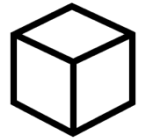
RGMII BFM – Quick Reference

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

BFM



rgmii_bfm_pkg.vhd

rgmii_write (data_array, action_when_transfer_is_done, msg, rgmii_tx_if, [scope, [msg_id_panel, [config]]]) [1]

Example: rgmii_write(v_data_array(0 to v_numBytes-1), HOLD_LINE_AFTER_TRANSFER , "Write v_numBytes bytes", rgmii_tx_if, C_SCOPE, shared_msg_id_panel, rgmii_bfm_config);
Example: rgmii_write((x"01", x"02", x"03", x"04"), "Write 4 bytes", rgmii_tx_if);

rgmii_read (data_array, data_len, msg, rgmii_rx_if, [scope, [msg_id_panel, [config, [ext_proc_call]]]]) [1]

Example: rgmii_read(v_data_array, v_numBytes, "Read v_numBytes bytes", rgmii_rx_if, C_SCOPE, shared_msg_id_panel, rgmii_bfm_config, "rgmii_expect()");
Example: rgmii_read(v_data_array, v_numBytes, "Read v_numBytes bytes", rgmii_rx_if);

rgmii_expect (data_exp, msg, rgmii_rx_if, [alert_level, [scope, [msg_id_panel, [config]]]]) [1]

Example: rgmii_expect(v_data_array(0 to v_numBytes-1), "Expect v_numBytes bytes", rgmii_rx_if, ERROR, C_SCOPE, shared_msg_id_panel, rgmii_bfm_config);
Example: rgmii_expect((x"01", x"02", x"03", x"04"), "Expect 4 bytes", rgmii_rx_if);

init_rgmii_if_signals (VOID)

Example: rgmii_tx_if <= init_rgmii_if_signals(VOID);

Note 1: the BFM configuration has to be defined and used when calling the RGMII BFM procedures. See section 5 for an example of how to define a local BFM config.

UVVM™

VHDL 2008 only

BFM Configuration record '**t_rgmii_bfm_config**'

Record element	Type	C_RGMII_BFM_CONFIG_DEFAULT
max_wait_cycles	integer	10
max_wait_cycles_severity	t_alert_level	ERROR
clock_period	time	-1 ns
rx_clock_skew	time	-1 ns
match_strictness	t_match_strictness	MATCH_EXACT
id_for_bfm	t_msg_id	ID_BFM
data_valid_on_both_clock_edges	boolean	true

Signal record **'t_rgmii_tx_if'**

Record element	Type
txc	std_logic
txd	std_logic_vector
tx_ctl	std_logic

Signal record **'t_rgmii_rx_if'**

Record element	Type
rxc	std_logic
rxd	std_logic_vector
rx_ctl	std_logic

BFM signal parameters

Name	Type	Description
txc	std_logic	TX reference clock
txd	std_logic_vector	TX data lines (to DUT)
tx_ctl	std_logic	TX enable
rxc	std_logic	RX reference clock
rxd	std_logic_vector	RX data lines (from DUT)
rx_ctl	std_logic	RX enable

Note: tx_ctl & rx_ctl only represent TXEN & RXEN respectively, the functionality of TXERR & RXERR is not implemented.
Also, there is no support for RGMII-ID (use of Tsetup & Thold). For more information see the specification "Reduced Gigabit Media Independent Interface (RGMII) Version 2.0".

BFM non-signal parameters

Name	Type	Example(s)	Description
data_array data_exp	t_byte_array	(x"D0", x"D1", x"D2", x"D3")	An array of bytes containing the data to be written/read. data_array(0) is written/read first, while data_array(data_array'high) is written/read last. For clarity, data_array is required to be ascending, for example defined by the test sequencer as follows: `variable v_data_array : t_byte_array(0 to C_MAX_BYTES-1);`
action_when_transfer_is_done	t_action_when_transfer_is_done	RELEASE_LINE_AFTER_TRANSFER	Whether to release (default) or hold the TXEN line after the procedure is finished. Useful when transmitting a packet of data through several procedures.
data_len	natural	v_data_len	The number of valid bytes in the data_array. Note that the data_array can be bigger and that is why the length is returned.
alert_level	t_alert_level	ERROR or TB_WARNING	Set the severity for the alert that may be asserted by the procedure.
msg	string	"Write bytes"	A custom message to be appended in the log/alert.
scope	string	"RGMII_BFM"	A string describing the scope from which the log/alert originates. In a simple single sequencer typically "RGMII_BFM". In a verification component typically "RGMII_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_rgmii_bfm_config	C_RGMII_BFM_CONFIG_DEFAULT	Configuration of BFM behaviour and restrictions. See section 2 for details.
ext_proc_call	string	"rgmii_expect()"	External procedure call. Only use when called from another BFM procedure.

BFM details

1 BFM procedure details

Procedure	Description
rgmii_write()	**rgmii_write (data_array, msg, rgmii_tx_if, [scope, [msg_id_panel, [config]]])** While config parameter data_valid_on_both_clock_edges = true (default): The rgmii_write() procedure writes 4 bits of data on each clock edge. The bits 3:0 are written on the rising edge and the bits 7:4 on the falling edge. While config parameter data_valid_on_both_clock_edges = false: The rgmii_write() procedure writes 4 bits of data on each rising clock edge. The bits 3:0 are written on the first rising edge and the bits 7:4 on the following rising edge. The length and data are defined by the “data_array” argument, which is a t_byte_array. data_array(0) is written first, while data_array(data_array’high) is written last. The default value of action_when_transfer_is_done is RELEASE_LINE_AFTER_TRANSFER which drives the TXEN low at the end of the procedure. However, if HOLD_LINE_AFTER_TRANSFER is used, the TXEN will be held at the end of the procedure.
rgmii_read()	**rgmii_read (data_array, data_len, msg, rgmii_rx_if, [scope, [msg_id_panel, [config, [ext_proc_call]]]])** While config parameter data_valid_on_both_clock_edges = true (default): The rgmii_read() procedure reads 4 bits of data on each clock edge. The bits 3:0 are read on the rising edge and the bits 7:4 on the falling edge. To avoid having to delay the receiver’s clock, the config rx_clock_skew is used to set the sampling time of the data. While config parameter data_valid_on_both_clock_edges = false: The rgmii_read() procedure reads 4 bits of data on each rising clock edge. The bits 3:0 are read on the first rising edge and the bits 7:4 on the following rising edge. The received data is stored in the data_array output, which is a t_byte_array. The number of valid bytes in the data_array is stored in data_len. data_array(0) is read first, while data_array(data_array’high) is read last.
rgmii_expect()	**rgmii_expect (data_exp, msg, rgmii_rx_if, [alert_level, [scope, [msg_id_panel, [config]]]])** Calls the rgmii_read() procedure, then compares the received data with data_exp.
init_rgmii_if_signals()	**init_rgmii_if_signals(VOID)** This function initializes the RGMII interface. All the BFM outputs are set to zeros ('0')

2 BFM Configuration record

Type name: t_rgmii_bfm_config

Record element	Type	C_RGMII_BFM_CONFIG_DEFAULT	Description
max_wait_cycles	integer	10	Used for setting the maximum cycles to wait before an alert is issued when waiting for signals from the DUT.
max_wait_cycles_severity	t_alert_level	ERROR	Severity if max_wait_cycles expires.
clock_period	time	-1 ns	Period of the clock signal.
rx_clock_skew	time	-1 ns	Skew of the sampling of the data in connection to the RX clock edges. Suggested value is clock_period/4.
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.
id_for_bfm	t_msg_id	ID_BFM	The message ID used as a general message ID in the BFM.
data_valid_on_both_clock_edges	boolean	true	Switch for changing between double data rate and single data rate on rgmii interface

3 Compilation

The RGMII 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 rgmii_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.

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.

```
rgmii_write(v_data_array(0 to 1), "msg");
```

rather than

```
rgmii_write(v_data_array(0 to 1), "msg", rgmii_tx_if, C_SCOPE, shared_msg_id_panel, C_RGMII_CONFIG_LOCAL);
```

By defining the local overload as e.g.:

```
procedure rgmii_write(
  constant data_array : in t_byte_array;
  constant msg        : in string) is
begin
    rgmii_write(data_array,                 -- keep as is
                msg,                        -- keep as is
                clk,                        -- Clock signal
```

```
                    rgmii_tx_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_RGMII_CONFIG_LOCAL);      -- Use locally defined configuration
  end;
```

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

- 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

See section 5 for defining a BFM configuration to use with the local overload and when calling the BFM procedures.

5 Local BFM configuration

The RGMII BFM requires that a local configuration is declared in the testbench and used in the BFM procedure calls. The default BFM configuration is defined with a clock period of -1 ns so that the BFM can detect and alert the user that the configuration has not been set. See section 2 for the RGMII BFM configuration record fields.

Defining a local RGMII BFM configuration:

```
constant C_RGMII_CONFIG_LOCAL : t_rgmii_bfm_config := (
  max_wait_cycles          => 10,
  max_wait_cycles_severity => ERROR,
  clock_period             => 8 ns,
  rx_clock_skew            => 2 ns,
  match_strictness         => MATCH_EXACT,
  id_for_bfm               => ID_BFM
);
```

See section 4 for how to define a local overload procedure and how to use a BFM config with the procedure call.

IMPORTANT
This is a simplified Bus Functional Model (BFM) for RGMII. The given BFM complies with the basic RGMII protocol and thus allows a normal access towards an RGMII interface. This BFM is not RGMII protocol checker. For a more advanced BFM please contact Bitvis AS at support@bitvis.no