AXI4-Lite BFM – Quick Reference

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.

BFM

axilite_bfm_pkg.vhd

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_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

UVVM™

VHDL 2008 only

			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.

1 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)
```

2 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.

3 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.

3.1 Simulator compatibility and setup

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

4 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