BFM Simulation

Zynq 14.2 Version
Objectives

After completing this module, you will be able to:

- Describe the Bus Functional Model (BFM) and how it helps in verification
- Attach a custom AXI peripheral to a BFM model
- Describe the basic tenets of the BFM Verilog API tasks
- Effectively debug an AXI peripheral
- Describe the two BFM design flows
Outline

¬ Debugging Custom IP
¬ BFM Simulation
¬ BFM Simulation Stimulus
¬ BFM Simulation Flow
¬ Summary
Debugging a custom AXI peripheral
  – What is the best way to simulate a custom peripheral?
  – Functional simulation of the entire processor system takes too long

Simulating a custom AXI peripheral
  – What is the best method to generate an AXI interface stimulus?
  – Perform read and write bus transactions

Peripheral debugging learning curve needs to be short

Two XPS BFM tool flows
Typical Bus Peripherals

Three components
- Bus interface
- IPIC Service Attachment
- User Design
Methods for Verifying Bus-Based Device Functionality

» Testbench-driven verification
  – Create a testbench that describes extensive (all different) combinations of bus transactions
    • Very time-consuming

» System verification
  – Simulate the entire system using a HDL simulator including the processor, memory, and all bus peripherals
    • Application software code is executed to provide stimulus

» Bus Functional Model (BFM) simulation
  – Useful for a bus-based device
  – Provides the ability to generate bus stimulus without needing the above two methods
Outline

- Debugging Custom IP
- BFM Simulation
- BFM Simulation Stimulus
- BFM Simulation Flow
- Summary
What is BFM Simulation?

- BFM simulation is a type of simulation that uses bus functional models to debug and verify hardware components that have a bus interface.
- BFM simulation is a quick and easy way to simulate a sub-section of the design without needing to simulate an entire processor system.
- BFMs provide the capability for easily generating, responding to, and analyzing bus transactions.
- Models and monitors bus-level transactions.
- BFMs verify that hardware bus components meet bus architecture specifications.
- Stimulus generated in a Verilog testbench file.
- Does not require processor boot sequence.
- Typically *much* faster than full simulation.
- Can easily be added into a full system design, removing the complexities of the processor block.
The AXI BFM solution is an optional product that is purchased separate from the ISE Design Suite software. Licensing is handled through the standard Xilinx licensing scheme.

A license feature, XILINX_AXI_BFM, is required in addition to the standard ISE license features. A license is checked out at simulation run time. While the Xilinx ISE software does not need to be running while the AXI BFM solution is in use, the AXI BFM only operates on a computer that has the Xilinx software installed and licensed.

For more information:

- View AXI Bus Functional Model Data Sheet http://www.xilinx.com/support/documentation/ip_documentation/cdn_axi_bfm/v3_00_a/ds824_axi_bfm.pdf
Why Platform Studio BFM?

- EDK can automatically build a BFM simulation project
  - MHS-defined system
  - No need to write HDL
- Replace any number of XPS IP components with “wrapped” BFM models or add new to your MHS
- Compilation is very quick
- Simulation time reduced significantly
Xilinx Platform Studio BFM Package

- Collection of bus functional models, Verilog API tasks, and documentation
- Capable of specifying bus connections via the MHS file
- Time and effort required to set up the testing environment is reduced
- Requires an additional license
- Requires use of Verilog testbench
  - Top-level peripheral design can be either Verilog or VHDL
- Simulation support via the ISim and ModelSim simulators

IP Catalog
Replace rest of the processor system with Master BFM agent

Allows focus on bus transactions and bus peripheral operation rather than entire system

Can monitor many aspects of the system.
  – Bus latency
  – Bridge transactions
  – Arbitration schemes
  – Component interaction

BFL simulation syntax can be written to mimic specific cases of any type of bus operation

Can often speed up simulation by over 100x
Slave IP verification use case
- Master BFM allows slave to be read/written
- Monitor BFM checks to ensure that all transactions are legal on the AXI interface
- BFL provides the stimulus instructions to the simulator

Master IP verification use case
- Slave BFM allows master to read/write
- Monitor BFM checks to ensure that all transactions are legal on the AXI interface
- BFL sets up the slave BFM responses
- Can also use standard EDK peripherals (block RAM or DDR SDRAM, for example)
Outline

- Debugging Custom IP
- BFM Simulation
- BFM Simulation Stimulus
- BFM Simulation Flow
- Summary
Bus Functional Stimulus

- Bus functional stimulus is provided by Cadence
- Provided as Verilog API tasks
  - Requires that top-level testbench is Verilog based
  - Top-level DUT can be Verilog or VHDL
- General command form
  - Task_command(parameters)
    - parameters can be one or more parameter=value pairs, separated by a comma
    - parameters can be specified in any order
    - Value can be scalar or enumerated type (string)
- Commands can be categorized as the following types
  - Read address channel
  - Write address channel
  - Read data channel
  - Write data channel
  - Write response channel
Set of Verilog tasks organized by interconnect type
- AXI3
- AXI4
  - Full
  - Lite
  - Streaming
Similar master and slave routines for each interface
Layered interface
- Configuration
- Function API
- Channel API
- Signal interface
AXI transaction tasks are issued separately for each of the five AXI channels

Various parameters are defined
- Channel address/data width
- User and bus ID width
- Maximum outstanding transactions allowed
- Various error and information messaging

Some channel command API tasks
- SEND_WRITE_ADDRESS
- SEND_WRITE_DATA
- RECEIVE_WRITE_RESPONSE
- SEND_WRITE_BURST

Refer to documentation for detailed explanation of enriched commands
Simple AXI single data phase write transaction
SEND_WRITE_ADDRESS('h30000000,3'b000);
SEND_WRITE_DATA(4'b1111, 'h12345678);
RECEIVE_WRITE_RESPONSE(response);

Simple AXI single data phase read transaction
SEND_READ_ADDRESS('h40000000,3'b000);
RECEIVE_READ_DATA(tb_rd_data, tb_rsp);

In the actual Verilog testbench, a complete path to an API task is required
– This path can vary with tool version and implementation
dut.bfm_processor.bfm_processor.cdn_axi4_lite_master_bfm_inst.SEND_WRITE_ADDRESS('h30000000,3'b000);
The default Verilog testbench is `bfm_system_tb.v`
- Auto-generated by SimGen in the `..\dev\bfmsim\simulation\behavioral` directory
- Will contain a DUT instance containing all user-defined signals
- Clock stimulus is provided
- User-defined signals must be added
- User-added reset and AXI transaction stimulus must be added
Outline

- Debugging Custom IP
- BFM Simulation
- BFM Simulation Stimulus
- BFM Simulation Flow
- Summary
Two approaches

– Create a peripheral device by using the Create and Import Peripheral Wizard and select the Generate BFM Simulation Platform option
  • Add additional peripherals which can interact with the created peripheral

– Create a blank project in XPS*
  • Use BFM models and create a system
  • Add the peripheral to be tested

* The Create or Import Peripheral Wizard provides the project
Start the wizard and select the appropriate peripheral

Follow the steps of the peripheral creation and select the Generate BFM Simulation Platform option
Locating the Generated BFM Project

- Go to the directory where the peripheral was created
  - Either the `<current project directory>`\pcores or repository directory
- Go to the `hdl\vhdl` directory
  - This is the same file used to hook your IP to the IPIC
- Edit the `user_logic.vhd` file to incorporate the peripheral functionality
  - Instantiate the top level of your user logic
- Go to the `devl\bfmsim` directory under the created peripheral directory
- Double-click `bfm_system.xmp` to start XPS
Launching the Generated BFM Project

- XPS project under bfmsim directory already includes user IP
- Add any other peripheral to the system if desired
- Make ports external if needed
- Modify the HDL testbench file, bfm_system_tb.v, in the ..\dev\bfmsim\simulation\behavioral directory to include custom defined parameters and ports
- Select XPS tool properties for the desired simulator
- Follow the simulation steps described next
MHS File Snippet

PORT sys_reset = sys_reset, DIR = I, SIGIS = RST
PORT sys_clk = sys_clk, DIR = I, SIGIS = CLK, CLK_FREQ = 100000000
PORT led_ip_inst_LED_pin = led_ip_inst_LED, DIR = O, VEC = [7:0]

BEGIN cdn_axi4_lite_master_bfm_wrap
PARAMETER INSTANCE = bfm_processor
PARAMETER HW_VER = 2.01.a
BUS_INTERFACE M_AXI_LITE = axi4lite_bus
PORT M_AXI_LITE_ACLK = sys_clk
END

BEGIN axi_interconnect
PARAMETER INSTANCE = axi4lite_bus
PARAMETER HW_VER = 1.06.a
PARAMETER C_INTERCONNECT_CONNECTIVITY_MODE = 0
PORT INTERCONNECT_ARESETN = sys_reset
PORT INTERCONNECT_ACLK = sys_clk
END

BEGIN led_ip
PARAMETER INSTANCE = led_ip_inst
PARAMETER HW_VER = 1.00.a
PARAMETER C_BASEADDR = 0x30000000
PARAMETER C_HIGHADDR = 0x3000ffff
BUS_INTERFACE S_AXI = axi4lite_bus
PORT S_AXI_ACLK = sys_clk
PORT LED = led_ip_inst_LED
END

DUT as a slave
BFM Simulation Setup Using SimGen

- From XPS, select Edit > Preferences and select Simulation
  - Select the ISIM simulator
- Select Project > Project Options and select behavioral simulation in the Design Flow tab
- Select Simulation > Generate Simulation HDL Files
- Develop a simulation testbench file (e.g., *bfm_system_tb.v* – as previously described) in the ..\simulation\behavioral directory
- Click the User Commands button ( ) in XPS
  - Launches and runs the simulator
    - Compiles all HDL files and use DO files
Outline

» Debugging Custom IP
» BFM Simulation
» BFM Simulation Stimulus
» BFM Simulation Flow
» Summary
A BFM package includes BFM models for AXI4 and AXI4-Lite

BFM simulation allows verification of bus-based master/slave device functionality

BFM simulation is fast and does not involve simulating all of the devices in a complete system

Traditional system simulation is a more complete simulation than BFM, but requires a lot of calendar time to run because the entire system is being simulated

Verilog API tasks generate AXI transactions and monitor responses

The Cadence-provided Verilog API tasks provide for system configuration via the use of parameters and layered tasks for channel and function operations