Ethernet HVVC – Quick Reference

This Ethernet Hierarchical-VVC is based on IEEE 802.3. It does not support optional fields or EtherType, only length is supported.
HVVCs are different than normal VVCs since they represent a higher protocol level than the physical layer, i.e. they have no physical connections. However due to similarities in the core code, the VVC term is used instead.
For general information see UVVM Essential Mechanisms located in uvvm_vvc_framework/doc. **CAUTION**: shaded code/description is preliminary

HVVC

ethernet_vvc.vhd

ethernet_transmit (VVCT, vvc_instance_idx, channel, [mac_destination], [mac_source], payload, msg, [scope])

Example: ethernet_transmit(ETHERNET_VVCT, 0, TX, v_mac_dest, v_mac_src, v_payload, “Transmit an ethernet packet”, C_SCOPE);

Example: ethernet_transmit(ETHERNET_VVCT, 0, TX, v_payload, “Transmit an ethernet packet using default MAC addresses”);

ethernet_receive (VVCT, vvc_instance_idx, channel, [TO_SB], msg, [scope])

Example: ethernet_receive(ETHERNET_VVCT, 1, RX, “Receive an ethernet packet and store it in the VVC. To be fetched later using fetch_result()”, C_SCOPE);

Example: ethernet_receive(ETHERNET_VVCT, 1, RX, TO_SB, “Receive an ethernet packet and send to Scoreboard for checking”);

ethernet_expect (VVCT, vvc_instance_idx, channel, [mac_destination], [mac_source], payload, msg, [alert_level, [scope]])

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

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_protocol_config	C_ETHERNET_PROTOCOL_CONFIG_DEFAULT
msg_id_panel	t_msg_id_panel	C_ETHERNET_VVC_MSG_ID_PANEL_DEFAULT

Ethernet Protocol Configuration record ´**t_ethernet_protocol_config**´

Record element	Type	C_ETHERNET_PROTOCOL_CONFIG_DEFAULT
mac_destination	unsigned(47 downto 0)	x”000000000000”
mac_source	unsigned(47 downto 0)	x”000000000000”
fcs_error_severity	t_alert_level	ERROR
interpacket_gap_time[2,3]	time	96 ns

[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 transmitted. Check of interpacket gap on receive is not implemented.
[3] If the physical VVC has a timeout, e.g. max_wait_cycles, it must be big enough to handle the interpacket gap and any other delays in the transmission.

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()

UVVM™
VHDL 2008 only

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)
payload_length	integer
payload	t_byte_array
fcs	std_logic_vector(31 downto 0)

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	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
mac_destination	unsigned(47 downto 0)	x”00_00_00_00_00_02”	The MAC address of destination.
mac_source	unsigned(47 downto 0)	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 of the packet.
alert_level	t_alert_level	ERROR or TB_WARNING	Set the severity for the alert that may be asserted by the procedure.
msg	string	“Send to DUT”	A custom message to be appended in the log/alert.
scope	string	“Ethernet_VVC”	A string describing the scope from which the log/alert originates.

VVC entity generic constants

Name	Type	Default	Description
GC_INSTANCE_IDX	natural	-	Instance number to assign the VVC.
GC_PHY_INTERFACE	t_interface	-	Physical VVC interface type, e.g. SBI, GMII. (see note below)
GC_PHY_VVC_INSTANCE_IDX	natural	-	Instance number of the physical VVC.
GC_PHY_MAX_ACCESS_TIME	time	1 us	Maximum time that the physical VVC takes to execute an access, e.g. GMII write 1 byte. It should also account for any margin it needs.
GC_DUT_IF_FIELD_CONFIG	t_dut_if_field_config_direction_array	C_DUT_IF_FIELD_CONFIG_DIRECTION_ARRAY_DEFAULT	Array of configurations for address based VVC interfaces. See chapter 0 for details.
GC_ETHERNET_PROTOCOL_CONFIG	t_ethernet_protocol_config	C_ETHERNET_PROTOCOL_CONFIG_DEFAULT	Configuration of the Ethernet protocol.
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 GC_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

Note that you can use any of the physical interfaces already implemented just by using the appropriate name in GC_PHY_INTERFACE and instantiating the corresponding VVC in the testbench (in addition to the HVVC). For more information see *UVVM Essential Mechanisms* located in uvvm_vvc_framework/doc.
If you however want to use an interface type which is not already included, see *HVVC_to_VVC_Bridge_Implementation_Guide* located in bitvis_vip_hvvc_to_vvc_bridge/doc for more info.

VVC details

All VVC procedures are defined in vvc_methods_pkg (dedicated to 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
ethernet_transmit()	**ethernet_transmit (VVCT, vvc_instance_idx, channel, [mac_destination], [mac_source], payload, msg, [scope])** The ethernet_transmit() VVC procedure adds a transmit command to the Ethernet VVC executor queue, which runs as soon as all preceding commands have completed. When the command is scheduled to run, the executor calls the priv_ethernet_transmit_to_bridge() procedure. This procedure builds an Ethernet packet and transmits each field using the HVVC-to-VVC bridge which then transfers the data to the lower level VVC (physical interface). After it has finished, it waits for the configured interpacket gap time.

ethernet_receive()	**ethernet_receive (VVCT, vvc_instance_idx, channel, [TO_SB], msg, [scope])**
	
	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. When the command is scheduled to run, the executor calls the priv_ethernet_receive_from_bridge() procedure. This procedure receives an Ethernet packet by requesting each field from the HVVC-to-VVC bridge which calls the lower level VVC (physical interface) to read the data. When the complete packet is received, it computes the FCS and checks that it corresponds to the one received in the packet.
	
	The received data from the DUT is not to be returned in this procedure call since it is non-blocking for the sequencer/caller, but it 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 Ethernet VVC dedicated scoreboard. There, it is checked against the expected value (provided by the testbench).
	
	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 : bitvis_vip_ethernet.vvc_cmd_pkg.t_vvc_result;` `-- Result from receive.`
	`(…)`
	`ethernet_receive(ETHERNET_VVCT, 1, RX, "Receive ethernet packet");`
	`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_result, "Fetching result from receive operation");`

ethernet_expect()	**ethernet_expect (VVCT, vvc_instance_idx, channel, [mac_destination], [mac_source], payload, msg, [alert_level, [scope]])** 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. When the command is scheduled to run, the executor calls the priv_ethernet_expect_from_bridge() procedure. This procedure performs a receive operation, then checks if the received data is equal to the expected data. The received data is not stored in this procedure.

2 VVC Configuration

Record element	Type	C_ETHERNET_VVC_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_protocol_config	C_ETHERNET_PROTOCOL_CONFIG_DEFAULT	Configuration of the Ethernet protocol.
msg_id_panel	t_msg_id_panel	C_ETHERNET_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.

Note: cmd/result queue parameters in the VVC Configuration are unused and will be removed in v3.0, use instead the entity generic constants.

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.interpacket_gap_time := 96 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_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 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 Ethernet transaction info record fields. Transaction type: t_base_transaction (BT) - accessible via **shared_ethernet_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.
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.

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. `ethernet_receive()`. 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.

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

7 Unwanted Activity Detection

Since HVVCs do not contain any physical ports, the unwanted activity detection is found in the physical layer VVC connected to the HVVC, e.g. GMII/RGMII/SBI. Thus, when the data is not expected from the DUT, i.e. Ethernet VVC receive/expect methods are not called, an alert of severity will be generated from the physical layer VVCs.

The unwanted activity detection can be configured from the Central Testbench Sequencer, where the severity of alert can be changed to a different value.
To disable this feature in the testbench, e.g. for GMII VVC:

```
shared_gmii_vvc_config(RX, C_VVC_INDEX).unwanted_activity_severity := NO_ALERT;
```

Note that the unwanted activity detection is enabled (`unwanted_activity_severity := ERROR`) by default for the GMII/RGMII/SBI VVC.
For a VVC specific description of this feature, see the Unwanted Activity Detection section in each physical layer VVC QuickRef.

8 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, e.g. SBI. The DUT IF field configuration array is a two-dimensional array (direction and index). If the same configuration is used for all fields, only one configuration per direction 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_field_config, see table below.

The Ethernet interface fields are associated with the following indexes

Ethernet field	Name	Index
Preamble & SFD	C_FIELD_IDX_PREAMBLE_AND_SFD	0
MAC destination	C_FIELD_IDX_MAC_DESTINATION	1
MAC source	C_FIELD_IDX_MAC_SOURCE	2
Payload length	C_FIELD_IDX_PAYLOAD_LENGTH	3
Payload	C_FIELD_IDX_PAYLOAD	4
FCS	C_FIELD_IDX_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	Width of the data per transfer, must be <= bus width.
use_field	boolean	Used by the HVVC to send/request fields to/from the HVVC-to-VVC bridge or ignore them when not applicable.
field_description	string	Description of the DUT IF field.

9 Additional Documentation

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

10 Compilation

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

- ***UVVM Utility Library (UVVM-Util), version 2.19.5 and up***
- ***UVVM VVC Framework, version 2.12.7 and up***
- ***Bitvis VIP Scoreboard***
- ***Library of the physical interface used (e.g. Bitvis VIP GMII)***
- ***HVVC-to-VVC Bridge***

Before compiling the Ethernet 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 Ethernet VVC:

Compile to library	File	Comment
bitvis_vip_ethernet	support_pkg.vhd	Ethernet support package
bitvis_vip_ethernet	transaction_pkg.vhd	Ethernet transaction package with DTT types, constants, etc.
bitvis_vip_ethernet	vvc_cmd_pkg.vhd	Ethernet VVC command types and operations
bitvis_vip_ethernet	ethernet_sb_pkg.vhd	Ethernet Scoreboard package
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	UVVM framework common methods compiled into bitvis_vip_ethernet library
bitvis_vip_ethernet	vvc_methods_pkg.vhd	Ethernet VVC methods
bitvis_vip_ethernet	../uvvm_vvc_framework/src_target_dependent/td_queue_pkg.vhd	UVVM queue package for the VVC
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_rx_vvc.vhd	Ethernet RX VVC
bitvis_vip_ethernet	ethernet_tx_vvc.vhd	Ethernet TX VVC
bitvis_vip_ethernet	ethernet_vvc.vhd	Ethernet VVC

11 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 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