AXI Bus Functional Model v1.1

PB001 December 14, 2010 **Product Brief**

Introduction

The AXI Bus Functional Models (BFMs), developed for Xilinx by Cadence Design Systems, support the simulation of customer-designed AXI-based IP. AXI BFMs support all versions of AXI (AXI3, AXI4, AXI4-Lite and AXI4-Stream). The BFMs are delivered as encrypted Verilog modules. BFM operation is controlled via a sequence of Verilog tasks contained in a Verilog-syntax text file. The API for the Verilog tasks is described in the AXI BFM User Guide.

Features

- Supports all protocol data widths and address widths, transfer types and responses
- Transaction level protocol checking (burst type, length, size, lock type, cache type)
- Behavioral Verilog Syntax
- Verilog Task-based API
- Delivered in ISE, enabled by a Xilinx-generated license

Overview

This chapter provides a high level, architectural overview of the general AXI BFM structure. It also shows how the AXI BFMs fit into an overall test environment.

The general AXI BFM architecture is shown in Figure 1.

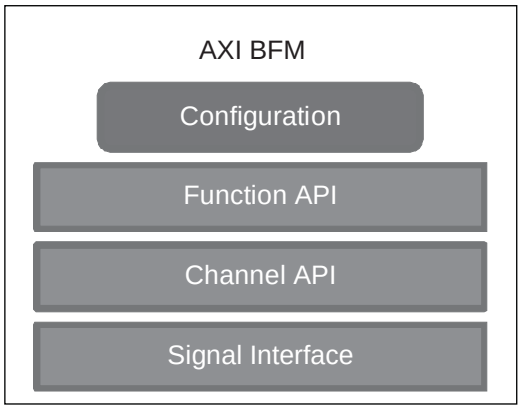

Figure 1: **AXI BFM Architecture**

All of the AXI BFMs consist of three main layers: the signal interface, the channel API and the function API. The signal interface includes the typical Verilog input/output ports and associated signals. The channel API is a set of defined Verilog tasks that operate at the basic transaction level inherent in the AXI protocol, including:

- Read Address Channel
- Write Address Channel
- Read Data Channel
- Write Data Channel
- Write Response Channel

This split enables the tasks associated with each channel to be executed concurrently or sequentially. This allows the test writer to control and implement out of order transfers, interleaved data transfers, and other features.

The next level up in the API hierarchy is the function level API. This level has complete transaction level control; for example, a complete AXI read burst process is encapsulated in a single Verilog task.

One final but important piece of the AXI BFM architecture is the configuration mechanism. This is implemented using Verilog parameters and/or BFM internal variables and is used to set the address bus

width, data bus width and other parameters. The reason Verilog parameters are used instead of defines is so that each BFM can be configured separately within a single test bench. For example, it is reasonable to have an AXI master that has a different data bus width than one of the slaves it is attached too (in this case the interconnect needs to handle this). BFM internal variables are used for configuration variables that maybe changed during simulation. For a complete list of configuration options, see UG783, *AXI Bus Functional Model User Guide*.

The intended usage of the AXI BFM is shown in Figure 2.

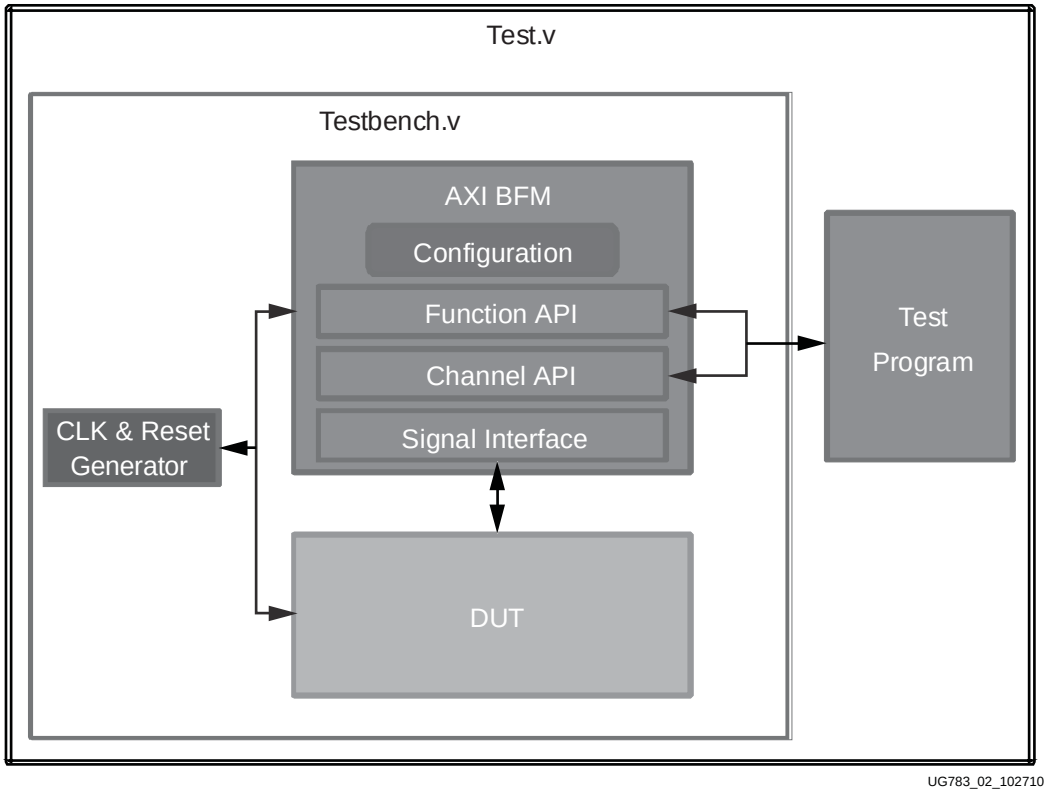

Figure 2: **AXI BFM Usage**

Figure 2 shows a single AXI BFM. However, the test bench can contain multiple instances of AXI BFMs. The DUT and the AXI BFMs are instantiated in a test bench that also contains a clock and reset generator. Then, the test writer instantiates the test bench into the test module and creates a test program using the BFM API layers. The test program would call API tasks either sequentially or concurrently using fork and join. See see UG783, *AXI Bus Functional Model User Guide*, for practical examples of test programs and test bench setups.

Support

Xilinx provides technical support for this product when used as described in the product documentation. Xilinx cannot guarantee functionality or support of this product not defined in the documentation, if modified in a way not described in the product documentation, or if changes are made to any section of the design labeled *DO NOT MODIFY*.

Ordering Information

The AXI Bus Functional Model is provided under the Xilinx Software End User License Agreement. A full license for the model must be purchased and obtained from Xilinx.

Please contact your local Xilinx sales representative for pricing and availability of additional Xilinx modules and software. Information about additional Xilinx solutions is available on the Xilinx IP Center.

Revision History

The following table shows the revision history for this document:

Date	Version	Description of Revisions
12/14/2010	1.0	Initial Xilinx release.

Notice of Disclaimer