Verification Continuum™

VC Verification IP
SPI
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 https://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 SPI FLASH (referred to as VIP) into testbenches that are compliant with the SystemVerilog Universal Verification Methodology (UVM). You are assumed to be familiar with the SPI FLASH protocol and UVM.

Web Resources

- Documentation through SolvNetPlus: 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:

- Go to https://solvnetplus.synopsys.com and open a case.
  - Enter the information according to your environment and your issue.
  - For simulation issues, provide a UVM_FULL verbosity log file of the VIP instance and a VPD or FSDB dump file of the VIP interface.
- 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.
- 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

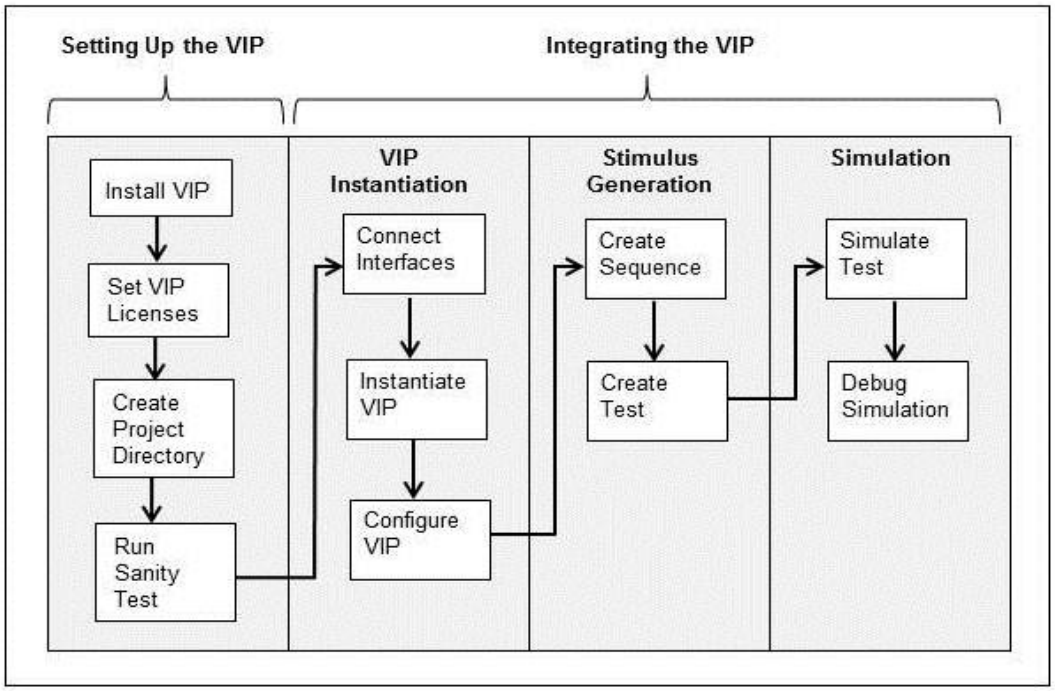
1 Overview of the Getting Started Guide

This Getting Started Guide presents information about integrating the VC VIP for SPI FLASH (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 Center (click here -> VC VIP Library -> Q-2020.03 -> Installation Guide) 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 SPI FLASH protocol and UVM. For more information on the VIP, refer to the *VC Verification IP SPI FLASH UVM User Guide* on SolvNet (click here) or in the VIP installation at the following location:

$DESIGNWARE_HOME/vip/svt/spi_svt/latest/doc/spi_svt_uvm_user_guide.pdf

2 Integrating VIP (Generic SPI) into User Testbench

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

$DESIGNWARE_HOME/vip/svt/spi_svt/latest/doc/spi_svt_uvm_class_reference/html/index.html

2.1 SPI VIP Testbench Integration Flow

The SPI agent (svt_spi_agent) is the top-level component provided by the VIP for modeling FLASH memory devices. This generic agent encapsulates the following components:

- Sequencer
- Monitor
- Driver

The SPI slave agent can be configured as a generic device by setting the frame_format to SPI_STD in the SPI configuration class (svt_spi_configuration) and loading the device part catalog file. You can instantiate and construct the SPI slave agent in the top-level environment of your UVM testbench.

Figure 2-1 Top-level Architecture of a SPI VIP Testbench (DUT as MASTER, VIP as SLAVE)

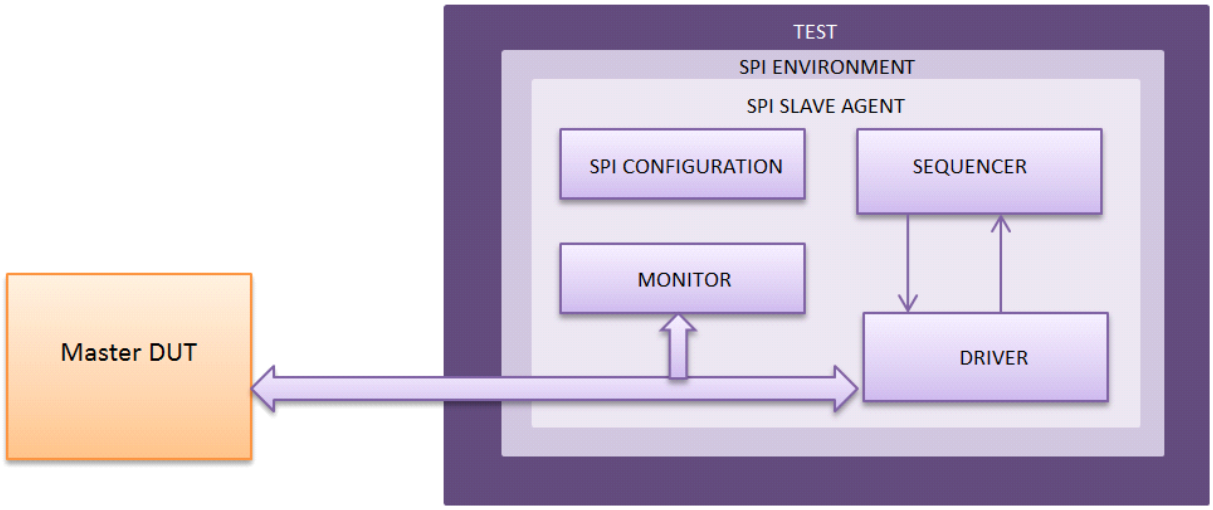


Figure 2-2 Top-level Architecture of a SPI VIP Testbench (DUT as SLAVE, VIP as MASTER)

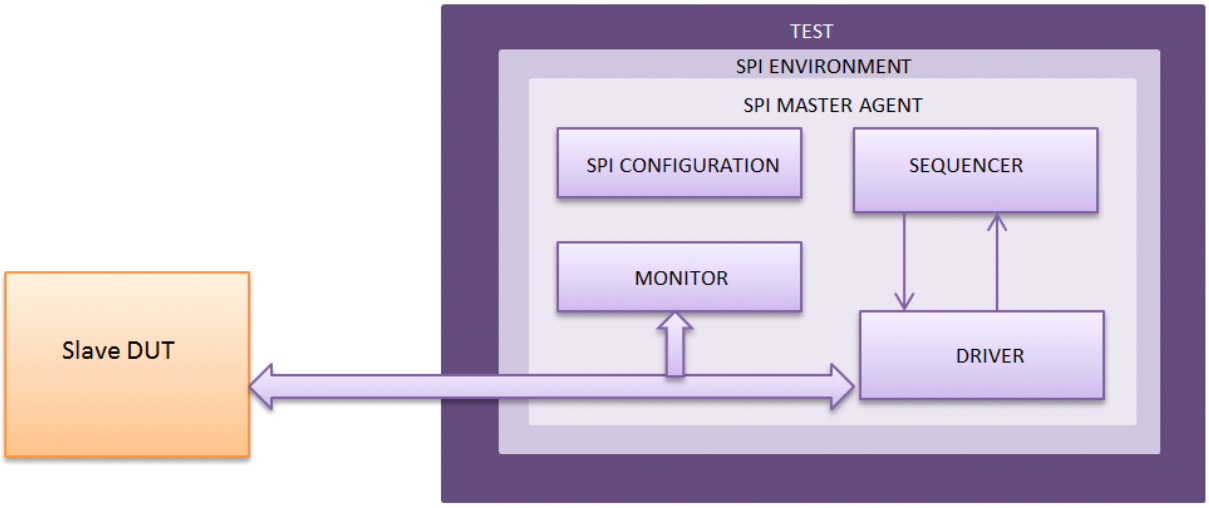


Figure 2-1 and Figure 2-2 shows top-level architecture of a simple VC VIP for SPI testbench. The steps for integrating the VIP into a UVM testbench are described in the following sections:

- 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 example:

```
$DESIGNWARE_HOME/vip/svt/spi_svt/latest/examples/sverilog/tb_spi_svt_uvm_basic_sys_1m_1s_sys
$DESIGNWARE_HOME/vip/svt/spi_svt/latest/examples/sverilog/tb_spi_svt_uvm_basic_multi_agent_sys
$DESIGNWARE_HOME/vip/svt/spi_svt/latest/examples/sverilog/tb_spi_svt_uvm_multilane_intermediate_sys
```

You can use these examples as guidance for creating UVM testbench environments with the VIP. Implementations can be customized for your requirements.

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 "svt_spi.uvm.pkg" //Top-level VIP package

import uvm_pkg::*;
`include "uvm_macro.svh"
import svt_uvm_pkg::*;
import svt_spi_uvm_pkg::*; //UVM SPI package
```

- Instantiate the top-level SPI interface.

```
svt_spi_if spi_slave_if (systemclock, reset);

  tran sclk (DUT.sclk, spi_slave_if.sclk);
  tran ssn_n(DUT.ss_n, spi_slave_if.ss_n);

Connect DUT's pins with SPI interface pins.

genvar i;
generate
for (i=0; i < `SVT_SPI_IO_WIDTH; i=i+1) begin :IO_CONNECT
tran mosi(DUT.mosi[i], spi_slave_if.mosi[i]);
tran miso(DUT.miso[i], spi_slave_if.miso[i]);
end
endgenerate
```

- Connect the top-level SPI interface to the DUT and the SPI system environment.

```
uvm_config_db#(virtual svt_spi_if)::set(uvm_root::get(),
     "uvm_test_top.env", "slave_if", spi_slave_if);
```

The `uvm_config_db` command connects the top-level SPI interface to the virtual interface of the SPI VIP environment. The "`uvm_test_top`" represents the top-level module in the UVM environment. The "`env`" is an instance of SPI environment that encapsulates the SPI agent (svt_spi_agent).

2.1.2 Instantiating and Configuring the VIP

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

- Instantiate the SPI agent (svt_spi_agent) and construct the agent in the build phase of your testbench environment.

```
svt_spi_agent slave_agent;

slave_agent = svt_spi_agent::type_id::create("slave_agent",this);
```

- Create a test configuration class by extending the SPI agent configuration class (svt_spi_agent_configuration). This configuration class has been provided for users to customize the configuration setting as per their requirements.

  For example,

```
class cust_svt_spi_agent_configuration extends
svt_spi_agent_configuration;

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

      endfunction: new
endclass: cust_svt_spi_agent_configuration
```

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

  $DESIGNWARE_HOME/vip/svt/spi_svt/latest/doc/spi_svt_uvm_class_reference/html/class_svt_spi_agent_configuration.html

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

```
slave_cfg =
cust_svt_spi_agent_configuration::type_id::create("slave_cfg");
```

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

2.1.3 Creating a Test

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

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

```
`include "spi_basic_env.sv" //Suggested UVM environment file
`include "cust_svt_spi_agent_configuration.sv" //Suggested test

//configuration file
//Build phase
slave_cfg = cust_svt_spi_agent_configuration::type_id::create("slave_cfg");
env = spi_basic_env::type_id::create("env", this);
```

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

```
$DESIGNWARE_HOME/vip/svt/spi_svt/latest/examples/sverilog/tb_spi_svt_uvm_basic_1m_1s_sys
/tests

$DESIGNWARE_HOME/vip/svt/spi_svt/latest/examples/sverilog/tb_spi_svt_uvm_basic_multi_age
nt_sys/tests

$DESIGNWARE_HOME/vip/svt/spi_svt/latest/examples/sverilog/tb_spi_svt_uvm_multilane_inter
mediate_sys/tests
```

3 Integrating SPI FLASH VIP into User Testbench

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

$DESIGNWARE_HOME/vip/svt/spi_svt/latest/doc/spi_svt_uvm_class_reference/html/index.html

3.1 SPI FLASH Device VIP Testbench Integration Flow

The SPI FLASH agent (svt_spi_agent) is the top-level component provided by the VIP for modeling FLASH memory devices. This generic agent encapsulates the following components:

- Sequencer and memory core
- Monitor
- Driver

The SPI slave agent can be configured as a FLASH memory device by setting the frame_format in the SPI configuration class (svt_spi_configuration) and loading the device part catalog file. You can instantiate and construct the SPI slave agent in the top-level environment of your UVM testbench.

Figure 3-1 Top-level Architecture of a SPI FLASH VIP Testbench

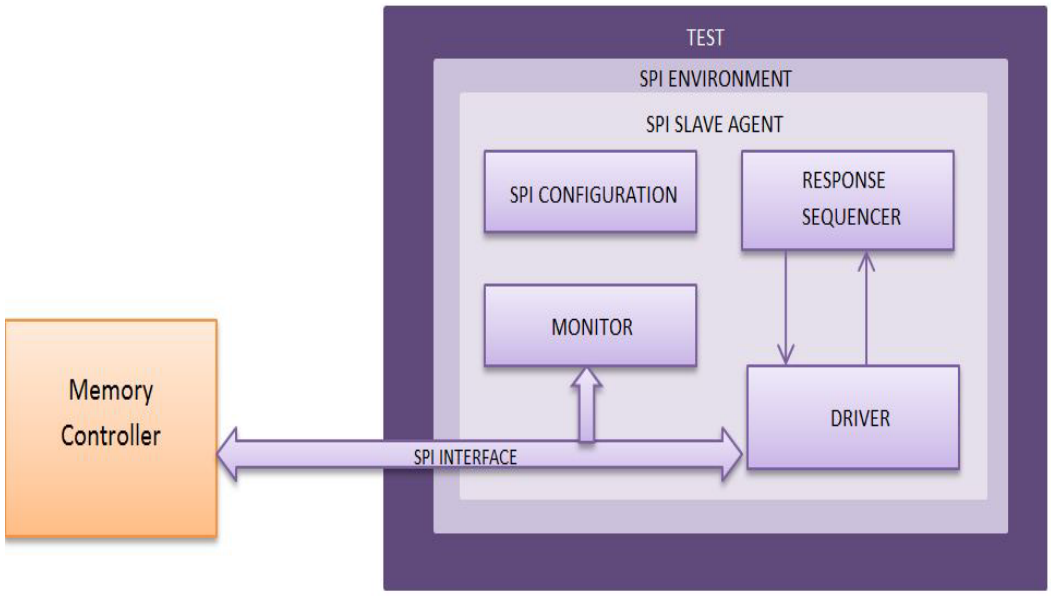


Figure 3-1 is a top-level architecture of a simple VC VIP for SPI FLASH testbench. The steps for integrating the VIP into a UVM testbench are described in the following sections:

- 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 example:

```
$DESIGNWARE_HOME/vip/svt/spi_svt/latest/examples/sverilog/tb_spiflash_svt_uvm_micron_basic_sys

$DESIGNWARE_HOME/vip/svt/spi_svt/latest/examples/sverilog/tb_spiflash_svt_uvm_winbond_basic_sys
```

You can use these examples as guidance for creating UVM testbench environments with the VIP. Implementations can be customized for your requirements.

3.1.1 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 "svt_spi.uvm.pkg" //Top-level VIP package

import uvm_pkg::*;
`include "uvm_macro.svh"
import svt_uvm_pkg::*;
import svt_mem_uvm_pkg::*; //UVM memory package
import svt_spi_uvm_pkg::*; //UVM SPI package
```

- Instantiate the top-level SPI FLASH interface.

```
svt_spi_if spi_slave_if (systemclock, reset);

  tran sclk (DUT.sclk, spi_slave_if.sclk);
  tran ssn_n(DUT.ss_n, spi_slave_if.ss_n);

assign slave_vcc = DUT_vcc; // if vcc port is not present then drive it to 1

if communication is through the flash pins then -
     tran dq0 (DUT.dq0,spi_slave_if.dq0);
     tran dq1 (DUT.dq1,spi_slave_if.dq1);
     tran dq2 (DUT.dq2,spi_slave_if.dq2);
     tran dq3 (DUT.dq3,spi_slave_if.dq3);

if communication is happening through SPI pins then -

 genvar i;
 generate
    for (i=0; i < `SVT_SPI_IO_WIDTH; i=i+1) begin :IO_CONNECT
      tran mosi(DUT.mosi[i], spi_slave_if.mosi[i]);
      tran miso(DUT.miso[i], spi_slave_if.miso[i]);
    end
 endgenerate
```

- Connect the top-level SPI FLASH interface to the DUT and the SPI FLASH system environment.

```
uvm_config_db#(virtual svt_spi_if)::set(uvm_root::get(),
     "uvm_test_top.env", "slave_if", spi_slave_if);
```

The `uvm_config_db` command connects the top-level SPI FLASH interface to the virtual interface of the flash VIP environment. The "`uvm_test_top`" represents the top-level module in the UVM environment. The "`env`" is an instance of SPI FLASH environment that encapsulates the SPI FLASH agent (svt_spi_agent).

3.1.2 Instantiating and Configuring the VIP

The following are the steps to instantiate and configure the SPI FLASH agent in your testbench environment.

- Instantiate the SPI FLASH agent (svt_spi_agent) and construct the agent in the build phase of your testbench environment.

```
svt_spi_agent slave_agent;

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

slave_agent = svt_spi_agent::type_id::create("slave_agent",this);
```

- Create a test configuration class by extending the SPI agent configuration class (svt_spi_agent_configuration). This configuration class has been provided for users to customize the configuration setting as per their requirements.

  For example,

```
class cust_svt_spi_agent_configuration extends svt_spi_agent_configuration;

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

      endfunction: new
endclass: cust_svt_spi_agent_configuration
```

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

  $DESIGNWARE_HOME/vip/svt/spi_svt/latest/doc/spi_svt_uvm_class_reference/html/class_svt_spi_agent_configuration.html

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

```
slave_cfg = cust_svt_spi_agent_configuration::type_id::create("slave_cfg");
slave_cfg.spi_mem_cfg = svt_spi_mem_configuration::type_id::create("spi_mem_cfg",
this);
```

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

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

SPI FLASH:

```
class svt_spi_part_number_policy;
  static function int weight(svt_spi_vendor_part part);
   ..................
    string spi_part_number;
    if ($value$plusargs("part_number=%s", spi_part_number)) begin
     ..............
      return part.get_part_number() == spi_part_number;
    end
    else begin
      return part.get_part_number() == "N25Q_1Gb_3V_65nm";
      ..............
    end
  endfunction : weight
endclass : svt_spi_part_number_policy

class spi_base_test extends uvm_test;
      ...
//Use a policy class to select the SPI FLASH Memory part number and print the catalog features
spi_vendor_part =
svt_mem_part_mgr#(svt_spi_vendor_part,svt_spi_part_number_policy)::pick();

//Use the "load_prop_vals()" method to load the configuration valuesof the selected part into the configuration
object
if (agent_cfg.spi_mem_cfg.load_prop_vals(spi_vendor_part.get_cfgfile()) begin
      ...
  end
      ...
endclass
```

3.1.3 Creating a Test

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

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

```
`include "spi_basic_env.sv" //Suggested UVM environment file
`include "cust_svt_spi_agent_configuration.sv" //Suggested test

//configuration file
//Build phase
slave_cfg = cust_svt_spi_agent_configuration::type_id::create("slave_cfg");
env = spi_basic_env::type_id::create("env", this);
```

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

```
$DESIGNWARE_HOME/vip/svt/spi_svt/latest/examples/sverilog/
tb_spiflash_svt_uvm_micron_basic_sys/tests

$DESIGNWARE_HOME/vip/svt/spi_svt/latest/examples/sverilog/tb_spiflash_svt_uvm_winbond_ba
sic_sys/tests
```

In the VIP examples, a dummy SPI FLASH controller agent component is used to model the controller. In a user testbench, actual SPI FLASH RTL controller will be used.

Note The SPI FLASH 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.

3.2 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:

```
class svt_spi_part_number_policy;
  static function int weight(svt_spi_vendor_part part);
    string spi_part_number;
    if ($value$plusargs("part_number=%s", spi_part_number)) begin
      ..................
      return part.get_part_number() == spi_part_number;
    end
else begin
      return part.get_part_number() == "N25Q_1Gb_3V_65nm";
      ..................
    end
  endfunction : weight
endclass : svt_spi_part_number_policy
```

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

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
```

4.2 VIP Compile-Time Options

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

```
+define+SVT_UVM_TECHNOLOGY
+define+UVM_PACKER_MAX_BYTES=16384
+define+SYNOPSYS_SV
+define+SVT_SPI_DATA_WIDTH=8
+define+SVT_SPI_IO_WIDTH=4   // for Generic SPI mode it should be 1 for Quad SPI Flash mode it should be 4
+define+SVT_MEM_ENABLE_DEPTH_INDEPENDENT_VALUES // applicable only in Quad SPI Flash mode
+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.

Macro	Description
SVT_UVM_TECHNOLOGY	Specifies SystemVerilog based VIPs that are compliant with UVM
UVM_PACKER_MAX_BYTES	Sets to 16384 or greater
SYNOPSYS_SV	Specifies SystemVerilog based VIPs that are compliant with UVM

- SVT_SPI_DATA_WIDTH=8 -> specifies the packet size, it should always be 8
- SVT_SPI_IO_WIDTH=4 -> defines the I/O width. For single data lane it should be 1, for dual mode 2 and for Quad lane 4.

  For FLASH mode it should be 4.
- SVT_MEM_ENABLE_DEPTH_INDEPENDENT_VALUES -> Applicable only for FLASH mode

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 Generic SPI example:

```
+UVM_TESTNAME=write_enable_test
```

For QSPI FLASH example:

```
+UVM_TESTNAME=write_enable_test +part_number=N25Q_1Gb_3V_65nm
```

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 SPI FLASH

VC Verification IP UVM Installation and Setup Guide:

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

VC VIP SPI FLASH UVM User Guide:

$DESIGNWARE_HOME/vip/svt/spi_svt/latest/doc/spi_svt_uvm_user_guide.pdf

VC VIP SPI FLASH Getting Started Guide:

$DESIGNWARE_HOME/vip/svt/spi_svt/latest/doc/spi_svt_uvm_getting_started.pdf

VC VIP SPI FLASH Class Reference:

$DESIGNWARE_HOME/vip/svt/spi_svt/latest/doc/spi_svt_uvm_class_reference/html/index.html

VC VIP SPI FLASH Verification Plans:

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

A.3 Example Home Directory

Directory that contains a list of VIP example directories:

```
$DESIGNWARE_HOME/vip/svt/spi_svt/latest/examples/sverilog/tb_spi_svt_uvm_basic_1m_1s_sys
$DESIGNWARE_HOME/vip/svt/spi_svt/latest/examples/sverilog/tb_spi_svt_uvm_basic_multi_agent_sys
$DESIGNWARE_HOME/vip/svt/spi_svt/latest/examples/sverilog/tb_spi_svt_uvm_multilane_intermediate_sys
$DESIGNWARE_HOME/vip/svt/spi_svt/latest/examples/sverilog/tb_spiflash_svt_uvm_micron_basic_sys

$DESIGNWARE_HOME/vip/svt/spi_svt/latest/examples/sverilog/tb_spiflash_svt_uvm_winbond_basic_sys
```

View simulation options for each example:

```
gmake help
```