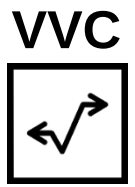
RGMII VVC – Quick Reference

NOTE: As of UVVM v3.x, all shared variables have been made protected. This means that any access to shared variables must be done using get- and set-methods. This documentation has not yet been updated with the methods for accessing these variables, but will be very soon. Please refer to section 2 of Avalon_mm_vvc_QuickRef for example usage of protected shared variables



For general information see UVVM VVC Framework Essential Mechanisms located in uvvm_vvc_framework/doc. **CAUTION**: shaded code/description is preliminary.

rgmii_vvc.vhd

rgmii_write (VVCT, vvc_instance_idx, channel, data_array, msg, [scope])

Example: rgmii_write(RGMII_VVCT, 0, TX, v_data_array(0 to v_numBytes-1), HOLD_LINE_AFTER_TRANSFER, “Write v_numBytes to DUT”, C_SCOPE);

Example: rgmii_write(RGMII_VVCT, 0, TX, (x”01”, x”02”, x”03”, x”04”), “Write 4 bytes to DUT”);

rgmii_read (VVCT, vvc_instance_idx, channel, [TO_SB,] msg, [scope])

Example: rgmii_read(RGMII_VVCT, 1, RX, “Read data which is stored in VVC and will be fetched later using fetch_result() “);
rgmii_read(RGMII_VVCT, 1, RX, TO_SB, “Read data which is stored in VVC and will be fetched later using fetch_result() “);

rgmii_expect (VVCT, vvc_instance_idx, channel, data_exp, msg, [alert_level, [scope]])

Example: rgmii_expect(RGMII_VVCT, 1, RX, v_data_array(0 to v_numBytes-1), “Expect v_numBytes from DUT”, ERROR, C_SCOPE);

Example: rgmii_expect(RGMII_VVCT, 1, RX, (x”01”, x”02”, x”03”, x”04”), “Expect 4 bytes from DUT”);

VHDL 2008 only

RGMII VVC Configuration record ´**vvc_config**´ **--** accessible via **shared_rgmii_vvc_config**

Record element	Type	C_RGMII_VVC_CONFIG_DEFAULT
inter_bfm_delay	t_inter_bfm_delay	C_RGMII_INTER_BFM_DELAY_DEFAULT
cmd_queue_count_max	natural	C_CMD_QUEUE_COUNT_MAX
cmd_queue_count_threshold	natural	C_CMD_QUEUE_COUNT_THRESHOLD
cmd_queue_count_threshold_severity	t_alert_level	C_CMD_QUEUE_COUNT_THRESHOLD_SEVERITY
result_queue_count_max	natural	C_RESULT_QUEUE_COUNT_MAX
result_queue_count_threshold	natural	C_RESULT_QUEUE_COUNT_THRESHOLD
result_queue_count_threshold_severity	t_alert_level	C_RESULT_QUEUE_COUNT_THRESHOLD_SEVERITY
bfm_config	t_rgmii_bfm_config	C_RGMII_BFM_CONFIG_DEFAULT
msg_id_panel	t_msg_id_panel	C_VVC_MSG_ID_PANEL_DEFAULT

RGMII VVC Status record signal ´**vvc_status**´ **--** accessible via **shared_rgmii_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_[any]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™

VVC target parameters

Name	Type	Example(s)	Description
VVCT	t_vvc_target_record	RGMII_VVCT	VVC target type compiled into each VVC in order to differentiate between VVCs.
vvc_instance_idx	integer	0	Instance number of the VVC
channel	t_channel	TX, RX	The VVC channel of the VVC instance

VVC functional 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.
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_VVC"	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 ".

VVC entity signals

Name	Type	Description
rgmii_vvc_tx_if	t_rgmii_tx_if	See RGMII BFM documentation.
rgmii_vvc_rx_if	t_rgmii_rx_if	See RGMII BFM documentation.

VVC entity generic constants

Name	Type	Default	Description
GC_INSTANCE_IDX	natural	-	Instance number to assign the VVC.
GC_RGMII_BFM_CONFIG	t_rgmii_bfm_config	C_RGMII_BFM_CONFIG_DEFAULT	Configuration for the RGMII BFM, see RGMII 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

Procedure	Description
rgmii_write()	**rgmii_write (VVCT, vvc_instance_idx, channel, data_array, msg, [scope])** The rgmii_write() VVC procedure adds a write command to the RGMII VVC executor queue, which will run as soon as all preceding commands have completed. When the command is scheduled to run, the executor calls the RGMII BFM rgmii_write() procedure, described in the RGMII BFM QuickRef.

rgmii_read()	**rgmii_read (VVCT, vvc_instance_idx, channel, [TO_SB,] msg, [scope])**
	
	The rgmii_read() VVC procedure adds a read command to the RGMII VVC executor queue, which will run as soon as all preceding commands have completed. When the command is scheduled to run, the executor calls the RGMII BFM rgmii_read() procedure, described in the RGMII BFM QuickRef.
	
	The value received from the DUT will not be returned in this procedure call since it is non-blocking for the sequencer/caller, but the received data and metadata will be stored in the VVC for a potential future fetch (see example with *fetch_result* below).
	
	If the option TO_SB is applied, the received data will be sent to the RGMII dedicated scoreboard. There it is checked against the expected value (provided by the testbench).
	
	Example with fetch_result() call: Result is placed in **`v_result`**
	`variable v_cmd_idx : natural;  -- Command index for the last receive`
	`variable v_result  : work.vvc_cmd_pkg.t_vvc_result; -- Result from read (data and metadata)`
	`(…)`
	`rgmii_read(RGMII_VVCT, 1, RX, "Read data in VVC");`
	`v_cmd_idx := get_last_received_cmd_idx(RGMII_VVCT, 1, RX);`
	`await_completion(RGMII_VVCT, 1, RX, 1 ms, "Wait for read to finish");`
	`fetch_result(RGMII_VVCT, 1, RX, v_cmd_idx, v_result, "Fetching result from read operation");`

rgmii_expect()	**rgmii_expect (VVCT, vvc_instance_idx, channel, data_exp, msg, [alert_level, [scope]])** The rgmii_expect() VVC procedure adds an expect command to the RGMII VVC executor queue, which will run as soon as all preceding commands have completed. When the command is scheduled to run, the executor calls the RGMII BFM rgmii_expect() procedure, described in the RGMII BFM QuickRef.

2 VVC Configuration

Record element	Type	C_RGMII_VVC_CONFIG_DEFAULT	Description
inter_bfm_delay	t_inter_bfm_delay	C_RGMII_INTER_BFM_DELAY_DEFAULT	Delay between any requested BFM accesses towards the DUT. - 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). - TIME_FINISH2START: Time from a BFM end to the next BFM start. 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_CMD_QUEUE_COUNT_MAX	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_rgmii_bfm_config	C_RGMII_BFM_CONFIG_DEFAULT	Configuration for RGMII BFM. See quick reference for RGMII BFM.
msg_id_panel	t_msg_id_panel	C_VVC_MSG_ID_PANEL_DEFAULT	VVC dedicated message ID panel. See section 16 of uvvm_vvc_framework/doc/UVVM_VVC_Framework_Essential_Mechanisms.pdf for how to use verbosity control.

The configuration record can be accessed from the Central Testbench Sequencer through the shared variable array, e.g.:

```
shared_rgmii_vvc_config(1).inter_bfm_delay.delay_in_time := 50 ns;
shared_rgmii_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_rgmii_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 Activity watchdog

The VVCs support a centralized VVC activity register which the activity watchdog uses to monitor the VVC activities. The VVCs will register their presence to the VVC activity register at start-up, and report when ACTIVE and INACTIVE, using dedicated VVC activity register methods, and trigger the global_trigger_vvc_activity_register signal during simulations. The activity watchdog is continuously monitoring the VVC activity register for VVC inactivity and raises an alert if no VVC activity is registered within the specified timeout period.

Include `activity_watchdog(num_exp_vvc, timeout, [alert_level, [msg]])` in the testbench to start using the activity watchdog.
Note that setting the exact number of expected VVCs in the VVC activity register can be omitted by setting `num_exp_vvc = 0`.

More information can be found in UVVM Essential Mechanisms PDF in the UVVM VVC Framework doc folder.

5 Transaction Info

This VVC supports transaction info, a UVVM concept for distributing transaction information in a controlled manner within the complete testbench environment. The transaction info may be used in many different ways, but the main purpose is to share information directly from the VVC to a DUT model.
See UVVM VVC Framework Essential Mechanisms PDF, section 6, for additional information.

*Table 1 RGMII transaction info record fields. Transaction type: t_base_transaction (BT) **-** accessible via **shared_rgmii_vvc_transaction_info.bt**.*

Info field	Type	Default	Description
operation	t_operation	NO_OPERATION	Current VVC operation, e.g. INSERT_DELAY, POLL_UNTIL, READ, WRITE.
data_array	t_byte_array(0 to 299)	(others => (others => '0'))	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.
vvc_meta	t_vvc_meta	C_VVC_META_DEFAULT	VVC meta data of the executing VVC command.
→ msg	string	" "	Message of executing VVC command.
→ cmd_idx	integer	-1	Command index of executing VVC command.
transaction_status	t_transaction_status	C_TRANSACTION_STATUS_DEFAULT	Set to INACTIVE, IN_PROGRESS, FAILED or SUCCEEDED during a transaction.

6 Scoreboard

This VVC has built in Scoreboard functionality where data can be routed by setting the `TO_SB` parameter in supported method calls. Note that the data is only stored in the scoreboard and not accessible with the `fetch_result()` method when the `TO_SB` parameter is applied.

See the Generic Scoreboard Quick Reference PDF in the Bitvis VIP Scoreboard document folder for a complete list of available commands and additional information. The RGMII VVC scoreboard is accessible from the testbench as a shared variable `RGMII_VVC_SB`, located in the `vvc_methods_pkg.vhd`. All of the listed Generic Scoreboard commands are available for the RGMII VVC scoreboard using this shared variable.

7 VVC Interface

In this VVC, the interface has been encapsulated in two signal records of type *t_rgmii_tx_if* for the signals going to the DUT and *t_rgmii_rx_if* for the signals coming from the DUT in order to improve readability of the code.

8 Additional Documentation

Additional documentation about UVVM and its features can be found under “/uvvm_vvc_framework/doc/”.

For additional documentation on the RGMII standard, see the specification “Reduced Gigabit Media Independent Interface (RGMII) Version 2.0”.

9 Compilation

RGMII VVC must be compiled with VHDL 2008.
It is dependent on the following libraries

- ***UVVM Utility Library (UVVM-Util), version 2.15.0 and up***
- ***UVVM VVC Framework, version 2.11.0 and up***
- ***RGMII BFM***
- ***Bitvis VIP Scoreboard***

Before compiling the RGMII VVC, assure that uvvm_vvc_framework, uvvm_util and bitvis_vip_scoreboard have been compiled.

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

Compile order for the RGMII VVC:

Compile to library	File	Comment
bitvis_vip_rgmii	rgmii_bfm_pkg.vhd	RGMII BFM
bitvis_vip_rgmii	transaction_pkg.vhd	RGMII transaction package with DTT types, constants etc.
bitvis_vip_rgmii	vvc_cmd_pkg.vhd	RMGII VVC command types and operations
bitvis_vip_rgmii	../uvvm_vvc_framework/src_target_dependent/td_target_support_pkg.vhd	UVVM VVC target support package, compiled into the RGMII VVC library.
bitvis_vip_rgmii	../uvvm_vvc_framework/src_target_dependent/td_vvc_framework_common_methods_pkg.vhd	UVVM framework common methods compiled into the RGMII VVC library
bitvis_vip_rgmii	vvc_methods_pkg.vhd	RGMII VVC methods
bitvis_vip_rgmii	../uvvm_vvc_framework/src_target_dependent/td_queue_pkg.vhd	UVVM queue package for the VVC
bitvis_vip_rgmii	../uvvm_vvc_framework/src_target_dependent/td_vvc_entity_support_pkg.vhd	UVVM VVC entity support compiled into the RGMII VVC library
bitvis_vip_rgmii	rgmii_tx_vvc.vhd	RGMII TX VVC
bitvis_vip_rgmii	rgmii_rx_vvc.vhd	RGMII RX VVC
bitvis_vip_rgmii	rgmii_vvc.vhd	RGMII VVC

10 Simulator compatibility and setup

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

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