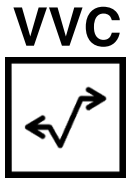
GMII VVC – Quick Reference

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



gmii_vvc.vhd

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

Example: gmii_write(GMII_VVCT, 0, TX, v_data_array(0 to v_numBytes-1), “Write v_numBytes to DUT”, C_SCOPE);

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

gmii_read (VVCT, vvc_instance_idx, channel, [num_bytes], 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 which is stored in VVC and will be fetched later using fetch_result()”);

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

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

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_THERSHOLD_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

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

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()

VHDL 2008 only

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_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);.`
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
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 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 and examples

Procedure	Description
gmii_write()	**gmii_write (VVCT, vvc_instance_idx, channel, data_array, msg, [scope])** The gmii_write() 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 gmii_write() procedure, described in the GMII BFM QuickRef.

gmii_read()	**gmii_read (VVCT, vvc_instance_idx, channel, [num_bytes], msg, [scope])**
	
	The gmii_read() 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 gmii_read() 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 *fetch_result* below).
	
	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)`
	`(…)`
	`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");`

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

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 an activity watchdog which monitors VVC activity and will alert if no VVC activity is registered within a selected timeout value. The VVCs will register their presence to the activity watchdog at start-up, and report when busy and not, using dedicated activity watchdog methods and triggering the global_trigger_activity_watchdog signal, during simulations.

Include `activity_watchdog(num_exp_vvc, timeout, alert_level, msg)` in the testbench to start using the activity watchdog.
More information can be found in UVVM Essential Mechanisms PDF in the UVVM VVC Framework doc folder.

5 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.

6 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.

7 Compilation

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

- ***UVVM Utility Library (UVVM-Util), version 2.12.0 and up***
- ***UVVM VVC Framework, version 2.7.3 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

8 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 Bitvis AS at support@bitvis.no