

User Guide: Integrating APB and eSPI BFM with ibl_espi_core

1 Overview

This guide explains how to integrate the APB Bus Functional Model (BFM) and the eSPI BFM with the ibl_espi_core. We will use a top-level module xnc_top that includes clock and reset signal generation, interface definitions, Device Under Test (DUT) instantiation, and UVM environment configuration.

2 Integration BFM with eSPI DUT

2.1 Define interface

1. Define APB Interfaces:

Use the amba3_apb_if interface for different APB peripherals.

In the xnc_top.sv:

```
amba3_apb_if #(ADDR_BITS, DATA_BITS) apb_plcc_intf (clk,reset_n);  
amba3_apb_if #(ADDR_BITS, DATA_BITS) apb_vw_intf (clk,reset_n);  
amba3_apb_if #(ADDR_BITS, DATA_BITS) apb_flash_intf (clk,reset_n);  
amba3_apb_if #(ADDR_BITS, DATA_BITS) apb_cfg_intf (clk,reset_n);
```

2. Define eSPI Interface:

Use the espi_ec_bfm_if interface for the eSPI BFM.

In the xnc_top.sv:

```
espi_ec_bfm_if ec_bfm_if(.clock(clk), .reset(reset_n));
```

2.2 Instantiate the eSPI DUT

1. Connect the APB and espi interface with the eSPI DUT

2.3 Interface Configuration

1. apb interface config

The APB interface configuration involves creating a configuration object (xnc_cfg) and setting it in the UVM configuration database. This configuration object is used to pass interface handles and other configuration parameters to the environment and its components.

```
xnc_cfg cfg;
```

```

cfg = new("config", apb_plcc_intf, apb_vw_intf, apb_flash_intf, apb_cfg_intf);

uvm_pkg::uvm_config_object::set(null, "*", "config", cfg);

```

2. eSPI interface config

The eSPI configuration involves setting virtual interface handles for the eSPI BFM agents in the UVM configuration database. This allows the eSPI BFM agents to access the eSPI interface signals.

```

uvm_config_db#(virtual espi_ec_bfm_if)::set(null, "uvm_test_top.t_env.espi_bfm_agent0",
"ec_bfm_vif", ec_bfm_if);

uvm_config_db#(virtual espi_ec_bfm_if)::set(null, "uvm_test_top.t_env.espi_bfm_agent1",
"ec_bfm_vif", ec_bfm_if);

```

3 Integrate BFM into the ENV

3.1 Instantiate the Agents

In the `build_phase` function in the `xnc_env.sv`, the APB master agent and eSPI BFM agents are instantiated. The APB master agent is created directly, while the eSPI BFM agents are created in the `build_espi` function.

```

function void build_phase(uvm_phase phase);

    uvm_object tmp;

    super.build_phase(phase);

    `uvm_info(get_full_name(),"START of build ",UVM_LOW);

    // Instantiate APB master agent
    apb_master_agent = ibl_apb_master_agent::type_id::create("apb_master_agent", this);

    // Retrieve configuration object
    assert(uvm_config_object::get(this, "", "config", tmp));

    assert($cast(cfg,tmp));

    `uvm_info(get_full_name(),"END of build ",UVM_LOW);

    // Build eSPI BFM agents
    build_espi();

endfunction

```

3.2 Build the eSPI BFM Agents

The `build_espi` function creates an array of eSPI BFM agents in the `xnc_env.sv`. The number of agents is determined by the size of the array (in this case, the size is 2). It means it instantiated the size number eSPI slave BFM.

```
function void build_espi();

    string name;

    espi_bfm_agent = new[2];

    for (int i=0; i<2; i++) begin

        $sformat(name, "espi_bfm_agent%1d", i);

        set_config_int(name,"is_active",UVM_ACTIVE); // EC BFM should be active most of the time.
        Tests can override this if needed.

        if($test$plusargs("ESPI_BFM_DISABLE")) begin

            set_config_int(name,"is_active",UVM_PASSIVE);

        end

        espi_bfm_agent[i] = espi_ec_bfm_agent::type_id::create(name, this);

    end

endfunction : build_espi
```

3.3 Connect the Agents

In the `connect_phase` function in the `xnc_env.sv`, the eSPI BFM agents are connected and configured. The `connect_espi` function is called to set up the necessary connections and enable the required features.

```
function void connect_phase(uvm_phase phase);

    super.connect_phase(phase);

    `uvm_info(get_full_name(),"START of connect ",UVM_LOW);

    connect_espi();

    `uvm_info(get_full_name(),"END of connect ",UVM_LOW);

endfunction
```

```
function void connect_espi();

    for (int i=0; i<2; i++) begin

        espi_bfm_agent[i].monitor.espi_enabled = 1;

        espi_bfm_agent[i].monitor.checks_enable = 1;

        espi_bfm_agent[i].enable_compliance_checker = 1;
```

```

        if(espi_bfm_agent[i].is_active == UVM_ACTIVE) begin
            espi_bfm_agent[i].driver.espi_enabled = 1;
            espi_bfm_agent[i].driver.set_valfw_enable(0);
        end
        espi_bfm_agent[i].espi_enabled = 1;
    end
endfunction : connect_espi

```

3.4 Configure the eSPI BFM Agents

The `configure_espi` function is called in the `end_of_elaboration_phase` to configure the eSPI BFM agents in the `xnc_env.sv`. This includes enabling various channels and setting specific register values.

```

function void end_of_elaboration_phase(uvm_phase phase);
    configure_espi();
endfunction

function void configure_espi();
    `uvm_info(get_name(), "\nConfigure eSPI Slave 0 General Cap Register\n", UVM_LOW)
    for (int i=0; i<2; i++) begin
        espi_bfm_agent[i].enable_periph_ch = 1'b1;
        espi_bfm_agent[i].enable_vwire_ch = 1'b1;
        espi_bfm_agent[i].enable_oob_ch = 1'b1;
        espi_bfm_agent[i].enable_flash_ch = 1'b1;
        if($test$plusargs("ESPI_FLASH_CH_DISABLE")) begin
            espi_bfm_agent[i].enable_flash_ch = 1'b0;
        end
        espi_bfm_agent[i].min_freq = _20M;
        espi_bfm_agent[i].min_iomode = SINGLE;
        espi_bfm_agent[i].driver.ec_spi_mem.WriteSpiDWord(32'h44, 32'hc0c133e8); // Flash Region 1(BIOS) Register
    end
endfunction : configure_espi

```

4 Compilation and Simulation

4.1 Folder Structure

The folder path is ./fe_collateral, the structure is as follows:

```
---source/--  
  
    --rtl/  # espi RTL  
  
    --val/  #verification header files  
  
---verify/---  
  
    --espi_standalone_tb/  #espi standalone testbench  
  
        --env/             #test env  
  
        --interface/       #AMBA APB interface  
  
        --registermap/     #espi register map file  
  
        --seqitem/         #APB seqitem  
  
        --seqlib/          #test sequence for all espi channels(cfg,plcc,vwcc,safcc)  
  
        --seqrlib/         #APB sequencer  
  
        --tests/           #test cases for all espi channels(cfg,plcc,vwcc,safcc)  
  
        --scripts/        #scripts for environment set  
  
        --tb/---  
  
            --apb_master_agent/  #APB mater BFM  
  
            --espi_bfm/          #espi slave BFM  
  
        --xnc_top.sv          #testbench top level file  
  
        --xnc_pkg.sv         #includes files package
```

4.2 Compilation and Simulation

There is a README in the espi_standalone_tb folder and please follow the steps described in README to update the scripts for environment set.

After the environment set, then you can compile the tb and run the simulation.

- run below command to build the model:

make clean && make compile

- run below command to run different test

1. run the cfg test

```
make ibl_espi_basic_cfg_test
```

2. run the Channel 0: Peripheral/LPC (PLCC) test

```
make ibl_espi_basic_lpc_test
```

3. run the Channel 1: Virtual Wire (VWCC) test

```
make ibl_espi_basic_vw_test
```

4. run the Channel 3: Slave Attached Flash test

```
make ibl_espi_basic_flash_test
```

- if load waveform through DVE, run command in this format

```
make xxx_test_waves
```

(e.g. make ibl_espi_basic_cfg_test_waves)

- check the log and tracker in path test_run_dir/xxx_test

xxx_test.log is the log and *.out is the tracker print by espi bfm

4.3 Tools and Package Version Information

Here lists the versions of the main tools and packages currently used in the project. To prevent errors, please ensure that the following tools and package versions are consistent across the project.

1. Synopsys VCS

- Tool Name: Synopsys VCS (Verilog Compiler Simulator)
- Version: Q-2020.03

2. UVM Package

- Package Name: UVM (Universal Verification Methodology)
- Version: 1.2

3. Synopsys Verdi

- Tool Name: Synopsys Verdi (Debug and Verification)
- Version: Q-2020.03