COMP.CE.420 System-on-Chip Verification

Santeri Laurila

Alper Özaslan

Integration-level testbench Exercise

1. General structure

1. Test name is **pss_spi_interrupt_test** (configurated in Makefile)

2. Tests are found in *ex_uvm/integration_tb/tb_build/sub_system_tbs/pss_tb/test*

Test names:

- **pss_gpio_outputs_test.svh**
- **pss_spi_interrupt_test.svh**
- **pss_spi_polling_test.svh**
- **pss_test_lib_pkg.svh**
- **pss_gpio_outputs_test.svh.bck**
- **pss_spi_interrupt_test.svh.bck**
- **pss_test_base.svh**
- **pss_test.svh**

3. DUT is located in /ex_uvm/integration_tb/tb_build/rtl/ahb2apb/rtl.
Our DUT is AHB2APB bridge. It basically converts host AHB cycles to target APB cycles. Data transfers are not buffered but done by one-to-one conversions. For bridge to support the AHB to APB transfer, some control signals must be piped to 'Response MUX'.

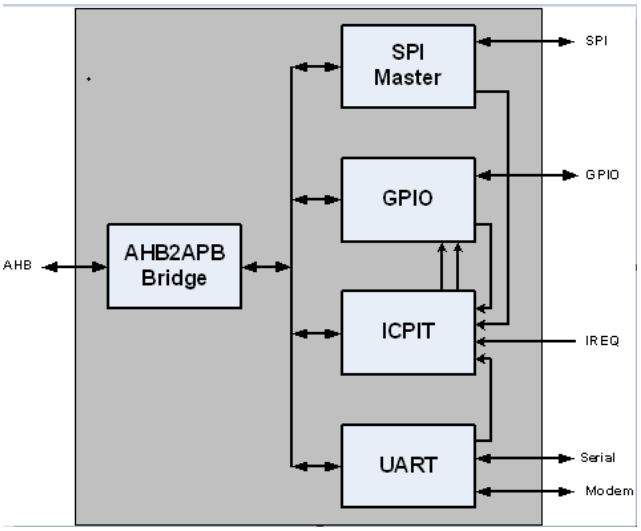


Figure 1 Internal scructure of DUT

4. There are testbenches for subsystems blocklevels. There are two block-level testbenches. gpio_tb and spi_tb. So, in DUT, AHB2APB-bridge, ICPIT and UART are not tested as block-level tests.

In tests-folder there are tests for GPIO and SPI IP:s. There are no tests for ICPIT or AHB2APB, so there is no coverage for those.

5. The example takes the SPI block level example and integrates it with another block level verification environment for a GPIO DUT. The hardware for the two blocks has been integrated into a Peripheral Sub-System (PSS) which uses an AHB to APB bus bridge to interface with the APB interfaces on the SPI and GPIO blocks.

The environments from the block level are encapsulated by the pss_env, which also includes an AHB agent to drive the exposed AHB bus interface. In this configuration, the block level APB bus interfaces are no longer exposed, and so the APB agents are put into passive mode to monitor the APB traffic. The stimulus needs to drive the AHB interface and register layering enables reuse of block level stimulus at the integration level. (from course material) UVM topology is shown in Fig 2.

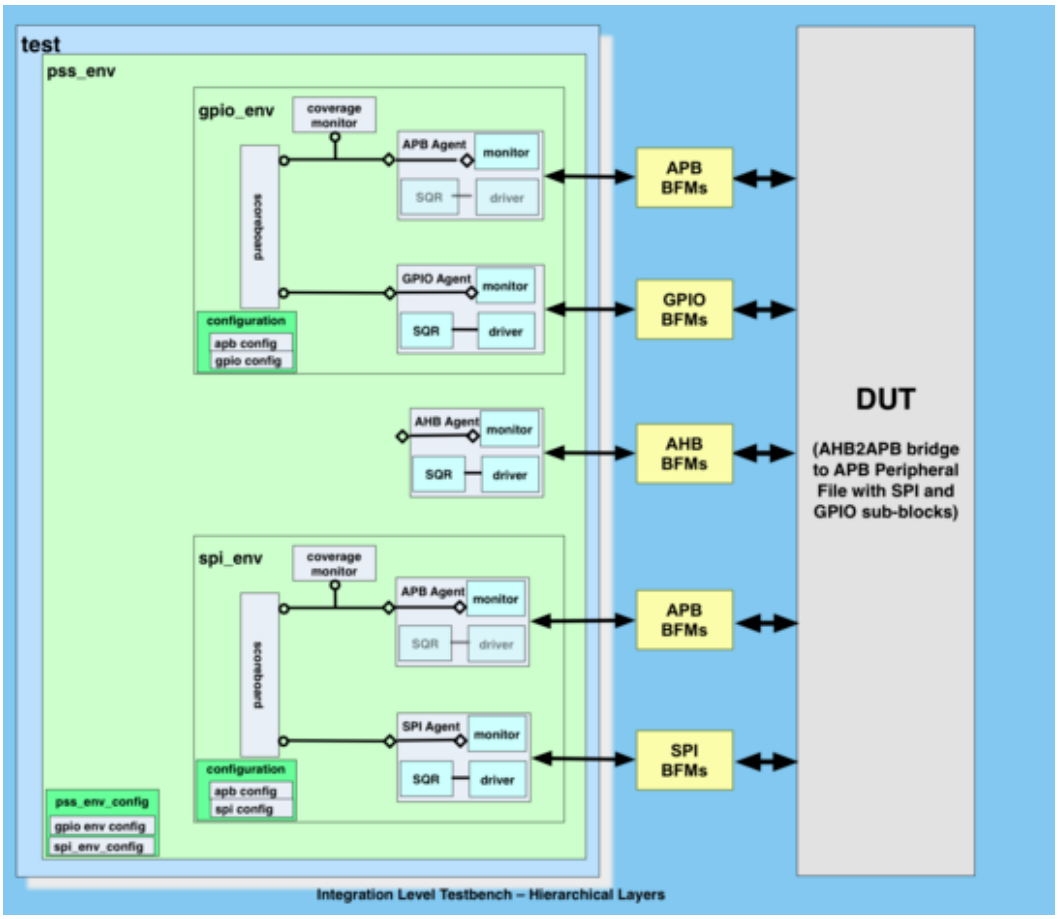


Figure 2 UVM topology

2. Connectivity

1. Top modules are:

- **pss_env_config.svh**
- **pss_env_pkg.sv**
- **pss_env.svh**

Modules are connected using phase-function in pss_env.

2. In DUT, there are different input/output interfaces for different IPs. In Fig. 3 interfaces are shown. DUT has interfaces for AHB Host, SPI, GPIO and UART.

```
module pss(// AHB Host interface
           input HCLK,
           input HRESETn,
           input[31:0] HADDR,
           input[1:0] HTRANS,
           input HWRITE,
           input[2:0] HSIZE,
           input[2:0] HBURST,
           input[3:0] HPROT,
           input[31:0] HWDATA,
           input HSEL,
           output [31:0] HRDATA,
           output HREADY,
           output [1:0] HRESP,
           // SPI interface
           output[7:0] spi_cs,
           output spi_clk,
           output spi_mosi,
           input spi_miso,
           // GPIO Interface
           input[31:0] gpi,
           output[31:0] gpo,
           output[31:0] gpoe,
           // UART Interface
           input rxd,
           output txd,
           // modem signals
           output rts,
           input cts,
           output dtr,
           input dsr,
           input ri,
           input dcd,
           output baud,
           // External Interrupts
           input[4:0] IREQ,
           output IRQ);
```

Figure 3 pss interfaces

3. Purpose of Bus Functional Model (BFM) is to mimic bus-like behavior. BFMs are usually defined as tasks. Typically, BFMs offer a two-sided interface: One interface side drivers and samples low-level signals according to the bus protocol, other side tasks are available to create and respond to bus transactions.

Instantiation of BFM interface is done in hdl_top. Pins of BFM is connected to the DUT. Similarly, BFMs for SPI, APB, AHB and GPIO are connected to the respective agent.

3. Configuration

1. Configuration for the testbench is done by passing config-objects around. In pss_env_config.svh, config is being set for the environment.

2. There are configuration objects for **spi_env, gpio_env and ahb_agent** (located in pss_env_config.svh)

3. Components affected by configuration are **spi, gpi** and **ahb_agent.**

4. Agents

1. There are six (6) types of Agents available: **ahb_agent, apb_agent, gpio_agent, modem_agent, spi_agent, uart_agent**.

2. Only **ahb_agent** is being used in PSS environment.

3. There are differences between tests. In some tests, some agents are set active and some passive.

→ In pss_spi_polling_test.svh spi_agent is set active. But in pss_spi_interrupt_test, spi_agent is set passive.

→ In pss_gpio_outputs_test, GPI_agent is set as active.

→ In pss_test_base, different agents are being set to different environments.

5. Constraints

We found many constraints in files using grep-command: **grep -nr 'constraint*'**

In **spi_bus_sequence_lib_pkg.sv:**

```
1
2  constraint div_values {data[15:0] inside {16'h0, 16'h1, 16'h2, 16'h4, 16'h8, 16'h10, 16'h20, 16'h40, 16'h80};}
3
```

'Inside' is used for checking if the specified values (in brackets) are existing in the data[16:0]

In Figure 4 is all the found constraints under tb_build/

```
[laurila2@linux-desktop3 integration_tb]$ grep -nr 'constraint*'
tb_build/block_level_tbs/spi_tb/sequences/spi_bus_sequence_lib_pkg.sv:146:  constraint div_values {data[15:0] inside {16'h0, 16'h1, 16'h2, 1
6'h4, 16'h8, 16'h10, 16'h20, 16'h40, 16'h80};}
Binary file tb_build/rtl/spi/doc/src/spi.doc matches
tb_build/agents/apb_agent/apb_seq_item.svh:47:constraint addr_alignment {
tb_build/agents/apb_agent/apb_seq_item.svh:51:constraint delay_bounds {
tb_build/agents/modem_agent/modem_agent_pkg.sv:45:  constraint clamp_top_bits {modem_bits[5:4] == 0;}
tb_build/agents/uart_agent/uart_seq_item.svh:31:// Need some constraints
tb_build/agents/uart_agent/uart_seq_item.svh:33:constraint BR_DIVIDE {baud_divisor == 16'h02;}
tb_build/agents/uart_agent/uart_seq_item.svh:35:constraint error_dists {fe_dist {1:= 1, 0:=99};
tb_build/agents/uart_agent/uart_seq_item.svh:39:constraint clks {delay inside {[0:20]};
tb_build/agents/uart_agent/uart_seq_item.svh:42:constraint lcr_setup {lcr == 8'h3f;}
tb_build/agents/ahb_agent/ahb_seq_item.svh:45:constraint addr_align {HADDR[1:0] == 0;}
```

Figure 4 constraints

6. Covergroups

Covergroups are located under:
/home/ozaslan/13/ex_uvm/integration_tb/tb_build/block_level_tbs/gpio_tb/env

```
gpio_in_scoreboard.svh:covergroup gpi_cov;
gpio_in_scoreboard.svh:covergroup ints_cov;
gpio_out_scoreboard.svh:covergroup gpoe_cov;
gpio_out_scoreboard.svh:covergroup gpo_cov;
```

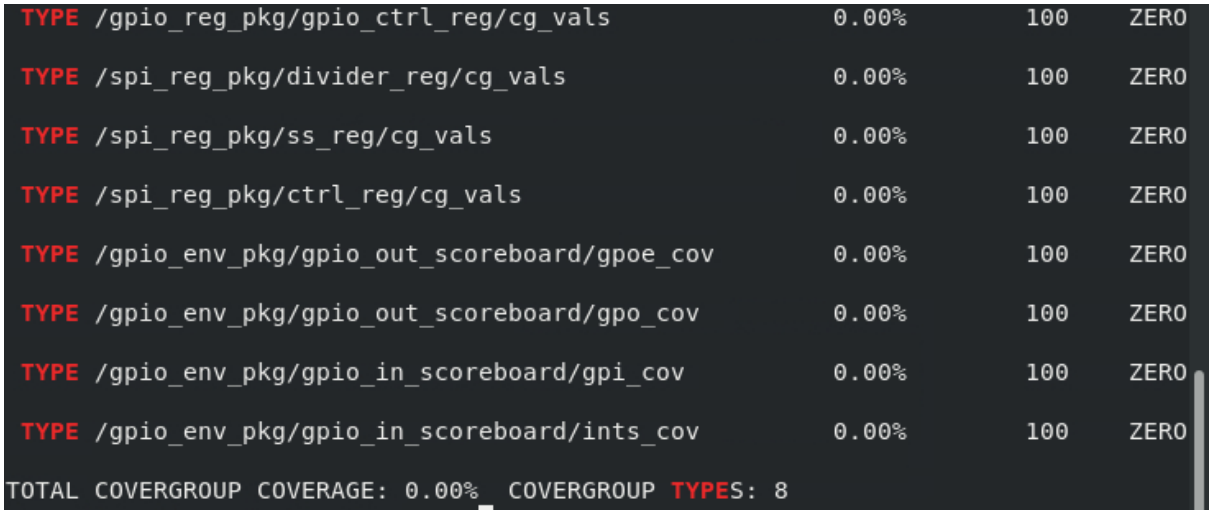
Every bit of GPI, ECLK_reg, NEC_reg are covered as 1 covergroup. INTS, INTE_reg and OTRIG_reg signals are covered as another group.

'Cross' coverage is applied to these two groups meaning that their 'cross product' of each group is checked. For example: cross product of GPO, AUX and AUX_in inside 'gpio_out_scoreboard.svh'

Path of Makefile: /home/laurila2/13/ex_uvm/integration_tb/tb_build/sub_system_tbs/pss_tb/sim

We ran different tests from /test/ folder, by modifying Makefile. With many tests TOTAL COVERGROUP COVERAGE shows 0%.

With test pss_spi_interrupt_test covergroup coverage shows 0%.



7. Register Abstraction Layer

The UVM Register Layer provides a standard base class libraries, that enable users to implement the object-oriented model to access the DUT registers and memories. UVM Register Layer is also referred to as UVM Register Abstraction Layer (UVM RAL) (source: chipverify.com).

Register Abstracton Layer is used in spi_test_base. RAL is used to enable all types of coverage available in the register model. Handles are used for the register model in the env config.

In uvm_register-2.0/examples/ -folder some examples of backdoors are found.

UVM Register Backdoor Access

UVM allows backdoor accesses, which uses a simulator database to directly access the signals within the DUT. Write operations deposit a value onto the signal and read operations sample the current value form the register signal. Since registers are the leaf nodes in a digital system, depositing a new value in the middle of any design transitions should not cause any problem (source: chipverify.com)

8. Tests

1. Inspection of the test pss_spi_interrupt_test.

Run sequence:

```
task pss_spi_interrupt_test::run_phase(uvm_phase phase);
  spi_int_vseq t_seq = spi_int_vseq::type_id::create("t_seq");
  assign_sequencers(t_seq);
//  t_seq.m_cfg = m_spi_env_cfg;
//  t_seq.spi = m_env.m_spi_env.m_spi_agent.m_sequencer;

  phase.raise_objection(this, "Starting PSS SPI interrupt test");

  repeat(10) begin
    t_seq.start(null);
  end

  phase.drop_objection(this, "Finishing PSS SPI interrupt test");
endtask: run_phase
```

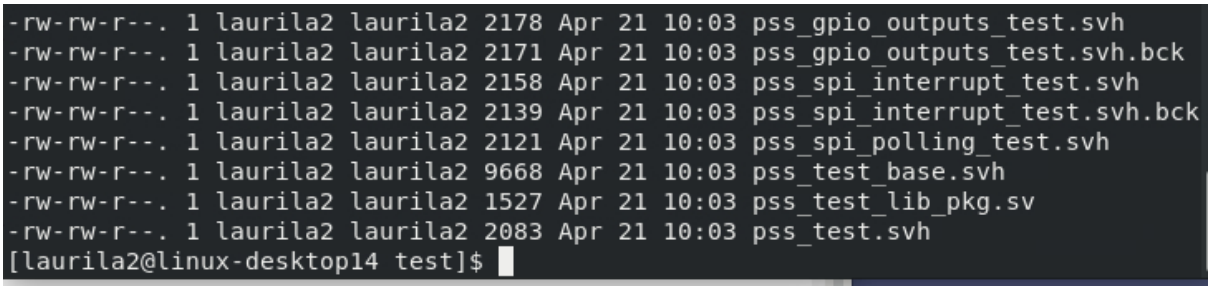
t_seq is ran 10 times

2. -
3. Errors in tests -part

```
# ** Report counts by severity
# UVM_INFO :  109
# UVM_WARNING :    2
# UVM_ERROR :    0
# UVM_FATAL :    0
# ** Report counts by id
# [** UVM TEST PASSED **]     2
# [GPIO_IN_SB]     1
# [GPIO_OUTPUT_SB]     2
# [RNTST]     1
# [SPI_DRV_RUN:]   100
# [SPI_SB_REPORT:]     1
# [TEST_DONE]     1
# [UVM/RELNOTES]     1
# [UVM/RSRC/NOREGEX]     2
#
# ** Note: $finish    : /opt/lintula/modelsim/10.7d/modeltech/verilog_src/uvm-1.
2/src/base/uvm_root.svh(517)
#    Time: 2688380 ns  Iteration: 85  Instance: /hvl_top
# Saving coverage database on exit...
# End time: 12:59:27 on May 03,2022, Elapsed time: 0:00:03
# Errors: 0, Warnings: 0
vcover report -details vsim.ucdb > report.txt
[laurila2@linux-desktop14 sim]$
```

➔ pss_spi_interrupt_test did not found any errors in the DUT.

4. List of different tests in the testbench


-rw-rw-r--. 1 laurila2 laurila2 2178 Apr 21 10:03 pss_gpio_outputs_test.svh
-rw-rw-r--. 1 laurila2 laurila2 2171 Apr 21 10:03 pss_gpio_outputs_test.svh.bck
-rw-rw-r--. 1 laurila2 laurila2 2158 Apr 21 10:03 pss_spi_interrupt_test.svh
-rw-rw-r--. 1 laurila2 laurila2 2139 Apr 21 10:03 pss_spi_interrupt_test.svh.bck
-rw-rw-r--. 1 laurila2 laurila2 2121 Apr 21 10:03 pss_spi_polling_test.svh
-rw-rw-r--. 1 laurila2 laurila2 9668 Apr 21 10:03 pss_test_base.svh
-rw-rw-r--. 1 laurila2 laurila2 1527 Apr 21 10:03 pss_test_lib_pkg.sv
-rw-rw-r--. 1 laurila2 laurila2 2083 Apr 21 10:03 pss_test.svh
[laurila2@linux-desktop14 test]$