

AXI4-Stream VVC – Quick Reference

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

axistream_transmit[_bytes] (VVCT, vvc_instance_idx, data_array, [user_array, [strb_array, id_array, dest_array]], msg)

Example: axistream_transmit(AXISTREAM_VVCT, 0, v_data_array(0 to v_numBytes-1), v_user_array(0 to v_numWords-1), " Send a 'v_numBytes' byte packet to DUT");

Example: axistream_transmit(AXISTERAM_VVCT, 0, v_data_array(0 to v_numBytes-1)(31 downto 0), v_user_array(0 to v_numWords-1), "Send a '4 x v_numBytes' byte packet to DUT");

*Note that this procedure can only be called when the AXISTREAM VVC is instantiated in master mode, i.e. setting the generic constant 'GC_MASTER_MODE' to true.
See page 5 for additional information.*

Note! Use axistream_transmit_bytes () when using t_byte_array.

VVC



axistream_vvc.vhd



UVVM™

AXI4-Stream VVC – Quick Reference

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

axistream_receive[_bytes] (VVCT, vvc_instance_idx, msg)

Example: axistream_receive (AXISTREAM_VVCT, 1, "Receive packet, which is stored in VVC and will be fetched later using fetch_result() ");

Note that this procedure can only be called when the AXISTREAM VVC is instantiated in slave mode, i.e. setting the generic constant 'GC_MASTER_MODE' to false. See page 6 for additional information.

Note! Use axistream_receive_bytes () when using t_byte_array.

VVC



axistream_vvc.vhd

axistream_expect[_bytes] (VVCT, vvc_instance_idx, exp_data_array, [exp_user_array, [exp_strb_array, exp_id_array, exp_dest_array]], msg, [alert_level])

Example: axistream_expect (AXISTREAM_VVCT, 0, v_data_array(0 to v_numBytes-1), v_user_array(0 to v_numWords-1), "Expect a packet, checking the tuser bits");

Example: axistream_expect (AXISTREAM_VVCT, 0, v_data_array(0 to v_numBytes-1)(16 downto 0), v_user_array(0 to v_numWords-1), "Expecting a packet, checking the tuser bits");

Note that this procedure can only be called when the AXISTREAM VVC is instantiated in slave mode, i.e. setting the generic constant 'GC_MASTER_MODE' to false. See page 5 and 6 for additional information.

Note! Use axistream_expect_bytes () when using t_byte_array



UVVM™

AXI4-Stream VVC Configuration record 'vvc_config' -- accessible via **shared_axistream_vvc_config**

| Record element | Type | C_AXISTREAM_VVC_CONFIG_DEFAULT |
|------------------------------------|------------------------|--------------------------------------|
| inter_bfm_delay | t_inter_bfm_delay | C_AXISTREAM_INTER_BFM_DELAY_DEFAULT |
| cmd_queue_count_max | natural | C_CMD_QUEUE_COUNT_MAX |
| cmd_queue_count_threshold | natural | C_CMD_QUEUE_COUNT_THRESHOLD |
| cmd_queue_count_threshold_severity | t_alert_level | C_CMD_QUEUE_COUNT_THRESHOLD_SEVERITY |
| bfm_config | t_axistream_bfm_config | C_AXISTREAM_BFM_CONFIG_DEFAULT |
| msg_id_panel | t_msg_id_panel | C_VVC_MSG_ID_PANEL_DEFAULT |

AXI4-Stream VVC Status record signal 'vvc_status' -- accessible via **shared_axistream_vvc_status**

| Record element | Type |
|------------------|---------|
| current_cmd_idx | natural |
| previous_cmd_idx | natural |
| pending_cmd_cnt | natural |

Common VVC procedures applicable for this VVC

- See UVVM Methods QuickRef for details.

await_[any_]completion()
enable_log_msg()
disable_log_msg()
fetch_result()
flush_command_queue()
terminate_current_command()
terminate_all_commands()
insert_delay()
get_last_received_cmd_idx()

VVC target parameters

| Name | Type | Example(s) | Description |
|------------------|---------------------|----------------|--|
| VVCT | t_vvc_target_record | AXISTREAM_VVCT | VVC target type compiled into each VVC in order to differentiate between VVCs. |
| vvc_instance_idx | integer | 0 | Instance number of the VVC |

VVC functional 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" | A byte array, SLV array or a single SLV containing the packet data to be sent or the data received. Note the name change when data_array is t_byte_array. SLV and t_slv_array data has to be a multiple of byte(s), e.g. x"AA", x"BEEF". t_byte_array is defined in axistream_bfm_pkg. Refer to the AXI4-Stream BFM documentation |
| user_array | t_user_array | x"1" & x"2" | Sideband data to send or has been received via the tuser signal. t_user_array is defined in axistream_bfm_pkg. Refer to the AXI4-Stream BFM documentation |
| strb_array | t_strb_array | x"1" & x"2" | Sideband data to send or has been received via the tstrb signal. t_strb_array is defined in axistream_bfm_pkg. Refer to the AXI4-Stream BFM documentation |
| id_array | t_id_array | x"1" & x"2" | Sideband data to send or has been received via the tid signal. t_id_array is defined in axistream_bfm_pkg. Refer to the AXI4-Stream BFM documentation |
| dest_array | t_dest_array | x"1" & x"2" | Sideband data to send or has been received via the tdest signal. t_dest_array is defined in axistream_bfm_pkg. Refer to the AXI4-Stream BFM documentation |
| msg | string | "Send data" | A custom message to be appended in the log/alert |
| alert-level | t_alert_level | ERROR or TB_WARNING | Set the severity for the alert that may be asserted by the method. |

VVC entity signals

| Name | Type | Description |
|-------------------------|----------------|-----------------------------------|
| clk | std_logic | VVC Clock signal |
| axistream_vvc_master_if | t_axistream_if | See AXI4-Stream BFM documentation |

VVC entity generic constants

| Name | Type | Default | Description |
|--|------------------------|--------------------------------|---|
| GC_VVC_IS_MASTER | boolean | - | Set to true when this VVC instance is an AXI4 Stream master (data is output from BFM). Set to false when this VVC is an AXI4 Stream slave (data is input to BFM.) |
| GC_DATA_WIDTH | integer | - | Width of the AXI4-Stream data bus |
| GC_USER_WIDTH | integer | - | Width of the AXI4-Stream TUSER signal. <i>Note 1:</i> if TUSER is wider than 8, increase the value of the constant C_MAX_TUSER_BITS in axistream_bfm_pkg. <i>Note 2:</i> If the TUSER signal is not used, refer to description in Section 4 |
| GC_ID_WIDTH | integer | - | Width of the AXI4-Stream TID signal. <i>Note 1:</i> if TID is wider than 8, increase the value of the constant C_MAX_TID_BITS in axistream_bfm_pkg. <i>Note 2:</i> If the TID signal is not used, refer to description in Section 4 |
| GC_DEST_WIDTH | integer | - | Width of the AXI4-Stream TDEST signal. <i>Note 1:</i> if TDEST is wider than 4, increase the value of the constant C_MAX_TDEST_BITS in axistream_bfm_pkg. <i>Note 2:</i> If the TDEST signal is not used, refer to description in Section 4 |
| GC_INSTANCE_IDX | natural | - | Instance number to assign the VVC |
| GC_AXISTREAM_CONFIG | t_axistream_bfm_config | C_AXISTREAM_BFM_CONFIG_DEFAULT | Configuration for the AXI4-Stream BFM, see AXI4-Stream BFM documentation. |
| GC_CMD_QUEUE_COUNT_MAX | natural | 1000 | Absolute maximum number of commands in the VVC command queue |
| GC_CMD_QUEUE_COUNT_THRESHOLD | natural | 950 | An alert will be generated when reaching this threshold to indicate that the command queue is almost full. The queue will still accept new commands until it reaches C_CMD_QUEUE_COUNT_MAX. |
| GC_CMD_QUEUE_COUNT_THRESHOLD_SEVERITY | t_alert_level | WARNING | Alert severity which will be used when command queue reaches GC_CMD_QUEUE_COUNT_THRESHOLD. |
| GC_RESULT_QUEUE_COUNT_MAX | natural | 1000 | Maximum number of unfetched results before result_queue is full. |
| GC_RESULT_QUEUE_COUNT_THRESHOLD | natural | 950 | An alert with severity 'result_queue_count_threshold_severity' will be issued if command queue exceeds this count. Used for early warning if result queue is almost full. Will be ignored if set to 0. |
| GC_RESULT_QUEUE_COUNT_THRESHOLD_SEVERITY | t_alert_level | WARNING | Severity of alert to be initiated if exceeding result_queue_count_threshold |

VVC details

All VVC procedures are defined in `vvm_methods_pkg` (dedicated this VVC), and `uvvm_vvc_framework.uvvm_methods_pkg` and `uvvm_vvc_framework.uvvm_support_pkg` (common VVC procedures).

It is also possible to send a multicast to all instances of a VVC with `ALL_INSTANCES` as parameter for `vvc_instance_idx`.

Note the procedure name change when using `t_byte_array`.

1 VVC procedure details

| Procedure | Description |
|--|---|
| <code>axistream_transmit[_bytes]()</code> | <p><code>axistream_transmit[_bytes]</code> (VVCT, <code>vvc_instance_idx</code>, <code>data_array</code>, [<code>user_array</code>, [<code>strb_array</code>, <code>id_array</code>, <code>dest_array</code>]], <code>msg</code>)</p> <p>The <code>axistream_transmit()</code> VVC procedure adds a transmit command to the AXI4-Stream VVC executor queue, which will run as soon as all preceding commands have completed. When the command is scheduled to run, the executor calls the AXI4-Stream BFM <code>axistream_transmit()</code> procedure, described in the AXI4-Stream BFM QuickRef.</p> <p>The <code>axistream_transmit()</code> procedure can only be called when the AXISTREAM VVC is instantiated in master mode, i.e. setting the generic constant 'GC_MASTER_MODE' to true.</p> <p>Examples:</p> <pre>axistream_transmit(AXISTREAM_VVCT, 0, v_data_array(0 to 1), "Send a 2 byte packet to DUT, tuser=0 each word / clock cycle"); axistream_transmit(AXISTREAM_VVCT, 0, v_data_array(0 to 1)(16 downto 0), "Send a 4 byte packet to DUT, tuser=0 each word / clock cycle"); axistream_transmit(AXISTREAM_VVCT, 0, v_data_array(0 to v_numBytes-1), v_user_array(0 to v_numWords-1), " Send a 'v_numBytes' byte packet to DUT"); axistream_transmit(AXISTREAM_VVCT, 0, 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.."); axistream_transmit(AXISTREAM_VVCT, 0, (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); -- (tdata'length = 16) axistream_transmit(AXISTREAM_VVCT, 0, (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); -- (tdata'length = 8)</pre> |
| <code>axistream_expect[_bytes]()</code> | <p><code>axistream_expect[_bytes]</code> (VVCT, <code>vvc_instance_idx</code>, <code>exp_data_array</code>, [<code>exp_user_array</code>, [<code>exp_strb_array</code>, <code>exp_id_array</code>, <code>exp_dest_array</code>]], <code>msg</code>, [<code>aler_level</code>])</p> <p>The <code>axistream_expect()</code> VVC procedure adds an expect command to the AXI4-Stream VVC executor queue, which will run as soon as all preceding commands have completed. When the command is scheduled to run, the executor calls the AXI4-Stream BFM <code>axistream_expect()</code> procedure, described in the AXI4-Stream BFM QuickRef.</p> <p>The <code>axistream_expect()</code> procedure can only be called when the AXISTREAM VVC is instantiated in slave mode, i.e. setting the generic constant 'GC_MASTER_MODE' to false.</p> <p>Examples:</p> <pre>axistream_expect(AXISTREAM_VVCT, 0, v_exp_data_array(0 to 1),</pre> |

```

        "Expect a 2 byte packet, ignoring the tuser bits");
axistream_expect(AXISTREAM_VVCT, 0, v_exp_data_array(0 to 1) (16 downto 0),
        "Expect a 4 byte packet, ignoring the tuser bits");
axistream_expect(AXISTREAM_VVCT, 0, v_exp_data_array(0 to v_numBytes-1), v_user_array(0 to v_numWords-1),
        "Expect a packet, checking the tuser bits");
axistream_expect(AXISTREAM_VVCT, 0, v_exp_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),
        "Check all sigs");
axistream_expect(AXISTREAM_VVCT, 0, (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); --(tdata'length = 16)
axistream_expect(AXISTREAM_VVCT, 0, (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); --(tdata'length = 8)

```

axistream_receive[_bytes] ()

axistream_receive[_bytes] (VVCT, vvc_instance_idx, addr, msg)

The axistream_receive() VVC procedure adds a receive command to the AXISTREAM VVC executor queue, which will run as soon as all preceding commands have completed. When the receive command is scheduled to run, the executor calls the AXISTREAM BFM axistream_receive() procedure, described in the AXISTREAM BFM QuickRef. The axistream_receive() procedure can only be called when the AXISTREAM VVC is instantiated in slave mode, i.e. setting the generic constant 'GC_MASTER_MODE' to false.

The value receive from DUT will not be returned in this procedure call since it is non-blocking for the sequencer/caller, but the received data and metadata will be stored in the VVC for a potential future fetch (see example with *fetch_result* below).

Note that the stored received data is *t_byte_array*.

Example:

```
axistream_receive(AXISTREAM_VVCT, 1, "Receive data to VVC");
```

Example with fetch_result() call: Result is placed in **v_result**

```

variable v_cmd_idx      : natural; -- Command index for the last receive
variable v_result       : work.vvc_cmd_pkg.t_vvc_result; -- Result from receive (data and metadata)
(...)
axistream_receive(AXISTREAM_VVCT, 1, "Receive data to VVC");
v_cmd_idx := shared_cmd_idx;
await_completion(AXISTREAM_VVCT, 1, 1 ms, "Wait for receive to finish");
fetch_result(AXISTREAM_VVCT, 1, v_cmd_idx, v_result, "Fetching result from receive operation");

```

2 VVC Configuration

| Record element | Type | C_AXISTREAM_BFM_CONFIG_DEFAULT | Description |
|---------------------------------------|------------------------|---|--|
| inter_bfm_delay | t_inter_bfm_delay | C_AXISTREAM_INTER_BFM_DELAY_DEFAULT | Delay between any requested BFM accesses towards the DUT. - TIME_START2START: Time from a BFM start to the next BFM start (A TB_WARNING will be issued if access takes longer than TIME_START2START). - TIME_FINISH2START: Time from a BFM end to the next BFM start. Any insert_delay() command will add to the above minimum delays, giving for instance the ability to skew the BFM starting time. |
| cmd_queue_count_max | natural | C_CMD_QUEUE_COUNT_MAX | Maximum pending number in command queue before queue is full. Adding additional commands will result in an ERROR. |
| cmd_queue_count_threshold | natural | C_CMD_QUEUE_COUNT_THRESHOLD | An alert with severity "cmd_queue_count_threshold_severity" will be issued if command queue exceeds this count. Used for early warning if command queue is almost full. Will be ignored if set to 0. |
| cmd_queue_count_threshold_severity | t_alert_level | C_CMD_QUEUE_COUNT_THRESHOLD_SEVERITY | Severity of alert to be initiated if exceeding cmd_queue_count_threshold |
| result_queue_count_max | natural | C_RESULT_QUEUE_COUNT_MAX | Maximum number of unfetched results before result_queue is full. |
| result_queue_count_threshold | natural | C_RESULT_QUEUE_COUNT_THRESHOLD | An alert with severity 'result_queue_count_threshold_severity' will be issued if command queue exceeds this count. Used for early warning if result queue is almost full. Will be ignored if set to 0. |
| result_queue_count_threshold_severity | t_alert_level | C_RESULT_QUEUE_COUNT_THRESHOLD_SEVERITY | Severity of alert to be initiated if exceeding result_queue_count_threshold |
| bfm_config | t_axistream_bfm_config | C_AXISTREAM_BFM_CONFIG_DEFAULT | Configuration for AXI4-Stream BFM. See quick reference for AXI4-Stream BFM |
| msg_id_panel | t_msg_id_panel | C_VVC_MSG_ID_PANEL_DEFAULT | VVC dedicated message ID panel |

The configuration record can be accessed from the Central Testbench Sequencer through the shared variable array, e.g.:

```
shared_axistream_vvc_config(1).inter_bfm_delay.delay_in_time := 50 ns;
shared_axistream_vvc_config(1).bfm_config.clock_period      := 10 ns;
```

3 VVC Status

The current status of the VVC can be retrieved during simulation. This is achieved by reading from the shared variable shared_axistream_vvc_status record from the test sequencer. The record contents can be seen below:

| Record element | Type | Description |
|------------------|---------|---|
| current_cmd_idx | natural | Command index currently running |
| previous_cmd_idx | natural | Previous command index to run |
| pending_cmd_cnt | natural | Pending number of commands in the command queue |

4 VVC Interface

In this VVC, the interface has been encapsulated in a signal record of type *t_axistream_if* in order to improve readability of the code. Since the AXI4-Stream interface busses can be of arbitrary size, the interface *std_logic_vectors* have been left unconstrained. These unconstrained SLVs needs to be constrained when the interface signals are instantiated. For this interface, the could look like:

```
signal axistream_if : t_axistream_if(tdata(C_DATA_WIDTH-1 downto 0),  
  tkeep((C_DATA_WIDTH/8)-1 downto 0),  
  tuser(C_USER_WIDTH-1 downto 0),  
  tstrb((C_DATA_WIDTH/8)-1 downto 0),  
  tid(C_ID_WIDTH-1 downto 0),  
  tdest(C_DEST_WIDTH-1 downto 0)  
  );
```

The widths of *tuser*, *tstrb*, *tid* and *tdest* are declared even when not used or connected to DUT.

Set the widths of unused signals to 1, for example *C_USER_WIDTH = 1*.

5 Additional Documentation

Additional documentation about UVVM and its features can be found under “/uvvm_vvc_framework/doc/”.

For additional documentation on the AXI4-Stream standard, refer to “AMBA 4 AXI4-Stream Protocol Specification (ARM IHI 0051)”, available from ARM.

6 Compilation

AXI4-Stream VVC must be compiled with VHDL 2008.

It is dependent on the following libraries

- **UVVM Utility Library (UVVM-Util), version 2.2.0 and up**
- **UVVM VVC Framework, version 2.1.0 and up**
- **AXI4-Stream BFM**

Before compiling the AXI4-Stream VVC, assure that uvvm_vvc_framework and uvvm_util have been compiled.

6.1.1.1 Compile order for the AXI4-Stream VVC:

| Compile to library | File | Comment |
|----------------------|--|---|
| bitvis_vip_axistream | axistream_bfm_pkg.vhd | AXI4-Stream BFM |
| bitvis_vip_axistream | vvc_cmd_pkg.vhd | AXI4-Stream VVC command types and operations |
| bitvis_vip_axistream | ../uvvm_vvc_framework/src_target_dependent/td_target_support_pkg.vhd | UVVM VVC target support package, compiled into the AXI4-Stream VVC library. |
| bitvis_vip_axistream | ../uvvm_vvc_framework/src_target_dependent/td_vvc_framework_common_methods_pkg.vhd | UVVM framework common methods compiled into the AXI4-Stream VVC library |
| bitvis_vip_axistream | vvc_methods_pkg.vhd | AXI4-Stream VVC methods |
| bitvis_vip_axistream | ../uvvm_vvc_framework/src_target_dependent/td_queue_pkg.vhd | UVVM queue package for the VVC |
| bitvis_vip_axistream | ../uvvm_vvc_framework/src_target_dependent/td_vvc_entity_support_pkg.vhd | UVVM VVC entity support compiled into the AXI4-Stream VVC library |
| bitvis_vip_axistream | axistream_vvc.vhd | AXI4-Stream VVC |

7 Simulator compatibility and setup

This VVC has been compiled and tested with Modelsim version 10.5b.

For required simulator setup see **UVVM-Util** Quick reference.

IMPORTANT

This is a simplified Verification IP (VIP) for AXI4-Stream. The given VIP complies with the basic AXI4-Stream protocol and thus allows a normal access towards an AXI4-Stream interface.

This VIP is not AXI4-Stream protocol checker. For a more advanced VIP please contact Bitvis AS at support@bitvis.no

INTELLECTUAL PROPERTY

Disclaimer: This IP and any part thereof are provided "as is", without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose and noninfringement. In no event shall the authors or copyright holders be liable for any claim, damages or other liability, whether in an action of contract, tort or otherwise, arising from, out of or in connection with this IP.