UVM SV Reference Flow User Guide

Version 1.1
April 2012

Printed in the United States of America.

Cadence Design Systems, Inc., 2655 Seely Avenue, San Jose, CA 95134, USA

Table of Contents

Overview

The UVM Reference Flow applies the Universal Verification Methodology (UVM) to the Block and Cluster Verification in a SoC Design. It begins by showing aspects of the verification of a block, a Universal Asynchronous Receiver Transmitter (UART).

It then shows how to verify a cluster design (an APB subsystem) into which the UART gets integrated along with other design components (viz. SPI, GPIO, Power Controller, Timers etc).

What's New

This release of the UVM Reference Flow is completely aligned with the Universal Verification Methodology UVM 1.1 as released by Accellera.

The UVM Reference Flow design is based on an Ethernet Switch System-on-Chip (SoC). The SoC has the following key design components:

- An Opencores Open RISC Processor
- Opencores Ethernet Media Access controller (MAC)
- AMBA AHB network interconnect
- Address look-up table (ALUT)
- Support and control functions. For instance, power management and peripherals like UART, SPI, GPO, timer, etc.
- On-chip memories and memory controller

The UVM Reference Flow also includes the following key verification components:

- AMBA AHB UVC
- AMBA APB UVC
- UART UVC
- GPIO UVC
- SPI UVC

This *UVM SV Reference Flow User Guide* can be found online here:

```
<UVM Reference Flow Installation
   Area>/doc/uvm_flow_topics/uvm_sv/uvm_sv_ref_flow_ug.pdf
```

Release Version: 1.1

The UVM Reference Flow is tested with the UVM 1.1 library (from Accellera) and with Incisive Enterprise Simulator (IES) 11.1. It should be possible to run the UVM Reference Flow on any IEEE 1800 Compliant Simulator that supports UVM.

For more information about using the UVM Reference Flow, please contact uvm_ref@cadence.com.

Setup and Installation Instructions

Licenses, Terms and Conditions

Refer to the `README_terms_and_conditions.txt` file located in the UVM Reference Flow installation directory.

Dependencies

- UVM 1.1 Library
  The UVM library is included in the IUS installation. There is no requirement to download it.
- Incisive Unified Simulator (IUS 11.1)

Note: All the flows in this release have been validated on UVM 1.1 library and IUS 11.1.

Setup Instructions

1. Set up the UVM Reference Flow using one of the following methods:
   - In csh

     ```
     % setenv UVM_REF_HOME <Reference Flow Installation Area>
     % setenv UVM_HOME <UVM Library Install Area>
     % source ${UVM_REF_HOME}/env.csh
     ```

   - In bash

     ```
     % UVM_REF_HOME=<Reference Flow Installation Area>
     % export UVM_REF_HOME
     % UVM_HOME=<UVM Library Install Area>
     % export UVM_HOME
     % ${UVM_REF_HOME}/env.sh
     ```

2. Ensure you have a simulation tool installed and properly set up.

Running a Simulation Using Incisive Unified Simulator

When the installation and setup of the UVM Reference Flow is complete and the Incisive Unified Simulator (IUS) or Incisive Enterprise Simulator (IES) are available, try a quick simulation to ensure everything is set up:

Module Level Simulation

```
% $UVM_REF_HOME/soc_verification_lib/sv_cb_ex_lib/uart_ctrl/demo.sh
```

Cluster Level Simulation

```
% $UVM_REF_HOME/soc_verification_lib/sv_cb_ex_lib/apb_subsystem/demo.sh
```

Further Information

Refer to the `README.txt` file in the Reference Flow installation area for information about:

- Available documentation for the UVM Reference Flow and other related items:
- Ethernet Switch SoC Design
- Opencores IP
- The Reference Design hierarchy and directory structure
- The Reference Flow verification environment and directory structure

UVM SystemVerilog Module-Level Verification

This document describes details of UVM standalone environments contained in the UVM Reference Flow for verifying the functionality of both a single module and simple subsystem. The Reference Flow starts with a basic module, the UART, and shows how an environment can be built around the UART module by using the Universal Verification Methodology (UVM).

This chapter illustrates an implementation based on UVM SystemVerilog.

UART DUV

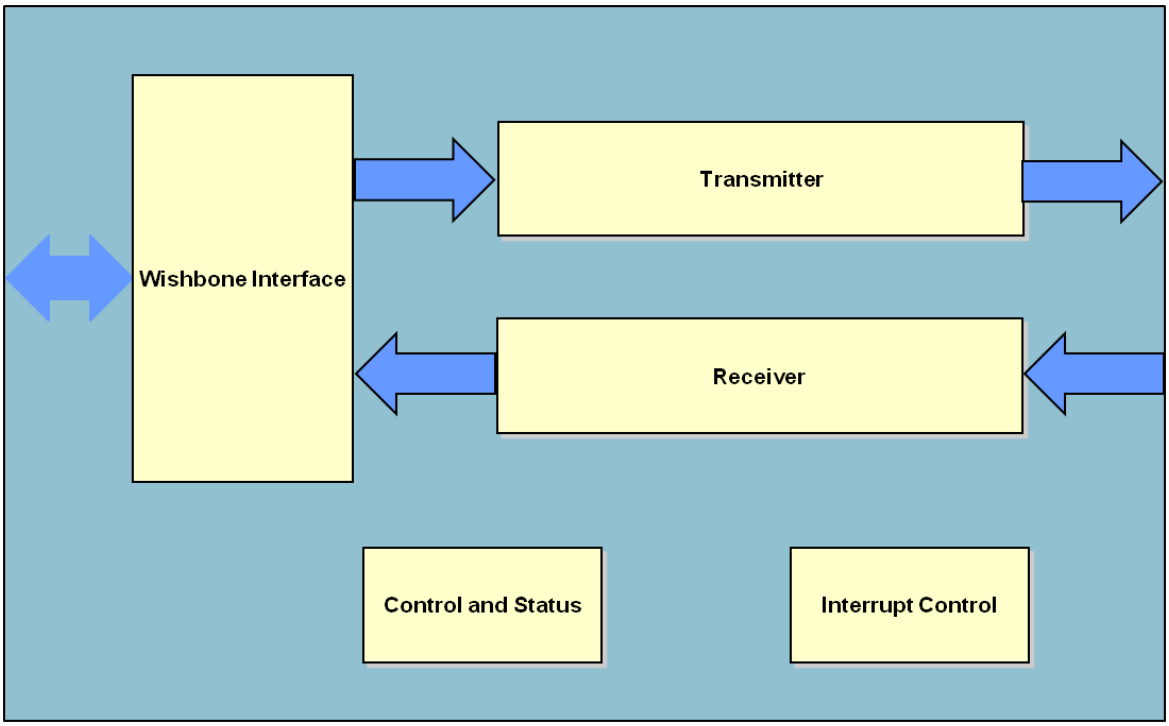
The UART module is a pre-verified soft IP from Opencores and is written in synthesizable Verilog RTL.

The DUV has the following interface which needs to be driven by the UVC:

- WISHBONE interface
- UART receiver interface
- UART transmitter interface
- UART interrupt interface
- Clocks and resets

The DUV is shown in the UART DUT figure below.

Figure 1. UART DUT



The following registers are available within UART DUV (WISHBONE Interface of the UART0):

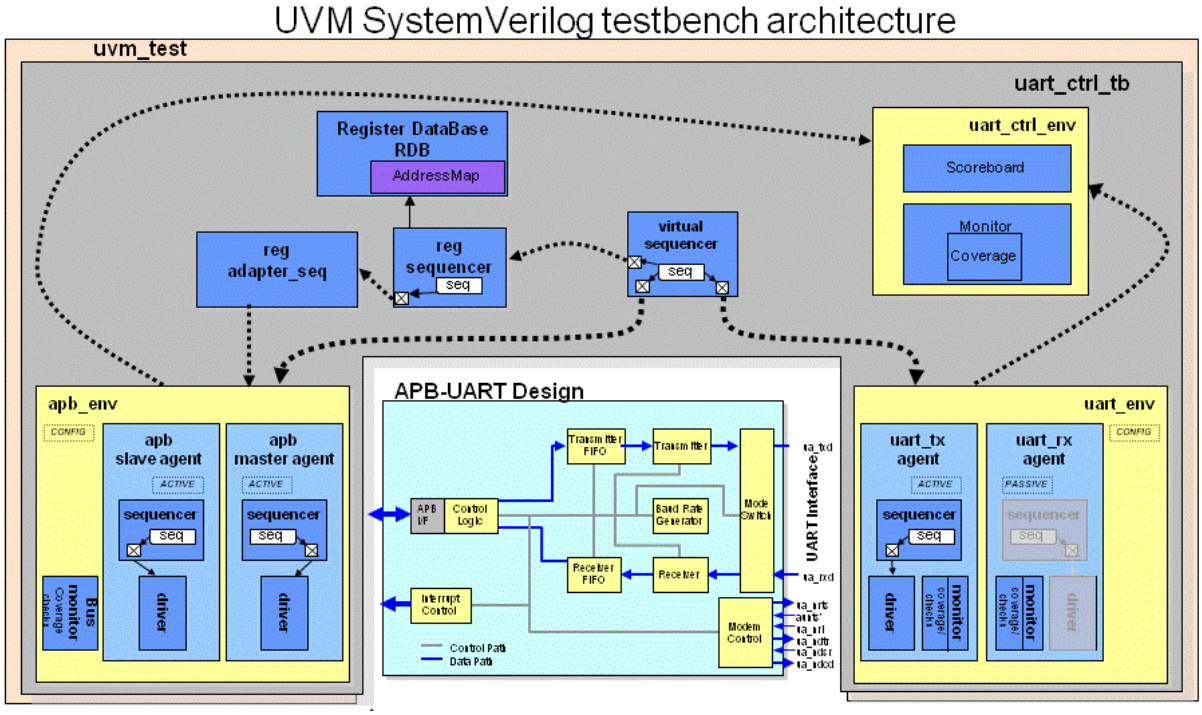
Table 1. WISHBONE Interface of the UART0

Offset	Function	Name	R/W	Reset Value
0x00	Receiver Buffer	UA_REC	R	0x00
0x00	Transmit Holding Register (THR)	UA_TRA	W	0x00
0x01	Interrupt Enable	UA_IER	R/W	0x00
0x02	Interrupt Identification	UA_IDR	R	0xC1
0x02	FIFO Control	UA_FCR	WO	0xC0
0x3	Line Control Register	UA_LCR	RW	0x03
0x4	Modem Control	UA_MCR	WO	0x00
0x6	Modem Status	UA_MSTS	RO	-
0x0	Divisor Latch Byte 1	UA_LDIV0	R/W	-
0x1	Divisor Latch Byte 2	UA_LDIV1	R/W	-

UART Module Verification – UVM Reference Flow Methodology

The UART module environment is implemented in compliance with the Universal Verification Methodology, and is constructed as shown in the UART Verification Environment figure below.

Figure 2. UART Verification Environment



Universal Verification Components (UVCs)

The UART module environment uses two UVCs: APB UVC (directly driving the WISHBONE interface) and a UART UVC constructed for the UVM Reference Flow according to the UVM SystemVerilog. These UVCs are used to provide stimulus and monitoring functionality for the DUV interfaces. The UVCs are provided in the following locations:

APB: $UVM_REF_HOME/soc_verification_lib/sv_cb_ex_lib/interface_uvc_lib/apb/sv

UART: $UVM_REF_HOME/soc_verification_lib/sv_cb_ex_lib/interface_uvc_lib/uart/sv

Test Sequence Library

The Test Sequence Library provides some common sequences for simplifying the writing of tests. It is defined to provide a simple sequence-based call for the test writer and thus provide a higher-level interface to the UART environment.

The Test Sequence Library provides the sequences shown in the Test Sequence Library table:

Table 1. Test Sequence Library

Sequence Name	Description
`concurrent_u2a_a2u_rand_trans`	The concurrent_u2a_a2u_rand_trans sequence first configures the UART DUV. Once the configuration sequence is completed, random read/write sequences from both the UVCs are enabled in parallel. At the end an Rx FIFO read sequence is executed.
`u2a_incr_payload`	The u2a_incr_payload sequence first configures the UART DUV. After configuration is done UART UVC starts sending incremented data to UART DUT. APB UVC starts interrupt sequence which reads RX FIFO.
`uart_1_stopbit_rx_trafic`	The uart_1_stopbit_rx_trafic sequence first configures the UART DUV with num of stop bits as one. APB UVC sequence writes data to the TX FIFO.

Scoreboards

A scoreboard is used to verify end-to-end data transformation through the DUV. The scoreboard collects data from the UVC monitors at the interfaces of the DUV and compares the results against the expected transformation. This transformation can be very simple or can be a complex model, depending on the type of DUV.

The UART environment implements two scoreboards:

- Checks whether data received by the DUV is successfully read out of the DUV RX FIFO.
- Checks whether data written into the DUV TX FIFO is successfully transmitted.

The transformation for these scoreboards is very simple in this example because the data input is the same as the data output.

The two scoreboards use standard synchronization events defined in the UVM-compliant UVC monitors to provide the data into and out of the DUV.

Coverage Module

The coverage module contains functional coverage DUV and top-level, environment-specific functional coverage collection. This coverage is collected in addition to protocol coverage provided by the UVCs.

The functional coverage is collected by specifying SV cover groups for the coverage points listed below in Coverage Points. These coverage points are implemented based on the requirements of the UART environment vPlan, which was created during the project verification planning phase.

Table 2. Coverage Points

Cover group	Cover Point	Description
dut_tx_fifo_cg	EMPTY, HALF_FULL, FULL	Coverage of the DUV TX FIFO fill level.
dut_rx_fifo_cg	EMPTY, HALF_FULL, FULL	Coverage of the DUV RX FIFO fill level.
uart_trans_frame	NUM_STOP_BITS : ONE, ONE_AND_HALF , TWO	Number of stop bits in UART protocol.
	DATA_LENGTH : EIGHT, SEVEN, SIX	Data Length in UART Protocol
	PARITY : ODD, EVEN, SPACE, MARK	Parity type
	ERROR BITS	Error bit types
	CROSS : DATA LENGTH X PARITY	Cross coverage
tx_trafic_cg	FRAME_DATA : ZERO, SMALLER, LARGER,MAX	Value of frame data
	FRAME_MSB_LSB : 00,01,10,11	Start and end bit in a frame
tx_protocol_cg	PARITY_ERROR : NORMAL, ERROR	Frame with parity error
	FRAME_BREAK : NORMAL, BREAK	Frame with break

Using the UART Module Verification Flow

Refer to the Setup and Installation Instructions if you have not already installed and set up the tools you need for this flow.

The rest of this section discusses:

- Package Directory Structure and Contents
- Run Scripts
- Test Cases

Package Directory Structure and Contents

Package Content and Description explains the UVM Reference Flow directory structure and contents. This package is located at the following location:

```
$UVM_REF_HOME/soc_verification_lib/sv_cb_ex_lib
```

Table 3. Package Content and Description

Directory	Filename	Description
`uart_ctrl/`	PACKAGE_README.txt	Describes UART environment package usage instructions.
`uart_ctrl/sv/`	uart_ctrl_defines.svh	UART DUV register offset defines
	uart_ctrl_scoreboard.sv	Scoreboard for data integrity check between APB UVC and UART UVC
	sequence_lib/uart_ctrl_seq_lib.sv	APB sequences specific to UART Tx/Rx FIFO write/read.
	uart_ctrl_virtual_sequencer.sv	This file implements the Virtual Sequencer for APB-UART environment
	sequence_lib/uart_ctrl_virtual_seq_lib.sv	This file contains virtual sequences for the APB-UART env
	uart_ctrl_env.sv	Implements the module UVC for the APB-UART Environment. Converts the UART Configuration writes from APB into a uart_config data structure

		to keep the UART UVC in sync with the DUT configuration.
	uart_ctrl_reg_model.sv	Contains the address map, register file, register and field definition.
	uart_ctrl_monitor.sv	Instantiates Rx and Tx scoreboards.
	uart_ctrl_reg_seq_lib.sv	Contains the register-memory sequences.
`uart_ctrl/sv/coverage/`	uart_ctrl_cover.sv	Cover groups to provide functional coverage collection on DUT and ENV specific points.
	uart_ctrl_internal_if.sv	UART interface for internal signals.
`uart_ctrl/tb/scripts/`	Makefile	Runs the test case
	covfile.cf	The irun coverage instructions.
	run.sh	Runs the test case with make file.
`uart_ctrl/tb/sv/`	uart_ctrl_tb.sv	This file implements the SVE for the APB-UART Environment.
	uart_ctrl_top.sv	Top-level testbench for the UART environment.
`uart_ctrl/tb/tests/`	Refer to Test Cases	Test case session.

Run Scripts

The simulation scripts for the UART module UVM SystemVerilog environment are available in the following directory:

```
$UVM_REF_HOME/soc_verification_lib/sv_cb_ex_lib/uart_ctrl/tb/scripts
```

The scripts are explained below.

- **run.sh** – This script should be executed to simulate the selected test in the UART UVM SystemVerilog environment.

  The testcase name can be passed as an argument. The default test case name is apb_uart_rx_tx.

- **covfile.cf** – A coverage configuration file. This file is used to inform irun how to elaborate the design so that coverage can be collected if desired. It is sourced during irun compilation using the

`+covfile+<covfile.cf>` command line option. The default version of this file enables the following coverage to be collected, if desired, during simulation:

- Block, expression, and toggle code coverage on the UART RTL modules.
- FSM coverage on the UART RTL modules.
- Functional coverage collection on the whole environment.

Test Cases

The UART module UVM SystemVerilog environment contains few test cases, which are described in the following table.

Table 4. Test Cases

S. No.	Test Name	Description
1.	apb_uart_rx_tx	apb_uart_rx_tx test class, configures DUV and starts concurrent transfers from APB writes to UART TX FIFO and UART UVC transmits to UART RX
2.	uart_apb_incr_data	Defines uart_apb_incr_data test class that first configures the UART DUV. After that UART UVC starts sending incremented data to UART DUV. APB UVC starts interrupt sequence which reads RX FIFO.
3.	apb_to_uart_1stopbit	apb_to_uart_1stopbit test class, configures the UART DUV. After that UART UVC starts sending random data with 1 stop bit

Running a Simulation

You can run a simulation either with or without Enterprise Manager.

Steps to Run Simulation

If you have not already done so, install the UVM Reference Flow and set up some environment variables as described in the Setup and Installation Instructions.

1. Change to your working directory.

   ```
   cd <path to your work area>
   ```

2. Execute the following script:

   ```
   $UVM_REF_HOME/soc_verification_lib/sv_cb_ex_lib/uart_ctrl/tb/scripts/run.sh –test
      <test_case>
   ```

3. Refer to Test Cases for various test cases.
   For example:

   ```
   $UVM_REF_HOME/soc_verification_lib/sv_cb_ex_lib/uart_ctrl/tb/scripts/run.sh –test
      apb_uart_rx_tx
   ```

UVM SystemVerilog Subsystem-Level Verification

This section describes how a verification environment (VE) can be constructed to verify the functionality at the cluster or subsystem level with UVM which reuses the existing module level components from the UART environment. For this demonstration, the chosen subsystem contains an AHB bus matrix, two UARTs, SPI and GPIO which are included within the UVM Reference Flow. A UVM environment is built around the APB subsystem by reusing the existing module/block-level UART verification environment components.

This section illustrates an implementation based on UVM SystemVerilog.

The APB Subsystem DUV

The APB subsystem contains UART, SPI, GPIO and other blocks as shown in the UART DUT figure. These blocks are written in synthesizable Verilog RTL. The Segment Representative Design (SRD) contains two instantiations of the UART IP module one to show the low power features.

The DUV has the following interface which needs to be driven/monitored by the UVCs:

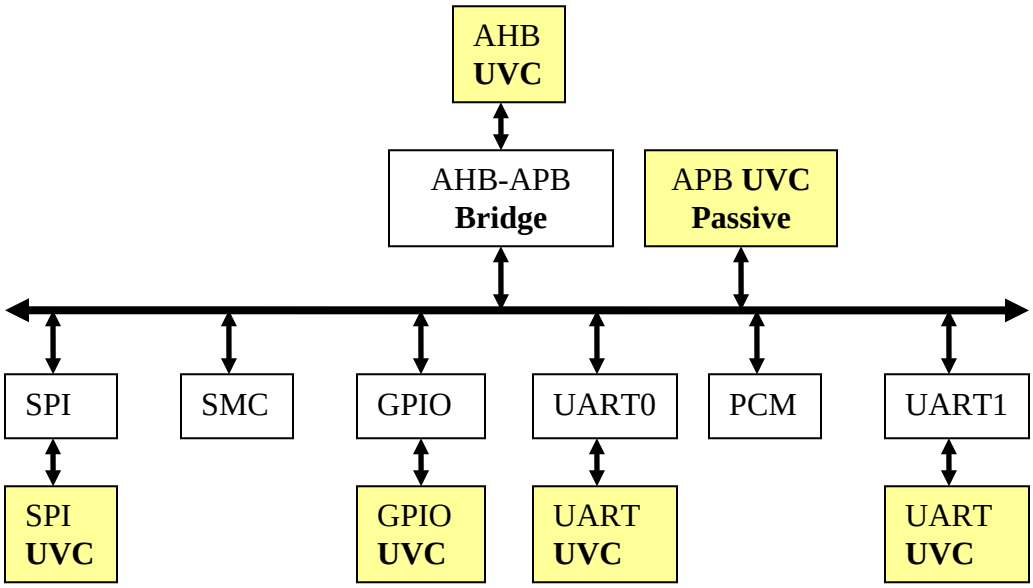
- AHB interface
- APB (WISHBONE) interface
- UART interface
- GPIO interface
- SPI interface
- Clocks and resets

APB Subsystem Verification—UVM Reference Flow Methodology

The APB subsystem environment is implemented with reference to coverage-driven methodology using constrained randomization.

The APB Subsystem environment is constructed as shown in the APB Subsystem Verification Environment figure.

Figure 3. APB Subsystem Verification Environment



Universal Verification Components (UVCs)

The APB Subsystem environment uses five UVCs: AHB, APB, UART, GPIO and SPI UVC constructed for the UVM Reference Flow according to the UVM SystemVerilog. These UVCs are used to provide stimulus and monitoring functionality for the DUV interfaces. The UVCs are provided in the following locations:

AHB: $UVM_REF_HOME/soc_verification_lib/sv_cb_ex_lib/interface_uvc_lib/ahb/sv

APB: $UVM_REF_HOME/soc_verification_lib/sv_cb_ex_lib/interface_uvc_lib/apb/sv

UART: $UVM_REF_HOME/soc_verification_lib/sv_cb_ex_lib/interface_uvc_lib/uart/sv

GPIO: $UVM_REF_HOME/soc_verification_lib/sv_cb_ex_lib/interface_uvc_lib/gpio/sv

SPI: $UVM_REF_HOME/soc_verification_lib/sv_cb_ex_lib/interface_uvc_lib/spi/sv

Test Sequence Library

The Test Sequence Library provides some common sequences for simplifying the writing of tests. The library is defined to provide a simple sequence-based call for the test writer and thus provide a higher-level interface to the APB subsystem environment.

The Test Sequence Library provides the following sequences:

Table 5. Test Sequence Library

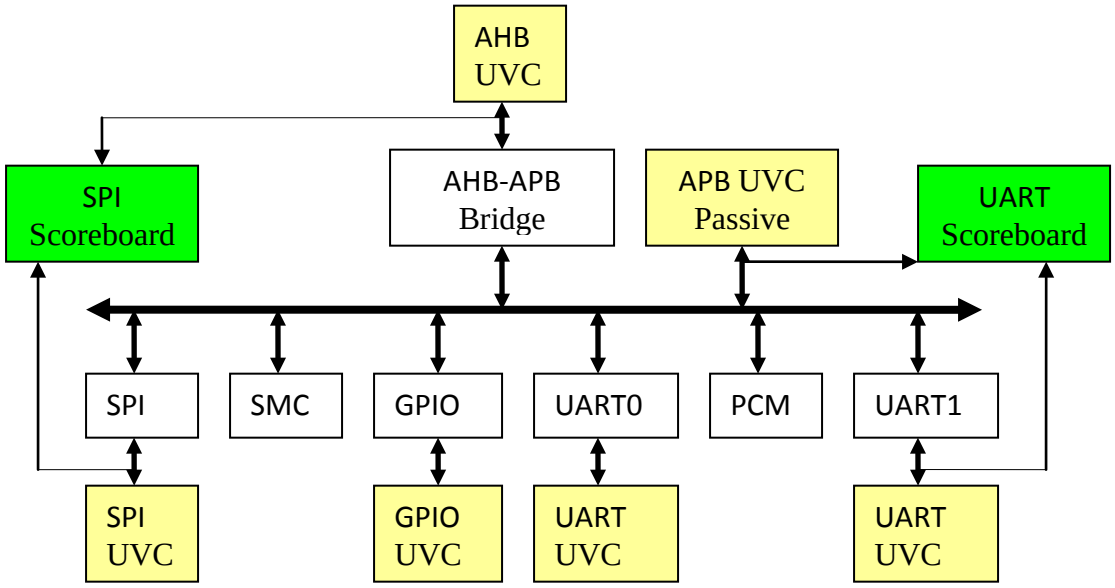
Sequence name	Description
`u2a_incr_payload`	This sequence exercises only the ahb-apb-uart0 and ahb-apb-uart1 path. This sequence first configures the UART0, UART1. After configuration is done UART UVC starts sending incremented data to UART* DUT. Simultaneously, AHB UVC starts filling the TX FIFO of the UART for transmitting it to UART UVC. When the above sequences complete, AHB UVC reads the UART's RX FIFO.
`apb_spi_incr_payload`	This sequence is same as the u2a_incr_payload sequence, except that the AHB-APB-SPI path is exercised instead of the AHB_APB_UART path.
`apb_gpio_simple_vseq`	This sequence exercises only the AHB-APB-GPIO path. It configures the GPIO module and then sends a transaction from AHB to GPIO. When this is completed, it sends a transaction from GPIO to AHB.
`apb_subsystem_vseq`	This sequence exercises all the three paths simultaneously: AHB-APB-UART*, AHB-APB-SPI, and AHB-APB-GPIO. It configures the UARTs, SPI and GPIO modules, and initiates transactions to and from all three modules.

Scoreboards

A scoreboard is used to verify end-to-end data transformation through the DUV. The scoreboard collects data from the UVC monitors at the interfaces of the DUV and compares the results against the expected transformation. This transformation can be very simple or can be a complex model, depending on the type of the DUV.

The Scoreboard implementation in the APB subsystem environment is as shown in Figure 4.

Figure 4. APB Subsystem Environment



The APB subsystem environment contains two scoreboards:

- The UART scoreboard (two instances, one for each UART) is reused from the module/block-level environment.
- The SPI scoreboard is used for end-to-end check between AHB and SPI UVCs

The transformation for these scoreboards is very simple in this example because the data input is the same as the data output.

The two scoreboards were developed to use standard synchronization events defined in the UVM-compliant UVC monitors and provide the data into and out of the DUV. The scoreboards are instantiated inside the module UVC.

Coverage Module

The Coverage Module is reused from the UART module/block-level environment. This coverage is collected in addition to protocol coverage provided by the individual UVCs.

These coverage points are implemented based on the requirements of the environment vPlan, which was created during the project verification planning phase.

Using the APB Subsystem Verification Flow

Refer to Setup and Installation Instructions if you have not already installed and set up the tools you need for this flow.

The rest of this section discusses

- Package Directory Structure and Contents

- Run Scripts
- Test Cases

Package Directory Structure and Contents

Package Content and Description explains the UVM Reference Flow directory structure and contents. This package is located at the following location:

$UVM_REF_HOME/soc_verification_lib/sv_cb_ex_lib

Table 6. Package Content and Description

Directory	Filename	Description
`apb_subsystem/`	PACKAGE_README.txt	Describes APB Subsystem environment package usage instructions.
`apb_subsystem/sv/`	spi_defines.svh gpio_defines.svh	SPI, GPIO DUV register offset defines
	ahb_user_monitor.sv	Implements an AHB user monitor which is extended from ahb_master_monitor. Whenever an AHB transaction is completed, it determines if it is a SPI or GPIO configuration register write. If so, it sends it on the appropriate port for the UVCs to get synced.
	sequence_lib/apb_subsystem_seq_lib.sv	This file contains a library of AHB sequences to initiate transactions to various peripherals.
	apb_subsystem_vir_sequencer.sv	This file implements the Virtual Sequencer for APB Subsystem environment
	apb_subsystem_vir_seq_lib.sv	This file contains virtual sequences for the APB subsystem environment. Refer to Table 2 for details.
	apb_subsystem_env.sv	This file implements the module UVC for the APB Subsystem Environment. Instantiates SPI RX

		and TX scoreboard. This is to be reused at the higher/chip level verification environments.
	spi_reg.sv gpio_reg.sv	Contains the register file, register and field definition.
	apb_subsystem_rgm_master_sequencer.sv	This file defines the register-memory master sequencer and the adapter sequence that converts the register-memory read/write operations into AHB specific operations.
	sequence_lib/spi_reg_seq_lib.sv sequence_lib/gpio_reg_seq_lib.sv	Contains the register sequences for SPI and GPIO CSR programming.
	apb_subsystem_reg_rdb.sv	Implements the address map for the APB subsystem and declares the RDB for reg package.
	apb_subsystem_scoreboard.sv	Implements the end to end data integrity check for AHB to SPI transfers and SPI to AHB transactions.
`apb_subsystem/tb/scripts/`	Makefile	Makefile to run the test case
	covfile.cf	The irun coverage instructions.
	run.sh	Runs the test case with make file
	nc_waves.tcl	tcl script to dump waveforms in interactive debug mode.
	assert_opt.tcl	tcl script to enable assertion summary to be printed at the end of the test run.

apb_subsystem/tb/sv/	apb_subsystem_tb.sv	This file implements the SVE for the APB Subsystem Environment.
	apb_subsystem_top.sv	Top-level testbench for the APB Subsystem environment.
apb_subsystem/tb/tests/	Refer to Test Cases	Test case session.

Run Scripts

The simulation scripts for the UART module UVM SystemVerilog environment are available in the following directory:

```
$UVM_REF_HOME/soc_verification_lib/sv_cb_ex_lib/apb_subsystem/tb/scripts
```

The scripts are explained below.

- **run.sh** – This script should be executed to simulate the selected test in the APB subsystem SystemVerilog environment in batch mode.

  The test case name can be passed as an argument. The default test case name is apb_subsystem_test.

- **covfile.cf** – A coverage file. This file is used to inform irun how to elaborate the design so that coverage can be collected, if desired. It is sourced during irun compilation using the `-covfile <covfile.cf>` command line option. The default version of this file enables the following coverage to be collected, if desired, during simulation:
  - Block, expression, and instance code coverage on the APB subsystem RTL modules
  - FSM coverage on the APB Subsystem RTL modules
  - Functional coverage collection on the whole environment

Test Cases

The APB Subsystem UVM SystemVerilog environment contains few test cases, which are described in the following table.

Table 7. Test Cases

S. No.	Test Case	Description
1.	apb_gpio_simple_test	This test sets the default sequence to apb_gpio_simple_vseq. Refer Table 6 for details of the apb_gpio_simple_vseq sequence.
2.	apb_spi_simple_test	This test sets the default sequence to apb_spi_incr_seq. Refer Table 6 for details of the apb_spi_incr_seq sequence

3.	apb_subsystem_test	This test sets the default sequence to apb_subsystem_test. Refer Table 6 for details of the apb_subsystem_test sequence
4.	apb_uart_simple_test	This test sets the default sequence tou2a_incr_payload. Refer Table 6 for details of the u2a_incr_payload sequence

Running a Simulation

Steps to Run Simulation

1. Install the UVM Reference Flow and set up some environment variables as described in Setup and Installation Instructions.
2. Change to your working directory to the area in which you want to run the simulation.

   ```
   cd <path to your work area>
   ```

3. Execute the following script:

   ```
   $UVM_REF_HOME/soc_verification_lib/sv_cb_ex_lib/apb_subsystem/tb/scripts/run.sh
      –test <test_case>
   ```

   Refer to the Test Cases table for various test cases. For example:

   ```
   $UVM_REF_HOME/soc_verification_lib/sv_cb_ex_lib/apb_subsystem/tb/scripts/run.sh
      -test apb_spi_simple_test
   ```