**AXI4-Lite BFM** –Quick Reference

**BFM**

Note: The AXI4-Lite BFM procedures do not access the AXI channels independently. However, this is sufficient for most use cases.

If independent channel access is required, for instance simultaneous read and write accesses, use the AXI4-Lite VVC.

|  |
| --- |
| axilite\_write (addr\_value, data\_value, [byte\_enable], msg, clk, axilite\_if, [scope, [msg\_id\_panel, [config]]]) |
| Example: axilite\_write(x"00456000", x”AA11”, “Writing data to Peripheral 1”, clk, axilite\_if); -- Without byte\_enable  Example: axilite\_write(C\_ADDR\_PERIPHERAL\_1, x”AA11”, “01”, “Writing data to Peripheral 1”, clk, axilite\_if); --With byte\_enable  *Suggested usage: axilite\_write(C\_ADDR\_DMA, x”AA11”, “Writing data to DMA”); -- Suggested usage requires local overload (see section 5)* |

*axilite\_bfm\_pkg.vhd*

|  |
| --- |
| axilite\_read (addr\_value, data\_value, msg, clk, axilite\_if, [scope, [msg\_id\_panel, [config, [proc\_name]]]]) |
| Example: axilite\_read(x"11355000", v\_data\_out, “Read from Peripheral 1”, clk, axilite\_if);  *Suggested usage: axilite\_read(C\_ADDR\_IO, v\_data\_out, “Read from IO”); -- Suggested usage requires local overload (see section 5)* |

|  |
| --- |
| axilite\_check (addr\_value, data\_exp, msg, clk, axilite\_if, [alert\_level, [scope, [msg\_id\_panel, [config]]]]) |
| Example: axilite\_check(x"6840A000", x”3B16”, *“Check data from Peripheral 1”*, clk, axilite\_if);  *Suggested usage: axilite\_check(C\_ADDR\_IO, x”3B16”, “Check data from IO”); -- Suggested usage requires local overload (see section 5)* |

|  |
| --- |
| init\_axilite\_if\_signals (addr\_width, data\_width) |
| Example: axilite\_if <= init\_axilite\_if\_signals(addr\_width, data\_width); |

BFM Configuration record ´**t\_axilite\_bfm\_config´**

|  |  |  |  |
| --- | --- | --- | --- |
| **Record element** | **Type** | **C\_AXILITE\_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 and valid signals from the DUT. |
| max\_wait\_cycles\_severity | t\_alert\_level | TB\_FAILURE | The above timeout will have this severity |
| clock\_period | time | -1 ns | Period of the clock signal. |
| 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 the severity |
| setup\_time | time | -1 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 | -1 ns | Hold time for generated signals. Suggested value is clock\_period/4. An alert is reported if hold\_time exceed clock\_period/2. |
| bfm\_sync | t\_bfm\_sync | SYNC\_ON\_CLOCK\_ONLY | When set to SYNC\_ON\_CLOCK\_ONLY the BFM will enter on the first falling edge, estimate the clock period, synchronise the output signals and exit ¼ clock period after a succeeding rising edge.  When set to SYNC\_WITH\_SETUP\_AND\_HOLD the BFM will use the configured setup\_time, hold\_time and clock\_period to synchronise output signals with clock edges. |
| match\_strictness | t\_match\_strictness | MATCH\_EXACT | Matching strictness for std\_logic values in check procedures.  MATCH\_EXACT requires both values to be the same. Note that the expected value  can contain the don’t care operator ‘-‘.  MATCH\_STD allows comparisons between ‘H’ and ‘1’, ‘L’ and ‘0’ and ‘-‘ in both values. |
| expected\_response | t\_xresp | OKAY | Sets the expected response for both read and write transactions. |
| expected\_response\_severity | t\_alert\_level | TB\_FAILURE | A response mismatch will have this severity. |
| protection\_setting | t\_axprot | UNPRIVILEGED\_NONSECURE\_DATA | Sets the AXI access permissions (e.g. write to data/instruction, privileged and secure access). |
| num\_aw\_pipe\_stages | natural | 1 | Write Address Channel pipeline steps |
| num\_w\_pipe\_stages | natural | 1 | Write Data Channel pipeline steps |
| num\_ar\_pipe\_stages | natural | 1 | Read Address Channel pipeline steps |
| num\_r\_pipe\_stages | natural | 1 | Read Data Channel pipeline steps |
| num\_b\_pipe\_stages | natural | 1 | Response Channel pipeline steps |
| id\_for\_bfm | t\_msg\_id | ID\_BFM | The message ID used as a general message ID in the AXI-Lite BFM |
| id\_for\_bfm\_wait | t\_msg\_id | ID\_BFM\_WAIT | The message ID used for logging waits in the AXI-Lite BFM |
| id\_for\_bfm\_poll | t\_msg\_id | ID\_BFM\_POLL | The message ID used for logging polling in the AXI-Lite BFM |

BFM non-signal parameters

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Example(s)** | **Description** |
| addr\_value | unsigned | x”125A” | The address of an AXI4-Lite accessible register. |
| data\_value | std\_logic\_vector | x”20D3” | The data value to be written to the addressed register |
| data\_exp | std\_logic\_vector | x”0D” | The data value to expect when reading the addressed register. A mismatch results in an alert ‘alert\_level’ |
| byte\_enable | std\_logic\_vector | x”11” | This argument selects which bytes to use (all ‘1’ means all bytes are updated) |
| 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 | "AXILITE\_BFM" | 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 ". |
| 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\_axilite\_bfm\_config | C\_AXILITE\_BFM\_CONFIG\_DEFAULT | Configuration of BFM behaviour and restrictions |

BFM signal parameters

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| clk | std\_logic | The clock signal used to read and write data in/out of the AXI4-Lite BFM. |
| axilite\_if | t\_axilite\_if | See table “Signal record ‘axilite\_if’” |

Signal record ‘axilite\_if’

|  |  |
| --- | --- |
| **Record element** | **Type** |
| **write\_address\_channel** | **t\_axilite\_write\_address\_channel** |
| * awaddr | * std\_logic\_vector |
| * awvalid | * std\_logic |
| * awprot | * std\_logic\_vector(2 downto 0) |
| * awready | * std\_logic |
| **write\_data\_channel** | **t\_axilite\_write\_data\_channel** |
| * wdata | * std\_logic\_vector |
| * wstrb | * std\_logic\_vector |
| * wvalid | * std\_logic |
| * wready | * std\_logic |
| **write\_response\_channel** | **t\_axilite\_write\_response\_channel** |
| * bready | * std\_logic |
| * bresp | * std\_logic\_vector(1 downto 0) |
| * bvalid | * std\_logic |
| **read\_address\_channel** | **t\_axilite\_read\_address\_channel** |
| * araddr | * std\_logic\_vector |
| * arvalid | * std\_logic |
| * arprot | * std\_logic\_vector(2 downto 0) |
| * arready | * std\_logic |
| **read\_data\_channel** | **t\_axilite\_read\_data\_channel** |
| * rready | * std\_logic |
| * rdata | * std\_logic\_vector |
| * rresp | * std\_logic\_vector(1 downto 0) |
| * rvalid | * std\_logic |

Note: All signals are active high. See AXI4-Lite documentation for protocol description.

For more information on the AXI4-Lite signals, please see the AXI4-Lite specification.

BFM details

# BFM procedure details and examples

|  |  |
| --- | --- |
| **Procedure** | **Description** |
| **axilite\_write()** | **axilite\_write(addr\_value, data\_value, [byte\_enable,] msg, clk, axilite\_if, [scope, [msg\_id\_panel, [config]]])**  The axilite\_write() procedure writes the given data to the given address of the DUT, using the AXI4-Lite protocol. For protocol details, see the AXI4-Lite specification.   * If the byte\_enable argument is not used, it will be set to all ‘1’, i.e. all bytes are used. * The default value of scope is C\_SCOPE (“AXILITE\_BFM”) * The default value of msg\_id\_panel is shared\_msg\_id\_panel, defined in UVVM-Util. * The default value of config is C\_AXILITE\_BFM\_CONFIG\_DEFAULT, see table on the first page. * A log message is written if ID\_BFM is enabled for the specified message ID panel.   The procedure reports an alert if:   * Data length is neither 32 bit nor 64 bit (alert level: TB\_ERROR) * wready does not occur within max\_wait\_cycles clock cycles (alert level: max\_wait\_cycles\_severity, set in the config) * awready does not occur within max\_wait\_cycles clock cycles (alert level: max\_wait\_cycles\_severity, set in the config) * bresp is not set to expected\_response (set in the config) when bvalid is set to ‘1’ (alert level: expected\_response\_severity, set in the config) * bvalid is not set within max\_wait\_cycles clock cycles (alert level: max\_wait\_cycles\_severity, set in the config)     Examples:  axilite\_write(x”00101155”, x”AAAA”, “Writing data to Peripheral 1”, clk, axilite\_if, C\_SCOPE, shared\_msg\_id\_panel,   C\_AXILITE\_BFM\_CONFIG\_DEFAULT);  axilite\_write(C\_ADDR\_PERIPHERAL\_1, x”00F1”, “01”, “Writing first byte to Peripheral 1”, clk, axilite\_if, C\_SCOPE,   shared\_msg\_id\_panel, C\_AXILITE\_BFM\_CONFIG\_DEFAULT);  Suggested usage (requires local overload, see section 5):  axilite\_write(C\_ADDR\_DMA, x”AAAA”, “Writing data to DMA”); |
| **axilite\_read()** | **axilite\_read(addr\_value, data\_value, msg, clk, axilite\_if, [scope, [msg\_id\_panel, [config, [proc\_name]]]])**  The axilite\_read() procedure reads data from the DUT at the given address, using the AXI4-Lite protocol. For protocol details, see the AXI4-Lite specification. The read data is placed on the output ‘data\_value’ when the read has completed.   * The default value of scope is C\_SCOPE (“AXILITE\_BFM”) * The default value of msg\_id\_panel is shared\_msg\_id\_panel, defined in UVVM-Util. * The default value of config is C\_AXILITE\_BFM\_CONFIG\_DEFAULT, see table on the first page. * The default value of proc\_name is “axilite\_read”. This argument is intended to be used internally, when the procedure is called by axilite\_check(). * A log message is written if ID\_BFM is enabled for the specified message ID panel. This will only occur if the argument proc\_name is left unchanged.   The procedure reports an alert if:   * The read data length (rdata) is neither 32 bit nor 64 bit (alert level: TB\_ERROR) * arready does not occur within max\_wait\_cycles clock cycles (alert level: max\_wait\_cycles\_severity, set in the config) * rresp is not set to expected\_response (set in the config) when rvalid is set to ‘1’ (alert level: expected\_response\_severity, set in the config) * rvalid is not set within max\_wait\_cycles clock cycles (alert level: max\_wait\_cycles\_severity, set in the config)   Example:  axilite\_read(C\_ADDR\_PERIPHERAL\_1, v\_data\_out, “Read from Peripheral 1”, clk, axilite\_if, C\_SCOPE, shared\_msg\_id\_panel,   C\_AXILITE\_BFM\_CONFIG\_DEFAULT);  Suggested usage (requires local overload, see section 5):  axilite\_read(C\_ADDR\_IO, v\_data\_out, “Reading from IO device”); |
| **axilite\_check()** | **axilite\_check(addr\_value, data\_exp, msg, clk, axilite\_if, [alert\_level, [scope, [msg\_id\_panel, [config]]]])**  The axilite\_check() procedure reads data from the DUT at the given address, using the AXI4-Lite protocol. For protocol details, see the AXI4-Lite specification. After reading data from the AXI4-Lite bus, the read data is compared with the expected data, ‘data\_exp’.   * The default value of alert\_level is ERROR * The default value of scope is C\_SCOPE (“AXILITE\_BFM”) * The default value of msg\_id\_panel is shared\_msg\_id\_panel, defined in UVVM-Util. * The default value of config is C\_AXILITE\_BFM\_CONFIG\_DEFAULT, see table on the first page. * If the check was successful, and the read data matches the expected data, a log message is written with ID\_BFM (if this ID has been enabled). * If the read data did not match the expected data, an alert with severity ‘alert\_level’ will be reported.   The procedure also report alerts for the same conditions as the axilite\_read() procedure.  Example:  axilite\_check(C\_ADDR\_PERIPHERAL\_1, x”3B”, “Check data from Peripheral 1”, clk, axilite\_if, C\_SCOPE, shared\_msg\_id\_panel, ERROR,   C\_AXILITE\_BFM\_CONFIG\_DEFAULT); -- All parameters included  Suggested usage (requires local overload, see section 5):  axilite\_check(C\_ADDR\_UART\_RX, x”3B”, “Checking data in UART RX register”); |
| **init\_axilite\_if\_signals()** | **init\_axilite\_if\_signals(addr\_width, data\_width)**  This function initializes the AXI4-Lite interface. All the BFM outputs are set to zeros ('0') and BFM inputs are set to 'Z'. awprot and arprot are set to UNPRIVILEGED\_NONSECURE\_DATA (“010”).  Example:  axilite\_if <= init\_axilite\_if\_signals(addr\_width, data\_width) |

# Additional Documentation

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.

# Compilation

The AXI4-Lite 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 axilite\_bfm\_pkg.vhd BFM can be compiled into any desired library.

See the UVVM Essential Mechanisms located in uvvm\_vvc\_framework/doc for information about compile scripts.

## Simulator compatibility and setup

See README.md for a list of supported simulators. For required simulator setup see UVVM-Util Quick reference.

# 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.

axilite\_write(C\_ADDR\_PERIPHERAL\_1, C\_TEST\_DATA, “Sending data to Peripheral 1”);

rather than

axilite\_write(C\_ADDR\_PERIPHERAL\_1, C\_TEST\_DATA, “Sending data to Peripheral 1”, clk, axilite\_if, C\_SCOPE,   
 shared\_msg\_id\_panel, C\_AXILITE\_BFM\_CONFIG\_DEFAULT);

By defining the local overload as e.g.:

procedure axilite\_write(

constant addr\_value : in unsigned;

constant data\_value : in std\_logic\_vector;

constant msg : in string) is

begin

axilite\_write(addr\_value, -- keep as is  
 data\_value, -- keep as is

msg, -- keep as is

clk, -- Clock signal

axilite\_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\_AXILITE\_BFM\_CONFIG\_LOCAL); -- Use locally defined configuration or C\_AXILITE\_BFM\_CONFIG\_DEFAULT

end;

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

* Have address value as natural – and convert in the overload
* 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-Lite. The given BFM complies with the basic AXI4-Lite protocol and thus allows a normal access towards an AXI4-Lite interface. This BFM is not AXI4-Lite protocol checker. For a more advanced BFM please contact Bitvis AS at support@bitvis.no

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.

**INTELLECTUAL**

**PROPERTY**