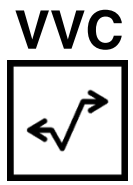
SBI VVC – Quick Reference

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



sbi_vvc.vhd

sbi_write (VVCT, vvc_instance_idx, addr, data | { num_words, randomisation}, msg, [scope])

Example: sbi_write(SBI_VVCT, 1, x"1000", x"40", "Set baud rate to 9600");
sbi_write(SBI_VVCT, 1, x"1001", 7, RANDOM, "Write 7 random bytes to UART TX");

sbi_read (VVCT, vvc_instance_idx, addr, [TO_SB,] msg, [scope])

Example: sbi_read(SBI_VVCT, 1, x"1000", "Read baud rate");
sbi_read(SBI_VVCT, 1, x"1002", TO_SB, "Read UART RX and send to Scoreboard", C_SCOPE);

sbi_check (VVCT, vvc_instance_idx, addr, data, msg, [alert_level, [scope]])

Example: sbi_check(SBI_VVCT, 1, x"1155, x"3B", "Check data from UART RX");

sbi_poll_until (VVCT, vvc_instance_idx, addr, data, msg, [max_polls, [timeout, [alert_level, [scope]]]])

Example: sbi_poll_until(SBI_VVCT, 1, x"1155", x"0D", "Read UART RX until CR is found");

SBI VVC Configuration record ´**vvc_config**´ -- accessible via **shared_sbi_vvc_config**

Record element	Type	C_SBI_VVC_CONFIG_DEFAULT
inter_bfm_delay	t_inter_bfm_delay	C_SBI_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_sbi_bfm_config	C_SBI_BFM_CONFIG_DEFAULT
msg_id_panel	t_msg_id_panel	C_VVC_MSG_ID_PANEL_DEFAULT

SBI VVC Status record signal ´**vvc_status**´ -- accessible via **shared_sbi_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_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™
VHDL 2008 only

VVC target parameters

Name	Type	Example(s)	Description
VVCT	t_vvc_target_record	SBI_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
addr	unsigned	x"5A"	The address of a SW accessible register. Could be offset or full address depending on the DUT
data	std_logic_vector	x"D3"	The data to be written (in sbi_write) or the expected data (in sbi_check/sbi_poll_until).
msg	string	"Read from DUT"	A custom message to be appended in the log/alert
timeout	time	100 ns	Timeout to be used in the sbi_poll_until BFM procedure. 0 ns means no timeout.
max_polls	integer	1	Maximum number of polls allowed in the sbi_poll_until procedure. 0 means no limit.
alert_level	t_alert_level	ERROR or TB_WARNING	Set the severity for the alert that may be asserted by the procedure.
scope	string	"SBI VVC"	A string describing the scope from which the log/alert originates. In a simple single sequencer typically "SBI BFM". In a verification component typically "SBI VVC ".

VVC entity signals

Name	Type	Direction	Description
clk	std_logic	Input	VVC Clock signal
sbi_vvd_master_if	t_sbi_if	Inout	See SBI BFM documentation

VVC entity generic constants

Name	Type	Default	Description
GC_ADDR_WIDTH	integer	8	Width of the SBI address bus
GC_DATA_WIDTH	integer	32	Width of the SBI data bus
GC_INSTANCE_IDX	natural	1	Instance number to assign the VVC
GC_SBI_CONFIG	t_sbi_bfm_config	C_SBI_BFM_CONFIG_DEFAULT	Configuration for the SBI BFM, see SBI 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

sbi_write()	**sbi_write(VVCT, vvc_instance_idx, addr, data | { num_words, randomisation}, msg, [scope])**
	
	The sbi_write() VVC procedure adds a write command to the SBI VVC executor queue, which will run as soon as all preceding commands have completed.
	The sbi_write() command has two variants using either just data for a basic single transaction, or num_words + randomisation for a more advanced version.
	When the basic write command is scheduled to run, the executor calls the SBI BFM sbi_write() procedure, described in the SBI BFM QuickRef.
	When the more advanced randomisation command is applied the basic BFM sbi_write() transaction is executed num_words times with new random data each time – according to the given randomisation profile.
	Current defined randomisation profiles are: RANDOM: Standard uniform random. This is provided as an example.
	
	Example:
	`sbi_write(SBI_VVCT, 1, x"1000", x"40", "Set UART baud rate to 9600", C_SCOPE);`
	`sbi_write(SBI_VVCT, 1, x"1001", 7, RANDOM, "Write 7 random bytes to UART TX");`
	It is recommended to use constants to improve the readability of the code, e.g.:
	`sbi_write(SBI_VVCT, 1, C_ADDR_UART_BAUDRATE, C_BAUDRATE_9600, "Set UART baud rate to 9600");`

sbi_read()

sbi_read (VVCT, vvc_instance_idx, addr, [TO_SB,] msg, [scope])

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

Example:

```
sbi_read(SBI_VVCT, 1, x"1000", "Read UART baud rate", C_SCOPE);
sbi_read(SBI_VVCT, 1, x"1002", TO_SB, "Read UART RX and send to Scoreboard", C_SCOPE);
```

It is recommended to use constants to improve the readability of the code, e.g.:

```
sbi_read(SBI_VVCT, 1, C_ADDR_UART_BAUDRATE, "Read UART baud rate");
```

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            : work.vvc_cmd_pkg.t_vvc_result; -- Result from read.
(…)
sbi_read(SBI_VVCT, 1, C_ADDR_UART_BAUDRATE, "Read from Peripheral 1");
v_cmd_idx := get_last_received_cmd_idx(SBI_VVCT, 1);
await_completion(SBI_VVCT,1, v_cmd_idx, 1 us, "Wait for read to finish");
fetch_result(SBI_VVCT,1, v_cmd_idx, v_data, "Fetching result from read operation");
```

sbi_check()

sbi_check (VVCT, vvc_instance_idx, addr, data, msg, [alert_level, [scope]])

The sbi_check() VVC procedure adds a check command to the SBI VVC executor queue, which will run as soon as all preceding commands have completed. When the check command is scheduled to run, the executor calls the SBI BFM sbi_check() procedure, described in the SBI BFM QuickRef. The sbi_check() procedure will perform a read operation, then check if the read data is equal to the expected data in the 'data' parameter. If the read data is not equal to the expected 'data' parameter, an alert with severity 'alert_level' will be issued. The read data will not be stored in this procedure.

Examples:

```
sbi_check(SBI_VVCT, 1, x"1155, x"3B", "Check data from UART RX");
sbi_check(SBI_VVCT, 1, x"1155, x"3B", "Check data from UART RX", TB_ERROR, C_SCOPE):
```

It is recommended to use constants to improve the readability of the code, e.g.:

```
sbi_check(SBI_VVCT, 1, C_ADDR_UART_RX, C_UART_START_BYTE, "Check data from UART RX");
```

sbi_poll_until()

sbi_poll_until (VVCT, vvc_instance_idx, addr, data, msg, [max_polls, [timeout, [alert_level, [scope]]]])

The sbi_poll_until() VVC procedure adds a poll_until command to the SBI VVC executor queue, which will run as soon as all preceding commands have completed. When the write command is scheduled to run, the executor calls the SBI BFM sbi_poll_until() procedure, described in the SBI BFM QuickRef. The sbi_poll_until() procedure will perform a read operation, then check if the read data is equal to the data in the 'data' parameter. If the read data is not equal to the expected 'data' parameter, the process will be repeated until the read data is equal to the expected data, or the procedure is terminated by either a terminate command, a timeout or the poll limit set in max_polls.
The read data will not be stored by this procedure.

Examples:

```
sbi_poll_until(SBI_VVCT, 1, x"1155", x"0D", "Read UART RX until CR is found");
sbi_poll_until(SBI_VVCT, 1, x"1155", x"0D", "Read UART RX until CR is found", 5, 0 ns, TB_WARNING, C_SCOPE);
```

It is recommended to use constants to improve the readability of the code, e.g.:

```
sbi_poll_until(SBI_VVCT, 1, C_ADDR_UART_RX, C_CR_BYTE, "Read UART RX until CR is found");
```

2 VVC Configuration

Record element	Type	C_SBI_BFM_CONFIG_DEFAULT	Description
inter_bfm_delay	t_inter_bfm_delay	C_SBI_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_sbi_bfm_config	C_SBI_BFM_CONFIG_DEFAULT	Configuration for SBI BFM. See quick reference for SBI 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_sbi_vvc_config(1).inter_bfm_delay.delay_in_time := 50 ns;
shared_sbi_vvc_config(1).bfm_config.id_for_bfm         := ID_BFM;
```

3 VVC Status

The current status of the VVC can be retrieved during simulation. This is achieved by reading from the shared variable shared_sbi_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 SBI transaction info record fields. Transaction Type: t_base_transaction (BT) - accessible via **shared_sbi_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.
address	unsigned(31 downto 0)	0x0	Address of the SBI read or write transaction.
data	slv(31 downto 0)	0x0	Data for SBI read or write transaction.
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.

*Table 5.2 SBI transaction info record fields. Transaction type: t_compound_transaction (CT)) - accessible via **shared_sbi_vvc_transaction_info.ct**.*

DTT field	Type	Default	Description
operation	t_operation	NO_OPERATION	Current VVC operation, e.g. INSERT_DELAY, POLL_UNTIL, READ, WRITE.
address	unsigned(31 downto 0)	0x0	Address of the SBI read or write transaction.
data	slv(31 downto 0)	0x0	Data for SBI read or write transaction.
randomisation	t_randomisation	NA	sbi_write() will generate random data when set to RANDOM.
num_words	natural	1	Use with randomisation to write a *num_words* number of random words using a single sbi_write() command.
max_polls	integer	1	Maximum number of polls allowed in the sbi_poll_until procedure. 0 means no limit.
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. `sbi_read()`. 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 SBI scoreboard is accessible from the testbench as a shared variable SBI_VVC_SB, located in the `vvc_methods_pkg.vhd`. E.g. `SBI_VVC_SB.add_expected(C_SBI_VVC_IDX, pad_sb_slv(v_expected), "Adding expected");`

The SBI scoreboard is per default a 128 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_sb_slv() function. E.g. `SBI_VVC_SB.add_expected(<SBI VVC instance number>, pad_sb_slv(<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. All of the listed Generic Scoreboard commands are available for the SBI VVC scoreboard using the `SBI_VVC_SB`.

7 VVC Interface

In this VVC, the interface has been encapsulated in a signal record of type *t_sbi_if* in order to improve readability of the code. Since the SBI interface busses can be of arbitrary size, the interface vectors have been left unconstrained. These unconstrained vectors need to be constrained when the interface signals are instantiated. For this interface, it could look like:

```
signal sbi_if_1 : t_sbi_if( addr (C_ADDR_WIDTH-1 downto 0),
                            wdata(C_DATA_WIDTH-1 downto 0),
                            rdata(C_DATA_WIDTH-1 downto 0) );
```

8 Additional Documentation

Additional documentation about UVVM and its features can be found under "/uvvm_vvc_framework/doc/".
For additional documentation on the SBI protocol, please see the SBI BFM QuickRef.

9 Compilation

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

- ***UVVM Utility Library (UVVM-Util), version 2.13.0 and up***
- ***UVVM VVC Framework, version 2.8.0 and up***
- ***SBI BFM***
- ***Bitvis VIP Scoreboard***

Before compiling the SBI 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 SBI VVC:

Compile to library	File	Comment
bitvis_vip_sbi	sbi_bfm_pkg.vhd	SBI BFM
bitvis_vip_sbi	transaction_pkg.vhd	SBI transaction package with DTT types, constants etc.
bitvis_vip_sbi	vvc_cmd_pkg.vhd	SBI VVC command types and operations
bitvis_vip_sbi	../uvvm_vvc_framework/src_target_dependent/td_target_support_pkg.vhd	UVVM VVC target support package, compiled into the SBI VVC library.
bitvis_vip_sbi	../uvvm_vvc_framework/src_target_dependent/td_vvc_framework_common_methods_pkg.vhd	Common UVVM framework methods compiled into the SBI VVC library
bitvis_vip_sbi	vvc_methods_pkg.vhd	SBI VVC methods
bitvis_vip_sbi	../uvvm_vvc_framework/src_target_dependent/td_queue_pkg.vhd	UVVM queue package for the VVC
bitvis_vip_sbi	../uvvm_vvc_framework/src_target_dependent/td_vvc_entity_support_pkg.vhd	UVVM VVC entity support compiled into the SBI VVC library
bitvis_vip_sbi	sbi_vvc.vhd	SBI 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 SBI.
The given VIP complies with the basic SBI protocol and thus allows a normal access towards a SBI interface. This VIP is not a SBI protocol checker.
For a more advanced VIP please contact Bitvis AS at support@bitvis.no