Verification of AMBA based AXI 4 Slave Interface

Krithi B1, Sudarshan Bhat2, Yogesh Panchaksharaiah3

1Final year M.Tech (VLSI Design & Embedded Systems) Student, Department of E&CE, Mangalore Institute of Technology and Engineering, Mangalore, Karnataka, India
2Associate Professor, Department of E&CE, Mangalore Institute of Technology and Engineering, Mangalore, Karnataka, India
3Technical Manager, Mindtree Ltd, Bangalore, Karnataka, India

Abstract— The increasing amount of logic that can be placed onto a single silicon die is driving the development of highly integrated SoC designs. An important feature for any of the SoC is based on how they interconnect. AMBA protocols are today the de facto standard for 32-bit embedded processors because they are well documented and can be used without royalties. The AMBA AXI 4 protocol supports high-performance, high-frequency system designs. It is suitable for high-bandwidth and low-latency designs and provides high-frequency operation without using complex bridges. It provides flexibility in the implementation of interconnect architectures and is backward-compatible with existing AHB and APB interfaces. This Project is aimed at the Verification of the AMBA based Design of the AXI4 Slave Interface and the Verification Environment is built using SystemVerilog coding. This slave interface can be used to connect different peripherals into AMBA based processors without using bridge. The developed slave interface can also be used to connect different peripherals like SPI, I2C, UART etc., into non AMBA based processors by developing wrapper around AXI4 slave interface.

Keywords— Advanced Microcontroller Bus Architecture (AMBA), Advanced Peripheral Bus (APB), AMBA High-performance Bus (AHB), Advanced Extensible Interface (AXI).

I. INTRODUCTION

The Advanced Extensible Interface (AXI) is a part of the Advanced Microcontroller Bus Architecture (AMBA) which is developed by ARM (Advanced RISC Machines) company. It is an On-Chip communication protocol. The AMBA AXI protocol supports high-performance, high-frequency system designs. The AXI protocol is suitable for high-bandwidth and low-latency designs. It provides high-frequency operation without using complex bridge. It meets the interface requirements of a wide range of components. AXI protocol provides flexibility in the implementation of interconnect architectures. It is backward-compatible with existing AHB and APB interfaces.

The key features of the AXI protocol are that it has separate address/control and data phases & support for unaligned data transfers, using byte strobes. It utilizes burst-based transactions with only the start address issued. It has separate read and write data channels that provide low-cost Direct Memory Access (DMA). It supports for issuing multiple outstanding addresses. It support for out-of-order transaction completion. It permits easy addition of register stages to provide timing closure.

II. LITERATURE REVIEW

The Advanced Microcontroller Bus Architecture (AMBA) is an open-standard, on-chip interconnect specification for the connection and management of functional blocks in system-on-a-chip (SoC) designs. It facilitates development of multi-processor designs with large numbers of controllers and peripherals. Since its inception, the scope of AMBA has, despite its name, gone far beyond micro controller devices. Today, AMBA is widely used on a range of ASIC and SoC parts including applications processors used in modern portable mobile devices like smartphones. AMBA was introduced by ARM in 1996. The first AMBA buses were Advanced System Bus (ASB) and Advanced Peripheral Bus (APB) [1&2]. In its second version, AMBA 2, ARM added AMBA High-performance Bus (AHB) that is a single clock-edge protocol [3-5]. In 2003, ARM introduced the third generation, AMBA 3 [6&7], including AXI to reach even higher performance interconnect and the Advanced Trace Bus (ATB) as part of the CoreSight on-chip debug and trace solution. In 2010 the AMBA 4 specifications were introduced starting with AMBA 4 AXI4, then in 2011 extending system wide coherency with AMBA 4 ACE with a re-designed high-speed transport layer and features designed to reduce congestion [8-10]. Advanced microcontroller bus architecture (AMBA) protocol family provides metric-driven verification of protocol compliance, enabling comprehensive testing of interface intellectual property (IP) blocks and system-on-chip (SoC) designs. The AMBA advanced extensible interface 4 (AXI4) update to AMBA AXI3 includes the following: support for burst lengths up to 256 beats, updated write response requirements, removal of locked transactions and AXI4 also includes information on the interoperability of components. AMBA AXI4 protocol system supports 16 masters and 16 slaves interfacing. The design is implemented using Verilog- HDL [11-13].

III. AXI PROTOCOL SPECIFICATIONS

A typical system consists of a number of master and slave devices connected together through the Interconnect. The AXI protocol provides a single interface definition, for the interfaces:

- Between a master and the interconnect
• Between a slave and the interconnect
• Between a master and a slave.

The AXI protocol is burst-based and defines the following independent transaction channels: Read Address Channel, Read Data Channel, Write Address Channel, Write Data Channel and the Write Response Channel.

An address channel carries control information that describes the nature of the data to be transferred. The data is transferred between master and slave using either:

A write data channel to transfer data from the master to the slave. In a write transaction, the slave uses the write response channel to signal the completion of the transfer to the master.

A read data channel to transfer data from the slave to the master.

The AXI protocol permits address information to be issued ahead of the actual data transfer. It supports multiple outstanding transactions. It also supports out-of-order completion of transactions.

The basic components of the verification environment are described below:

1. **Testbench**
   Testbench mimics the environment in which the design resides. It checks whether the RTL implementation meets the design specification or not. This environment creates invalid and unexpected as well as valid and expected conditions to test the design.

2. **Test Cases**
   Test cases are written for different scenarios, which cover the functionality, corner cases. Basic read write transactions, various burst modes are verified. Parameterization will be randomized for different test cases and are written to check all the possible scenarios. These test cases are run in regression with multiple seeds.

3. **DUT**
   The verification environment is organized in a hierarchical layered structure which helps to maintain and reuse it with the DUT.

4. **Virtual Interface**
   Virtual interfaces provide a mechanism for separating abstract models and test programs from the actual signals that make up the design. It allows the same subprogram to operate on different portions of design, and to dynamically control the set of signals associated with the subprogram. Instead of referring to the actual set of signals directly, users are able to manipulate a set of virtual signals.

5. **Scoreboard**
   The scoreboard collects write address and control information on address channel, and data on write data channel at the output of AXI slave and compares it with the address, data and control information at the output of DUT.
It collects the responses at the DUT side and compares it with that on the input side. Also, it collects read address and control information on address channel at the AXI slave input side and compares it with the address and control information at the output of the DUT. It collects the read response and data at the DUT side and compares it with that on the input side.

The IP architecture for the AXI 4 Verification is shown in Fig 4-2.

---

**ACKNOWLEDGMENT**

The authors wish to thank Mindtree Limited. This work was supported by a grant from Mindtree Limited.

**REFERENCES**


