp_strb_array, exp_id_array, exp_dest_array are compared to the received user_array, strb_array, id_array and dest_array respectively. If some signals are unused, the checks can by skipped by filling the corresponding exp_*_array with don't cares. For example: `v_dest_array := (others => (others => '-'));`
init_axistream_if_signals()	**init_axistream_if_signals(is_master, data_width, user_width, id_width, dest_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

Record element	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	ERROR	The above timeout will have this severity
clock_period	time	0 ns	Period of the clock signal. Default is 0 ns to detect if not set by user.
clock_period_margin	time	0 ns	Input clock period margin to specified clock_period
clock_margin_severity	t_alert_level	TB_ERROR	The above margin will have this severity
setup_time	time	0 ns	Setup time for generated signals. Suggested value is clock_period/4. An alert is reported if setup_time exceed clock_period/2.
hold_time	time	0 ns	Hold time for generated signals. Suggested value is clock_period/4. An alert is reported if hold_time exceed clock_period/2.
byte_endianness	t_byte_endianness	FIRST_BYTE_LEFT	Little-endian or big-endian endianness byte ordering.
check_packet_length	boolean	false	When true, receive() will check that tlast is set at data_array'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), "msg");
```

rather than

```
axistream_transmit(v_data_array(0 to 1), "msg", clk, axistream_if_m, C_SCOPE, shared_msg_id_panel, axistream_bfm_config);
```

By defining the local overload as e.g.:

```
procedure axistream_transmit_bytes (
  constant data_array   : in t_byte_array;
  constant msg          : in string) is
begin
    axistream_transmit_bytes(data_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