**GMII BFM** –Quick Reference

**BFM**

For general information see UVVM Essential Mechanisms located in uvvm\_vvc\_framework/doc.

|  |
| --- |
| gmii\_write (data, msg, gmii\_to\_dut\_if, [scope, [msg\_id\_panel, [config]]]) |
| Example: gmii\_write(v\_data\_bytes, “Send 10 bytes of data”, gmii\_to\_dut\_if); |

*gmii\_bfm\_pkg.vhd*

|  |
| --- |
| gmii\_read (data, msg, dut\_to\_gmii\_if, [scope, [msg\_id\_panel, [config, [proc\_name]]]]) |
| Example: gmii\_read(v\_received\_data\_bytes, “Read 10 bytes of data”, dut\_to\_gmii\_if); |

|  |
| --- |
| init\_gmii\_if\_signals (VOID) |
| Example: t\_gmii\_if <= init\_gmii\_if\_signals(VOID); |



|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | BFM Configuration record ´**t\_gmii\_bfm\_config´** | | | | **Record element** | **Type** | **C\_GMII\_BFM\_CONFIG\_DEFAULT** | | clock\_period | time | 8 ns | | clock\_period\_margin | time | 0 ns | | clock\_margin\_severity | t\_alert\_level | TB\_ERROR | | setup\_time | time | 2 ns | | hold\_time | time | 2 ns | | id\_for\_bfm | t\_msg\_id | ID\_BFM | | id\_for\_bfm\_wait | t\_msg\_id | ID\_BFM\_WAIT | | |  |  | | --- | --- | | Signal record ´**t\_gmii\_to\_dut\_if´** | | | **Record element** | **Type** | | gtxclk | std\_logic | | txd | std\_logic\_vector | | txen | std\_logic |  |  |  | | --- | --- | | Signal record ´**t\_dut\_to\_gmii\_if´** | | | **Record element** | **Type** | | rxclk | std\_logic | | rxd | std\_logic\_vector | | rxdv | std\_logic |  |  |  | | --- | --- | | Signal record ´**t\_gmii\_if´** | | | **Record element** | **Type** | | gmii\_to\_dut | t\_gmii\_to\_dut | | dut\_to\_gmii | t\_dut\_to\_gmii | |

BFM non-signal parameters

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Example(s)** | **Description** |
| data | t\_byte\_array | (x”1A”, x”1B”, x”2A”) | The data value written to or read from the DUT. |
| data\_exp | std\_logic\_vector | x”0D” | The data value to expect when reading the addressed register. A mismatch results in an alert with severity ‘alert\_level’ |
| timeout | time | 100 ns | The maximum time to pass before the expected data must be found. Exceeding this limit results in an alert with severity ‘alert\_level’. |
| alert\_level | t\_alert\_level | ERROR or TB\_WARNING | Set the severity for the alert that may be asserted by the BFM procedure. |
| msg | string | “Write to Peripheral 1” | A custom message to be appended in the log/alert. |
| scope | string | "GMII BFM" | A string describing the scope from which the log/alert originates. In a simple single sequencer typically "GMII BFM". In a verification component typically "GMII\_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 ID panel defined in the adaptations package. |
| config | t\_gmii\_bfm\_config | C\_GMII\_BFM\_CONFIG\_DEFAULT | Configuration of BFM behaviour and restrictions. See section 2 for details. |

BFM signal parameters

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| gtxclk | std\_logic | Clock signal for TX signals. |
| txd | std\_logic\_vector | Data from DUT. |
| txen | std\_logic | Transmit enable from DUT. |
| rxclk | std\_logic | Clock signal for RX signals. |
| rxd | std\_logic\_vector | Data to DUT. |
| rxdv | std\_logic | Data valid signal to DUT, |

BFM details

# BFM procedure details and examples

|  |  |
| --- | --- |
| **Procedure** | **Description** |
| **gmii\_write()** | **gmii\_write(data, msg, gmii\_to\_dut\_if, [scope, [msg\_id\_panel, [config]]])**  The gmii\_write() procedure writes the given data to the DUT, using the GMII protocol: |
| **gmii\_read()** | **gmii\_read(data, msg, dut\_to\_gmii\_if, [scope, [msg\_id\_panel, [config, [proc\_name]]]])**  The gmii\_read() procedure reads data from the DUT, using the GMII protocol: |
| **init\_gmii\_if\_signals()** | **init\_gmii\_if\_signals(VOID)**  This function initializes the GMII interface. All the BFM signals are set to 'Z'.  Example:  gmii\_if <= init\_gmii\_if\_signals(VOID) |

# BFM Configuration record

Type name: t\_sbi\_bfm\_config

|  |  |  |  |
| --- | --- | --- | --- |
| **Record element** | **Type** | **C\_SBI\_BFM\_CONFIG\_DEFAULT** | **Description** |
| max\_wait\_cycles | integer | 10 | The maximum number of clock cycles to wait for the DUT ready signal before reporting a timeout alert. |
| max\_wait\_cycles\_severity | t\_alert\_level | failure | The above timeout will have this severity |
| use\_fixed\_wait\_cycles\_read | boolean | false | When true, wait 'fixed\_wait\_cycles\_read' after asserting ‘rena’ signal, before sampling ‘rdata from DUT’ |
| fixed\_wait\_cycles\_read | natural | 0 | Number of clock cycles to wait after asserting ‘rena’ signal, before sampling ‘rdata’ from DUT. |
| clock\_period | time | 10 ns | Period of the clock signal. |
| clock\_period\_margin | time | 0 ns | Input clock period margin to specified clock\_period. Will check T/2 if input clock is low when BFM is called and T if input clock is high |
| clock\_margin\_severity | t\_alert\_level | TB\_ERROR | The above margin will have this severity |
| setup\_time | time | 2.5 ns | Generated signals setup time. Suggested value is clock\_period/2.  An alert is reported if setup\_time exceed clock\_period/2. |
| hold\_time | time | 2.5 ns | Generated signals hold time. Suggested value is clock\_period/2.  An alert is reported if hold\_time exceed clock\_period/2. |
| id\_for\_bfm | t\_msg\_id | ID\_BFM | The message ID used as a general message ID in the SBI BFM |
| id\_for\_bfm\_wait | t\_msg\_id | ID\_BFM\_WAIT | The message ID used for logging waits in the SBI BFM |
| id\_for\_bfm\_poll | t\_msg\_id | ID\_BFM\_POLL | The message ID used for logging polling in the SBI BFM |
| use\_ready\_signal | boolean | true | Whether or not to use the interface ‘ready’ signal |
| setup\_time | time | 0 ns | Generated signals setup time. |
| hold\_time | time | 0 ns | Generated signals hold time. |

# Compilation

The GMII 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 sbi\_bfm\_pkg.vhd BFM can be compiled into any desired library.

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

## 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.

\*1 <https://www.aldec.com/en/support/resources/multimedia/webinars/1673>

# 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.

sbi\_write(C\_ADDR\_UART\_BAUDRATE, C\_BAUDRATE\_9600, “Set Baudrate to 9600”);

rather than

sbi\_write(C\_ADDR\_UART\_BAUDRATE, C\_BAUDRATE\_9600, “Set Baudrate to 9600”, clk, sbi\_if,

C\_CLK\_PERIOD, C\_SCOPE, shared\_msg\_id\_panel, C\_SBI\_CONFIG\_DEFAULT);

By defining the local overload as e.g.:

procedure sbi\_write(

constant addr\_value : in unsigned;

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

constant msg : in string) is

begin

sbi\_write(addr\_value, -- keep as is

data\_value, -- keep as is

msg, -- keep as is

sbi\_if, -- Signal must be visible in local process scope

C\_CLK\_PERIOD, -- Just use the default

C\_SCOPE, -- Just use the default

shared\_msg\_id\_panel, -- Use global, shared msg\_id\_panel

C\_SBI\_CONFIG\_LOCAL); -- Use locally defined configuration or C\_SBI\_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 SBI.  
The given BFM complies with the basic SBI protocol and thus allows a normal access towards a SBI interface. This BFM is not a SBI protocol checker.   
For a more advanced BFM please contact Bitvis AS at [support@bitvis.no](mailto: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**