

# **VVC Implementation Guide**

This guide is meant for users that want to make their own VVC. Users that only write test cases that are using the VVCs do NOT need to read this guide.

Making test cases using available VVCs is very easy. Implementing new VVCs is slightly more complex, but fast, safe and efficient once you understand the VVC mechanisms.

# Understanding and Modifying a VVC

This guide goes through all relevant files needed to make a complete VVC. The intention is to allow a VVC implementer to go through file by file and understand and modify as needed. All code objects and functionality given in the VVC and UVVM files are considered mandatory unless otherwise noted.

# To implement your own VVC

### Prerequisites:

- 1. Go through 'The\_critically\_missing\_VHDL\_TB\_feature.ppsx' for a presentation on cycle related corner cases and the need for a far more structured verification approach.
- 2. Read the 'VVC Framework manual pdf' to understand the basic concepts, the communication and handshake between the central sequencer and the VVCs.
- 3. If your VVC is going to access a DUT interface, you need to have a BFM (Bus Functional Model) for that interface independent of whether you are going to make a VVC.

### Implementing your new VVC:

- 1. Use the script vvc\_generator located in the uvvm\_vvc\_framework/script/vvc\_generator/ to generate a new VVC. Notice that the name length is limited by C\_LOG\_SCOPE\_WIDTH (default =20) in uvvm\_util/src/adaptations\_pkg.
- 2. Then see comments in the code for where to make required changes.
- 3. If the new VVC uses multiple channels other than TX and RX, modify the t\_channel type under UVVM-Util adaptations\_pkg.
- 4. See this guide for an explanation to all the various sections you need to evaluate or modify file by file.





## 1 Dependent and independent source

One of the key concepts of the UVVM VVC Framework is the compilation strategy, and how some packages in the UVVM VVC Framework directory are compiled into each of the individual VVC libraries. To avoid confusion about this for future VVC designers, the VVC dependent and VVC independent sources have been marked and split into two source directories. The target dependent source, also known as packages that are compiled into each of the individual VVCs, are placed in the *src\_target\_dependent*/folder. These files are also prefixed with "*td\_*\*" for "target dependent".

The target independent files are compiled into the uvvm\_vvc\_framework library. These files are placed in the src/ folder and prefixed with "ti\_\*" for "target independent".

For examples of how the compile order should be handled, please see the example VIPs' QuickRef and Modelsim compile scripts.

## 2 <name>\_vvc.vhd

## 2.1 For single channel VVCs

Code Section	Description
Entity	GC_INSTANCE_IDX:       Needed in case there are multiple instances of a given VVC. (E.g. DUT with 2 <vvc-name>S). Default is 1, but any natural type is ok.         GC_<name>_CONFIG:       Recommended. Allows predefined BFM behaviour to be set up for every VVC.         GC_CMD_QUEUE_*:       Needed to limit the queue size and to generate a warning if more elements in the queue than ever expected.         Other generics:       Optional and VVC dependent. These generics could for example contain widths of BFM signals.         The interface to the DUT and any other needed I/O. The examples show DUT interfaces as single signals, records and records of records. This is optional.</name></vvc-name>
Declarations	C_SCOPE: Used for logs and alerts. C_VVC_NAME is defined in the VVC 'vvc_methods_pkg.vhd' C_VVC_LABELS: A record of constants, e.g. name and channel, used in multiple procedures. Various status signals used as flags between the processes Command termination record. (Fields: <i>set, reset, is_active</i> . Where <i>set</i> and <i>reset</i> signal fields are used to toggle <i>is_active</i> . Used as inter process flags. <i>command_queue</i> is the queue of commands to be executed in sequence towards the DUT. The aliases are defined to allow common and simplified names.
Constructor	The constructor is run once only – immediately when starting the simulation. The procedure: <ul> <li>Initialises VVC with BFM config and the queue with queue name</li> <li>Allows constructor log for VVC info (using ID_CONSTRUCTOR), and VVC Queue info (using ID_CONSTRUCTOR_SUB)</li> </ul> The procedure will report alerts with severity TB_FAILURE if one of the following occurs: <ul> <li>The instantiated VVC index is higher or equal to the maximum allowed number of VVC instances, given by C_MAX_VVC_INSTANCE_NUM in UVVM-Util 'adaptations_pkg.vhd'</li> <li>UVVM has not been initialized</li> </ul>



### Command Interpreter

Waits for commands from the central test sequencer distributed to this VVC, then puts the command on the queue for execution or immediately performs the required action – depending on command type. Afterwards, it acknowledges the command and waits for the next command from the sequencer.

#### Step 0

Initialize\_interpreter():

- Initialises parameters to default passive/initial values (e.g. terminate\_current\_cmd.set := '0')

#### Step 1

await\_cmd\_from\_sequencer():

- Waits for a command from the central sequencer. Continues on matching VVC, instance index, name and channel
- Log at start using ID CMD INTERPRETER WAIT and at the end using ID CMD INTERPRETER.
- Will only accept exact matches of instance index and name, and either the correct address or "ALL\_CHANNELS"

Step 2a (Only if command type is QUEUED)

put\_command\_on\_queue():

- Puts the received command on the VVC queue (for later retrieval by the Command Executor)

#### Step 2b (Only if command type is IMMEDIATE)

Execute the requested command/operation.

- For the VVC methods these procedures will correspond to the UVVM methods, but prepended with "interpreter\_", e.g. "interpreter\_await\_completion". These UVVM methods are documented in the Common\_VVC\_methods.pdf document. Other commands are documented in their respective QuickRefs.
- format\_command\_idx(): (ti\_vvc\_framework\_support\_pkg) Converts the command index to string, enclosed by C\_CMD\_IDX\_PREFIX and C\_CMD\_IDX\_SUFFIX (found in UVVM-Util adaptations\_pkg).

#### Step 3

acknowledge\_cmd():

- Acknowledges the command from the sequencer by driving global\_vvc\_ack signal (to '1') for 1 delta cycle, then setting it back to 'Z'. This lets the central sequencer know that it can continue execution.

### Command

Executor

Fetches commands from the command queue - if any. Then executes the command and fetches or waits for the next command in the command queue

#### Step 0

-

initialize\_executor():

Initialises parameters to default passive/initial values (e.g. terminate\_current\_cmd.reset := '0')

#### Step 1

fetch\_command\_and\_prepare\_executor():

- Fetches a command from the queue (waits until available if needed).
- Sets relevant flag parameters.
- Log command using ID\_CMD\_EXECUTOR (or Log using ID\_CMD\_EXECUTOR\_WAIT if queue is empty)

Set transaction information for wave-view.

- transaction\_info\_for\_waveview (from vvc\_methods\_pkg.vhd) is a shared variable intended for use in a wave-view – to yield a better overview of transaction info. Setting this information is optional. The pad\_string and to\_string procedures are documented in the UVVM-Util QuickRef.

Insert inter BFM delay if requested.

- insert\_inter\_bfm\_delay\_if\_requested(): Inserts either start-to-start or finish-to-start delay between BFM accesses if this is set in the inter\_bfm\_delay parameter in 'vvc\_config'. Logs information using ID\_CMD\_EXECUTOR.
- If the command currently being processed by the executor is a BFM access, a timestamp will be stored in v\_timestamp\_start\_of\_current\_bfm\_access.

Step 2



Executes a command depending on the requested command/operation.

- terminate\_current\_cmd is only checked inside operations that require multiple BFM accesses like for instance a POLL\_UNTIL command.
- store\_result() is executed for any BFM, where it makes sense for you to store the result of a BFM access. In our example for SBI we think it only makes sense for 'READ'.
- Logging as defined by your BFM.
- Transaction info can be stored in the transaction\_info\_for\_waveview struct for each command type, but this is optional.
- Update the BFM access timestamps if this was a BFM access.
  - v\_timestamp\_of\_last\_bfm\_access is set to now
  - v\_timestamp\_start\_of\_last\_bfm\_access is set to v\_timestamp\_start\_of\_current\_bfm\_access
- The terminate\_current\_cmd flag is reset if it has been active.

Update the *last\_cmd\_idx\_executed* variable with the current command index, *v\_cmd.cmd\_idx*.

Command	
Terminator	The command terminator concurrent procedure sets the <i>is_active</i> flag based on the set and reset flags.

## 2.2 Additional for multi-channel VVCs

Please note that we strongly recommend implementing the VVCs such that each leaf VVC handles one independent DUT communication thread (here: 'Channel'). No more; no less. This allows a single command queue and a single executor handling DUT communication. (Additional processes to handle other characteristics is fine. E.g., a parallel bit-rate check thread.)

Note that SBI\_VVC must handle both read and write accesses, but never simultaneously and always in the given order.

Multi-channel VVCs may be implemented in many different ways – depending on your preferences and priorities. Some examples:

1. As unique VVC implementation

Unique VVCs may be used in order to omit the channel input, e.g. UART RX VVC and UART TX VVC. UART TX VVC would only contain TX specific BFM procedures, while UART RX VVC would only contain RX specific BFM procedures. With this approach the test bench sequencer calls would look like e.g. (assuming both VVCs in this pair are set to instance index 1):

- a. uart\_transmit(UART\_TX\_VVCT,1,...)
- b. uart\_receive(UART\_RX\_VVCT,1,...)

### 2. As shared VVC implementation with usage restricted by user, and multiple VVC instances

A combined VVC with different VVC instances for different channels e.g. RX and TX. The TX instance could e.g. be instance 1, and the RX instance could be e.g. instance 2. Using this UART VVC with this implementation would look like:

- a. uart transmit(UART VVCT,1,...)
- b. uart receive(UART VVCT,2,...)
- 3. As shared VVC implementation with GC\_CHANNEL generic input

A combined VVC with the same combined VVC implementation, but separate instances for different channels e.g. RX and TX (both functionalities inside the same leaf VVC). The downside of this implementation is that it would be possible to call TX BFM procedures when calling the RX VVC channel. Using this UART VVC would look like:

- a. uart\_transmit(UART\_VVCT,TX,1,...)
- b. uart\_receive(UART\_VVCT,RX,1,...)
- 4. As unique VVC implementation with GC\_CHANNEL generic input



This approach uses unique VVC implementations for each channel, e.g. in uart\_rx\_vvc.vhd and uart\_tx\_vvc.vhd, but they both share the VVC target parameter, UART\_VVCT. They both use the GC\_CHANNEL generic input to specify their channel, i.e. TX or RX. This is similar to the method described in 3., but with restrictions that ensure that e.g. the UART TX VVC can't use the UART RX BFM procedures. The included bitvis\_vip\_uart example is implemented with this method. Using this UART VVC would look like:

- a. uart\_transmit(UART\_VVCT,TX,1,...)
- b. uart\_receive(UART\_VVCT,RX,1,...)

When using multiple leaf VVCs it is recommended to use a wrapper architecture to encapsulate the channels. This way, it is possible to instantiate a single VVC rather than each VVC channel individually. For more information about the wrapper architecture, see the uart\_vvc.vhd example in the bitvis\_vip\_uart/src/ directory.

# 3 vvc\_cmd\_pkg.vhd

Section	Comment
t_operation	Contains all UVVM common operations, e.g. AWAIT_COMPLETION and ENABLE_LOG_MSG, in addition to the VVC specific operations such as e.g. WRITE and READ. The VVC specific will have to be evaluated and potentially replaced when implementing a new VVC. The t_operation type is used when relaying commands from the sequencer to the VVC. The t_operation type also has its own to_string() function in this package.
t_vvc_cmd_record	Record type used for relaying a command from the test bench sequencer to the VVC. The record contains fields needed in the common UVVM procedures (listed under the "Common UVVM fields" comment), and VVC specific fields needed to relay data to the VVC executor. The VVC specific data fields should contain any data fields that the BFM procedures might need, e.g. data, address, timeouts etc. There is also a default for this type called C_VVC_CMD_DEFAULT in this package.
Constants	The vvc_cmd_pkg.vhd should contain constants that needs to be set for the entire VVC. In most VVCs this will include the C_VVC_CMD_STRING_MAX_LENGTH which determines the maximum size of msg variables in the VVC. It is also a good idea to declare constants for maximum VVC data/address bus sizes here. It will be possible to construct VVCs with bus sizes up to and including the sizes declared here.
Shared Variables	The shared_vvc_cmd shared variable (type t_vvc_cmd_record) is used for relaying commands between sequencer methods and the VVC. It is default set to C_VVC_CMD_DEFAULT, which is also declared in this file.



# 4 vvc\_methods\_pkg.vhd

Constants and aliases         The wc_methods_pkg contain constants for the VVC name, e.g. *NAME>_VVC". There are also aliases created to make the code more readable.           SNAME>_VVCT         The wc_methods_pkg contain constants for the VVC name, e.g. *NAME>_VVC". There are also aliases created to make the code more readable.           Iters         The wc_methods_pkg contain constants for the VVC name, e.g. *NAME>_VVC". There are also aliases created to make the code more readable.           Iters         The wc_methods_pkg contains the needed configuration for setting up the VVC and BFM. In Bitvis VVCs the BFM configuration will differ for each VVC. Record contents:	Section	Comment
The vvc_methods_pkg contain constants for the VVC name, e.g. " <name>_VVC". There are also aliases created to make the code more readable.         <name>_VVCT       The <name>_VVCT is the VVC target record signal. The target type t_vvc_target_record is a record that contains the parameters needed to trigger a VVC, and to identify the correct target of a VVC cand BFM. In Bilvis VVCs the BFM configuration is encapsulated in a bfm_config record, of type L_vbm_nameffm_config. This torcord is placed in this file and completed into each VVC since the VVCBFM configuration will differ for each VVC. Record contents:         <ul> <li>inter_ffm_delay. A record containing the potential inter-fine delay specifications.</li> <li>mmg_ut_parts. The OB panel that the VVC shall use</li> <li>thm config. A record containing all settings for the BFM eq. dicks periods, message IDs etc.</li> <li>Account C_config SFAULT is defined for this type to use a default value.</li> </ul> <li>Levc_status</li> <li>The optional status record is created in order for the test bench sequence to have access to the default value.</li> <li>a current_mail_dix: The current ormand index being processed in the executor</li> <li>periodig_mail_dix: The current ormand index being processed in the executor</li> <li>periodig_mail_dix: The current ormand index being processed in the executor</li> <li>periodig_mail_dix: The current ormand index being processed in the executor</li> <li>periodig_mail_dix: The current ormand index being processed in the executor</li> <li>periodig_mail_dix: The current ormand index being processed by the executor</li> <li>periodig_mail_dix: The current ormand index being processed by the executor</li> <li>periodig_mail_dix: The current ormand index being processed by the executor.</li> <li>periodig_mail_dix: The curenter or periodig_mail_dix the current or per</li></name></name></name>	Constants and aliases	
NAME>_VVCT       The <\NME>_VVCT isgnal (e.g. SBL_VVCT) is the VVC target record signal. The target type L_vvc_target_record is a record that contains the parameters needed to trigger a VVC, and to identify the correct target of a VVC command.         t_vvc_config       This type contains the needed configuration for setting up the VVC and BPM. In Bitvis VVCS the BFM configuration vial differ for each VVC. Record contents: <ul> <li>inter_bfm_delay. A record containing the potential inter-dfm delay specifications, e.g. if BFM accesses shall be separated with a given time config_view count; 'Command queue specifications</li> <li>mag_id_panel: The ID panel that the VVC shall use</li> <li>bfm_config. A record containing all settings for the BFM. eq. (deck periods, message IDs etc.)</li> <li>A constant QconremVVC_QOV/FIQ_DEFAULT is defined for this type to use as default value.</li> </ul> <li>the optional status record is created in order for the test bench sequence to the vace.</li> <li>current_ormi_ldx: The current command index being processed in the executor</li> <li>previous_ormi_d/x: The previous command index being processed in the executor</li> <li>previous_ormi_d/x: The number of panding commands to be use as default value.</li> <li>t_vvc_status</li> <li>The t_transaction_info_for_waveview</li> <li>meding_com_status record is created in order for the test bench sequence to back secutor:</li> <ul> <li>previous_ormi_d/x: The previous command index being processed in the executor:</li> <ul> <li>previous_ormi_d/x: The previous command index being processed in the executor:</li> <li>previous_mi_d/x: The previous_command index being processed by the executor.</li> <li>previous_mi_d/x: The previous_command index being processed by the executo</li></ul></ul>		The vvc_methods_pkg contain constants for the VVC name, e.g. " <name>_VVC". There are also aliases created to make the code more readable.</name>
NAME>_VVCT       The <namesvvct (e.g.="" a="" and="" command.<="" contains="" correct="" identify="" is="" l_vvc_target_record="" needed="" of="" parameters="" record="" sbl_vvct)="" signal="" signal.="" target="" td="" that="" the="" to="" trigger="" type="" vvc="" vvc,="">         t_vvc_config       This type contains the needed configuration for setting up the VVC and BFM. In Bivis VVCs the BFM configuration is encapsulated in a bfm_config record, of type L_tofm_name_bfm_config theorem containing the potential liner-bfm_delay specifications, e.g. if BFM accesses shall be separated with a given time - orng_dueue_count, 'C command gueue specifications - orng_dueueue_count, 'C command gueue specifications - orng_dueueueueueueue_count, 'C command gueueueueueueueueueueueueueueueueueueue</namesvvct>		
The <names_vvct (e.g.="" <ul="" a="" and="" bfm="" bfm_config="" commend.="" compiled="" configuration="" contains="" contents:="" correct="" differ="" each="" encapsulated="" file="" for="" identify="" in="" is="" needed="" of="" parameters="" placed="" record="" sbl_vvct)="" signal="" signal.="" t_v4m_names_bfm_config="" t_vvc_target_record="" target="" that="" the="" this="" to="" treeord,="" trigger="" type="" vvc="" vvc,="" vvc.="" vvcs="" will="" wvcs=""> <li>inter_bfm_delay: A record containing the optiential inter-thm delay specifications, e.g. if BFM accesses shall be separated with a given time</li> <li>emd_queu_count ?: Command queu specifications</li> <li>mmg_id_panei. The ID panel that the VVC shall use</li> <li>bfm_config A record containing all settings for the BFM, e.g. clock periods, message IDs etc.</li> <li>A constant C_vname_vVC_config shared_rames_wc_config is declared and all elements are set to the default value.</li> <li>the optional status record is created in order for the test bench sequence to have access to the status of the VVC. The status record can contain anything that is relevant for the outside, and it is recommendin direx being processed in the executor</li> <li>pervicus_ord_dir. The previous command index being processed in the executor</li> <li>constant C_VVC_STATUS_DEFAULT is defined for this type to use as default value.</li> <li>t_transaction_info_for_waveview</li> <li>the t_transaction_info_for_waveview type is an optional status record to be processed by the executor.</li> <li>A shared variable array of t_vvc_status shared_rammes_vvc_status is declared and all elements are set to the default value.</li> <li>t_transaction_info_for_waveview<th><name>_VVCT</name></th><th></th></li></names_vvct>	<name>_VVCT</name>	
tigger a VVC, and to identify the correct target of a VVC command.         t_vvc_config         This type contains the needed configuration for setting up the VVC and BFM. In Bitvis VVCs the BFM configuration is encapsulated in a bfm_config record, of type t_cdm_name>bfm_config_This record is placed in this file and compiled into each VVC since the VVC/BFM configuration will differ for each VVC. Record contains the people containing the population target in the the VVC field feators, e.g. if BFM accesses shall be separated with a given time         - intert bfm_config_A Traceord containing the population target in the sequence on the VVC field feators.       - ording use access to the set the VVC. The status record contain anything that is relevant.         t_vvc_status       The optional status record is created in order for the test bench sequencer to have access to the status of the VVC. The status record can contain anything that is relevant.         to runned_cm_differ_the revolus command index being processed in the executor       - pervious command index being processed in the executor         - providuam_differ_the previous command index being processed in the executor       - pervious_command index being processed in the executor         - providuam_differ_the previous command index being processed by the executor       - pervious_command index being processed in the executor         - providuam_differ_the previous command index being processed by the executor       - providuam_differ_the previous command index being processed by the executor         - providuam_differ_the previous command index being processed by the executor       - providuam_differ_the previous comm	—	The <name>_VVCT signal (e.g. SBI_VVCT) is the VVC target record signal. The target type t_vvc_target_record is a record that contains the parameters needed to</name>
t_vvc_config       This type contains the needed configuration for setting up the VVC and BFM. In Bitvis VVCs the BFM configuration is encapsulated in a bfm_config record, of type         t_vtm_clean_annex_bfm_config. This record is placed in this file and compiled into each VVC since the VVCBMC onfiguration will differ for each VVC. Record contents: <ul> <li>inter_bfm_delay. A record containing the potential inter-bfm delay specifications, e.g. if BFM accesses shall be separated with a given time</li> <li>cmd_uper.config. Command queue specifications</li> <li>mg_id_panet: The ID panet that the VVC shall use</li> <li>bfm_config. A record containing all settings for the BFM, e.g. clock periods, message IDs etc.</li> <li>A constant CnamesVVC_CONFIG_DEFAULT is defined for this type to use as default value.</li> </ul> <li>A shared variable array of t_vvc_config shared_<name>_vvc_config is declared and all elements are set to the default value.</name></li> <li>c current_om_loit. The cord current command index being processed in the executor</li> <li>perding_om_loit. The revious command index being processed in the executor</li> <li>perding_om_loit. The number of pending commands to be processed by the executor</li> <li>A constant C_VVC_STATUS_DEFAULT is defined for this type to use as default value.</li> <li>t_transaction_info_for_waveviewiew</li> <li>The t_transaction_info_for_waveview type is an optional status record to be used in the wave-view. This record should be modified to suit the BFM fields, but it can also contain the VVC field to paration, which can be updated with the VVC paration currently being processed by the executor. A constant CVVC_STATUS_DEFAULT is defined for this type to use as default value.</li> <li>t_transaction_info_for_waveviewie</li> <li>a current_</li>		trigger a VVC, and to identify the correct target of a VVC command.
Lyve_borning       This type contains the needed configuration for setting up the VVC and BFM. In Bitvis VVCs the BFM configuration will differ for each VVC. Record contentings: <ul> <li>inter_bim_delay. A record containing the potential inter-bim each VVC since the VVC/BFM configuration will differ for each VVC. Record contents:                 <ul></ul></li></ul>	t une confin	
this type dotains the interact continguation to setting up to this VVC share to Michiguation is endapatated in the full control will differ for each VVC. Record contents:         -       inter_form_conte_This record is placed in placed specifications, e.g. (FBFM accesses shall be separated with a given time         -       inter_form_delay. A record containing the potential inter-form delay specifications, e.g. (FBFM accesses shall be separated with a given time         -       inter_form_delay. A record containing at the potential inter-form delay specifications, e.g. (FBFM accesses shall be separated with a given time         -       inter_form_delay. Control, 'Command queue specifications         -       inter_form_delay. Control, 'Command queue specifications         -       inter_form_delay. Control, 'Record containing at settings for the BFM. e.g. clock periods, message IDs etc.         A constant C_manze_VVC_CONFIG_DEFAUL Tis defined for this type to use as default value.         A shared variable array of L_vvc_config shared_rname_vvc_config is declared and all elements are set to the default value.         t_vvc_status       The optional status record is created in order for the test bench sequencer to have access to the status of the VVC. The status record to an contain anything that is relevant for the outside, and it is recommand index being processed in the executor         -       previous_cmd_dx: The previous command index being processed in the executor         -       previous_cmd_dx: The previous command index being processed in the executor         -       previous_cm	t_vvc_conng	This type contains the needed configuration for setting up the VVC and REM. In Bityis VVCs the REM configuration is approximated in a bitml configuration of type
<ul> <li>inter_bm_edia/. A record containing the potential inter-binace into being work user volume to subject the separated with a given time         <ul> <li>cmd_queue_count_?: Command queue specifications</li> <li>mg_qi_q panet: The D panel that the VVC shall use</li> <li>bfm_config. A record containing all settings for the BFM, e.g., clock periods, message IDs etc.</li> </ul> </li> <li>A constant Crame&gt;_VVC_CONFIG_DEFALL T is defined for this type to use as default value.</li> <li>A shared variable array of t_vvc_config shared_<reame>_vvc_config is declared and all elements are set to the default value.</reame></li> <li>I explore a status record is created in order for the test bench sequencer to have access to the status of the VVC. The status record can contain anything that is relevant for the outside, and it is recommand index being processed in the executor</li> <li>pendingmididx: The number of pending to the vector device a default value.</li> <li>A constant CvVC_STATUS_DEFALLT is defined for this type to use as default value.</li> <li>A constant CvVC_STATUS_DEFALLT is defined for this type to use as default value.</li> <li>A shared variable array of t_vvc_status shared_<reame>_vvc_status is declared and all elements are set to the default value.</reame></li> <li>A constant CvTANSACTION_INFO_FOR_WARWARWEWED DEFALLT is defined for this type to use as default value.</li> <li>A shared variable array of t_vvc_status shared_<reame>_vvc_status is declared and all elements are set to the default value.</reame></li> <li>A constant CVTANSACTION_INFO_FOR_WARWEWEW DEFAULT is defined for this type to use as default value.</li> <li>A shared variable of t_transaction_info_for_waveview t_crame&gt;_vvc_status is declared and all elements are set to the default value.</li> <li>A constant CVTANSACTION_INFO_FOR_WARWEWEWED DEFAULT is defined for this type to use as default value.</li> <li>A shared variable of t_transaction_info_f</li></ul>		t shift names the rectain the rectain of the second in the second of the second s
<ul> <li>Intel_Dim_Queue_Count_: Command queues specifications</li> <li>msg_id_panel: The ID panel that the VCC shall use</li> <li>msg_id_panel: The ID panel that the VCC shall use</li> <li>bm_config: A record containing all settings for the BFM, e.g. clock periods, message IDs etc.</li> <li>A constant C_rames_VVC_CONFIG_DEFAULT is defined for this type to use as default value.</li> <li>A shared variable array of t_vvc_config shared_rames_vvc_config is declared and all elements are set to the default value.</li> <li>A shared variable array of t_vvc_config shared_rames_vvc_config is declared and all elements are set to the default value.</li> <li>A constant C_rame_UVC_CONFIG_DEFAULT is defined for this type to use as default value.</li> <li>Current_ord jdx: The current command index being processed in the executor</li> <li>previous, cmd_jdx: The previous command index being processed in the executor</li> <li>previous, cmd_jdx: The previous command index being processed in the executor</li> <li>previous, cmd_jdx: The previous command index being processed in the executor</li> <li>previous, cmd_jdx: The previous command index being processed in the executor</li> <li>previous, cmd_jdx: The previous command index being processed in the executor</li> <li>previous, cmd_jdx: The previous command index being processed in the executor</li> <li>previous, cmd_jdx: The previous command index being processed in the executor</li> <li>A shared variable array of t_vvc_status shared_ramme_vvc_status is declared and all elements are set to the default value.</li> <li>t_transaction_info_for_waveview type is an optional status record to be used in the wave-view. This record should be modified to suit the BFM fields, but it can also contain the VVC field t_operation, which can be updated with the VVC operation currently being processed by the executor. The transaction_info_for_waveview is meant as a way of improving the readability of wave-views.</li> <li>A</li></ul>		inter bim delay. A record containing the protection inter bind delay experimentations of a fit BEM accesses shall be senarated with a citizen time.
Interprete Configure		- intel_direction of the potential inter-one deay specifications, e.g. in Driviaccesses shall be separated with a given unite
Inst_in_parter. The tup parter that the VVC_CONFIG_DEFAULT is defined for this type to use as default value.     A constant C_ <names_vvc_config_default a="" all="" and="" are="" array="" as="" c_<="" c_<names_vvc_config="" constant="" declared="" default="" defined="" elements="" for="" is="" of="" set="" shared="" shared_<names_vvc_config="" t_vvc_config="" td="" the="" this="" to="" type="" use="" value.="" variable="">       t_vvc_status     The optional status record is created in order for the test bench sequencer to have access to the status of the VVC. The status record can contain anything that is relevant for the outside, and it is recommended to have at least these three fields:         <ul> <li>urent_cmd_idx: The orumetor command index being processed in the executor</li> <li>previous_emd_idx: The number of pending commands to be processed by the executor</li> <li>A constant C_VVC_STATUS_DEFAULT is defined for this type to use as default value.</li> </ul> <li>t_transaction_info_for_waveview</li> <li>The t_transaction_info_for_waveview type is an optional status record to be used in the wave-view. This record should be modified to suit the BFM fields, but it can also contain the VVC field t_operation, which can be updated with the VVC operation currently being processed by the executor. The transaction_info_for_waveview is an ay of improving the readability of wave-views. A constant C_TRANSACTION_INFO_FOR_WAVEVEW_DEFAULT is defined for this type to use as default value.</li> <li>YVC Dedicated Methods</li> <li>The vvc_methods_pkg whd file also contains the</li></names_vvc_config_default>		- citil _ ueee_coult command duede specifications
Control of the containing an seture prive config is for the prive e.g. cock periods, message lus etc.     A constant Crame>_tywc_CONFIG_DEFAULT is defined for this type to use as default value.     A shared variable array of two_config is declared and all elements are set to the default value.     Control to the control of the control of the test bench sequencer to have access to the status of the VVC. The status record can contain anything that is relevant for the outside, and it is recommended to have at least these three fields:         - current_orm_idx: The current command index being processed in the executor         - pending_cmd_idx: The number of pending commands to be processed in the executor         - pending_cmd_idx: The number of pending commands to be processed in the executor         A constant C_WC_STATUS_DEFAULT is defined for this type to use as default value.         A shared variable array of t_vvc_status shared_ <name>_wvc_status is declared and all elements are set to the default value.         A shared variable array of t_vvc_status shared_variable array of t_vvc_status is declared and all elements are set to the default value.         A shared variable of t_vvc_vection, which can be updated with the VVC operation currently being processed by the executor. The transaction_info_for_waveview is meant as a way of improving the readability of wave-views.         A constant to _TRANSACTION_INFO_FOR_WAVEVIEW_DEFAULT is defined for this type to use as default value.         A shared variable of t_transaction_info_for_waveview t_<name>_transaction_info_for_waveview is declared and all elements are set to the default value.         A constant to _TRANSACTION_INFO_FOR_WAVEVIEW_DEFAULT is defined for this type to use as default va</name></name>		- misg_o_partiel. The to partie that the VVC shall use
A constant C_VVC_CONFIG_DEFAULT is defined for this type to use as default value.         A shared variable array of t_vvc_config shared_ <a more="">wcc_config is declared and all elements are set to the default value.         t_vvc_status       The optional status record is created in order for the test bench sequencer to have access to the status of the VVC. The status record can contain anything that is relevant for the outside, and it is recommended to have at least these three fields:         • current_end_idx: The previous command index being processed in the executor         • previous_cmd_idx: The previous command index type to use as default value.         A constant C_VC_STATUS_DEFAULT is defined for this type to use as default value.         A shared variable array of t_vvc_status shared_<a more="">wc_status is declared and all elements are set to the default value.         t_transaction_info_for_waveview       The t_transaction_info_for_waveview type is an optional status record to be used in the wave-view. This record should be modified to suit the BFM fields, but it can also contain the VVC field to operation, which can be updated with the VVC operation currently being processed by the executor. The transaction_info_for_waveview is meant as a way of improving the readability of wave-views.         A constant C_TRANSACTION_INFO_FOR_WAVEVIEW_DEFAULT is defined for this type to use as default value.         VVC Dedicated Methods       The vc_methods_pkg whd file also contains the VVC procedures that are called from the test bench sequencer. These procedures should reflect the procedures in the BFM, e.g. <name>_write or <name>_receive. The parameters of these procedures are mostly up to the user, but it is recommended tha</name></name></a></a>		- btm_contig: A record containing all settings for the BFM, e.g. clock periods, message IDs etc.
t_vvc_status       A shared variable array of t_vvc_contig shared_shared_variable_vvc_contig is declared and all elements are set to the default value.         t_vvc_status       The optional status record is created in order for the test bench sequencer to have access to the status of the VVC. The status record can contain anything that is relevant for the outside, and it is recommended to have at least these three fields: <ul> <li>current_cmd_idx: The previous command index being processed in the executor</li> <li>peroid_idx: The previous command index being processed by the executor</li> <li>pending_cmd_idx: The number of pending commands to be processed by the executor</li> <li>A constant C_VVC_STATUS_DEFAULT is defined for this type to use as default value.</li> </ul> <li>t_transaction_info_for_waveview</li> <li>The t_transaction_info_for_waveview type is an optional status record to be used in the wave-view. This record should be modified to suit the BFM fields, but it can also contain the VVC field t_operation, which can be updated with the VVC operation currently being processed by the executor. The transaction_info_for_waveview is meant as a way of improving the readballity of wave-views.</li> <li>A constant C_TRANSACTION_INFO_FOR_WAVEVIEW_DEFAULT is defined for this type to use as default value.</li> <li>         VVC Dedicated Methods       <ul> <li>The vc_methods_pkg.vhd file also contains the VVC procedures that are called from the test bench sequencer. These procedures should reflect the procedures in the BFM, e.g. <ul> <li>mame&gt;_receive. The parameters of these procedures are mostly up to the user, but it is recommended that the BFM arguments that are rarely altered be placed into the bfm_config parameters, while the parameters that changes often are used as input arguments.</li></ul></li></ul></li>		A constant C_ <name>_VVC_CONFIG_DEFAULT is defined for this type to use as default value.</name>
t_vvc_status       The optional status record is created in order for the test bench sequencer to have access to the status of the VVC. The status record can contain anything that is relevant for the outside, and it is recommended to have at least these three fields: <ul> <li>current_cmd_idx: The current command index being processed in the executor</li> <li>previous_cmd_idx: The number of pending commands to be processed by the executor</li> <li>previous_cmd_idx: The number of pending commands to be processed by the executor</li> <li>A constant C_VVC_STATUS_DEFAULT is defined for this type to use as default value.</li> </ul> <li>t_transaction_info_for_waveview</li> <li>The t_transaction_info_for_waveview type is an optional status record to be used in the wave-view. This record should be modified to suit the BFM fields, but it can also contain the VVC field t_operation, which can be updated with the VVC operation currently being processed by the executor. The transaction_info_for_waveview is meant as a way of improving the readability of wave-views. A constant C_TRANSACTION_INFO_FOR_WAVEVIEW_DEFAULT is defined for this type to use as default value.</li> <li>A constant C_TRANSACTION_INFO_FOR_WAVEVIEW_DEFAULT is defined for this type to use as default value.</li> <li>A shared variable of t_transaction_info_for_waveview t_<name>_transaction_info_for_waveview is declared and all elements are set to the default value.</name></li> <li>VVC Dedicated Methods</li> <li>The vc_methods_pkg.vhd file also contains the VVC procedures that are called from the test bench sequencer. These procedures should reflect the procedures in the BFM e.g. canne&gt;_write or <name>_receive. The parameters of these procedures are mostly up to the user, but it is recommended that the BFM arguments that are rarely altered be placed into the bfm_confg parameter, while the parameters that changes often are used as inpu</name></li>		A shared variable array of t_vvc_config shared_ <name>_vvc_config is declared and all elements are set to the default value.</name>
The optional status record is created in order for the test bench sequencer to have access to the status of the VVC. The status record can contain anything that is relevant for the outside, and it is recommended to have at least these three fields: <ul> <li>current_cmd_idx: The current command index being processed in the executor</li> <li>previous_cmd_idx: The number of pending commands to be processed in the executor</li> <li>premious_cmd_idx: The number of pending commands to be processed by the executor</li> <li>A constant C_VVC_STATUS_DEFAULT is defined for this type to use as default value.</li> </ul> <li>t_transaction_info_for_waveview</li> The t_transaction_info_for_waveview type is an optional status record to be used in the wave-view. This record should be modified to suit the BFM fields, but it can also contain the VVC field t_operation, which can be updated with the VVC operation currently being processed by the executor. The transaction_info_for_waveview is meant as a way of improving the readability of wave-views. A constant C_TRANSACTION_INFO_FOR_WAVEVIEW_DEFAULT is defined for this type to use as default value. VVC Dedicated Methods The vvc_methods_pkg.vhd file also contains the VVC procedures that are called from the test bench sequencer. These procedures should reflect the procedures in the BFM arguments that are rarely altered be placed into the fir_coning parameter, while the parameters that changes often are used as input arguments. Since these VVC methods are reused for all instances of this VVC, it is necessary with some extra parameters in order to specify which VVC instance to forward the call to. This is done with the first two (or three) parameters:	t vvc status	
for the outside, and it is recommended to have at least these three fields:         - current_cmd_idx: The current command index being processed in the executor         - previous_cmd_idx: The pervious command index being processed by the executor         - pending_cmd_idx: The pervious command index being processed by the executor         - pending_cmd_idx: The pervious command index being processed by the executor         A constant C_VVC_STATUS_DEFAULT is defined for this type to use as default value.         A shared variable array of t_vvc_status shared_ <name>_vvc_status is declared and all elements are set to the default value.         t_transaction_info_for_waveview         The t_transaction_info_for_waveview type is an optional status record to be used in the wave-view. This record should be modified to suit the BFM fields, but it can also contain the VVC field t_operation, which can be updated with the VVC operation currently being processed by the executor. The transaction_info_for_waveview is meant as a way of improving the readability of wave-views.         A constant C_TRANSACTION_INFO_FOR_WAVEVIEW_DEFAULT is defined for this type to use as default value.         A shared variable of t_transaction_info_for_waveview t_<name>_transaction_info_for_waveview is declared and all elements are set to the default value.         VVC Dedicated Methods       The vvc_methods_pkg.vhd file also contains the VVC procedures that are called from the test bench sequencer. These procedures should reflect the procedures in the BFM, e.g., -name&gt;_write or name&gt;_crecieve. The parameters of these procedures are mostly up to the user, but it is recommended that the BFM arguments that are rarely altered b</name></name>		The optional status record is created in order for the test bench sequencer to have access to the status of the VVC. The status record can contain anything that is relevant
Current_cmd_idx: The current command index being processed in the executor     previous_cmd_idx: The previous command index being processed in the executor     previous_cmd_idx: The number of pending commands to be processed by the executor     A constant C_VVC_STATUS_DEFAULT is defined for this type to use as default value.     A shared variable array of t_vvc_status shared_ <name>_vvc_status is declared and all elements are set to the default value.     The t_transaction_info_for_waveview type is an optional status record to be used in the wave-view. This record should be modified to suit the BFM fields, but it can also contain the VVC field t_operation, which can be updated with the VVC operation currently being processed by the executor. The transaction_info_for_waveview is meant as a way of improving the readability of wave-views.     A constant C_TRANSACTION_INFO_FOR_WAVEVIEW_DEFAULT is defined for this type to use as default value.     A shared variable of t_transaction_info_for_waveview t_<name>_transaction_info_for_waveview is declared and all elements are set to the default value.     XVC Dedicated Methods     The vvc_methods_pkg.vhd file also contains the VVC procedures that are called from the test bench sequencer. These procedures should reflect the procedures in the BFM, e.g. <name>_receive. The parameters of these procedures are mostly up to the user, but it is recommended that the BFM arguments that are rarely altered be placed into the bfm_config parameters.     Since these VVC methods are reused for all instances of this VVC, it is necessary with some extra parameters in order to specify which VVC instance to forward the call to. This is done with the first two (or three) parameters:</name></name></name>		for the outside, and it is recommended to have at least these three fields:
Previous_cmd_idx: The previous command index being processed in the executor     Pending_cmd_idx: The previous commands to be processed by the executor     A constant C_VVC_STATUS_DEFAULT is defined for this type to use as default value.     A shared variable array of t_vvc_status shared_ <name>_vvc_status is declared and all elements are set to the default value.     The t_transaction_info_for_waveview type is an optional status record to be used in the wave-view. This record should be modified to suit the BFM fields, but it can also contain the VVC field t_operation, which can be updated with the VVC operation currently being processed by the executor. The transaction_info_for_waveview is meant as a way of improving the readability of wave-views.     A constant C_TRANSACTION_INFO_FOR_WAVEVIEW_DEFAULT is defined for this type to use as default value.     A shared variable of t_transaction_info_for_waveview t_<name>_transaction_info_for_waveview is declared and all elements are set to the default value.     VVC Dedicated Methods     The vvc_methods_pkg.vhd file also contains the VVC procedures that are called from the test bench sequencer. These procedures should reflect the procedures in the BFM, e.g. <name>_receive. The parameters of these procedures are mostly up to the user, but it is recommended that the BFM arguments that are rarely altered be placed into the bfm_config parameter, while the parameters that changes often are used as input arguments.     Since these VVC methods are reused for all instances of this VVC, it is necessary with some extra parameters in order to specify which VVC instance to forward the call to. This is done with the first two (or three) parameters:</name></name></name>		- current cmd idx: The current command index being processed in the executor
pending_cmd_idx: The number of pending commands to be processed by the executor     A constant C_VVC_STATUS_DEFAULT is defined for this type to use as default value.     A shared variable array of t_vvc_status shared_ <name>_vvc_status is declared and all elements are set to the default value.     The t_transaction_info_for_waveview type is an optional status record to be used in the wave-view. This record should be modified to suit the BFM fields, but it can also     contain the VVC field t_operation, which can be updated with the VVC operation currently being processed by the executor. The transaction_info_for_waveview is meant     as a way of improving the readability of wave-views.     A constant C_TRANSACTION_INFO_FOR_WAVEVIEW_DEFAULT is defined for this type to use as default value.     A shared variable of t_transaction_info_for_waveview t_<name>_transaction_info_for_waveview is declared and all elements are set to the default value.     VVC Dedicated Methods     The vvc_methods_pkg.vhd file also contains the VVC procedures that are called from the test bench sequencer. These procedures should reflect the procedures in the     BFM, e.g. <name>_write or <name>_receive. The parameters of these procedures are mostly up to the user, but it is recommended that the BFM arguments that are     rarely altered be placed into the bfm_config parameter, while the parameters that changes often are used as input arguments.     Since these VVC methods are reused for all instances of this VVC, it is necessary with some extra parameters in order to specify which VVC instance to forward the call to.     This is done with the first two (or three) parameters:</name></name></name></name>		- previous cmd idx: The previous command index being processed in the executor
A constant C_VVC_STATUS_DEFAULT is defined for this type to use as default value.         A shared variable array of t_vvc_status shared_ <name>_vvc_status is declared and all elements are set to the default value.         t_transaction_info_for_waveview         The t_transaction_info_for_waveview type is an optional status record to be used in the wave-view. This record should be modified to suit the BFM fields, but it can also contain the VVC field t_operation, which can be updated with the VVC operation currently being processed by the executor. The transaction_info_for_waveview is meant as a way of improving the readability of wave-views.         A constant C_TRANSACTION_INFO_FOR_WAVEVIEW_DEFAULT is defined for this type to use as default value.         A shared variable of t_transaction_info_for_waveview t_<name>_transaction_info_for_waveview is declared and all elements are set to the default value.         VVC Dedicated Methods       The vvc_methods_pkg.vhd file also contains the VVC procedures that are called from the test bench sequencer. These procedures should reflect the procedures in the BFM, e.g. <name>_write or <name>_receive. The parameters of these procedures are mostly up to the user, but it is recommended that the BFM arguments that are rarely altered be placed into the bfm_config parameter, while the parameters that changes often are used as input arguments.         Since these VVC methods are reused for all instances of this VVC, it is necessary with some extra parameters in order to specify which VVC instance to forward the call to. This is done with the first two (or three) parameters:</name></name></name></name>		- pending cm dix: The number of pending commands to be processed by the executor
the start of		A constant C VVC STATUS DEFAULT is defined for this type to use as default value
t_transaction_info_for_waveview         The t_transaction_info_for_waveview         The t_transaction_info_for_waveview type is an optional status record to be used in the wave-view. This record should be modified to suit the BFM fields, but it can also contain the VVC field t_operation, which can be updated with the VVC operation currently being processed by the executor. The transaction_info_for_waveview is meant as a way of improving the readability of wave-views.         A constant C_TRANSACTION_INFO_FOR_WAVEVIEW_DEFAULT is defined for this type to use as default value.         A shared variable of t_transaction_info_for_waveview t_ <name>_transaction_info_for_waveview is declared and all elements are set to the default value.         VVC Dedicated Methods         The vvc_methods_pkg.vhd file also contains the VVC procedures that are called from the test bench sequencer. These procedures should reflect the procedures in the BFM, e.g. <name>_write or <name>_receive. The parameters of these procedures are mostly up to the user, but it is recommended that the BFM arguments that are rarely altered be placed into the bfm_config parameter, while the parameters that changes often are used as input arguments.         Since these VVC methods are reused for all instances of this VVC, it is necessary with some extra parameters in order to specify which VVC instance to forward the call to. This is done with the first two (or three) parameters:</name></name></name>		A shared variable array of t vvc status shared < name> vvc status is declared and all elements are set to the default value
t_transaction_info_for_waveview The t_transaction_info_for_waveview type is an optional status record to be used in the wave-view. This record should be modified to suit the BFM fields, but it can also contain the VVC field t_operation, which can be updated with the VVC operation currently being processed by the executor. The transaction_info_for_waveview is meant as a way of improving the readability of wave-views. A constant C_TRANSACTION_INFO_FOR_WAVEVIEW_DEFAULT is defined for this type to use as default value. A shared variable of t_transaction_info_for_waveview t_ <name>_transaction_info_for_waveview is declared and all elements are set to the default value. VVC Dedicated Methods The vvc_methods_pkg.vhd file also contains the VVC procedures that are called from the test bench sequencer. These procedures should reflect the procedures in the BFM, e.g. <name>_write or <name>_receive. The parameters of these procedures are mostly up to the user, but it is recommended that the BFM arguments that are rarely altered be placed into the bfm_config parameter, while the parameters that changes often are used as input arguments. Since these VVC methods are reused for all instances of this VVC, it is necessary with some extra parameters in order to specify which VVC instance to forward the call to. This is done with the first two (or three) parameters:</name></name></name>		
The t_transaction_info_for_waveview type is an optional status record to be used in the wave-view. This record should be modified to suit the BFM fields, but it can also contain the VVC field _operation, which can be updated with the VVC operation currently being processed by the executor. The transaction_info_for_waveview is meant as a way of improving the readability of wave-views. A constant <i>C_TRANSACTION_INFO_FOR_WAVEVIEW_DEFAULT</i> is defined for this type to use as default value. A shared variable of t_transaction_info_for_waveview t_< <i>name&gt;_transaction_info_for_waveview</i> is declared and all elements are set to the default value. WVC Dedicated Methods The vvc_methods_pkg.vhd file also contains the VVC procedures that are called from the test bench sequencer. These procedures should reflect the procedures in the BFM, e.g. <name>_write or <name>_receive. The parameters of these procedures are mostly up to the user, but it is recommended that the BFM arguments that are rarely altered be placed into the bfm_config parameter, while the parameters that changes often are used as input arguments. Since these VVC methods are reused for all instances of this VVC, it is necessary with some extra parameters in order to specify which VVC instance to forward the call to. This is done with the first two (or three) parameters:</name></name>	t_transaction_info_for_waveview	
contain the VVC field t_operation, which can be updated with the VVC operation currently being processed by the executor. The transaction_info_for_waveview is meant as a way of improving the readability of wave-views.         A constant C_TRANSACTION_INFO_FOR_WAVEVIEW_DEFAULT is defined for this type to use as default value.         A shared variable of t_transaction_info_for_waveview t_ <name>_transaction_info_for_waveview is declared and all elements are set to the default value.         VVC Dedicated Methods         The vvc_methods_pkg.vhd file also contains the VVC procedures that are called from the test bench sequencer. These procedures should reflect the procedures in the BFM, e.g. <name>_write or <name>_receive. The parameters of these procedures are mostly up to the user, but it is recommended that the BFM arguments that are rarely altered be placed into the bfm_config parameter, while the parameters that changes often are used as input arguments.         Since these VVC methods are reused for all instances of this VVC, it is necessary with some extra parameters in order to specify which VVC instance to forward the call to. This is done with the first two (or three) parameters:</name></name></name>		The t_transaction_info_for_waveview type is an optional status record to be used in the wave-view. This record should be modified to suit the BFM fields, but it can also
as a way of improving the readability of wave-views. A constant <i>C_TRANSACTION_INFO_FOR_WAVEVIEW_DEFAULT</i> is defined for this type to use as default value. A shared variable of t_transaction_info_for_waveview <i>t_<name>_transaction_info_for_waveview</name></i> is declared and all elements are set to the default value. <b>VVC Dedicated Methods</b> The vvc_methods_pkg.vhd file also contains the VVC procedures that are called from the test bench sequencer. These procedures should reflect the procedures in the BFM, e.g. <name>_write or <name>_receive. The parameters of these procedures are mostly up to the user, but it is recommended that the BFM arguments that are rarely altered be placed into the bfm_config parameter, while the parameters that changes often are used as input arguments. Since these VVC methods are reused for all instances of this VVC, it is necessary with some extra parameters in order to specify which VVC instance to forward the call to. This is done with the first two (or three) parameters:</name></name>		contain the VVC field t_operation, which can be updated with the VVC operation currently being processed by the executor. The transaction_info_for_waveview is meant
A constant C_TRANSACTION_INFO_FOR_WAVEVIEW_DEFAULT is defined for this type to use as default value.         A shared variable of t_transaction_info_for_waveview t_ <name>_transaction_info_for_waveview is declared and all elements are set to the default value.         VVC Dedicated Methods         The vvc_methods_pkg.vhd file also contains the VVC procedures that are called from the test bench sequencer. These procedures should reflect the procedures in the BFM, e.g. <name>_write or <name>_receive. The parameters of these procedures are mostly up to the user, but it is recommended that the BFM arguments that are rarely altered be placed into the bfm_config parameter, while the parameters that changes often are used as input arguments.         Since these VVC methods are reused for all instances of this VVC, it is necessary with some extra parameters in order to specify which VVC instance to forward the call to. This is done with the first two (or three) parameters:</name></name></name>		as a way of improving the readability of wave-views.
A shared variable of t_transaction_info_for_waveview t_ <name>_transaction_info_for_waveview is declared and all elements are set to the default value.         VVC Dedicated Methods       The vvc_methods_pkg.vhd file also contains the VVC procedures that are called from the test bench sequencer. These procedures should reflect the procedures in the BFM, e.g. <name>_write or <name>_receive. The parameters of these procedures are mostly up to the user, but it is recommended that the BFM arguments that are rarely altered be placed into the bfm_config parameter, while the parameters that changes often are used as input arguments.         Since these VVC methods are reused for all instances of this VVC, it is necessary with some extra parameters in order to specify which VVC instance to forward the call to. This is done with the first two (or three) parameters:</name></name></name>		A constant C_TRANSACTION_INFO_FOR_WAVEVIEW_DEFAULT is defined for this type to use as default value.
VVC Dedicated Methods       The vvc_methods_pkg.vhd file also contains the VVC procedures that are called from the test bench sequencer. These procedures should reflect the procedures in the BFM, e.g. <name>_write or <name>_receive. The parameters of these procedures are mostly up to the user, but it is recommended that the BFM arguments that are rarely altered be placed into the bfm_config parameter, while the parameters that changes often are used as input arguments.         Since these VVC methods are reused for all instances of this VVC, it is necessary with some extra parameters in order to specify which VVC instance to forward the call to. This is done with the first two (or three) parameters:</name></name>		A shared variable of t_transaction_info_for_waveview t_ <name>_transaction_info_for_waveview is declared and all elements are set to the default value.</name>
The vvc_methods_pkg.vhd file also contains the VVC procedures that are called from the test bench sequencer. These procedures should reflect the procedures in the BFM, e.g. <name>_write or <name>_receive. The parameters of these procedures are mostly up to the user, but it is recommended that the BFM arguments that are rarely altered be placed into the bfm_config parameter, while the parameters that changes often are used as input arguments. Since these VVC methods are reused for all instances of this VVC, it is necessary with some extra parameters in order to specify which VVC instance to forward the call to. This is done with the first two (or three) parameters:</name></name>	VVC Dedicated Mothods	
BFM, e.g. <name>_write or <name>_receive. The parameters of these procedures are mostly up to the user, but it is recommended that the BFM arguments that are rarely altered be placed into the bfm_config parameter, while the parameters that changes often are used as input arguments. Since these VVC methods are reused for all instances of this VVC, it is necessary with some extra parameters in order to specify which VVC instance to forward the call to. This is done with the first two (or three) parameters:</name></name>		The vice methods, now which the also contains the VVC procedures that are called from the test hanch sequencer. These procedures should reflect the procedures in the
Since these VVC methods are reused for all instances of this VVC, it is necessary with some extra parameters in order to specify which VVC instance to forward the call to. This is done with the first two (or three) parameters:		REM a discovery write and over charges received the proceedings and over an excitivity in to the user built is proceeding that the BEM arguments that are
Since these VVC methods are reused for all instances of this VVC, it is necessary with some extra parameters in order to specify which VVC instance to forward the call to. This is done with the first two (or three) parameters:		rarely altered be placed into the bfm_config parameter, while the parameters that changes often are used as input arguments.
This is done with the first two (or three) parameters:		Since these VVC methods are reused for all instances of this VVC, it is necessary with some extra parameters in order to specify which VVC instance to forward the call to.
		This is done with the first two (or three) parameters:



- signal VVCT : inout t\_vvc\_target\_record;
- constant vvc\_instance\_idx : in integer;
- constant channel : in t\_channel; -- Only if the VVC is multi-channel.

The method bodies are quite similar for all VVC commands:

- 1. First, the shared\_vvc\_cmd record is set to its default value, resetting the data from any potential previous command.
- 2. The general VVC fields (e.g. name and instance index) are set using the UVVM method set\_general\_target\_and\_command\_fields()
- 3. The VVC specific fields are set in the shared\_vvc\_cmd shared variable. This means e.g. address and data fields.
- 4. The command is sent to all VVCs using the UVVM method send\_command\_to\_vvc(VVCT)

All VVC instances and channels of this type receive the command, but only the VVC with the correct instance index, channel and name will accept it and acknowledge it.

## 5 BFM prerequisites

There are no firm restrictions of how to implement the BFM in order for the VVC to function, but if the VVC generated with the vvc\_generator script is to work out of the box, it is necessary to have some components in the BFM:

- The BFM needs to be called <name>\_bfm\_pkg.vhd. If this is not the case, the package use clauses in each of the VVC files needs to be altered.
- The BFM needs to contain a bfm\_config record type with an associated default constant. The generated VVC file assumes that this bfm config type is called t\_<name>\_bfm\_config and the constant is called C\_<NAME>\_BFM\_CONFIG\_DEFAULT. In order to support the delay operation in the VVC executor the BFM config type will also need to have a parameter clock\_period. If this is not needed, the "INSERT\_DELAY" case in the generated VVC can be removed.

A BFM skeleton that contains the necessary structure is created by the vvc\_generator script, and can be used as a base for a BFM that includes the necessary structure for the VVC to work out of the box.



# 6 UVVM Framework Packages

## 6.1 td\_target\_support\_pkg.vhd

The UVVM VVC dedicated support package contains VVC support that is common for all VVCs, but needs to be compiled specifically into each of the VVC libraries.

Section	Comment
Target record	The target record type, <i>t_vvc_target_record</i> , is used to target a VVC command to a specific VVC implementation. This is needed since many of the UVVM common commands are shared between all VVCs, e.g. await_completion() which is compiled into each VVC library. For a sequencer with two VVCs, A and B, there must be a way of determining if await_completion is to be executed in VVC A or VVC B. To resolve this, each VVC has a signal in their vvc_methods_pkg that is compiled into their own library. For VVC A and B this signal will be called A_VVCT and B_VVCT. When await_completion(A_VVCT,) is called from the sequencer, the compiler will understand that this await_completion is called with target type library_a.t_vvc_target_record, which only complies with the await_completion procedure in the VVC A library.
	td_target_support_pkg also contains a default value for the t_vvc_target_record type, and a function set_vvc_target_defaults for setting the VVC target based on the VVC name.
String methods	The package contains two string methods: - to_string(): This function converts a t_vvc_target_record, vvc_instance and vvc_channel into a string - format_command_idx(): Function which encapsulates a command record index.
send_command_to_vvc	Sends command to VVC and waits for ACK or timeout <ul> <li>Logs with ID_UVVM_SEND_CMD when sending to VVC</li> <li>Logs with ID_UVVM_CMD_ACK when ACK or timeout occurs</li> </ul>
Setting the command field	Sets target index and channel, and updates shared_vvc_cmd which is used to transport VVC commands from the central test bench sequencer to VVC.



## 6.2 td\_vvc\_entity\_support\_pkg.vhd

The VVC support package contains procedures that are compiled into and used in the VVC. This includes initializers for the executor and interpreter, and the interpreter procedures called *interpreter\_\**, e.g. *interpreter\_await\_completion*. For more information about the *interpreter\_\** procedures, please see the Common\_VVC\_Methods under doc/. For more information about the other methods in this package, see the <name>\_vvc.vhd section in this document.

In addition to the procedures, the td\_vvc\_entity\_support\_pkg also contains types for VVC labels and executor results. The result array is also defined and its shared variable is instantiated in this package.

## 7 Additional Documentation

Additional documentation about UVVM and its features can be found under "uvvm\_vvc\_framework/doc/".



Disclaimer: This IP and any part thereof are provided "as is", without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose and noninfringement. In no event shall the authors or copyright holders be liable for any claim, damages or other liability, whether in an action of contract, tort or otherwise, arising from, out of or in connection with this IP.