Verification Continuum™

VC Verification IP DDR Memory UVM Getting Started Guide

Version Q-2020.06, June 2020

SYNOPSYS®

Contents

Preface

About This Document

This Getting Started Guide presents information about integrating the VC VIP for DDR 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 DDR Memory protocol and UVM.

Web Resources

- ❖ Documentation through SolvNet: https://solvnetplus.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 Product version for which you want to register the problem.
   - ✦ If applicable, provide the information noted in Appendix A, "Reporting Problems" on page 59.
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:

     http://www.synopsys.com/Support/GlobalSupportCenters

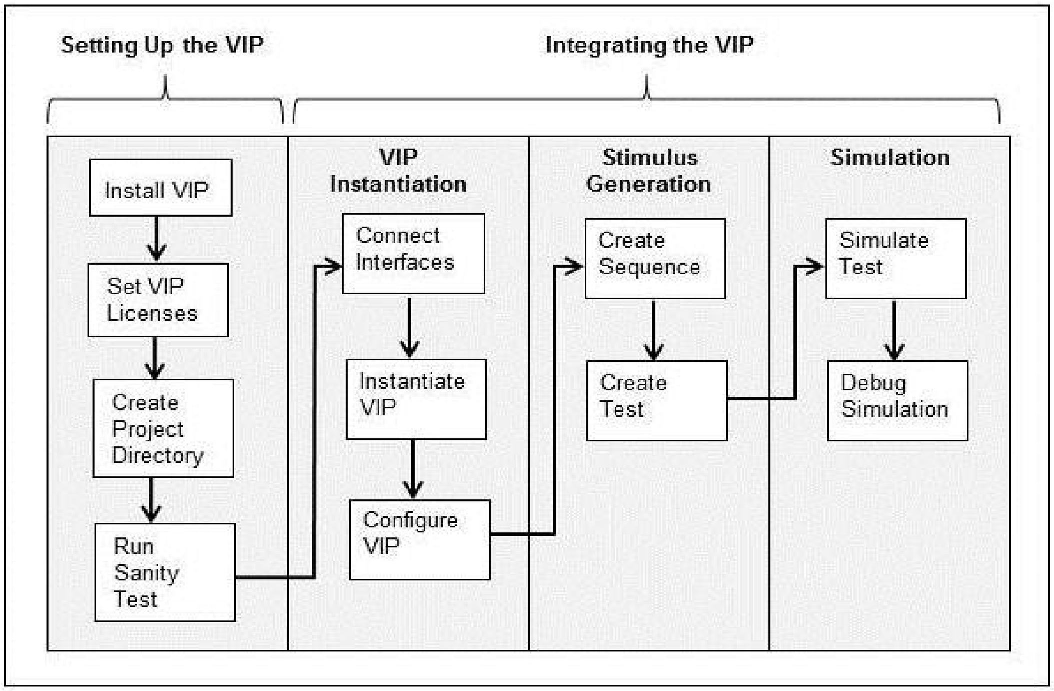
1 Overview of the Getting Started Guide

This Getting Started Guide presents information about integrating the VC VIP for DDR 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/common/latest/doc/uvm_install.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 DDR Memory protocol and UVM. For more information on the VIP, refer to the *VC Verification IP DDR Memory UVM User Guide* on SolvNet (click here) or in the VIP installation at the following location:

```
$DESIGNWARE_HOME/vip/svt/ddr_svt/latest/doc/ddr_svt_uvm_user_guide.pdf
```

2 Integrating the VIP into a User Testbench

The VC VIP for DDR 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 DDR Memory Class Reference* (only in HTML format) at the following location:

```
$DESIGNWARE_HOME/vip/svt/ddr_svt/latest/doc/ddr_svt_uvm_class_reference/html/
index.html
```

2.1 DDR Memory Device VIP Testbench Integration Flow

The DDR Memory agent (svt_ddr_agent) is the top-level component provided by the VIP for modeling DDR2, DDR3, DDR4, DDR5 and NVDIMM-P devices. This generic agent encapsulates the following components:

- Sequencer and memory core
- Monitor
- Driver

The agent can be configured as a DDR2, DDR3, DDR4 , DDR5 and NVDIMM-Pdevice by setting the protocol kind in the DDR configuration class (svt_ddr_configuration) and loading the device part catalog file. You can instantiate and construct the DDR Memory agent in the top-level environment of your UVM testbench.

Figure 2-1 Top-level Architecture of a DDR Memory VIP Testbench

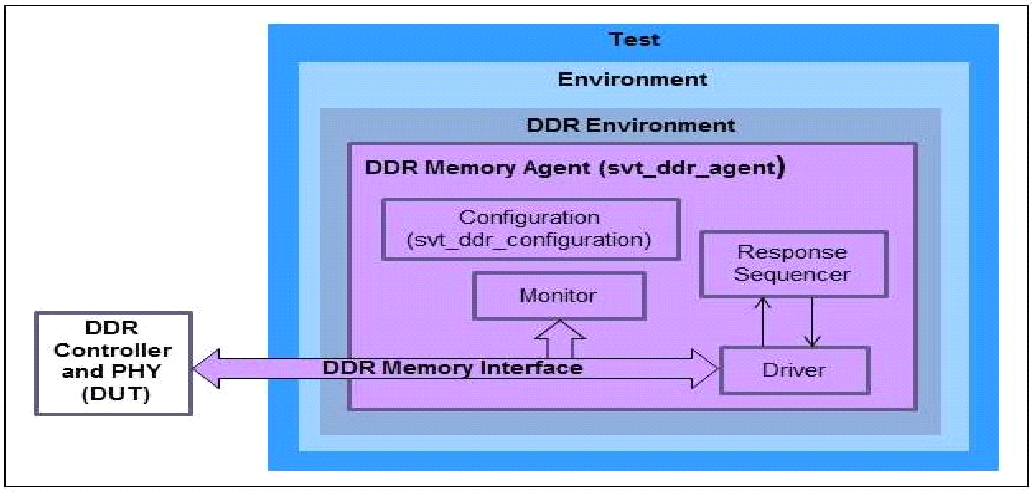

Test
Environment
DDR Environment
DDR Memory Agent (svt_ddr_agent)
Configuration (svt_ddr_configuration)
Response Sequencer
Monitor
DDR Controller and PHY (DUT)
DDR Memory Interface
Driver

Figure 2-1 is a top-level architecture of a simple VC VIP for DDR 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/ddr_svt/latest/examples/sverilog/
tb_ddr4_svt_uvm_basic_sys

$DESIGNWARE_HOME/vip/svt/ddr_svt/latest/examples/sverilog/
tb_ddr3_svt_uvm_basic_sys

$DESIGNWARE_HOME/vip/svt/ddr_svt/latest/examples/sverilog/
tb_ddr2_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_ddr.uvm.pkg”
`include "svt_ddr_if.svi" //Top-level DDR Memory interface
`include "svt_ddr_controller.uvm.pkg"

import uvm_pkg::*;
import svt_uvm_pkg::*;
import svt_mem_uvm_pkg::*; //UVM memory package
import svt_ddr_uvm_pkg::*; //UVM DDR package
import svt_ddr_controller_uvm_pkg::*; // UVM controller package
`include "svt_ddr4_catalog.svi" //For DDR4 catalog files
`include "svt_ddr3_catalog.svi" //For DDR3 catalog files
`include "svt_ddr2_catalog.svi" // For DDR2 catalog files
```

✦ Instantiate the top-level DDR Memory interface.

DDR4 3DS

```
svt_ddr4_jedec_if memory_if();
```

DDR3:

```
svt_ddr3_jedec_if memory_if();
```

✦ Connect the top-level DDR Memory interface to the DUT and the DDR Memory system environment.

DDR4 3DS:

```
dut dut_inst(memory_if);

uvm_config_db#(svt_ddr4_jedec_vif)::set(uvm_root::get(),
"uvm_test_top.env.ddr_env", "memory_if", memory_if);
```

DDR3:

```
dut dut_inst(memory_if);

uvm_config_db#(svt_ddr3_jedec_vif)::set(uvm_root::get(),
"uvm_test_top.env.ddr_env", "memory_if", memory_if);
```

The `uvm_config_db` command connects the top-level DDR Memory interface to the virtual interface of the DDR 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 "`ddr_env`" is an instance of the DDR Memory environment that encapsulates the DDR Memory agent (svt_ddr_agent).

2.1.2 Instantiating and Configuring the VIP

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

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

```
svt_ddr_agent memory_agent;

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

memory_agent = svt_ddr_agent::type_id::create("memory_agent",
this);
```

- Create a test configuration class by extending the DDR configuration class (svt_ddr_configuration). This configuration class specifies the kind of protocol, DDR2, DDR4 3DS, DDR4, or DDR3.

  For example:

```
class cust_svt_ddr_configuration extends
svt_ddr_configuration;

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

      this.protocol_kind = DDR4;
      //OR
      this.protocol_kind = DDR3;

// OR
this.protocol_kind = DDR2;

   endfunction: new
endclass: cust_svt_ddr_configuration
```

  For more information on the configuration class, refer to the *svt_ddr_configuration Class Reference* at the following location:

```
$DESIGNWARE_HOME/vip/svt/ddr_svt/latest/doc/
ddr_svt_uvm_class_reference/html/class_svt_ddr_configuration.html
```

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

```
cfg = cust_svt_ddr_configuration::type_id::create("cfg");

uvm_config_db#(svt_ddr_configuration)::set(this,
"memory_agent",
"cfg", cfg);
```

The "`cust_svt_ddr_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.

DDR4:

```
class svt_ddr4_part_number_policy;
  static function int weight(svt_ddr4_vendor_part part);
            …
      //Part number is "jedec_ddr4_4G_x16_1600L_1_6"
   return part.get_part_number()=="jedec_ddr4_4G_x16_1600L_1_6";
    end
  endfunction: weight
endclass: svt_ddr4_part_number_policy

class ddr_base_test extends uvm_test;
            …
   //Use a policy class to select the DDR4 Memory part number and print
  //the catalog features
  ddr4_vendor_part = svt_mem_part_mgr#(svt_ddr4_vendor_part,
                     svt_ddr4_part_number_policy)::pick();

   //Use the "load_prop_vals()" method to load the configuration values
 //of the selected part into the configuration object
  if (cfg.load_prop_vals(ddr4_vendor_part.get_cfgfile())) begin
            …
  end
            …
endclass
```

DDR3:

```
class svt_ddr3_part_number_policy;
  static function int weight(svt_ddr3_vendor_part part);
            …
   //Part number is "jedec_ddr3_2G_x16_1066F_1_875"
   return part.get_part_number()=="jedec_ddr3_2G_x16_1066F_1_875";
            …
    end
  endfunction: weight
endclass: svt_ddr3_part_number_policy

class ddr_base_test extends uvm_test;
            …
  //Use a policy class to select the DDR3 Memory part number and print
 //the catalog features
  ddr3_vendor_part = svt_mem_part_mgr#(svt_ddr3_vendor_part,
                     svt_ddr3_part_number_policy)::pick();

  //Use the "load_prop_vals()" method to load the configuration values
 //of the selected part into the configuration object
  if (cfg.load_prop_vals(ddr3_vendor_part.get_cfgfile())) begin
            …
  end
            …
endclass
```

```
class svt_ddr2_part_number_policy;

  static function int weight(svt_ddr2_vendor_part part);
    ...
      return part.get_part_number() == "jedec_ddr2_256M_x16_800C_4_0";
    ...
  endfunction: weight

endclass: svt_ddr2_part_number_policy


//----------------------------------------------------------------------
---
/** Base test class extended by every test in this example */
class ddr_base_test extends uvm_test;

   //Use a policy class to select the DDR2 Memory part number and print
 //the catalog features
   ddr2_vendor_part = svt_mem_part_mgr#(svt_ddr2_vendor_part,
   svt_ddr2_part_number_policy)::pick();

   //Use the "load_prop_vals()" method to load the configuration values
 //of the selected part into the configuration object
   if (cfg.load_prop_vals(ddr2_vendor_part.get_cfgfile())) begin
   …
   end

endclass: ddr_base_test
```

2.1.3 Creating a Test

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

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

```
`include "ddr_basic_env.sv" //suggested UVM environment file
`include "cust_svt_ddr_configuration.sv" //suggested test
                                         //configuration file
//build_phase
cfg = cust_svt_ddr_configuration::type_id::create("cfg");

uvm_config_db#(cust_svt_ddr_configuration)::set(this,
"ddr_env",
"cfg", this.cfg);

ddr_env = svt_basic_env::type_id::create("ddr_env", this);
```

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

DDR4-3DS:

```
$DESIGNWARE_HOME/vip/svt/ddr_svt/latest/examples/sverilog/tb_ddr4_3d
s_svt_uvm_basic_sys/tests
```

DDR4:

```
$DESIGNWARE_HOME/vip/svt/ddr_svt/latest/examples/sverilog/
tb_ddr4_svt_uvm_basic_sys/tests
```

DDR3:

```
$DESIGNWARE_HOME/vip/svt/ddr_svt/latest/examples/sverilog/
tb_ddr3_svt_uvm_basic_sys/tests
```

DDR2:

```
$DESIGNWARE_HOME/vip/svt/ddr_svt/latest/examples/sverilog/
tb_ddr2_svt_uvm_basic_sys/tests
```

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

Note
The DDR 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 DDR DIMM Memory VIP Testbench Integration Flow

The DDR DIMM Memory environment (svt_ddr_dimm_env) is the top-level component provided by the VIP for modeling DDR4 or DDR4 3DS or DDR3 DIMMs (UDIMM, SODIMM, RDIMM and LRDIMM). This generic system environment component encapsulates the following components:

- Instances of the DRAM agent with a single sequencer and memory core
- DIMM monitor
- RCD component

  DDR3: LRDIMM, RDIMM
- Command address buffer

  DDR4: LRDIMM, RDIMM
  DDR5 : RDIMM
- Data buffers

  DDR4: LRDIMM
- SPD agent

The DDR DIMM Memory environment implicitly constructs the required components for modeling a DDR3, DDR5 or DDR4 or DDR4 3DS DIMM (UDIMM, RDIMM and LRDIMM) as specified by the protocol kind of the DIMM configuration (svt_ddr_dimm_configuration) and the DIMM part catalog file. You can instantiate and construct the DDR DIMM Memory environment in the top-level environment of your UVM testbench.

Note
Currently DDR2 DIMMs and DDR4 NVDIMM DRAM are not supported in Synopsys DDR VIP.

Figure 2-2 Top-level Architecture of a DDR DIMM Memory VIP Testbench

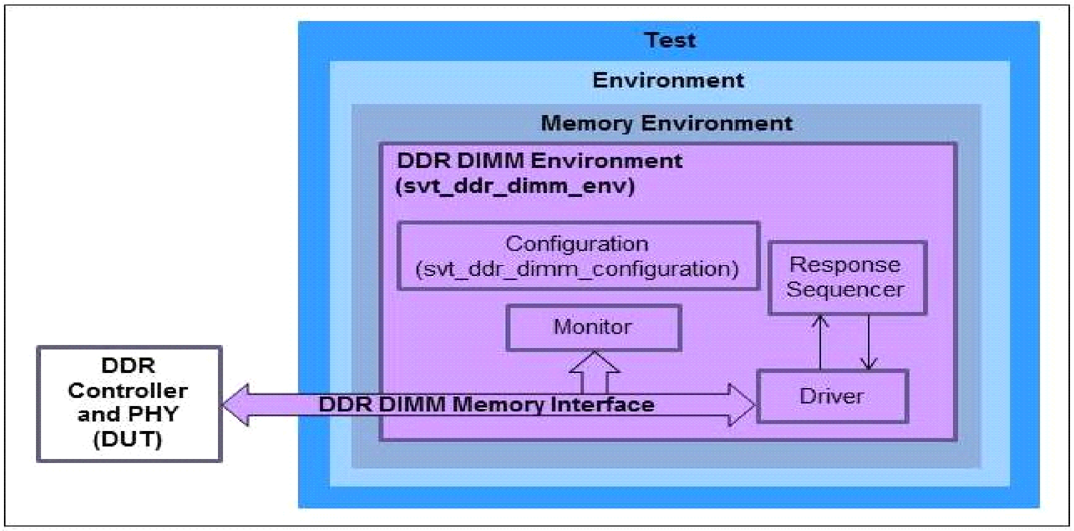

Test
Environment
Memory Environment
DDR DIMM Environment (svt_ddr_dimm_env)
Configuration (svt_ddr_dimm_configuration)
Response Sequencer
Monitor
Driver
DDR DIMM Memory Interface
DDR Controller and PHY (DUT)

Figure 2-2 is a top-level architecture of a simple DDR DIMM Memory VIP 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.

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_ddr_dimm.uvm.pkg"
`include "svt_ddr_dimm_if.svi"    //Top-level DIMM interfaces
`include "svt_ddr_controller.uvm.pkg"

import uvm_pkg::*;
import svt_uvm_pkg::*;
import svt_mem_uvm_pkg::*;   //UVM memory package
import svt_ddr_dimm_uvm_pkg::*; //UVM DIMM package
import svt_ddr_controller_uvm_pkg::*;

`include "svt_ddr4_catalog.svi" //For DDR4 catalog files
`include "svt_ddr3_catalog.svi" //For DDR3 catalog files
`include "svt_ddr5_catalog.svi"//For DDR5 catalog files
```

- Instantiate the required DDR DIMM Memory interface.

DDR4:

```
    svt_ddr4_dimm_if dimm_if();
```

DDR4 NVDIM:

```
    svt_ddr4_dimm_if dimm_if();
```

DDR5:

```
    svt_ddr5_dimm_if dimm_if();
```

DDR3:

```
    svt_ddr3_dimm_if dimm_if();
```

✦ Connect the top-level DDR DIMM Memory interface to the DUT and the DDR DIMM Memory environment based on the number of DIMM interfaces.

DDR4:

```
        dut dut_inst(dimm_if);

        uvm_config_db#(svt_ddr4_dimm_vif)::set(uvm_root::get(),
        "uvm_test_top.env.dimm_env", "dimm_if", dimm_if);
```

DDR4 NVDIMM:

```
        dut dut_inst(dimm_if);

        uvm_config_db#(svt_ddr4_nvdimm_vif)::set(uvm_root::get(),
        "uvm_test_top.env.dimm_env", "dimm_if", dimm_if);
```

DDR5:

```
        dut dut_inst(dimm_if);

        uvm_config_db#(svt_ddr5_dimm_vif)::set(uvm_root::get(),
        "uvm_test_top.env.dimm_env", "dimm_if", dimm_if);
```

DDR3:

```
        dut dut_inst(dimm_if);

        uvm_config_db#(svt_ddr3_dimm_vif)::set(uvm_root::get(),
        "uvm_test_top.env.dimm_env", "dimm_if", dimm_if);
```

The `uvm_config_db` command connects the top-level DDR DIMM Memory interface to the virtual interface of the DDR DIMM 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 "`dimm_env`" is an instance of the DDR DIMM Memory environment (`svt_ddr_dimm_env`).

2.2.2 Instantiating and Configuring the VIP

The following are the steps to instantiate and configure the DDR DIMM Memory environment in your testbench environment.

- ✦ Instantiate the DDR DIMM Memory environment (svt_ddr_dimm_env) and construct the environment in the build phase of your testbench environment.

```
svt_ddr_dimm_env dimm_env;

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

dimm_env = svt_ddr_dimm_env::type_id::create("dimm_env",
this);
```

- ✦ Create a test configuration class by extending the DDR DIMM Memory configuration class (`svt_ddr_dimm_configuration`). This configuration class specifies the required DDR5, DDR4 or DDR3 DIMM (UDIMM, RDIMM and LRDIMM).

For example:

```
class cust_svt_ddr_dimm_configuration extends
svt_ddr_dim_configuration;

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


this.set_protocol_kind(svt_ddr_dimm_configuration::DDR5_RDIMM
);

//OR

this.set_protocol_kind(svt_ddr_dimm_configuration::DDR5_UDIMM
);

//OR
this.set_protocol_kind(svt_ddr_dimm_configuration::DDR4
                              UDIMM);
//OR
this.set_protocol_kind(svt_ddr_dimm_configuration::DDR4
                              RDIMM);
//OR
this.set_protocol_kind(svt_ddr_dimm_configuration::DDR4
                              LRDIMM);
//OR
this.set_protocol_kind(svt_ddr_dimm_configuration::DDR3
                              UDIMM);
//OR
this.set_protocol_kind(svt_ddr_dimm_configuration::DDR3
                              RDIMM);
//OR
this.set_protocol_kind(svt_ddr_dimm_configuration::DDR3
                              LRDIMM);
   endfunction: new
endclass: cust_svt_ddr_dimm_configuration
```

For more information on the configuration class, refer to the *svt_ddr_dimm_configuration Class Reference* at the following location:

```
$DESIGNWARE_HOME/vip/svt/ddr_svt/latest/doc/
ddr_svt_uvm_class_reference/html/
class_svt_ddr_dimm_configuration.html
```

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

```
cfg =
cust_svt_ddr_dimm_configuration::type_id::create("cfg");

uvm_config_db#(svt_ddr_dimm_configuration)::set(this,
"dimm_env",
"cfg", cfg);
```

The “`cust_svt_ddr_dimm_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.

DDR5:

```
class svt_ddr5_part_number_policy_rdimm;
  static function int weight(svt_ddr5_vendor_part part);
    string ddr5_part_number;
    if ($value$plusargs("part_number=%s", ddr5_part_number)) begin
      return part.get_part_number() == ddr5_part_number;
      `uvm_info ("weight",$sformatf("User provided part_number=%s
selected", ddr5_part_number),UVM_HIGH);
    end
    else  begin
      return part.get_part_number() ==
"jedec_ddr5_rdimm_16G_x8_3200W_0_625_2R_noecc";
     `uvm_info ("weight",$sformatf("Default part_number=%s selected
", ddr5_part_number),UVM_LOW);
    end
  endfunction: weight
endclass: svt_ddr5_part_number_policy_rdimm
class ddr_base_test extends uvm_test;

    /** Use a policy class to select DIMM part number and print
various catalog features */
    ddr5_vendor_part =
svt_mem_part_mgr#(svt_ddr5_vendor_part,svt_ddr5_part_number_policy_
rdimm)::pick();



    if (cfg.load_prop_vals(ddr5_vendor_part.get_cfgfile())) begin
if (cfg.is_valid(0)) begin
          `uvm_info("build_phase", "Loaded
svt_ddr_dimm_configuration is VALID ...",UVM_LOW);
       end
       else begin
          `uvm_fatal("build_phase", "Loaded
svt_ddr_dimm_configuration is NOT VALID ...");
        end
end
    else begin
      `uvm_fatal("create_cfg", $sformatf("Unable to load the
configuration values for %s", ddr5_vendor_part.get_part_number()));
    end
endclass
```

DDR4:

```
class svt_ddr4_part_number_policy;
  static function int weight(svt_ddr4_vendor_part part);
          …
  string ddr4_part_number;

    if ($value$plusargs("part_number=%s", ddr4_part_number))
begin
         //Select a DDR4 part_number during runtime
         return part.get_part_number() == ddr4_part_number;

         `uvm_info("weight",$sformatf("User provided
part_number=%s
                  selected", ddr4_part_number, UVM_LOW);
    end
    else begin
         //Part number is
"jedec_ddr4_lrdimm_2G_x4_2400P_1R_noecc"
         return part.get_part_number() ==

"jedec_ddr4_lrdimm_2G_x4_2400P_1R_noecc";
         `uvm_info("weight",$sformatf("Default part_number=%s
                  selected", ddr4_part_number, UVM_LOW);
          …
end
  endfunction: weight
endclass: svt_ddr4_part_number_policy

class ddr_base_test extends uvm_test;
          …
  //Use a policy class to select the DDR DIMM Memory part number
 //and print the catalog features
  ddr4_vendor_part = svt_mem_part_mgr#(svt_ddr4_vendor_part,

svt_ddr4_part_number_policy)::pick();

 //Use the "load_prop_vals()" method to load the configuration values
 //of the selected part into the configuration object
  if (cfg.load_prop_vals(ddr4_vendor_part.get_cfgfile())) begin
          …
end
          …
endclass
```

DDR3:

```
class svt_ddr3_part_number_policy;
  static function int weight(svt_ddr3_vendor_part part);
             …
string ddr3_part_number;

    if ($value$plusargs("part_number=%s", ddr3_part_number))
begin
         //Select a DDR3 part_number during runtime
         return part.get_part_number() == ddr3_part_number;

         `uvm_info("weight",$sformatf("User provided
part_number=%s
                  selected", ddr3_part_number, UVM_LOW);
    end
    else begin
         //Part number is "jedec_ddr3_lrdimm_1G_x8_1600G_1_250_1R_noecc"
         return part.get_part_number() ==

"jedec_ddr3_lrdimm_1G_x8_1600G_1_250_1R_noecc";
         `uvm_info("weight",$sformatf("Default part_number=%s
                  selected", ddr3_part_number, UVM_LOW);
            …
end
  endfunction: weight
endclass: svt_ddr3_part_number_policy

class ddr_base_test extends uvm_test;
            …
 //Use a policy class to select the DDR DIMM Memory part number
 //and print the catalog features
  ddr3_vendor_part = svt_mem_part_mgr#(svt_ddr3_vendor_part,

svt_ddr3_part_number_policy)::pick();

  //Use the "load_prop_vals()" method to load the configuration values
 //of the selected part into the configuration object
  if (cfg.load_prop_vals(ddr3_vendor_part.get_cfgfile())) begin
            …
end
            …
endclass
```

Note
NVDIMM protocol doesn't support part catalogs currently.

2.2.3 Creating a Test

You can create a base test class (ddr_base_test) to specify the default test behaviors and to use this class as the base class for other DDR DIMM Memory tests. The following code snippets can be used in the base test class file (ddr_base_test.sv).

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

```
`include "ddr_basic_env.sv" //suggested UVM environment file
`include "cust_svt_ddr_dimm_configuration.sv" //suggested test
//configuration file

//build_phase
cfg =
cust_svt_ddr_dimm_configuration::type_id::create("cfg");

uvm_config_db#(cust_svt_ddr_dimm_configuration)::set(this,
"dimm_env", "cfg", this.cfg);

dimm_env = svt_ddr_dimm_env::type_id::create("dimm_env",
this);
```

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

DDR4:

```
$DESIGNWARE_HOME/vip/svt/ddr_svt/latest/examples/sverilog/
tb_ddr4_dimm_svt_uvm_basic_sys/tests
```

DDR4 NVDIMM:

```
$DESIGNWARE_HOME/vip/svt/ddr_svt/latest/examples/tb_nvdimmp_dimm_svt
_uvm_basic_sys/tests
```

DDR5:

```
$DESIGNWARE_HOME/vip/svt/ddr_svt/latest/examples/tb_ddr5_dimm_svt_uv
m_basic_sys/tests
```

DDR3:

```
$DESIGNWARE_HOME/vip/svt/ddr_svt/latest/examples/sverilog/
tb_ddr3_dimm_svt_uvm_basic_sys/tests
```

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

Note
The DDR 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:

DDR5:

```
class svt_ddr5_part_number_policy_rdimm;
  static function int weight(svt_ddr5_vendor_part part);
    string ddr5_part_number;
    if ($value$plusargs("part_number=%s", ddr5_part_number)) begin
      return part.get_part_number() == ddr5_part_number;
      `uvm_info ("weight",$sformatf("User provided part_number=%s 
selected", ddr5_part_number),UVM_HIGH);
    end
    else  begin
      return part.get_part_number() == 
"jedec_ddr5_rdimm_16G_x8_3200W_0_625_2R_noecc";
     `uvm_info ("weight",$sformatf("Default part_number=%s selected ", 
ddr5_part_number),UVM_LOW);
    end
  endfunction: weight
endclass: svt_ddr5_part_number_policy_rdimm
```

DDR4:

```
class svt_ddr4_part_number_policy;
  static function int weight(svt_ddr4_vendor_part part);
    string ddr4_part_number;

    if ($value$plusargs("part_number=%s", ddr4_part_number)) 
begin
         return part.get_part_number() == ddr4_part_number;
    end
    else begin
             …
end
  endfunction: weight
endclass: svt_ddr4_part_number_policy
```

DDR3:

```
class svt_ddr3_part_number_policy;
  static function int weight(svt_ddr3_vendor_part part);
    string ddr3_part_number;

    if ($value$plusargs("part_number=%s", ddr3_part_number)) 
begin
         return part.get_part_number() == ddr3_part_number;
    end
    else begin
             …
end
  endfunction: weight
endclass: svt_ddr3_part_number_policy
```

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 DDR Memory:

```
+define+SVT_UVM_TECHNOLOGY
+define+UVM_PACKER_MAX_BYTES=1500000
+define+SYNOPSYS_SV
project_directory_path/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
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=same_bank_rd_wr_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
```

A.2 Primary Documentation for VC VIP DDR Memory

VC Verification IP UVM Installation and Setup Guide:

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

VC VIP DDR Memory UVM User Guide:

`$DESIGNWARE_HOME/vip/svt/ddr_svt/latest/doc/ddr_svt_uvm_user_guide.pdf`

VC VIP DDR Memory Getting Started Guide:

`$DESIGNWARE_HOME/vip/svt/ddr_svt/latest/doc/ddr_svt_uvm_getting_started.pdf`

VC VIP DDR Memory QuickStarts:

`$DESIGNWARE_HOME/vip/svt/ddr_svt/latest/examples/sverilog/`
`tb_ddr3_svt_uvm_basic_sys/doc/tb_ddr3_svt_uvm_basic_sys/index_basic.html`

```
$DESIGNWARE_HOME/vip/svt/ddr_svt/latest/examples/sverilog/
tb_ddr4_svt_uvm_basic_sys/doc/tb_ddr4_svt_uvm_basic_sys/index_basic.html
```

VC VIP DDR Memory Class Reference:

```
$DESIGNWARE_HOME/vip/svt/ddr_svt/latest/doc/ddr_svt_uvm_class_reference/html/
index.html
```

VC VIP DDR Memory Verification Plans:

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

A.3 Example Home Directory

Directory that contains a list of VIP example directories:

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

View simulation options for each example:

```
gmake help
```