GPIO 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

VVC

gpio_vvc.vhd

gpio_set (VVCT, vvc_instance_idx, data, msg, [scope])

Example: gpio_set(GPIO_VVCT, 1, C_BAUDRATE_9600, “Set baudrate to 9600”);

gpio_get (VVCT, vvc_instance_idx, [TO_SB,] msg, [scope])

Example: gpio_get(GPIO_VVCT, 1, “Read GPIO baudrate, and store result in VVC. To be retrieved using fetch_result()”);
gpio_get(GPIO_VVCT, 1, TO_SB, “Read GPIO baudrate and send result to scoreboard for checking”);

gpio_check (VVCT, vvc_instance_idx, data_exp, msg, [alert_level, [scope]])

Example: gpio_check(GPIO_VVCT, 1, x”3B”, “Check data from UART RX”, ERROR);

gpio_check_stable (VVCT, vvc_instance_idx, data_exp, stable_req, msg, [alert_level, [scope]])

Example: gpio_check_stable(GPIO_VVCT, 1, x”3B”, 100 us, “Check data from UART RX has been stable for 100 us”, ERROR);

gpi_expect (VVCT, vvc_instance_idx, data_exp, timeout, msg, [alert_level, [scope]])

Example: gpio_expect(GPIO_VVCT, 1, x"0D", 2 ms, “Read UART RX until CR is found or timeout”, ERROR);

gpi_expect_stable (VVCT, vvc_instance_idx, data_exp, stable_req, stable_req_from, timeout, msg, [alert_level, [scope]])

Example: gpio_expect_stable(GPIO_VVCT, 1, x"0D", 100 us, FROM_NOW, 2 ms, “Read UART RX until CR is found and check that it remains stable for 100 us”, ERROR);

Common VVC procedures applicable for this VVC
- See UVVM Methods QuickRef for details.

Name
await_[any]completion()
enable_log_msg()
disable_log_msg()
fetch_result()
flush_command_queue()
terminate_current_command()

GPIO VVC Configuration record ‘vvc_config’
- Accessible via **shared_gpio_vvc_config** – see section 2.

Record element
inter_bfm_delay
[cmd/result]_queue_count_max
[cmd/result]_queue_count_threshold
[cmd/result]_queue_count_threshold_severity
bfm_config
msg_id_panel

GPIO VVC Status record signal ‘vvc_status’
- Accessible via **shared_gpio_vvc_status** – see section 3.

Record element
current_cmd_idx
previous_cmd_idx
pending_cmd_cnt

VHDL 2008 only

terminate_all_commands()
insert_delay()
get_last_received_cmd_idx()

VVC target parameters

Name	Type	Example(s)	Description
VVCT	t_vvc_target_record	GPIO_VVCT	VVC target type compiled into each VVC in order to differentiate between VVCs.
vvc_instance_idx	integer	1	Instance number of the VVC

VVC functional parameters

Name	Type	Example(s)	Description
data	std_logic_vector	x"FF"	The data to be written.
data_exp	std_logic_vector	x"FF"	The expected data to be read.
stable_req	time	1 ms	The time that the expected data value should remain stable in the register.
stable_req_from	t_from_point_in_time	FROM NOW	The point in time where stable_req starts.
timeout	time	10 ms	The maximum time to pass before the expected data must be found. A timeout result in an alert 'alert_level'.
msg	string	"Set baudrate"	A custom message to be appended in the log/alert.
alert_level	t_alert_level	ERROR or TB_WARNING	Set the severity for the alert that may be asserted by the method.
scope	string	"GPIO_VVC"	A string describing the scope from which the log/alert originates. In a simple single sequencer typically "GPIO_BFM". In a verification component typically "GPIO_VVC ".

VVC entity signals

Name	Type	Direction	Description
gpio_vvc_if	std_logic_vector	Inout	This interface can be used as input, output or inout. If the GC_DEFAULT_LINE_VALUE is set to 'Z', 'H' or 'L', the interface can be used as an input, output or inout. If the default is set to '1', '0' or 'U', the interface must be used as an output. When using the interface as inout, the process currently driving the signal (DUT or VVC) must set it to 'Z' after it is finished in order to release the interface so that the other process can drive it.

VVC entity generic constants

Name	Type	Default	Description
GC_DATA_WIDTH	natural	-	Bits in the GPIO data word
GC_INSTANCE_IDX	natural	-	Instance number to assign the VVC

GC_DEFAULT_LINE_VALUE	std_logic_vector	-	Default value of input or output GPIO.
GC_GPIO_BFM_CONFIG	t_gpio_bfm_config	C_GPIO_BFM_CONFIG_DEFAULT	Configuration for the GPIO BFM, see GPIO 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
gpio_set()	**gpio_set (VVCT, vvc_instance_idx, data, msg, [scope])** The gpio_set() VVC procedure adds a SET command to the GPIO VVC executor queue, that will run as soon as all preceding commands have completed. When the command is scheduled to run, the executor calls the GPIO BFM gpio_set() procedure, described in the GPIO BFM QuickRef. Example: `gpio_set(GPIO_VVCT, 1, C_BAUDRATE_9600, "Set baudrate to 9600", C_SCOPE);`

gpio_get()

gpio_get (VVCT, vvc_instance_idx, [TO_SB,] msg, [scope])

The gpio_get() VVC procedure adds a GET command to the GPIO VVC executor queue, that will run as soon as all preceding commands have completed. When the command is scheduled to run, the executor calls the GPIO BFM gpio_get() procedure, described in the GPIO BFM QuickRef. The received data from DUT will not be returned in this procedure call since it is non-blocking for the sequencer/caller, but the received data 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 GPIO VVC dedicated scoreboard. There, it is checked against the expected value (provided by the testbench).

Example:

```
gpio_get(GPI_VVCT, 1, "Read baudrate", C_SCOPE);
```

Example with fetch_result() call: Result is placed in **v_data**

```
variable v_cmd_idx         : natural;                       -- Command index for the last read
variable v_data            : std_logic_vector(31 downto 0); -- Result from read
(…)
gpio_get(GPIO_VVCT, 1, "Read baudrate");
v_cmd_idx := get_last_received_cmd_idx(GPIO_VVCT, 1);
await_completion(GPIO_VVCT,1, v_cmd_idx, 1 us, "Wait for receive to finish");
fetch_result(GPIO_VVCT,1, v_cmd_idx, v_data, "Fetching result from receive operation");
```

gpio_check()

gpio_check (VVCT, vvc_instance_idx, data_exp, msg, [alert_level, [scope]])

The gpio_check() VVC procedure adds a CHECK command to the GPIO VVC executor queue, that will run as soon as all preceding commands have completed. When the command is scheduled to run, the executor calls the GPIO BFM gpio_check() procedure, described in the GPIO BFM QuickRef.

Example:

```
gpio_check(GPIO_VVCT, 1, x"F5", "Check data from UART RX", ERROR, C_SCOPE);
```

gpio_check_stable()

gpio_check_stable (VVCT, vvc_instance_idx, data_exp, stable_req, msg, [alert_level, [scope]])

The gpio_check_stable() VVC procedure adds a CHECK_STABLE command to the GPIO VVC executor queue, that will run as soon as all preceding commands have completed. When the command is scheduled to run, the executor calls the GPIO BFM gpio_check_stable() procedure, described in the GPIO BFM QuickRef.

Example:

```
gpio_check_stable(GPIO_VVCT, 1, x"F5", 100 us, "Check data from UART RX has been stable for 100 us", ERROR, C_SCOPE);
```

gpio_expect()

gpio_expect (VVCT, vvc_instance_idx, data_exp, timeout, msg, [alert_level, [scope]])

The gpio_expect() VVC procedure adds a EXPECT command to the GPIO VVC executor queue, that will run as soon as all preceding commands have completed. When the command is scheduled to run, the executor calls the GPIO BFM gpio_expect() procedure, described in the GPIO BFM QuickRef.

Example:

```
gpio_expect(GPIO_VVCT, 1, x"0D", 2 ms, "Read UART RX until CR is found or timeout", ERROR, C_SCOPE);
```

gpio_expect_stable()

gpio_expect_stable (VVCT, vvc_instance_idx, data_exp, stable_req, stable_req_from, timeout, msg, [alert_level, [scope]])

The gpio_expect_stable() VVC procedure adds a EXPECT_STABLE command to the GPIO VVC executor queue, that will run as soon as all preceding commands have completed. When the command is scheduled to run, the executor calls the GPIO BFM gpio_expect_stable() procedure, described in the GPIO BFM QuickRef.

Example:

```
gpio_expect_stable(GPIO_VVCT, 1, x"0D", 100 us, FROM_NOW, 2 ms, "Read UART RX until CR is found and check that it remains stable
                   for 100 us", ERROR, C_SCOPE);
```

2 VVC Configuration

Record element	Type	C_GPIO_VVC_CONFIG_DEFAULT	Description
inter_bfm_delay	t_inter_bfm_delay	C_GPIO_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 triggered if command count 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_gpio_bfm_config	C_GPIO_BFM_CONFIG_DEFAULT	Configuration for GPIO BFM. See QuickRef for GPIO 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_gpio_vvc_config(C_VVC_IDX).inter_bfm_delay.delay_in_time := 10 ms;
```

3 VVC Status

The current status of the VVC can be retrieved during simulation. This is achieved by reading from the shared variable shared_gpio_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 4.1 GPIO transaction info record fields. Transaction type: t_base_transaction (BT) **- accessible via shared_gpio_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	slv(31 downto 0)	0x0	The data to be written (in gpio_set).
data_exp	slv(31 downto 0)	0x0	The expected data to be read (in gpio_check or gpio_expect).
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.

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 `TO_SB` parameter in supported method calls, i.e. `gpio_get()`. 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.

The GPIO scoreboard is per default a 32 bits wide standard logic vector. When sending expected data to the scoreboard, where the data width is smaller than the default scoreboard width, we recommend zero-padding the data with the pad_gpio_sb() function. I.e. `GPIO_VVC_SB.add_expected(<GPIO VVC instance number>, pad_gpio_sb(<exp data>));`

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 GPIO VVC scoreboard is accessible from the testbench as a shared variable `GPIO_VVC_SB`, located in the `vvc_methods_pkg.vhd`. All of the listed Generic Scoreboard commands are available for the GPIO VVC scoreboard using this shared variable.

7 Additional Documentation

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

8 Compilation

The GPIO 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***
- ***GPIO BFM***
- ***Bitvis VIP Scoreboard***

Before compiling the GPIO VVC, make sure 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 GPIO VVC:

Compile to library	File	Comment
bitvis_vip_gpio	gpio_bfm_pkg.vhd	GPIO BFM
bitvis_vip_gpio	transaction_pkg.vhd	GPIO transaction package with DTT types, constants etc.
bitvis_vip_gpio	vvc_cmd_pkg.vhd	GPIO VVC command types and operations
bitvis_vip_gpio	../uvvm_vvc_framework/src_target_dependent/td_target_support_pkg.vhd	UVVM VVC target support package, compiled into the GPIO VVC library.
bitvis_vip_gpio	../uvvm_vvc_framework/src_target_dependent/td_vvc_framework_common_methods_pkg.vhd	UVVM framework common methods compiled into the GPIO VVC library
bitvis_vip_gpio	vvc_methods_pkg.vhd	GPIO VVC methods
bitvis_vip_gpio	../uvvm_vvc_framework/src_target_dependent/td_queue_pkg.vhd	UVVM queue package for the VVC
bitvis_vip_gpio	../uvvm_vvc_framework/src_target_dependent/td_vvc_entity_support_pkg.vhd	UVVM VVC entity methods compiled into the GPIO VVC library
bitvis_vip_gpio	gpio_vvc.vhd	GPIO VVC

9 Simulator compatibility and setup

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