Verification Continuum™

VC Verification IP
LPDDR Memory
UVM Getting Started Guide

Version Q-2020.06, June 2020

SYNOPSYS®

Copyright Notice and Proprietary Information

Destination Control Statement

All technical data contained in this publication is subject to the export control laws of the United States of America. Disclosure to nationals of other countries contrary to United States law is prohibited. It is the reader's responsibility to determine the applicable regulations and to comply with them.

Disclaimer

SYNOPSYS, INC., AND ITS LICENSORS MAKE NO WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, WITH REGARD TO THIS MATERIAL, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE.

Trademarks

Synopsys and certain Synopsys product names are trademarks of Synopsys, as set forth at http://www.synopsys.com/company/legal/trademarks-brands.html.
All other product or company names may be trademarks of their respective owners.

Free and Open-Source Software Licensing Notices

If applicable, Free and Open-Source Software (FOSS) licensing notices are available in the product installation.

Third-Party Links

Any links to third-party websites included in this document are for your convenience only. Synopsys does not endorse and is not responsible for such websites and their practices, including privacy practices, availability, and content.

www.synopsys.com

Contents

Preface

About This Document

This Getting Started Guide presents information about integrating the VC VIP for LPDDR Memory (referred to as VIP) into testbenches that are compliant with the SystemVerilog Universal Verification Methodology (UVM). You are assumed to be familiar with the LPDDR Memory protocol and UVM.

Web Resources

- Documentation through SolvNet: https://solvnet.synopsys.com (Synopsys password required)
- Synopsys Common Licensing (SCL): http://www.synopsys.com/keys

Customer Support

To obtain support for your product, choose one of the following:

1. Go to https://solvnetplus.synopsys.com and open a case.

   Enter the information according to your environment and your issue.

2. Send an e-mail message to support_center@synopsys.com.

   Include the Product name, Sub Product name, and Tool Version in your e-mail so it can be routed correctly.

3. Telephone your local support center.
   - North America:

     Call 1-800-245-8005 from 7 AM to 5:30 PM Pacific time, Monday through Friday.
   - All other countries:

     https://www.synopsys.com/support/global-support-centers.html

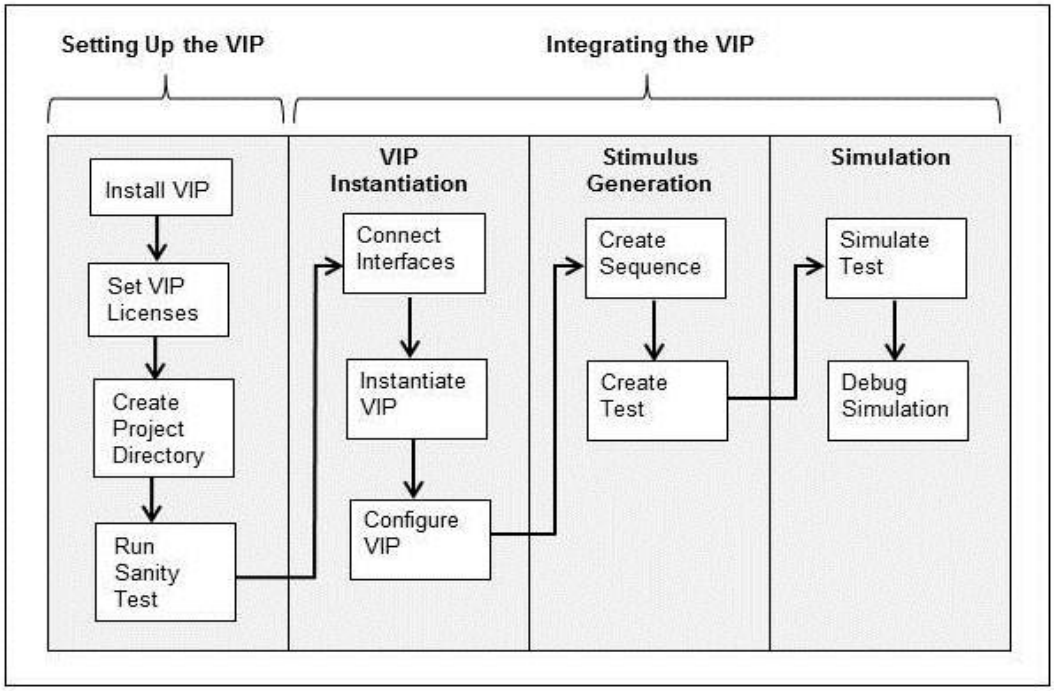
1 Overview of the Getting Started Guide

This Getting Started Guide presents information about integrating the VC VIP for LPDDR Memory (referred to as VIP) into testbenches that are compliant with the SystemVerilog Universal Verification Methodology (UVM). Figure 1-1 is the VIP integration and test work flow presented in this document. The steps for setting up the VIP are documented in the *VC Verification IP Installation and Setup Guide*. This guide is available on the SolvNet Download page and in the VIP installation at the following location:

```
$DESIGNWARE_HOME/vip/svt/lpddr_svt/latest/doc/lpddr_svt_uvm_getting_started.pdf
```

The VIP setup should be completed before executing the steps in this document.

Figure 1-1 VIP Integration and Test Work Flow



You are assumed to be familiar with the LPDDR Memory protocol and UVM. For more information on the VIP, refer to the *VC Verification IP LPDDR Memory UVM User Guide* on SolvNet (https://solvnet.synopsys.com/dow_retrieve/latest/snps_vip_lib/doc/lpddr_svt_uvm_user_guide.pdf) or in the VIP installation at the following location:

```
$DESIGNWARE_HOME/vip/svt/lpddr_svt/latest/doc/lpddr_svt_uvm_user_guide.pdf
```

2
Integrating the VIP into a User Testbench

The VC VIP for LPDDR Memory provides a suite of advanced SystemVerilog verification components and data objects that are compliant to UVM. Integrating these components and objects into any UVM compliant testbench is straightforward. For a complete list of VIP components and data objects, refer to the main page of the *VC VIP LPDDR Memory Class Reference* (only in HTML format) at the following location:

`$DESIGNWARE_HOME/vip/svt/lpddr_svt/latest/doc/lpddr_svt_uvm_class_reference/html/index.html`

2.1 LPDDR Memory Device VIP Testbench Integration Flow

The LPDDR Memory agent (svt_lpddr_agent) is the top-level component provided by the VIP for modeling Single channel LPDDR 5 , Single channel LPDDR 4, LPDDR 3, and LPDDR 2 devices.For Dual channel LPDDR4 and Dual channel LPDDR5 , see *Dual Channel LPDDR4/Dual Channel LPDDR5 Memory Device VIP Testbench Integration Flow* section. This generic agent encapsulates the following components:

- Sequencer and memory core
- Monitor
- Driver

The agent can be configured as a LPDDR2 or LPDDR3 or Single Channel LPDDR4 or Single Channel LPDDR5 device by setting the protocol kind in the LPDDR configuration class (svt_lpddr_configuration) and loading the device part catalog file. You can instantiate and construct the LPDDR Memory agent in the top-level environment of your UVM testbench.

Figure 2-1 Top-level Architecture of a Single Channel LPDDR Memory VIP Testbench

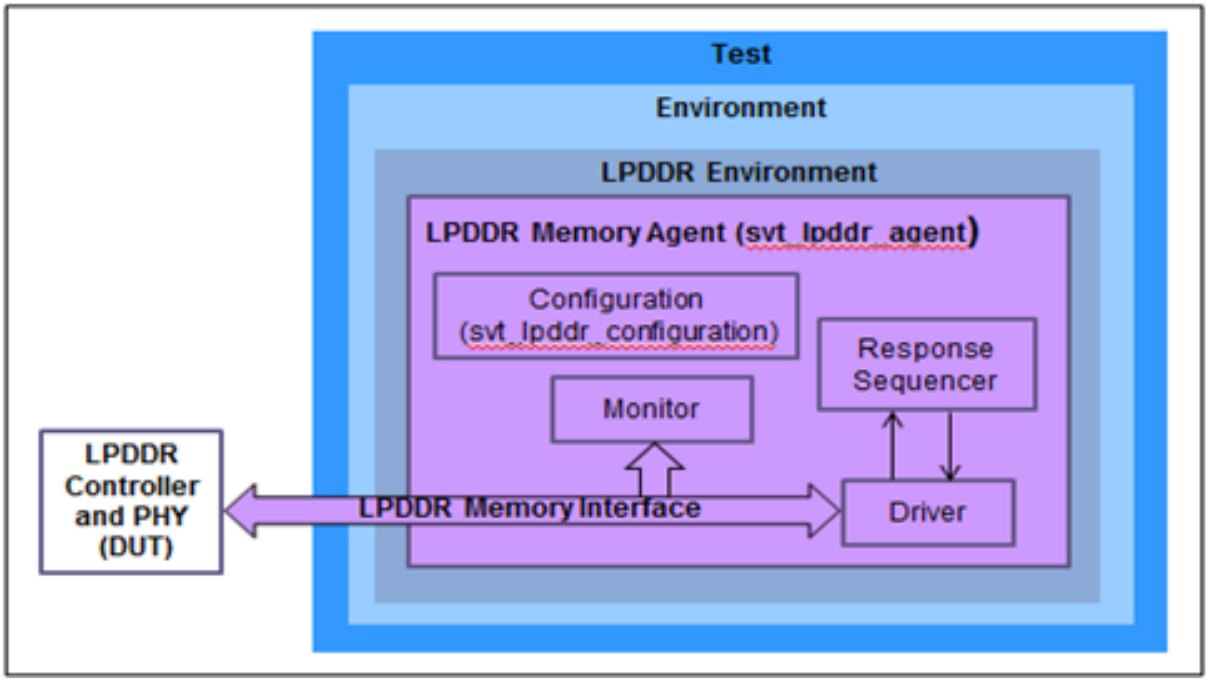


Figure 2-1 is a top-level architecture of a simple VC VIP for LPDDR Memory testbench. The steps for integrating the VIP into a UVM testbench are described in the following sections:

- "Connecting the VIP to the DUT"
- "Instantiating and Configuring the VIP"
- "Creating a Test"

The code snippets presented in this chapter are generic and can be applied to any UVM compliant testbench. For more information on the code usage, refer to the following examples:

```
$DESIGNWARE_HOME/vip/svt/lpddr_svt/latest/examples/sverilog/
tb_lpddr5_svt_uvm_basic_sys
$DESIGNWARE_HOME/vip/svt/lpddr_svt/latest/examples/sverilog/
tb_lpddr4_svt_uvm_basic_sys

$DESIGNWARE_HOME/vip/svt/lpddr_svt/latest/examples/sverilog/
tb_lpddr3_svt_uvm_basic_sys
$DESIGNWARE_HOME/vip/svt/lpddr_svt/latest/examples/sverilog/
tb_lpddr2_svt_uvm_basic_sys
```

2.1.1 Connecting the VIP to the DUT

The following are the steps to establish a connection between the VIP to the DUT in your top-level testbench:

- Include the standard UVM and VIP files and packages.

```
`include "uvm_pkg.sv"
`include "svt_lpddr_full.uvm.pkg"

import uvm_pkg::*;
`include "uvm_macros.svh"
import svt_uvm_pkg::*;
import svt_mem_uvm_pkg::*; //UVM memory package
import svt_lpddr_full_uvm_pkg::*; //UVM LPDDR full package
```

- Instantiate the top-level Single Channel LPDDR 5 Memory interface.

```
svt_lpddr5_jedec_chip_if memory_if (.ck_t    (dut.SystemClock),
                      .ck_c    (dut.SystemClock_bar),
                      .wck_t   (dut.wck_t_int),
                      .wck_c   (dut.wck_c_int),
                      .cs_p    (dut.cs_p),
                      .reset_n (dut.reset_n),
                      .zq      (dut.zq_unconn),
                      .ca      (dut.ca),
                      .dq      (dut.dq_int),
                      .rdqs_t  (dut.rdqs_t_int),
                      .rdqs_c  (dut.rdqs_c_int),
                      .dmi     (dut.dmi_int));
```

- Instantiate the top-level Single channel LPDDR 4, Single channel LPDDR3 and LPDDR 2 Memory interface.

```
svt_lpddr_jedec_chip_if memory_if  (.ck_t    (DUT.ck_t_int),
                                    .ck_c    (DUT.ck_c_int),
                                    .cke     (DUT.cke_int),
                                    .cs_p    (DUT.cs_p_int),
                                    .odt     (DUT.odt_int),
                                    .reset_n DUT.reset_n_int),
                                    .cs_n (DUT.cs_n_unconn),
                                    .zq      (DUT.zq_unconn),
                                    .dm      (DUT.dm_unconn),
                                    .ca      (DUT.ca_int),
                                    .dqs_t   (DUT.dqs_t_int),
                                    .dqs_c   (DUT.dqs_c_int),
                                    .dq      (DUT.dq_int),
                                    .dmi     (DUT.dmi_int));
```

- Connect the top-level Single channel LPDDR 5 Memory interface to the DUT and the LPDDR Memory system environment.

```
uvm_config_db#(svt_lpddr5_jedec_chip_vif)::set(uvm_root::g
et(), uvm_test_top.lpddr5_env", "lpddr5_vif", memory_if);
```

- Connect the top-level Single channel LPDDR 4, Single channel LPDDR 3,and LPDDR 2 Memory interface to the DUT and the LPDDR Memory system environment.

```
uvm_config_db#(svt_lpddr_jedec_chip_vif)::set(uvm_root::
get(), "uvm_test_top.lpddr4_env", "lpddr4_vif",memory_if);
```

The `uvm_config_db` command connects the top-level LPDDR Memory interface to the virtual interface of the LPDDR Memory environment. The `uvm_test_top` represents the top-level module in the UVM environment. The env is an instance of your testbench environment. The l`pddr_env/lpddr5_env` is an instance of the LPDDR/Single channel LPDDR5 Memory environment that encapsulates the LPDDR Memory agent (`svt_lpddr_agent`).

2.1.2 Instantiating and Configuring the VIP

The following are the steps to instantiate and configure the LPDDR Memory agent in your testbench environment.

- Instantiate the LPDDR Memory agent (svt_lpddr_agent) and construct the agent in the build phase of your testbench environment.

```
svt_lpddr_agent mem_agent;

//For backdoor access to memory core inside the VIP
svt_mem_backdoor mem_backdoor;

mem_agent = svt_lpddr_agent::type_id::create("mem_agent",
this);
```

- Create a test configuration class by extending the LPDDR configuration class (`svt_lpddr_configuration`). It is for Single channel LPDDR 5,Single channel LPDDR4, LPDDR3,and LPDDR2.This configuration class specifies the VIP type, LPDDR5, LPDDR 4, LPDDR 3 or LPDDR 2. For example,

```
class cust_svt_lpddr4/3/2_configuration extends
svt_lpddr_configuration;

function new (string
name="cust_svt_lpddr5/4/3/2_configuration");
super.new(name);
this.vip_type = svt_lpddr_type::LPDDR5; // For LPDDR5
Or
this.vip_type = svt_lpddr_type::LPDDR4; // For LPDDR4
Or
this.vip_type = svt_lpddr_type::LPDDR3; // For LPDDR3
Or
this.vip_type = svt_lpddr_type::LPDDR2; // For LPDDR2
……

endfunction: new
endclass: cust_svt_lpddr5/4/3/2_configuration
```

For more information on the configuration class, see the *svt_lpddr_configuration Class Reference* at the following location:

```
$DESIGNWARE_HOME/vip/svt/lpddr_svt/latest/doc/lpddr_svt_uvm_class_re
ference/html/configuration/class_svt_lpddr_configuration.html
```

- Configure the VIP in the build phase of your testbench environment.

Single channel LPDDR 5

```
   mem_cfg =
   cust_svt_lpddr5_configuration::type_id::create("mem_cfg");
uvm_config_db#(cust_svt_lpddr5_configuration)::set(this,
"lpddr5_env","mem_cfg", this.mem_cfg);
```

Single channel LPDDR 4

```
mem_cfg =
cust_svt_lpddr4_configuration::type_id::create("mem_cfg");

uvm_config_db#(cust_svt_lpddr4_configuration)::set(this,
"lpddr4_env","mem_cfg", this.mem_cfg);
```

LPDDR 3

```
mem_cfg =
cust_svt_lpddr3_configuration::type_id::create("mem_cfg");
 uvm_config_db#(cust_svt_lpddr3_configuration)::set(this,
"lpddr_env","mem_cfg", this.mem_cfg);
```

LPDDR 2

```
mem_cfg =
cust_svt_lpddr2_configuration::type_id::create("mem_cfg");

uvm_config_db#(cust_svt_lpddr2_configuration)::set(this,
"lpddr_env","mem_cfg", this.mem_cfg);
```

The `cust_svt_lpddr_configuration`is the test configuration as defined in the previous step. The `cfg` is an instance of this configuration.

- Specify a part number to load the catalog information in the configuration file.

LPDDR4

```
class lpddr4_part_number_policy;
 static function int weight(svt_lpddr4_vendor_part part);
    string lpddr4_part_number;
    if ($value$plusargs("part_number=%s", lpddr4_part_number)) begin
      return part.get_part_number() == lpddr4_part_number;
    end
    else  begin
      return part.get_part_number() ==
"jedec_lpddr4_8G_x16_1600_1_25";
    end
  endfunction: weight
endclass: lpddr4_part_number_policy


class lpddr4_base_test extends uvm_test;

// Select a part number from the catalog
          if (lpddr4_ext == 1'b1)
        lpddr4A_vendor_part =
svt_mem_part_mgr#(svt_lpddr4A_vendor_part,lpddr4A_part_number_policy):
:pick();
      else
      lpddr4_vendor_part =
svt_mem_part_mgr#(svt_lpddr4_vendor_part,lpddr4_part_number_policy)::p
ick();
    end
    else begin
    lpddr4_vendor_part =
svt_mem_part_mgr#(svt_lpddr4_vendor_part,lpddr4_part_number_policy)::p
ick();
    end

 if ((lpddr4_vendor_part != null)&&(lpddr4A_vendor_part == null))
begin
 if (mem_cfg.load_prop_vals(lpddr4_vendor_part.get_cfgfile())) begin
...
end

else if ((lpddr4_vendor_part == null)&&(lpddr4A_vendor_part != null))
begin
      if (mem_cfg.load_prop_vals(lpddr4A_vendor_part.get_cfgfile()))
begin
...
end
...
endclass
```

LPDDR3

```
class lpddr3_part_number_policy;
static function int weight(svt_lpddr3_vendor_part part);
…
//Part number is "jedec_lpddr3_4G_x32_1333_1_5"
return part.get_part_number()=="jedec_lpddr3_4G_x32_1333_1_5";
…
end
endfunction: weight
endclass: lpddr3_part_number_policy

class lpddr3_base_test extends uvm_test;
…
//Use a policy class to select the LPDDR3 Memory part number and print
//the catalog features
lpddr3_vendor_part = svt_mem_part_mgr#(svt_lpddr3_vendor_part,
lpddr3_part_number_policy)::pick();

//Use the "load_prop_vals()" method to load the configuration values
//of the selected part into the configuration object
if (mem_cfg.load_prop_vals(lpddr3_vendor_part.get_cfgfile()))
begin
…
end
…
endclass
```

LPDDR2

```
class lpddr2_part_number_policy;
static function int weight(svt_lpddr2_vendor_part part);
…
//Part number is " jedec_lpddr2_s2_1G_x32_1066_1_875"
return part.get_part_number()=="jedec_lpddr2_s2_1G_x32_1066_1_875";
…
end
endfunction: weight
endclass: lpddr2_part_number_policy

class lpddr2_base_test extends uvm_test;
…
//Use a policy class to select the LPDDR2 Memory part number and print
//the catalog features
Lpddr2_vendor_part = svt_mem_part_mgr#(svt_lpddr2_vendor_part,
Lpddr2_part_number_policy)::pick();

//Use the "load_prop_vals()" method to load the configuration values
//of the selected part into the configuration object
if (mem_cfg.load_prop_vals(lpddr2_vendor_part.get_cfgfile()))
begin
…
end
…
endclass
```

2.1.3 Creating a Test

You can create a base test class (`lpddr_base_test`) to specify the default test behaviors and to serve as the base class for other LPDDR Memory tests. The following code snippets can be used in the base test class file (*lpddr_base_test.sv*).

- Instantiate the testbench environment and the test configuration, and construct these components in the build phase.

For Single channel LPDDR5

```
`include "lpddr5_basic_env.sv"
`include "cust_svt_lpddr5_configuration.sv"
// Allocate the configuration object
mem_cfg = cust_svt_lpddr5_configuration::type_id::create("mem_cfg");
// Send the memory cfg to the testbench ENV
    uvm_config_db#(cust_svt_lpddr5_configuration)::set(this,
"lpddr5_env","mem_cfg", this.mem_cfg);
// Create the testbench ENV
    lpddr5_env = lpddr5_basic_env::type_id::create("lpddr5_env",this);
```

For Single channel LPDDR4

```
`include "lpddr4_basic_env.sv"
`include "cust_svt_lpddr4_configuration.sv"
// Allocate the configuration object
  mem_cfg = cust_svt_lpddr4_configuration::type_id::create("mem_cfg");
// Send the memory cfg to the testbench ENV
    uvm_config_db#(cust_svt_lpddr4_configuration)::set(this,
"lpddr4_env","mem_cfg", this.mem_cfg);
// Create the testbench ENV
    lpddr4_env = lpddr4_basic_env::type_id::create("lpddr4_env",this);
```

For LPDDR3

```
`include "lpddr3_basic_env.sv"
`include "cust_svt_lpddr3_configuration.sv"
// Allocate the configuration object
  mem_cfg = cust_svt_lpddr3_configuration::type_id::create("mem_cfg");

 // Send the memory cfg to the testbench ENV
    uvm_config_db#(cust_svt_lpddr3_configuration)::set(this,
"lpddr_env","mem_cfg", this.mem_cfg);
// Create the testbench ENV
    lpddr_env = lpddr3_basic_env::type_id::create("lpddr_env",this);
```

For LPDDR2

```
`include "lpddr2_basic_env.sv"
`include "cust_svt_lpddr2_configuration.sv"
// Allocate the configuration object
    mem_cfg =
cust_svt_lpddr2_configuration::type_id::create("mem_cfg");

// Send the memory cfg to the testbench ENV
    uvm_config_db#(cust_svt_lpddr2_configuration)::set(this,
"lpddr_env","mem_cfg", this.mem_cfg);
// Create the testbench ENV
    lpddr_env = lpddr2_basic_env::type_id::create("lpddr_env",this);
```

Examples from the VIP installation include a set of basic LPDDR Memory tests. These tests are extended from the base test (`lpddr_base_test`) to create different test scenarios. For more information on the VIP tests, see the test files in the following directories:

Single channel LPDDR5

```
$DESIGNWARE_HOME/vip/svt/lpddr_svt/latest/examples/sverilog/tb_lpddr5_svt_uvm_
basic_sys/tests
```

Single channel LPDDR4

```
$DESIGNWARE_HOME/vip/svt/lpddr_svt/latest/examples/sverilog/
tb_lpddr4_svt_uvm_basic_sys/tests
```

LPDDR3:

```
$DESIGNWARE_HOME/vip/svt/lpddr_svt/latest/examples/sverilog/
tb_lpddr3_svt_uvm_basic_sys/tests
```

LPDDR2:

```
$DESIGNWARE_HOME/vip/svt/lpddr_svt/latest/examples/sverilog/
tb_lpddr2_svt_uvm_basic_sys/tests
```

In the VIP examples, a dummy LPDDR Memory controller agent component is used to model the controller and PHY. In a user testbench, actual LPDDR Memory RTL controller and PHY are used.

Note
The LPDDR Memory VIP is a reactive component and does not raise and drop objections. Users must implement the raise and drop of objections. Otherwise, tests will finish at simulation time 0.

2.2 Dual Channel LPDDR4/Dual Channel LPDDR5 Memory Device VIP Testbench Integration Flow

The `svt_lpddr4_env/svt_lpddr5_env` is top-level component provided by the VIP for modeling Dual Channel LPDDR4/ Dual Channel LPDDR5 devices. This class encapsulates the following components:

- Two agents(channel_a and channel_b) of `svt_lppdr_agent` class type to model Channel A and Channel B.

Note
To make it consistent with LPDDR4, Synopsys LPDDR5 VIP provides provision to have a single wrapper consisting of two LPDDR5 single channels.

The env is be configured as a Dual Channel LPDDR4/ Dual Channel LPDDR5 device by setting the VIP type of the `cfg_channel_a` and `cfg_channel_b` configuration class (`svt_lpddr_configuration`) handles within the `svt_lpddr4_configuration/ svt_lpddr5_configuration` class and loading the device part catalog file. You can instantiate and construct the `svt_lpddr4_env` in the top-level environment of your UVM testbench.

Figure 2-2 is a top-level architecture of a simple VC VIP for LPDDR Memory testbench. The steps for integrating the VIP into a UVM testbench are described in the following sections:

The code snippets presented in this chapter are generic and can be applied to any UVM compliant testbench. For more information on the code usage, refer to the following examples:

The steps for integrating the VIP into a UVM testbench are described in the following sections:

- “Connecting the VIP to the DUT”
- “Instantiating and Configuring the VIP”
- “Creating a Test”

The code snippets presented in this chapter are generic and can be applied to any UVM compliant testbench. . For more information on the code usage, see the following examples of LPDDR 4 and LPDDR Dual channel:

```
$DESIGNWARE_HOME/vip/svt/lpddr_svt/latest/examples/sverilog/
tb_lpddr4_dual_ch_svt_uvm_basic_sys
```

```
$DESIGNWARE_HOME/vip/svt/lpddr_svt/latest/examples/sverilog/
tb_lpddr5_dual_ch_svt_uvm_basic_sys
```

2.2.1 Connecting the VIP to the DUT

The following are the steps to establish a connection between the VIP to the DUT in your top-level testbench:

- Include the standard UVM and VIP files and packages.

```
`include "uvm_pkg.sv"
`include "svt_lpddr_full.uvm.pkg"

import uvm_pkg::*;
`include "uvm_macros.svh"
import svt_uvm_pkg::*;
import svt_mem_uvm_pkg::*; //UVM memory package
import svt_lpddr_full_uvm_pkg::*; //UVM LPDDR full package
```

- Instantiate the Dual Channel LPDDR5 Memory interface

```
`include "uvm_pkg.sv"
`include "svt_lpddr_full.uvm.pkg"

import uvm_pkg::*;
`include "uvm_macros.svh"
import svt_uvm_pkg::*;
import svt_mem_uvm_pkg::*; //UVM memory package
import svt_lpddr_full_uvm_pkg::*; //UVM LPDDR full package
```

- (`svt_lpddr5_dual_chan_jedec_chip_if`) and connect it with DUT ports.

```
svt_lpddr5_dual_chan_jedec_chip_if   memory_if (.ck_t_a
(dut0.SystemClock),
                                       .ck_c_a (dut0.SystemClock_bar),
                                       .ck_t_b    (dut1.SystemClock),
                                       .ck_c_b (dut1.SystemClock_bar),
                                       .wck_t_a   (dut0.wck_t_int),
                                       .wck_c_a   (dut0.wck_c_int),
                                       .wck_t_b   (dut1.wck_t_int),
                                       .wck_c_b   (dut1.wck_c_int),
                                       .cs_p_a    (dut0.cs_p),
                                       .cs_p_b    (dut1.cs_p),
                                       .reset_n (dut0.reset_n),
                                       .zq       (dut0.zq_unconn),
                                       .ca_a      (dut0.ca),
                                       .ca_b      (dut1.ca),
                                       .dq_a      (dut0.dq_int),
                                       .dq_b      (dut1.dq_int),
                                       .rdqs_t_a  (dut0.rdqs_t_int),
                                       .rdqs_c_a  (dut0.rdqs_c_int),
                                       .rdqs_t_b  (dut1.rdqs_t_int),
                                       .rdqs_c_b  (dut1.rdqs_c_int),
                                       .dmi_a     (dut0.dmi_int),
                                       .dmi_b     (dut1.dmi_int)
                                     );
```

- Connect the top-level LPDDR Memory interface with LPDDR Memory system environment.

```
uvm_config_db#(svt_lpddr5_dual_chan_jedec_chip_vif)::set(uvm_roo
t::get(), "uvm_test_top.lpddr5_env", "lpddr5_vif", memory_if).
```

- Instantiate the Dual Channel LPDDR4 Memory interface (svt_lpddr4_jedec_chip_if) and connect it with DUT ports.

```
svt_lpddr4_jedec_chip_if memory_if(.ck_t_a   (ck_t_int_a),
                                   .ck_c_a   (ck_c_int_a),
                                   .cke_a    (cke_int_a),
                                   .cs_p_a   (cs_p_int_a),
                                   .ck_t_b   (ck_t_int_b),
                                   .ck_c_b   (ck_c_int_b),
                                   .cke_b    (cke_int_b),
                                   .cs_p_b   (cs_p_int_b),
                                   .odt      (odt_int),
                                   .reset_n  (reset_n_int),
                                   .zq       (zq_unconn),
                                   .ca_a     (ca_int_a),
                                   .dqs_t_a  (dqs_t_int_a),
                                   .dqs_c_a  (dqs_c_int_a),
                                   .dq_a     (dq_int_a),
                                   .dmi_a    (dmi_int_a),
                                   .ca_b     (ca_int_b),
                                   .dqs_t_b  (dqs_t_int_b),
                                   .dqs_c_b  (dqs_c_int_b),
                                   .dq_b     (dq_int_b),
                                   .dmi_b    (dmi_int_b));
```

- Connect the top-level LPDDR Memory interface with LPDDR Memory system environment.

```
uvm_config_db#(svt_lpddr4_jedec_chip_vif)::set(uvm_root::get(
), "uvm_test_top.lpddr4_env", "lpddr4_vif", memory_if);
```

The `uvm_config_db` command connects the top-level LPDDR Memory interface to the virtual interface of the LPDDR Memory environment. The `uvm_test_top` represents the top-level module in the UVM environment. The env is an instance of your testbench environment. The `lpddr_env` is an instance of the LPDDR Memory environment that encapsulates the Dual Channel LPDDR4 Memory environment (`svt_lpddr4_env`) and the `lpddr5_env` is an instance of the LPDDR Memory environment that encapsulates the Dual Channel LPDDR5 Memory environment.

2.2.2 Instantiating and Configuring the VIP

The following are the steps to instantiate and configure the Dual Channel LPDDR4/ / Dual Channel LPDDR5 Memory env in your testbench environment.

- Create a test configuration class by extending the Dual Channel LPDDR4// Dual Channel LPDDR5 configuration class (`svt_lpddr4_configuration/ svt_lpddr5_configuration`).

This configuration class specifies the VIP type, LPDDR4/LPDDR5 for `cfg_channel_a` and `cfg_channel_b` handles within the `svt_lpddr4_configuration`. For example:

```
class cust_svt_lpddr4_configuration extends
svt_lpddr4_configuration;

function new (string name="cust_svt_lpddr_configuration");
super.new(name);

this.cfg_channel_a.vip_type = svt_lpddr_type::LPDDR4;
this.cfg_channel_b.vip_type = svt_lpddr_type::LPDDR4;
……

endfunction: new
endclass: cust_svt_lpddr4_configuration
```

For Dual Channel LPDDR5

```
class cust_svt_lpddr5_configuration extends
svt_lpddr5_configuration;
function new (string name="cust_svt_lpddr5_configuration");

    this.cfg_channel_a.vip_type = svt_lpddr_type::LPDDR5;
    this.cfg_channel_b.vip_type = svt_lpddr_type::LPDDR5;
……

endfunction: new
endclass: cust_svt_lpddr5_configuration
```

For more information on the configuration class, see the *svt_lpddr4_configuration / svt_lpddr5_configuration Class Reference* at the following location:

For LPDDR 4:

```
$DESIGNWARE_HOME/vip/svt/lpddr_svt/latest/doc/lpddr_svt_uvm_class_re
ference/html/configuration/class_svt_lpddr4_configuration.html
```

For LPDDR5

```
$DESIGNWARE_HOME/vip/svt/lpddr_svt/latest/doc/lpddr_svt_uvm_class_re
ference/html/configuration/class_svt_lpddr5_configuration.html
```

- Configure the VIP in the build phase of your testbench environment.

For Dual Channel LPDDR4:

```
mem_cfg =
cust_svt_lpddr4_configuration::type_id::create("mem_cfg");

// Send the memory cfg to the testbench ENV
      uvm_config_db#(cust_svt_lpddr4_configuration)::set(this,
   "lpddr4_env","mem_cfg", this.mem_cfg);
```

The `cust_svt_lpddr4_configuration` is the test configuration as defined in the previous step. The "`mem_cfg`" is an instance of this configuration.

For Dual Channel LPDDR5

```
mem_cfg =
cust_svt_lpddr5_configuration::type_id::create("mem_cfg");
// Send the memory cfg to the testbench ENV
uvm_config_db#(cust_svt_lpddr5_configuration)::set(this,
   "lpddr5_env","mem_cfg", this.mem_cfg);
```

- Specify a part number to load the catalog information in the configuration file.

For Dual Channel LPDDR4:

```
class lpddr4_part_number_policy;
static function int weight(svt_lpddr4_vendor_part part);
…
//Part number is " jedec_lpddr4_8G_x16_1600_1_25"
return part.get_part_number()=="jedec_lpddr4_8G_x16_1600_1_25";
…
end
endfunction: weight
endclass: lpddr4_part_number_policy

class lpddr4_base_test extends uvm_test;
…
//Use a policy class to select the LPDDR4 Memory part number and print
//the catalog features
Lpddr4_vendor_part =
svt_mem_part_mgr#(svt_lpddr4_vendor_part,
lpddr4_part_number_policy)::pick();

//Use the "load_prop_vals()" method to load the configuration values
//of the selected part into the configuration object
if(mem_cfg.cfg_channel_a.load_prop_vals(lpddr4_vendor_part.ge
t_cfgfile())) begin
…
if
(mem_cfg.cfg_channel_b.load_prop_vals(lpddr4_vendor_part.get_
cfgfile())) begin
       ...
      end
end
…
if(mem_cfg.cfg_channel_b.load_prop_vals(
    lpddr4_vendor_part.get_cfgfile()))
begin
…
end
…

endclass
```

For Dual Channel LPDDR5

```
class lpddr5_part_number_policy;
static function int weight(svt_lpddr5_vendor_part part);
…

//Part number is " jedec_lpddr5_8G_x16_5800_1_38"
return part.get_part_number()=="jedec_lpddr5_8G_x16_5800_1_38";
…
end
endfunction: weight
endclass: lpddr5_part_number_policy

class lpddr5_base_test extends uvm_test;
lpddr5_vendor_part =
svt_mem_part_mgr#(svt_lpddr5_vendor_part,lpddr5_part_number_p
olicy)::pick();
// Load the configuration object with the configuration values for this part number
    if (lpddr5_vendor_part != null) begin
      if
(mem_cfg.cfg_channel_a.load_prop_vals(lpddr5_vendor_part.get_
cfgfile())) begin
        `uvm_info("generate_configuration",
$sformatf("Successfully loaded the memory configuration
values for %s", lpddr5_vendor_part.get_part_number()),
UVM_LOW);
      end
     else begin
        `uvm_fatal("generate_configuration",
$sformatf("Unable to load the memory configuration values for
%s", lpddr5_vendor_part.get_part_number()));
      end

      if
(mem_cfg.cfg_channel_b.load_prop_vals(lpddr5_vendor_part.get_
cfgfile())) begin
        `uvm_info("generate_configuration",
$sformatf("Successfully loaded the memory configuration
values for %s", lpddr5_vendor_part.get_part_number()),
UVM_LOW);
      end
     else begin
        `uvm_fatal("generate_configuration",
$sformatf("Unable to load the memory configuration values for
%s", lpddr5_vendor_part.get_part_number()));
      end

    end
   else begin
      `uvm_fatal("generate_configuration", "Unable to select
a part number from the catalog");
    end
```

2.2.3 Creating a Test

You can create a base test class to specify the default test behaviors and to serve as the base class for other LPDDR Memory tests. The following code snippets can be used in the base test class file

- Instantiate the testbench environment and the test configuration, and construct these components in the build phase.

  For LPDDR4

```
`include "lpddr4_basic_env.sv" //suggested UVM environment file
`include "cust_svt_lpddr4_configuration.sv" //suggested test
//configuration file
//build_phase
mem_cfg =
cust_svt_lpddr4_configuration::type_id::create("mem_cfg");

uvm_config_db#(cust_svt_lpddr4_configuration)::set(this,
"lpddr4_env","mem_cfg", this.mem_cfg);

 lpddr4_env =
lpddr4_basic_env::type_id::create("lpddr4_env",this);
```

  For LPDDR5

```
`include "lpddr5_basic_env.sv" //suggested UVM environment file `include
"cust_svt_lpddr5_configuration.sv" //suggested test //configuration file
//build_phase
mem_cfg =
cust_svt_lpddr5_configuration::type_id::create("mem_cfg");

uvm_config_db#(cust_svt_lpddr5_configuration)::set(this,
"lpddr5_env","mem_cfg", this.mem_cfg);
lpddr5_env =
lpddr5_basic_env::type_id::create("lpddr5_env",this);
```

Examples from the VIP installation include a set of basic LPDDR Memory tests. These tests are extended from the base test to create different test scenarios. For more information on the VIP tests, see the test files in the following directories:

For LPDDR4:

```
$DESIGNWARE_HOME/vip/svt/lpddr_svt/latest/examples/sverilog/
tb_lpddr4_dual_ch_svt_uvm_basic_sys/tests
```

For LPDDR5:

```
$DESIGNWARE_HOME/vip/svt/lpddr_svt/latest/examples/sverilog/
tb_lpddr5_dual_ch_svt_uvm_basic_sys/tests
```

In the VIP examples, a dummy LPDDR Memory controller agent component is used to model the controller and PHY. In a user testbench, actual LPDDR Memory RTL controller and PHY are used.

Note

The LPDDR Memory VIP is a reactive component and does not raise and drop objections. Users must implement the raise and drop of objections. Otherwise, tests will finish at simulation time 0.

2.3 Changing the Part Number at Runtime

You can use the $value$plusarg method in the policy class to select a different part number at runtime without recompiling the testbench.

For example:

LPDDR3:

```
class lpddr3_part_number_policy;
static function int weight(svt_lpddr3_vendor_part part);
string lpddr3_part_number;

if ($value$plusargs("part_number=%s", lpddr3_part_number))
begin return part.get_part_number() == lpddr3_part_number;

end

end
else begin
…

endfunction: weight
endclass: lpddr3_part_number_policy
```

Note
Similarly this is applicable for LPDDR 2,3,4and 5.

2.4 Compiling and Simulating a Test with the VIP

The steps for compiling and simulating a test with the VIP are described in the following sections:

- "Directory Paths for VIP Compilation"
- "VIP Compile-time Options"
- "VIP Runtime Option"

2.4.1 Directory Paths for VIP Compilation

You need to specify the following directory paths in the compilation commands for the compiler to load the VIP files.

```
+incdir+project_directory_path/include/sverilog
+incdir+project_directory_path/src/sverilog/simulator
```

Where, *project_directory_path* is your project directory and *simulator* is vcs, ncv or mti.

For example:

```
+incdir+/home/project1/testbench/vip/include/sverilog
+incdir+/home/project1/testbench/vip/src/sverilog/vcs
```

2.4.2 VIP Compile-time Options

The following are the required compile-time options for compiling a testbench with the VC VIP for LPDDR Memory:

```
+define+SVT_UVM_TECHNOLOGY
+define+UVM_PACKER_MAX_BYTES=1500000
+define+SYNOPSYS_SV
+incdir+project_directory_path/vip/svt/common/latest/C/lib/platform/
libmemserver.so
```

Where, *project_directory_path* is your project directory and *platform* is linux, amd64 or suse64.

Note

UVM_PACKER_MAX_BYTES define needs to be set to maximum value as required by each VIP title in your testbench. For example, if VIP title 1 needs UVM_PACKER_MAX_BYTES to be set to 8192, and VIP title 2 needs UVM_PACKER_MAX_BYTES to be set to 500000, you need to set UVM_PACKER_MAX_BYTES to 500000.

Macro	Description
SVT_UVM_TECHNOLOGY	Specifies SystemVerilog based VIPs that are compliant with UVM
UVM_PACKER_MAX_BYTES	Sets to 1500000 or greater
UVM_DISABLE_AUTO_ITEM_RECORDING	Disables the UVM automatic transaction begin and end event triggering and recording
SYNOPSYS_SV	Specifies SystemVerilog based VIPs that are compliant with UVM

2.4.3 VIP Runtime Option

No VIP specific runtime option is required to run simulations with the VIP. Only relevant UVM runtime options are required.

For example:

```
+UVM_TESTNAME=basic_wr_rd_test
```

A Summary of Commands, Documents, and Examples

A.1 Commands in This Document

Display VIP models and examples under the VIP installation directory specified by $DESIGNWARE_HOME:

```
% $DESIGNWARE_HOME/bin/dw_vip_setup -info home
```

Add VIP models to the project directory:

```
% $DESIGNWARE_HOME/bin/dw_vip_setup  -path project_directory -add VIP_model
-svlog
```

Add VIP examples to the directory where the command is executed:

```
% $DESIGNWARE_HOME/bin/dw_vip_setup -e VIP_example -svlog -doc
```

A.2 Primary Documentation for VC VIP LPDDR Memory

VC Verification IP UVM Installation and Setup Guide:

`$DESIGNWARE_HOME/vip/svt/common/latest/doc/uvm_install.pdf`

VC VIP LPDDR Memory UVM User Guide:

`https://solvnet.synopsys.com/dow_retrieve/latest/snps_vip_lib/doc/lpddr_svt_uvm_user_guide.pdf`

VC VIP LPDDR Memory Getting Started Guide:

`https://solvnet.synopsys.com/dow_retrieve/latest/snps_vip_lib/doc/lpddr_svt_uvm_getting_started.pdf`

VC VIP LPDDR Memory QuickStarts:

```
https://solvnet.synopsys.com/dow_retrieve/latest/snps_vip_lib/lpddr_quickstart_doc.pdf
```

VC VIP LPDDR Memory Class Reference:

```
https://solvnet.synopsys.com/dow_retrieve/latest/snps_vip_lib/doc/lpddr_svt_uvm_class_reference/html/index.html
```

VC VIP LPDDR Memory Verification Plans:

```
$DESIGNWARE_HOME/vip/svt/lpddr_svt/latest/doc/VerificationPlans
```

A.3 Example Home Directory

Directory that contains a list of VIP example directories:

```
$DESIGNWARE_HOME/vip/svt/lpddr_svt/latest/examples/sverilog
```

View simulation options for each example:

```
gmake help
```