

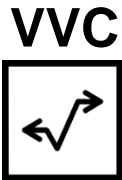
GMII VVC – Quick Reference

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

gmii_write (VVCT, vvc_instance_idx, channel, data_array, msg, [action_when_transfer_is_done], [scope])

Example: `gmii_write(GMII_VVCT, 0, TX, v_data_array(0 to v_numBytes-1), "Write v_numBytes to DUT", HOLD_LINE_AFTER_TRANSFER, C_SCOPE);`

Example: `gmii_write(GMII_VVCT, 0, TX, (x"01", x"02", x"03", x"04"), "Write 4 to DUT");`



gmii_vvc.vhd

gmii_read (VVCT, vvc_instance_idx, channel, [num_bytes], [TO_SB,] msg, [scope])

Example: `gmii_read(GMII_VVCT, 1, RX, 10, "Read 10 bytes of data", C_SCOPE);`

Example: `gmii_read(GMII_VVCT, 1, RX, "Read data and store it in the VVC. To be fetched later using fetch_result()");`
`gmii_read(GMII_VVCT, 1, RX, TO_SB, "Read data and send to Scoreboard for checking");`

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

Example: `gmii_expect(GMII_VVCT, 1, RX, v_data_array(0 to v_numBytes-1), "Expect v_numBytes from DUT", ERROR, C_SCOPE);`

Example: `gmii_expect(GMII_VVCT, 1, RX, (x"01", x"02", x"03", x"04"), "Expect 4 bytes from DUT");`

GMII VVC Configuration record `'vvc_config'` -- accessible via `shared_gmii_vvc_config`

Record element	Type	C_GMII_VVC_CONFIG_DEFAULT
inter_bfm_delay	t_inter_bfm_delay	C_GMII_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_gmii_bfm_config	C_GMII_BFM_CONFIG_DEFAULT
msg_id_panel	t_msg_id_panel	C_VVC_MSG_ID_PANEL_DEFAULT
unwanted_activity_severity	t_alert_level	C_UNWANTED_ACTIVITY_SEVERITY

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()`

GMII VVC Status record signal `'vvc_status'` -- accessible via `shared_gmii_vvc_status`

Record element	Type
current_cmd_idx	natural
previous_cmd_idx	natural
pending_cmd_cnt	natural



VVC target parameters

Name	Type	Example(s)	Description
VVCT	t_vvc_target_record	GMII_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_slv_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: <pre>variable v_data_array : t_slv_array(0 to C_MAX_BYTES-1) (7 downto 0);</pre>
num_bytes	positive	16	Number of bytes to be read.
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 to DUT"	A custom message to be appended in the log/alert
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, e.g. from an Ethernet HVVC.
scope	string	"GMII_VVC"	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".

VVC entity signals

Name	Type	Description
gmii_vvc_tx_if	t_gmii_tx_if	See GMII BFM documentation.
gmii_vvc_rx_if	t_gmii_rx_if	See GMII BFM documentation.

VVC entity generic constants

Name	Type	Default	Description
GC_INSTANCE_IDX	natural	-	Instance number to assign the VVC.
GC_GMII_BFM_CONFIG	t_gmii_bfm_config	C_GMII_BFM_CONFIG_DEFAULT	Configuration for the GMII BFM, see GMII 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 GC_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 and examples

Procedure	Description
gmii_write()	gmii_write (VVCT, vvc_instance_idx, channel, data_array, msg, [action_when_transfer_is_done], [scope]) The <code>gmii_write()</code> VVC procedure adds a write command to the GMII 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 GMII BFM <code>gmii_write()</code> procedure, described in the GMII BFM QuickRef.
gmii_read()	gmii_read (VVCT, vvc_instance_idx, channel, [num_bytes], [TO_SB,] msg, [scope]) The <code>gmii_read()</code> VVC procedure adds a read command to the GMII 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 GMII BFM <code>gmii_read()</code> procedure, described in the GMII 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 <i>fetch_result</i> below). If the option <code>TO_SB</code> is applied, the received data will be sent to the GMII VVC dedicated scoreboard. There, it is checked against the expected value (provided by the testbench). Example with <code>fetch_result()</code> call: Result is placed in <code>v_result</code> <pre>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) (...) gmii_read(GMII_VVCT, 1, RX, "Read data in VVC"); v_cmd_idx := get_last_received_cmd_idx(GMII_VVCT, 1, RX); await_completion(GMII_VVCT, 1, RX, 1 ms, "Wait for read to finish"); fetch_result(GMII_VVCT, 1, RX, v_cmd_idx, v_result, "Fetching result from read operation");</pre>
gmii_expect()	gmii_expect (VVCT, vvc_instance_idx, channel, data_exp, msg, [alert_level, [scope]]) The <code>gmii_expect()</code> VVC procedure adds an expect command to the GMII 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 GMII BFM <code>gmii_expect()</code> procedure, described in the GMII BFM QuickRef.

2 VVC Configuration

Record element	Type	C GMII VVC CONFIG DEFAULT	Description
inter_bfm_delay	t_inter_bfm_delay	C_GMII_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_gmii_bfm_config	C_GMII_BFM_CONFIG_DEFAULT	Configuration for GMII BFM. See quick reference for GMII 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.
unwanted_activity_severity	t_alert_level	C_UNWANTED_ACTIVITY_SEVERITY	Severity of alert to be initiated if unwanted activity on the DUT TX outputs is detected. Unwanted activity detection is enabled (ERROR) by default.

Note: cmd/result queue parameters in the VVC Configuration are unused and will be removed in v3.0, use instead the entity generic constants.

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

```
shared_gmii_vvc_config(1).inter_bfm_delay.delay_in_time := 50 ns;
shared_gmii_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_gmii_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.

Table 5.1 GMII transaction info record fields. Transaction type: `t_base_transaction (BT)` - accessible via ***shared_gmii_vvc_transaction_info.bt***.

Info field	Type	Default	Description
operation	<code>t_operation</code>	<code>NO_OPERATION</code>	Current VVC operation, e.g. <code>INSERT_DELAY</code> , <code>POLL_UNTIL</code> , <code>READ</code> , <code>WRITE</code> .
data_array	<code>t_slv_array(0 to 2047)(7 downto 0)</code>	(others => (others => '0'))	An array of bytes containing the data to be written/read. <code>data_array(0)</code> is written/read first, while <code>data_array(data_array'high)</code> is written/read last.
vvc_meta	<code>t_vvc_meta</code>	<code>C_VVC_META_DEFAULT</code>	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	<code>t_transaction_status</code>	<code>C_TRANSACTION_STATUS_DEFAULT</code>	Set to <code>INACTIVE</code> , <code>IN_PROGRESS</code> , <code>FAILED</code> or <code>SUCCEEDED</code> during a transaction.

See UVVM VVC Framework Essential Mechanisms PDF, section 6, for additional information about transaction types and transaction info usage.

6 Scoreboard

This VVC has built in Scoreboard functionality where data can be routed by setting the `T0_SB` parameter in supported method calls, i.e. `gmii_read()`. Note that the data is only stored in the scoreboard and not accessible with the `fetch_result()` method when the `T0_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 GMII scoreboard is accessible from the testbench as a shared variable `GMII_VVC_SB`, located in the `vvc_methods_pkg.vhd`. All of the listed Generic Scoreboard commands are available for the GMII VVC scoreboard using this shared variable.

7 VVC Interface

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

8 Unwanted Activity Detection

This VVC supports detection of unwanted activity from the DUT. This mechanism will give an alert if the DUT generates any unexpected bus activity. It assures that no data is output from the DUT when it is not expected, i.e. GMII read/expect VVC methods are not called. Once the VVC is inactive, it starts to monitor continuously on the DUT TX outputs. When unwanted activity is detected, the VVC issues an alert of severity. Note that this feature is only implemented on `gmii_rx_vvc`.

The unwanted activity detection can be configured from the central testbench sequencer, where the severity of alert can be changed to a different value.

To disable this feature in the testbench, e.g.:

```
shared_gmii_vvc_config(RX, C_VVC_INDEX).unwanted_activity_severity := NO_ALERT;
```

The unwanted activity detection is ignored when the `rxdv` signal goes low within one clock period after the VVC becomes inactive. This is to handle the situation when the read command exits before the next rising edge, causing signal transitions during the first clock cycle after the VVC is inactive.

For GMII VVC, the unwanted activity detection is enabled (`unwanted_activity_severity := ERROR`) by default.

9 Additional Documentation

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

For additional documentation on the GMII standard, please see the GMII BFM QuickRef

10 Compilation

The GMII 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**
- **GMII BFM**
- **Bitvis VIP Scoreboard**

Before compiling the GMII VVC, assure that `uvvm_vvc_framework`, `uvvm_util` and `bitvis_vip_scorebord` have been compiled.

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

Compile order for the GMII VVC:

Compile to library	File	Comment
bitvis_vip_gmii	gmii_bfm_pkg.vhd	GMII BFM
bitvis_vip_gmii	transaction_pkg.vhd	GMII transaction package with DTT types, constants, etc.
bitvis_vip_gmii	vvc_cmd_pkg.vhd	GMII VVC command types and operations
bitvis_vip_gmii	../uvvm_vvc_framework/src_target_dependent/td_target_support_pkg.vhd	UVVM VVC target support package, compiled into the GMII VVC library.
bitvis_vip_gmii	../uvvm_vvc_framework/src_target_dependent/td_vvc_framework_common_methods_pkg.vhd	UVVM framework common methods compiled into the GMII VVC library
bitvis_vip_gmii	vvc_methods_pkg.vhd	GMII VVC methods
bitvis_vip_gmii	../uvvm_vvc_framework/src_target_dependent/td_queue_pkg.vhd	UVVM queue package for the VVC
bitvis_vip_gmii	../uvvm_vvc_framework/src_target_dependent/td_vvc_entity_support_pkg.vhd	UVVM VVC entity support compiled into the GMII VVC library
bitvis_vip_gmii	gmii_tx_vvc.vhd	GMII TX VVC
bitvis_vip_gmii	gmii_rx_vvc.vhd	GMII RX VVC
bitvis_vip_gmii	gmii_vvc.vhd	GMII VVC

11 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 GMII. The given VIP complies with the basic GMII protocol and thus allows a normal access towards a GMII interface. This VIP is not a GMII protocol checker. For a more advanced VIP please contact UVVM at info@uvvm.no

INTELLECTUAL PROPERTY

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.