# AXI4-Lite VVC – Quick Reference

For general information see UVVM Essential Mechanisms located in uvvm_vvc_framework/doc.


*axilite_vvc.vhd*

**axilite_write** (VVCT, vvc_instance_idx, addr, data, [byte_enable], msg, [scope])

**Example**: axilite_write(AXILITE_VVCT, 1, x"6000", x"F102", "Writing data to Peripheral 1");

**axilite_read** (VVCT, vvc_instance_idx, addr, msg, [scope])

**Example**: axilite_read(AXILITE_VVCT, 1, x"6000", "Read from Peripheral 1");

**axilite_check** (VVCT, vvc_instance_idx, addr, data, msg, [alert_level, [scope]])

**Example**: axilite_check(AXILITE_VVCT, 1, x"6000", x"393B", "Check data from Peripheral 1");

AXI4-Lite VVC Configuration record ´**vvc_config**´ -- accessible via **shared_axilite_vvc_config**

| Record element | Type | C_AXILITE_VVC_CONFIG_DEFAULT |
|---|---|---|
| inter_bfm_delay | t_inter_bfm_delay | C_AXILITE_INTER_BFM_DELAY_DEFAULT |
| [cmd/result]_queue_count_max | natural | C_[CMD/RESULT]_QUEUE_COUNT_MAX |
| [cmd/result]_queue_count_threshold | natural | C_[CMD/RESULT]_QUEUE_COUNT_THRESHOLD |
| [cmd/result]_queue_count_threshold_severity | t_alert_level | C_[CMD/RESULT]_QUEUE_COUNT_THRESHOLD_SEVERITY |
| bfm_config | t_axilite_bfm_config | C_AXILITE_BFM_CONFIG_DEFAULT |
| msg_id_panel | t_msg_id_panel | C_VVC_MSG_ID_PANEL_DEFAULT |

AXI4-Lite VVC Status record signal ´**vvc_status**´ -- accessible via **shared_axilite_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_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()**

UVVM™



VHDL 2008 only

## VVC target parameters

| Name | Type | Example(s) | Description |
|---|---|---|---|
| VVCT | t_vvc_target_record | AXILITE_VVCT | VVC target type compiled into each VVC in order to differentiate between VVCs. |
| vvc_instance_idx | integer | 1 | Instance number of the VVC |

## VVC functional parameters

| Name | Type | Example(s) | Description |
|---|---|---|---|
| addr | unsigned | x"325A" | The address of a SW accessible register. Could be offset or full address depending on the DUT |
| data | std_logic_vector | x"F1A332D3" | The data to be written (in axilite_write) or the expected data (in axilite_check). |
| byte_enable | std_logic_vector | (others => '1') | This argument selects which bytes to use (all '1' means all bytes are updated) |
| msg | string | "Send to peripheral 1" | 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. |
| scope | string | "AXILITE VVC" | A string describing the scope from which the log/alert originates. In a simple single sequencer typically "AXILITE BFM". In a verification component typically "AXILITE VVC ". |

## VVC entity signals

| Name | Type | Description |
|---|---|---|
| clk | std_logic | VVC Clock signal |
| axilite_vvc_master_if | t_axilite_if | See AXI4-Lite BFM documentation |

## VVC entity generic constants

| Name | Type | Default | Description |
|---|---|---|---|
| GC_ADDR_WIDTH | integer | 8 | Width of the AXI4-Lite address bus |
| GC_DATA_WIDTH | integer | 32 | Width of the AXI4-Lite data bus |
| GC_INSTANCE_IDX | natural | 1 | Instance number to assign the VVC |
| GC_AXILITE_CONFIG | t_axilite_bfm_config | C_AXILITE_BFM_CONFIG_DEFAULT | Configuration for the AXI4-Lite BFM, see AXI4-Lite 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 result 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 vvc_methods_pkg (dedicated this VVC), and uvvm_vvc_framework.td_vvc_framework_common_methods_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: Every procedure here can be called without the optional parameters enclosed in [ ].*

## 1 VVC procedure details and examples

| Procedure | Description |
|---|---|
| **axilite_write()** | **axilite_write(VVC, instance_idx, addr, data, [byte_enable,] msg, [scope])**<br><br>The axilite_write() VVC procedure adds a write command to the AXI4-Lite VVC executor queue, which will run as soon as all preceding commands have completed. When the write command is scheduled to run, the executor calls the AXI4-Lite BFM axilite_write() procedure, described in the AXI4-Lite BFM QuickRef.<br>axilite_write can be called with or without byte_enable constant. When not set, byte_enable is set to all '1', indicating that all bytes are valid.<br><br>Examples:<br>`axilite_write(AXILITE_VVCT, 1, x"0011A000", x"F102", "Writing data to Peripheral 1", C_SCOPE);`<br>`axilite_write(AXILITE_VVCT, 1, C_ADDR_PERIPHERAL_1, x"F102", b"11", "Writing data to Peripheral 1", C_SCOPE);`<br>`axilite_write(AXILITE_VVCT, 1, C_ADDR_DMA, x"1155F102", "Writing data to DMA", C_SCOPE);` |
| **axilite_read()** | **axilite_read(VVC, instance_idx, addr, msg, [scope])**<br><br>The axilite_read() VVC procedure adds a read command to the AXI4-Lite VVC executor queue, which will run as soon as all preceding commands have completed. When the read command is scheduled to run, the executor calls the AXI4-Lite BFM axilite_read() procedure, described in the AXI4-Lite BFM QuickRef.<br>The value read from the DUT will not be returned in this procedure call since it is non-blocking for the sequencer/caller, but the read data will be stored in the VVC for a potential future fetch (see example with fetch_result() below).<br><br>Examples:<br>`axilite_read(AXILITE_VVCT, 1, x"00099555", "Read from Peripheral 1" C_SCOPE);`<br>`axilite_read(AXILITE_VVCT, 1, C_ADDR_IO, "Read from IO device" C_SCOPE);`<br><br>**Example with fetch_result() call. Result is placed in `v_data`**<br>`variable v_cmd_idx : natural; -- Command index for the last read`<br>`variable v_data : work.vvc_cmd_pkg.t_vvc_result; -- Result from read`<br>(…)<br>`axilite_read(AXILITE_VVCT, 1, x"112252AA", "Read from Peripheral 1");`<br>`v_cmd_idx := get_last_received_cmd_idx(AXILITE_VVCT, 1);`<br>`await_completion(AXILITE_VVCT,1, v_cmd_idx, 100 ns, "Wait for read to finish");`<br>`fetch_result(AXILITE_VVCT,1, v_cmd_idx, v_data, "Fetching result from read operation");` |

**axilite_check()**

**axilite_check(VVC, instance_idx, addr, data, msg, [alert_level, [scope]])**

The axilite_check() VVC procedure adds a check command to the AXI4-Lite VVC executor queue, which will run as soon as all preceding commands have completed. When the check command is scheduled to run, the executor calls the AXI4-Lite BFM axilite_check() procedure, described in the AXI4-Lite BFM QuickRef. The axilite_check() procedure will perform a read operation, then check if the read data is equal to the 'data' parameter. If the read data is not equal to the expected 'data' parameter, an alert with severity 'alert_level' will be issued. The read data will not be stored by this procedure.

Example:

```
axilite_check(AXILITE_VVCT, 1, x"00099555", x"393B", "Check data from Peripheral 1", ERROR, C_SCOPE);
```

## 2 VVC Configuration

| Record element | Type | C_AXILITE_BFM_CONFIG_DEFAULT | Description |
|---|---|---|---|
| inter_bfm_delay | t_inter_bfm_delay | C_AXILITE_INTER_BFM_DELAY_DEFAULT | Delay between any requested BFM accesses towards the DUT.<br>- 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).<br>- TIME_FINISH2START: Time from a BFM end to the next BFM start.<br>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_MAX_COMMAND_QUEUE | 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 result 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_axilite_bfm_config | C_AXILITE_BFM_CONFIG_DEFAULT | Configuration for AXI4-Lite BFM. See quick reference for AXI4-Lite 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_axilite_vvc_config(1).inter_bfm_delay.delay_in_time := 50 ns;
shared_axilite_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_axilite_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_axilite_if* in order to improve readability of the code. Since the AXI4-Lite 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 axilite_if_1 : t_axilite_if( write_address_channel( awaddr( C_ADDR_WIDTH   -1 downto 0)),
                                    write_data_channel   ( wdata ( C_DATA_WIDTH   -1 downto 0),
                                                           wstrb(( C_DATA_WIDTH/8)-1 downto 0)),
                                    read_address_channel ( araddr( C_ADDR_WIDTH   -1 downto 0)),
                                    read_data_channel    ( rdata ( C_DATA_WIDTH   -1 downto 0)) );
```

## 5 Additional Documentation

Additional documentation about UVVM and its features can be found under “/uvvm_vvc_framework/doc/”.
For additional documentation on the AXI4-Lite standard, please see the AXI4-Lite specification “AMBA® AXI™ and ACE™ Protocol Specification - AXI3™, AXI4™, and AXI4-Lite™ ACE and ACE-Lite™”, available from ARM.

## 6 Compilation

AXI4-Lite 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-Lite BFM***

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

See UVVM Essential Mechanisms located in uvvm_vvc_framework/doc for information about compile scripts.

**Compile order for the AXI4-Lite VVC:**

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

# 7 Simulator compatibility and setup

This VVC 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.

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