SPI VVC – Quick Reference

VVC

spi_vvc.vhd

spi_master|slave_transmit_and_receive (VVCT, vvc_instance_idx, data, msg)

Master example: spi_master_transmit_and_receive(SPI_VVCT, 1, x"AF", “Sending data to Peripheral 1 and receiving data from Peripheral 1”);

spi_master|slave_transmit_only (VVCT, vvc_instance_idx, data, msg)

Slave example: spi_slave_transmit_only(SPI_VVCT, 1, x"AF", “Sending data to Peripheral 1”);

spi_master|slave_receive_only (VVCT, vvc_instance_idx, msg)

Master example: spi_master_receive_only(SPI_VVCT, 1, “Receive from Peripheral 1”);

spi_master|slave_transmit_and_check (VVCT, vvc_instance_idx, data, data_exp, msg, [alert_level])

Slave example: spi_slave_transmit_and_check(SPI_VVCT, 1, x"42", x"AF", “Sending data to Peripheral 1 and expecting data from Peripheral 1”);

spi_master|slave_check_only (VVCT, vvc_instance_idx, data_exp, msg, [alert_level])

Master example: spi_master_check_only(SPI_VVCT, 1, x"42",“Expect data from Peripheral 1”);

Common VVC procedures applicable for this VVC

- See UVVM Methods QuickRef for details.

Name
await_completion()
await_any_completion()
enable_log_msg()
disable_log_msg()
flush_command_queue()
terminate_current_command()
fetch_result()
insert_delay()

SPI VVC Configuration record ‘t_vvc_config’

- Accessible via **shared_spi_vvc_config** – se section 2.

Name
inter_bfm_delay
[cmd/result]_queue_count_max
[cmd/result]_queue_count_threshold
[cmd/result]_queue_count_threshold_severity
bfm_config
msg_id_panel

SPI VVC Status record signal ‘t_vvc_status’

- Accessible via **shared_spi_vvc_status** – se section 3.

Name
current_cmd_idx
previous_cmd_idx
pending_cmd_idx

UVVM™

VHDL 2008 only

VVC target parameters

Name	Type	Example(s)	Description
VVCT	t_vvc_target_record	SPI_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 transmitted (in spi_<master/slave>_transmit_and_check or spi_<master/slave>_transmit_only).
data_exp	std_logic_vector	x"FF"	The expected data to be received (in spi_<master/slave>_transmit_and_check or spi_<master/slave>_check_only).
msg	string	"Send to peripheral 1"	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.

VVC entity signals

Name	Type	Direction	Description
spi_vvc_if	t_spi_if	Inout	See SPI BFM documentation

VVC entity generic constants

Name	Type	Default	Description
GC_DATA_WIDTH	natural	8	Bits in the SPI data word
GC_INSTANCE_IDX	natural	1	Instance number to assign the VVC
GC_MASTER_MODE	boolean	TRUE	Whether the VVC shall act as an SPI master or an SPI slave on the bus.
GC_SPI_CONFIG	t_spi_bfm_config	C_SPI_BFM_CONFIG_DEFAULT	Configuration for the SPI BFM, see SPI 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 command 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.uvvm_methods_pkg and uvvm_vvc_framework.uvvm_support_pkg (common VVC procedures)
It is also possible to send a multicast to all instances of a VVC with C_VVCT_ALL_INSTANCES as parameter for vvc_instance_idx.

1 VVC procedure details and examples

Procedure	Description
spi_master_transmit_and_receive()	The spi_master_transmit_and_receive() VVC procedure adds a transmit and receive command to the SPI VVC executor queue, that will run as soon as all preceding commands have completed. When the transmit and receive command is scheduled to run, the executor calls the SPI BFM spi_master_transmit_and_receive() procedure, described in the SPI BFM QuickRef. There is one requirement for running the spi_master_transmit_and_receive() procedure: - The VVC entity with instance index corresponding to the 'instance_idx' parameter must have the generic constant GC_MASTER_MODE set to TRUE. **spi_master_transmit_and_receive (VVCT, instance_idx, data, msg)** e.g.: - spi_master_transmit_and_receive (SPI_VVCT, 1, x"0D", "Transmitting carriage return to Peripheral 1 and receiving data from Peripheral 1");
spi_master_transmit_only()	The spi_master_transmit_only() VVC procedure adds a transmit command to the SPI VVC executor queue, that will run as soon as all preceding commands have completed. When the transmit command is scheduled to run, the executor calls the SPI BFM spi_master_transmit_and_receive() procedure, described in the SPI BFM QuickRef. The SPI BFM spi_master_transmit() procedure will ignore the received data from the slave DUT. There is one requirement for running the spi_master_transmit_only() procedure: - The VVC entity with instance index corresponding to the 'instance_idx' parameter must have the generic constant GC_MASTER_MODE set to TRUE. **spi_master_transmit_only (VVCT, instance_idx, data, msg)** e.g.: - spi_master_transmit_only (SPI_VVCT, 1, x"0D", "Transmitting carriage return to Peripheral 1");

spi_master_receive_only()

The spi_master_receive_only() VVC procedure adds a receive command to the SPI VVC executor queue, that will run as soon as all preceding commands have completed. When the receive command is scheduled to run, the executor calls the SPI BFM spi_master_receive() procedure, described in the SPI 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). The SPI BFM spi_master_transmit() procedure will transmit dummy data (0x0) while receiving data from the slave DUT.

There is one requirement for running the spi_master_receive_only() procedure:

- The VVC entity with instance index corresponding to the 'instance_idx' parameter must have the generic constant GC_MASTER_MODE set to TRUE.

spi_master_receive_only (VVCT, instance_idx, msg)
e.g.

- spi_master_receive_only (SPI_VVCT, 1, "Receiving from Peripheral 1");

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
(…)
    spi_master_receive_only(SPI_VVCT, 1, "Receiving from Peripheral 1");
    v_cmd_idx := shared_cmd_idx;
    await_completion(SPI_VVCT,1, v_cmd_idx, 1 us, "Wait for receive to finish");
    fetch_result(SPI_VVCT,1, v_cmd_idx, v_data, "Fetching result from receive operation");
```

spi_master_transmit_and_check()

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

There is one requirement for running the spi_master_transmit_and_check() procedure:

- The VVC entity with instance index corresponding to the 'instance_idx' parameter must have the generic constant GC_MASTER_MODE set to TRUE.

spi_master_transmit_and_check (VVCT, instance_idx, data, data_exp, msg)
e.g.:

- spi_master_transmit_and_check (SPI_VVCT, 1, x"0D", x"5F", "Transmitting carriage return to Peripheral 1 and expecting data from Peripheral 1");

The procedure can also be called with the optional parameters, e.g.:

- spi_master_transmit_and_check (SPI_VVCT, 1, C_CR_BYTE, x"5F", "Transmitting carriage return to Peripheral 1 and expecting data from Peripheral 1", ERROR);

spi_master_check_only()

The spi_master_check_only() VVC procedure adds a check command to the SPI 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 SPI BFM spi_master_check() procedure, described in the SPI BFM QuickRef. The received data will not be stored by this procedure and the SPI BFM spi_master_check() procedure will transmit dummy data (0x0) while receiving data from the slave DUT.

There is one requirement for running the spi_master_check_only() procedure:

- The VVC entity with instance index corresponding to the 'instance_idx' parameter must have the generic constant GC_MASTER_MODE set to TRUE.

spi_master_check_only (VVCT, instance_idx, data, msg, [alert_level])
e.g.

- spi_master_check_only (SPI_VVCT, 1, x"0D", "Expecting carriage return from Peripheral 1");

The procedure can also be called with the optional parameters, e.g.:

- spi_master_check_only (SPI_VVCT, 1, C_CR_BYTE, "Expecting carriage return from Peripheral 1", ERROR);

spi_slave_transmit_and_receive()

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

There is one requirement for running the spi_slave_transmit_and_reveice () procedure:

- The VVC entity with instance index corresponding to the 'instance_idx' parameter must have the generic constant GC_MASTER_MODE set to FALSE.

spi_slave_transmit_and_receive (VVCT, instance_idx, data, msg)
e.g.:

- spi_slave_transmit_and_receive (SPI_VVCT, 1, x"0D", "Transmitting carriage return to Peripheral 1 and receiving data from Peripheral 1");

spi_slave_transmit_only()

The spi_slave_transmit_only() VVC procedure adds a transmit command to the SPI VVC executor queue, that will run as soon as all preceding commands have completed. When the transmit command is scheduled to run, the executor calls the SPI BFM spi_slave_transmit () procedure, described in the SPI BFM QuickRef. The SPI BFM spi_slave_transmit() procedure will ignore the data received from the master DUT.

There is one requirement for running the spi_slave_transmit () procedure:

- The VVC entity with instance index corresponding to the 'instance_idx' parameter must have the generic constant GC_MASTER_MODE set to FALSE.

spi_slave_transmit_only (VVCT, instance_idx, data, msg)
e.g.:

- spi_slave_transmit_only (SPI_VVCT, 1, x"0D", "Transmitting carriage return to Peripheral 1");

spi_slave_receive_only()

The spi_slave_receive_only() VVC procedure adds a receive command to the SPI RX VVC executor queue, that will run as soon as all preceding commands have completed. When the receive command is scheduled to run, the executor calls the SPI BFM spi_slave_receive () procedure, described in the SPI BFM QuickRef.
The received data 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). The SPI BFM spi_slave_receive() procedure will transmit dummy data (0x0) while receiving data from the master DUT.

There is one requirement for running the spi_slave_receive_only() procedure:

- The VVC entity with instance index corresponding to the 'instance_idx' parameter must have the generic constant GC_MASTER_MODE set to FALSE.

spi_slave_receive_only (VVCT, instance_idx, msg)
e.g.

- spi_slave_receive_only (SPI_VVCT, 1, "Receiving from Peripheral 1");

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
 (…)
    spi_slave_receive_only(SPI_VVCT, 1, "Receiving from Peripheral 1");
    v_cmd_idx := shared_cmd_idx;
    await_completion(SPI_VVCT,1, v_cmd_idx, 1 us, "Wait for receive to finish");
    fetch_result(SPI_VVCT,1, v_cmd_idx, v_data, "Fetching result from receive operation");
```

spi_slave_transmit_and_check()

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

There is one requirement for running the spi_slave_transmit_and_check() procedure:

- The VVC entity with instance index corresponding to the 'instance_idx' parameter must have the generic constant GC_MASTER_MODE set to FALSE.

spi_slave_transmit_and_check (VVCT, instance_idx, data, data_exp, msg)
e.g.:

- spi_slave_transmit_and_check (SPI_VVCT, 1, x"0D", x"5F", "Transmitting carriage return to Peripheral 1 and expecting data from Peripheral 1");

The procedure can also be called with the optional parameters, e.g.:

- spi_slave_transmit_and_check (SPI_VVCT, 1, C_CR_BYTE, x"5F", "Transmitting carriage return to Peripheral 1 and expecting data from Peripheral 1", ERROR);

spi_slave_check_only()

The spi_slave_check_only() VVC procedure adds a check command to the SPI 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 SPI BFM spi_slave_check() procedure, described in the SPI BFM QuickRef. The received data will not be stored by this procedure and the SPI BFM spi_slave_check() procedure will transmit dummy data (0x0) while receiving data from the master DUT.

There is one requirement for running the spi_slave_check_only() procedure:

- The VVC entity with instance index corresponding to the 'instance_idx' parameter must have the generic constant GC_MASTER_MODE set to FALSE.

spi_slave_check_only (VVCT, instance_idx, data, msg, [alert_level])
e.g.

- spi_slave_check_only(SPI_VVCT, 1, x"0D", "Expecting carriage return from Peripheral 1");

The procedure can also be called with the optional parameters, e.g.:

- spi_slave_check_only(SPI_VVCT, 1, C_CR_BYTE, "Expecting carriage return from Peripheral 1", ERROR);

2 VVC Configuration

Name	Type	C_SPI_VVC_CONFIG_DEFAULT	Description
inter_bfm_delay	t_inter_bfm_delay	C_SPI_INTER_BFM_DELAY_DEFAULT	Specified delay between BFM accesses from the VVC. If parameter delay_type is set to NO_DELAY, BFM accesses will be back to back, i.e. no delay.
cmd_queue_count_max	natural	C_MAX_COMMAND_QUEUE	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 command 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_spi_bfm_config	C_SPI_BFM_CONFIG_DEFAULT	Configuration for SPI BFM. See QuickRef for SPI 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_spi_vvc_config(C_VVC_IDX_MASTER_1).inter_bfm_delay.delay_in_time := 10 ms;
shared_spi_vvc_config(C_VVC_IDX_SLAVE_1).bfm_config.CPOL                := '1';
```

3 VVC Status

The current status of the VVC can be retrieved during simulation. This is done by reading from the shared variable shared_spi_vvc_status record from the test sequencer. The record contains status for both channels, specified with the channel axis of the shared_spi_vvc_status array.The record contents can be seen below:

Name	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 Additional Documentation

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

5 Compilation

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

- ***UVVM Utility Library (UVVM-Util), version 1.0.0 and up***
- ***UVVM VVC Framework, version 2.0.0 and up***
- ***SPI BFM***

Before compiling the SPI VVC, make sure that uvvm_vvc_framework and uvvm_util have been compiled.

Compile order for the SPI VVC:

Compile to library	File	Comment
bitvis_vip_spi	spi_bfm_pkg.vhd	SPI BFM
bitvis_vip_spi	vvc_cmd_pkg.vhd	SPI VVC command types and operations
bitvis_vip_spi	../uvvm_vvc_framework/src_target_dependent/td_target_support_pkg.vhd	UVVM VVC target support package, compiled into the SPI VVC library.
bitvis_vip_spi	../uvvm_vvc_framework/src_target_dependent/td_vvc_framework_common_methods_pkg.vhd	UVVM framework common methods compiled into the SPI VVC library
bitvis_vip_spi	vvc_methods_pkg.vhd	SPI VVC methods
bitvis_vip_spi	../uvvm_vvc_framework/src_target_dependent/td_queue_pkg.vhd	UVVM queue package for the VVC
bitvis_vip_spi	../uvvm_vvc_framework/src_target_dependent/td_vvc_entity_support_pkg.vhd	UVVM VVC entity methods compiled into the SPI VVC library
bitvis_vip_spi	spi_vvc.vhd	SPI VVC wrapper for the RX and TX VVCs

6 Simulator compatibility and setup

This VVC has been compiled and tested with Modelsim version 10.3d and Riviera-PRO version 2015.10.85.
For required simulator setup see ***UVVM-Util*** Quick reference.

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