Ethernet HVVC – Quick Reference

This ethernet VVC is based on IEEE 802.3. It does not support optional fields or EtherType, only length is supported.
For general information see UVVM Essential Mechanisms located in uvvm_vvc_framework/doc. **CAUTION**: shaded code/description is preliminary

HVVC

ethernet_hvvc.vhd

ethernet_send (VVCT, vvc_instance_idx, channel, [mac_dest], [mac_src], payload, msg, [scope, [use_provided_msg_id_panel, [msg_id_panel]]])

Example: ethernet_send(ETHERNET_VVCT, 1, TX, v_mac_dest, v_mac_src, v_payload, “Send ethernet packet”);

ethernet_receive (VVCT, vvc_instance_idx, channel, msg, [scope, [use_provided_msg_id_panel, [msg_id_panel]]])

Example: ethernet_receive(ETHERNET_VVCT, 1, RX, “Receive ethernet packet”);

ethernet_expect (VVCT, vvc_instance_idx, channel, [mac_dest], [mac_src], payload, msg, [alert_level])

Example: ethernet_expect(ETHERNET_VVCT, 1, RX, v_mac_dest, v_mac_src, v_payload, “Expect ethernet packet”, ERROR);

Ethernet VVC Configuration record ´**vvc_config**´ -- accessible via **shared_ethernet_vvc_config**

Record element	Type	C_ETHERNET_VVC_CONFIG_DEFAULT
inter_bfm_delay	t_inter_bfm_delay	C_ETHERNET_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[1]	t_ethernet_bfm_config	C_ETHERNET_BFM_CONFIG_DEFAULT
msg_id_panel	t_msg_id_panel	C_ETHERNET_HVVC_MSG_ID_PANEL_DEFAULT

BFM Configuration record ´**t_ethernet_bfm_config**´

Record element	Type	C_ETHERNET_BFM_CONFIG_DEFAULT
mac_destination	unsigned(47 downto 0)	x”000000000000”
mac_source	unsigned(47 downto 0)	x”000000000000”
fcs_error_severity	t_alert_level	TB_ERROR
interpacket_gap_time[2]	time	96 ns

Ethernet VVC Status record signal ´**vvc_status**´ -- accessible via **shared_ethernet_vvc_status**

Record element	Type
current_cmd_idx	natural
previous_cmd_idx	natural
pending_cmd_cnt	natural

Record ´**t_ethernet_frame**´

Record element	Type
mac_destination	unsigned(47 downto 0)
mac_source	unsigned(47 downto 0)
length	integer
payload	t_byte_array
fcs	std_logic_vector(31 downto 0)

[1] Not strictly a bus functional model (BFM), but holds BFM-like configuration data.
[2] Interpacket gap is implemented as a wait statement after the ethernet packet has been sent. Check of interpacket gap on receive is not implemented.

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	ETHERNET_VVCT	VVC target type compiled into each VVC in order to differentiate between VVCs.
vvc_instance_idx	integer	1	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
mac_dest	unsigned	x"00_00_00_00_00_02"	The MAC address of destination.
mac_src	unsigned	x"00_00_00_00_00_01"	The MAC address of source.
payload	t_byte_array	(x"01", x"23", x"45", x"AB", x"CD")	The payload containing data.

VVC transaction info

Name	Type	Description
operation	t_operation	Command operation type.
msg	string	Command message.
ethernet_frame	t_ethernet_frame	Ethernet frame.

VVC entity generic constants

Name	Type	Default	Description
GC_INSTANCE_IDX	natural	-	Instance number to assign the VVC
GC_INTERFACE	t_interface	-	VVC interface type, e.g. GMII.
GC_VVC_INSTANCE_IDX	natural		Instance number of the sub-VVC.
GC_DUT_IF_FIELD_CONFIG	t_dut_if_field_config_channel_array	C_DUT_IF_FIELD_CONFIG_CHANNEL_ARRAY_DEFAULT	Array of configurations for address-based VVC interfaces. See chapter 4 for details.
GC_ETHERNET_BFM_CONFIG	t_ethernet_bfm_config	C_ETHERNET_BFM_CONFIG_DEFAULT	Configuration of Ethernet.
GC_CMD_QUEUE_COUNT_MAX	natural	1000	Absolute maximum number of commands in the HVVC 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 to 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 ALL_INSTANCES as parameter for vvc_instance_idx.

1 VVC procedure details and examples

Procedure	Description

ethernet_send()	**ethernet_send (VVCT, vvc_instance_idx, channel, [mac_dest], [mac_src], payload, msg)**
	
	The ethernet_send() VVC procedure adds a send command to the Ethernet VVC executor queue, which runs as soon as all preceding commands have completed.
	
	Example:
	`ethernet_send(ETHERNET_VVCT, 1, TX, v_mac_dest, v_mac_src, v_payload, "Send ethernet packet");`
	`ethernet_send(ETHERNET_VVCT, 1, TX, v_payload, "Send ethernet packet");`

ethernet_receive()	**ethernet_receive (VVCT, vvc_instance_idx, channel, msg)**
	
	The ethernet_receive() VVC procedure adds a receive command to the Ethernet VVC executor queue, which runs as soon as all preceding commands have completed.
	The value read from the DUT is not be returned in this procedure call since it is non-blocking for the sequencer/caller, but the read data is stored in the VVC for a potential future fetch (see example with *fetch_result* below).
	
	Example:
	`ethernet_receive(ETHERNET_VVCT, 1, RX, "Receive ethernet packet");`
	
	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 : bitvis_vip_ethernet.vvc_cmd_pkg.t_vvc_result; -- Result from read.`
	`(…)`
	`ethernet_receive(ETHERNET_VVCT, 1, RX, "Receive ethernet package");`
	`v_cmd_idx := get_last_received_cmd_idx(ETHERNET_VVCT, 1, RX);`
	`await_completion(ETHERNET_VVCT,1, RX, v_cmd_idx, 1 us, "Wait for receive to finish");`
	`fetch_result(ETHERNET_VVCT, 1, RX, v_cmd_idx, v_data, "Fetching result from receive operation");`

ethernet_expect()	**ethernet_expect (VVCT, vvc_instance_idx, channel, [mac_dest], [mac_src], payload, msg, [alert_level])** The ethernet_expect() VVC procedure adds an expect command to the Ethernet VVC executor queue, which runs as soon as all preceding commands have completed. The ethernet_expect() procedure performs a read operation, then checks if the read data is equal to the expected data. If the read data is not equal to the expected data, an alert with severity 'alert_level' is issued. The read data is not be stored in this procedure. Examples: `ethernet_expect(ETHERNET_VVCT, 1, RX, v_mac_dest, v_mac_src, v_payload", "Expect data from ethernet");`

2 VVC Configuration

Record element	Type	C_ETHERNET_BFM_CONFIG_DEFAULT	Description
inter_bfm_delay	t_inter_bfm_delay	C_ETHERNET_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 adds 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_ethernet_bfm_config	C_ETHERNET_BFM_CONFIG_DEFAULT	Configuration for Ethernet BFM.
msg_id_panel	t_msg_id_panel	C_ETHERNET_HVVC_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_ethernet_vvc_config(1).inter_bfm_delay.delay_in_time := 50 ns;
shared_ethernet_vvc_config(1).bfm_config.id_for_bfm         := ID_BFM;
```

The index in shared_ethernet_vvc_config corresponds with the instance number of the VVC.

See section 16 of uvvm_vvc_framework/doc/UVVM_VVC_Framework_Essential_Mechanisms.pdf for how to use verbosity control when debugging simulations.

3 VVC Status

The current status of the VVC can be retrieved during simulation. This is achieved by reading from the shared variable shared_ethernet_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_testcase_inactivity_watchdog signal, during simulations.

Include `activity_watchdog(num_exp_vvc, timeout, alert_level, msg)` in the testbench to start using the activity watchdog.
Note that each channel is counted in the number of registered VVCs in the activity watchdog register, thus the Ethernet VVC is counted as two VVCs.
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.
See UVVM VVC Framework Essential Mechanisms PDF, section 6, for additional information.

Info field	Type	Default	Description
operation	t_operation	NO_OPERATION	Current VVC operation, e.g. INSERT_DELAY, POLL_UNTIL, READ, WRITE.
ethernet_frame	t_ethernet_frame	C_ETHERNET_FRAME_DEFAULT	Ethernet Frame.
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.

6 DUT interface field configuration

The table below shows which index in the DUT IF field configuration array the Ethernet fields are associated with. These configurations are only necessary when the lower level VVC is address-based. The DUT IF field configuration array is a two-dimensional array, dimensions channel and index. If the same configuration is used for all fields, only one configuration per channel is needed. The highest indexed configuration is used for indexes higher than those supplied. E.g. if the array consists of two configurations the first configuration, index 0, is used for the field preamble & SFD and the other fields use the last configuration, index 1. Each index holds an element of type t_dut_if_config, see table below.

The Ethernet interface fields are associated with the following indexes

Ethernet field	Name	Index
Preamble & SFD	C_IF_FIELD_NUM_ETHERNET_PREAMBLE_SFD	0
MAC destination	C_IF_FIELD_NUM_ETHERNET_MAC_DESTINATION	1
MAC source	C_IF_FIELD_NUM_ETHERNET_SOURCE	2
Length	C_IF_FIELD_NUM_ETHERNET_LENGTH	3
Payload	C_IF_FIELD_NUM_ETHERNET_PAYLOAD	4
FCS	C_IF_FIELD_NUM_ETHERNET_FCS	5

Record ´t_dut_if_field_config´

Record element	Type	Description
dut_address	unsigned	Address of the DUT IF field.
dut_address_increment	integer	Incrementation of the address on each access.
data_width	positive	The width of the data per transfer, must be <= bus width.
field_description	string	Description of the DUT IF field.

7 Additional Documentation

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

8 Compilation

The Ethernet HVVC must be compiled with VHDL 2008.
It is dependent on the following libraries

- ***UVVM Utility Library***
- ***UVVM VVC Framework***
- ***HVVC-to-VVC Bridge***
- ***Bitvis VIP Scoreboard***

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

Compile order for the Ethernet HVVC:

Compile to library	File	Comment
bitvis_vip_ethernet	ethernet_bfm_pkg.vhd	Ethernet BFM
bitvis_vip_ethernet	hvvc_cmd_pkg.vhd	Ethernet HVVC command types and operations
bitvis_vip_ethernet	../uvvm_vvc_framework/src_target_dependent/td_target_support_pkg.vhd	UVVM VVC target support package, compiled into bitvis_vip_ethernet library.
bitvis_vip_ethernet	../uvvm_vvc_framework/src_target_dependent/td_vvc_framework_common_methods_pkg.vhd	Common UVVM framework methods compiled into bitvis_vip_ethernet library
bitvis_vip_ethernet	hvvc_methods_pkg.vhd	Ethernet HVVC methods
bitvis_vip_ethernet	../uvvm_vvc_framework/src_target_dependent/td_queue_pkg.vhd	UVVM queue package for the HVVC
bitvis_vip_ethernet	../uvvm_vvc_framework/src_target_dependent/td_vvc_entity_support_pkg.vhd	UVVM VVC entity support compiled into bitvis_vip_ethernet library
bitvis_vip_ethernet	ethernet_transmit_hvvc.vhd	Ethernet transmit HVVC
bitvis_vip_ethernet	ethernet_receive_hvvc.vhd	Ethernet receive HVVC
bitvis_vip_ethernet	ethernet_hvvc.vhd	Ethernet HVVC

9 Simulator compatibility and setup

This VVC has been compiled and tested with Modelsim version 10.5b and Riviera-PRO version 2018.02.111.6909.
For required simulator setup see ***UVVM-Util*** Quick reference.

IMPORTANT
This is a simplified Verification IP (VIP) for Ethernet.
This Ethernet VVC is based on IEEE 802.3. It does not support optional fields or EtherType, only length is supported. This VIP is not an Ethernet protocol checker.
For a more advanced VIP please contact Bitvis AS at support@bitvis.no