Training on
IP & SoC Functional Verification Methodology
Using UVM

LAB
Scoreboarding

Objectives

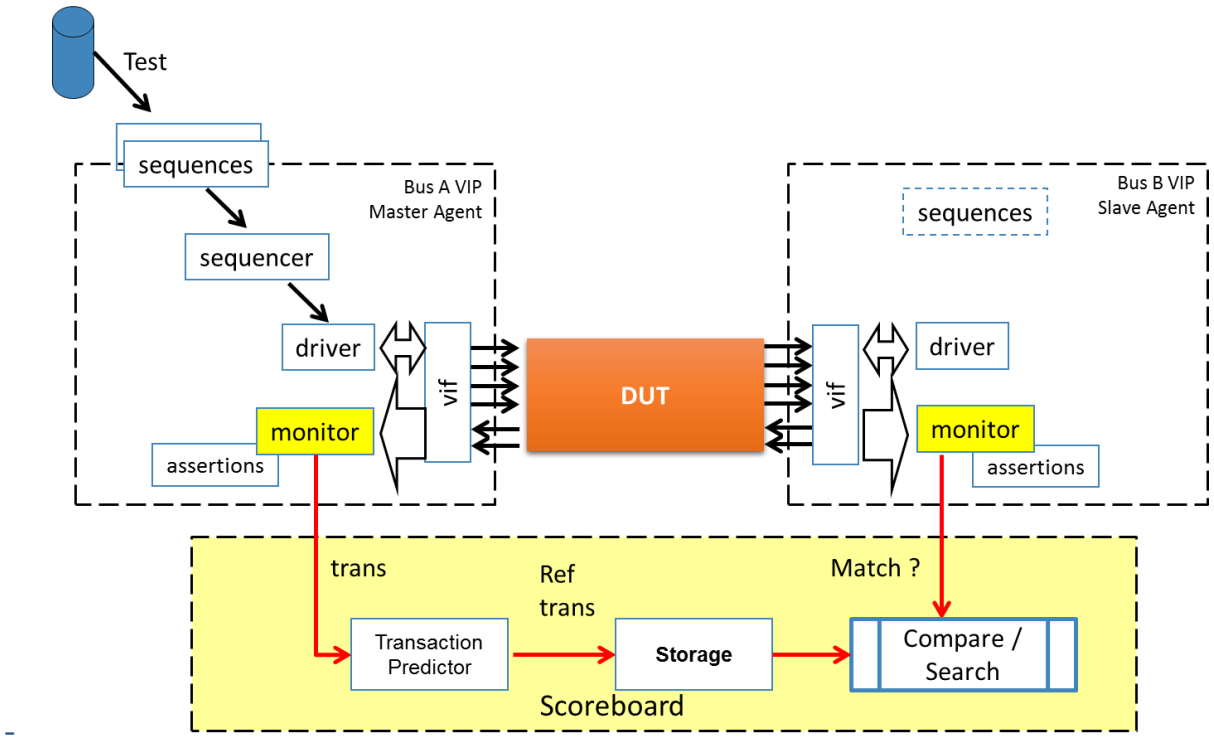
The goal of this lab is to implement a scoreboard checking mechanism between the register interface of the UART and the external serial interface of the UART on the TX line.
It focuses on:

- Understanding how the scoreboard is connected to multiple monitors
- Understanding data checking mechanisms.

Introduction

The design under test (DUT) is a UART 16550 from opencores.org.
The goal of this design is to perform serial transfers on the Tx line and receive Rx transfers from the Rx line, given registers programmable via a APB interface.

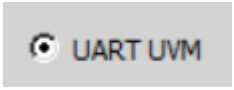
In this lab we will implement a scoreboard between the APB Register programming interface of the UART and the Tx data that are actually sent by the DUT.



Instructions

Step 0: Run a first existing test:

Follow instructions given in “aedv_training_labs_intructions_for_questa.pdf”.
Open the file “<SANDBOX>/labs-Xdays/labNN-uvm_scoreboard/lab.sv”
Select



Step 1: Implement a simple assertion to check the protocol

In the file platform/aedvices/uart_tb/uart_ip_tb.sv , the design is instantiated and connected to the signals of the APB Protocol:

```
uart_apb uart0 (

    .PCLK_i     ( apbif0.PCLK    )   ,
    .PRESETn_i  ( apbif0.PRESETn )   ,
    .PADDR_i    ( apbif0.PADDR   )   ,
    .PPROT_i    ( apbif0.PPROT   )   ,
    .PSEL_i     ( apbif0.PSEL    )   ,
    .PENABLE_i  ( apbif0.PENABLE )   ,
    .PWRITE_i   ( apbif0.PWRITE  )   ,
    .PWDATA_i   ( apbif0.PWDATA  )   ,
    .PSTRB_i    ( apbif0.PSTRB   )   ,
    .PREADY_o   ( apbif0.PREADY  )   ,
    .PRDATA_o   ( apbif0.PRDATA  )   ,
    .PSLVERR_o  ( apbif0.PSLVERR )   ,
    // Serial Signals
    .rx_i       ( uartif0.rx.sig ) ,
    .tx_o       ( uartif0.tx.sig ) ,
    .int_o      ( ) // open
  );
```

The APB protocol defines the following waveforms:

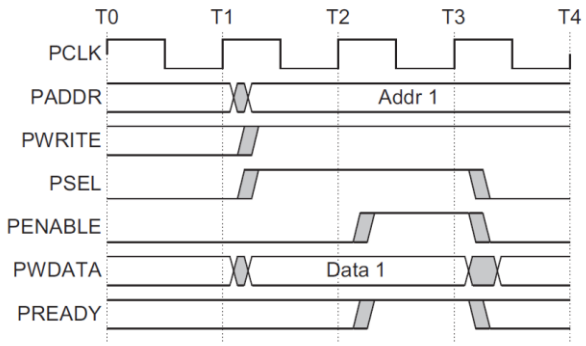


Figure 3-1 Write transfer with no wait states

In the file platform/aedvices/uart_tb/uart_ip_tb.sv, write an assertion which will check that after reset, PSEL is 0.

```
assert property (
  @(posedge apbif0.PCLK)
  apbif0.PRESETn |=> apbif0.PSEL
);
```

Using this example, write an assertion than checks that:

- A first cycle of “PSEL” == 1, is followed by PENABLE == 1.

Step 2: Connect the scoreboard

The lab_scoreboard class will hold the scoreboard mechanisms.
In order to connect the scoreboard to the register APB Verification IP and the UART verification IP, we first need to create input analysis port.

Instructions:

- Search for LAB-TODO-STEP-2-a
- Declare the input analysis port extensions:
  `uvm_analysis_imp_decl(_apb)
  `uvm_analysis_imp_decl(_tx_frame)
- Search for LAB-TODO-STEP-2-b
- Add the input analysis port members to the lab4_scoreboard class:
  uvm_analysis_imp_apb #(apb_transfer , lab_scoreboard) apb_import;
  uvm_analysis_imp_tx_frame #(uart_frame , lab_scoreboard) tx_frame_import;
- Search for LAB-TODO-STEP-2-c
- Create the 2 associated instances
- Search for LAB-TODO-STEP-2-d
- Connect the scoreboard to the VIP monitors
- Add breakpoints to:
  - write_apb() (around line 50)
- Recompile and Rerun

Step 3: Implement the TX scoreboard

In the above code, we have connected the scoreboard to the APB monitor and to the TX monitor.
We can therefore store and compare data that are coming these monitors.

As the APB interface is used for register configuration, read the status register (LCR) as well as writing to the TX FIFO of the UART, we need to filter out unwanted transactions and only store the TX data when appropriate.
Furthermore, when the UART is configure to use only characters of 5 bits, only these 5 bits should be compared.

The scoreboard should therefore perform the following:

- Detect when LCR bit 8 is written to 1→ mark the TX FIFO as disable
- Detect when LCR bit 8 is written to 0 → mark the TX FIFO as enable
- Store the character width LCR[2:0] to know how many bits to compare.
- When the TX FIFO is enable and a write occurs to the TX FIFO, store the data.
- When a UART character is received, compare with previously written characters.

Instructions

- Search for LAB-TODO-STEP-3-a
- In write_apb:
  - Set the variable “scoreboard_enable” to 0 when
    - There is a write access to address == `UART_LCR with data[7:7] == 1
  - Set the variable “scoreboard_enable” to 1 when
    - There is a write access to address == `UART_LCR with data[7:7] == 0

    - Push the transaction data to the queue “tx_scoreboard” when:
        - The scoreboard is enable
        - There is a write transaction to address 0x00
- Search for LAB-TODO-STEP-3-b
- In write_tx_frame:
    - Check the previously pushed data is matching the one you are receiving.
    - Check the first element of the queue is the received packets, else issue a UVM_ERROR using the macro:
      `uvm_error("MESSAGE_ID","SOME ERROR MESSAGE")

Debug:

- It is expected to find functional bugs in the design, what is it ?
- Can you identify and fix it ?

Notes:

- In order to tests the scoreboard with more configurations, you may replace the content of the sequence_lib.svh file with your LAB 2 development.
- In steps 3, you can use a scalar variable, you can use a SystemVerilog FIFO:

```
int my_fifo[$];
int val;
my_fifo.push_front(12);
val = my_fifo.pop_back();
```