Training on
IP & SoC Functional Verification Methodology
Using UVM

LAB
Virtual Interfaces
In SystemVerilog

Objectives

In the last lab the students had the opportunity to know the Interface SytemVerilog feature. Now, they will know a new concept, the Virtual Interface

Global Explanation

In this lab the *driver* is no longer a program, but a class now, responsible just to drive the DUT signals to execute the addition and get the result. A second class was attached to the program, called *monitor* responsible to get the transaction from the DUT interface and check the results, to do this we connect this class, as the *driver* one too, to the DUT through the Virtual Interface get by the class as a constructor argument. It is important to remark that the checker should not be a role played by monitor, since the monitor must just monitor, but, since it is a small project, we choose this approach

Furthermore, the program called *lab_prog* is responsible to instantiate these classes and "connect" them to the respective interfaces, besides calling the *run* task from each class to perform the drive and monitor role.

Instructions

Follow instructions given in "aedv_training_labs_intructions_for_questa.pdf".
Select

● System Verilog

Open the file: <SANDBOX>/labs-Xdays/labNN-systemverilog_virtual_interface /lab.sv

Step 1: Complete the interface.

- Open the file: <SANDBOX>/labs-Xdays/labNN-systemverilog_virtual_interface /adder_if.sv
- Search for ***LAB-TODO-STEP1-a***
- Declare the *modport* for the monitor.

Step 2: Complete the driver.

- Open the file: <SANDBOX>/labs-Xdays/labNN-systemverilog_virtual_interface /lab_prog.sv
- Search for ***LAB-TODO-STEP2-a***
- Declare the virtual interface as a class attribute.

- Search for ***LAB-TODO-STEP2-b***
- Declare the constructor function.
- Search for ***LAB-TODO-STEP2-c***
- Uncomment the task content.
    - Why we must use a virtual interface instead of a common interface within the class?

Step 3: Complete the monitor.

- Open the file: <SANDBOX>/labs-Xdays/labNN-systemverilog_virtual_interface /lab_prog.sv
- Search for ***LAB-TODO-STEP3-a***
- Declare the virtual interface as a class attribute.
- Search for ***LAB-TODO-STEP3-b***
- Declare the constructor function.
- Search for ***LAB-TODO-STEP3-c***
- Wait by the clock and reset signals:

```
@(posedge this.vif.i_clk or negedge this.vif.i_rstn);
```

- Search for ***LAB-TODO-STEP3-d***
- Reset the model registers when required
- Search for ***LAB-TODO-STEP3-e***
- Set the *addRegs*

```
else if(vif.i_we) begin
    $display($psprintf("[%d] Writting transaction Addr:%2x Data:%2x ",
$realtime, vif.i_addr, vif.i_data));
    case (vif.i_addr)
        'h1: begin
            addRegs.data1       = this.vif.i_data;
            this.expec.data1    = this.vif.i_data;
        end
        'h2: begin
            addRegs.data2       = this.vif.i_data;
            this.expec.data2    = this.vif.i_data;
        end
        'h3: begin

            addRegs.cin         = this.vif.i_data;
            this.expec.cin      = this.vif.i_data;
        end
        default : $display($psprintf("**ERROR : Writting in R/O register\n
%2x ",vif.i_addr));
    endcase
end
```

- Search for ***LAB-TODO-STEP3-f***
- Call the function to compute the expected sum when the start signal is set
- Search for ***LAB-TODO-STEP3-g***
- Uncomment the remain content of the task

Step 4: Use virtual functions.

- Open the file: <SANDBOX>/labs-Xdays/labNN-systemverilog_interfaces /tb.sv
- Search for ***LAB-TODO-STEP4-a***

- Instantiate the Interface.
- Search for ***LAB-TODO-STEP4-b***
- Instantiate the DUT module.
- Search for ***LAB-TODO-STEP4-c***
- Instantiate the program

Step 5: Driver/Monitor instantiation.

- Open the file: <SANDBOX>/labs-Xdays/labNN-systemverilog_virtual_interface /lab_prog.sv
- Search for ***LAB-TODO-STEP4-a***
- Instantiate the driver object passing the interface driver *modport* as argument
- Search for ***LAB-TODO-STEP4-b***
- Perform 10 additions. Modify, if you wish to see other transactions.
- Compile and run
    - Were the signals driven? (See the waveform)
- Search for ***LAB-TODO-STEP4-c***
- Instantiate the monitor object passing the interface monitor *modport* as argument
- Search for ***LAB-TODO-STEP4-d***
- Call the run method from monitor to start monitoring
- Compile and run
    - Were the transactions displayed?