Development of Verification Environment for AXI Bus Using SystemVerilog

Xu Chen, Zheng Xie, and Xin-An Wang
Key Lab of Integrated Micro-Systems Science Engineering and Applications
Peking University Shenzhen Graduate School, Shenzhen, China
Email: chenxu211@gmail.com; anxinwang@pku.edu.cn

***Abstract*—System-on-a-Chip (SoC) design has become more and more complexly. How to verify a design effectively has become a serious challenge. In this paper, how to build up the effective verification environment of AXI using SystemVerilog is introduced. Firstly, the design under verify (DUV) AXI bus is introduced. Then a comprehensive analysis of the verification plan has been made according to the protocol. The proposed integrated verification environment with Functional coverage, score-boarding, assertions and constrained random vectors generation is implemented. With this environment, a high coverage and less time spending verification has been achieved.**

***Index Terms*—SystemVerilog, SoC, AXI, Verification Environment**

I. Introduction

Verification has become the dominant cost in the design process. On current projects, verification engineers outnumber designers, with this ratio reaching two or three to one for the most complex designs [1]. So an effective verification environment is needed.

As a set of extensions to the Verilog HDL, SystemVerilog gains the advantage of OOP, which can play a role of hardware description language as well as hardware verification language [2]. It can conveniently describe design functions and scenarios, which is considerably suitable and effective for verification engineer.

In this paper, how to build up the verification environment of AXI bus using SystemVerilog is introduced. Functional coverage, score-boarding, assertions and constrained random vectors generation is implemented with the proposed integrated verification environment.

The remaining part of the paper is organized into the following sections. In section II, a brief introduction of verification plan of the AXI bus is described, while section III addresses the proposed verification environment in details. The verification results are presented in section IV and summaries are drawn in section V.

Manuscript received March 19, 2013; revised May 25, 2013.

II. The Verification Plan

The AMBA AXI protocol is targeted at high-performance, high-frequency system designs and includes a number of features that make it suitable for a high-speed submicron interconnect [3]. The SoC architecture is shown in the Fig. 1. It is an AMBA based SoC that include the AXI bus with two masters and three slaves.

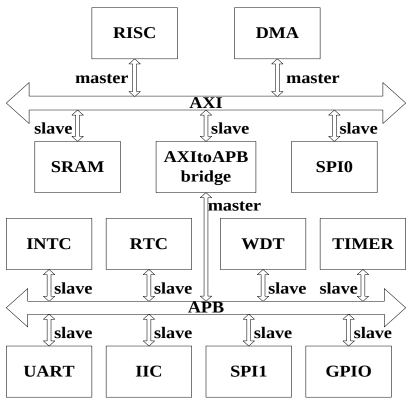

Figure 1. SoC architecture

Following the specification of the AXI bus, the functional verification plan of AXI in this paper include:

Two kinds of address mode: aligned and unaligned.

Three kinds of burst type: FIXED, INCR and WRAP.

Sixteen choices of burst length in the range of 1-16.

Several choices of burst size according to the data bus width.

Four kinds of response types: OKAY, EXOKAY, SLVERR and DECERR.

Priorities between two masters and three slaves.

The test plan or verification plan acts as a coverage model describing the functionality of the design to be covered through its individual test points which serve as coverage points in that coverage space [4].

III. The Verification Environment

The verification environment is shown in Fig. 2. This environment is organized in a hierarchical layered structure which helps to maintain and reuse it with different designs under verify, following the recommendations show in [5]. The following subsections explain the functionality of four primary components in the verification environment.

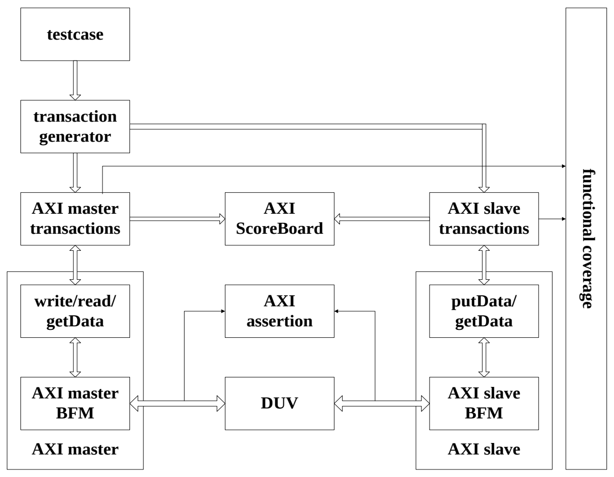

Figure 2. The testbench architecture

A. AXI Master

The AXI master mainly contains two classes, AXI_m_busBFM and AXI_m_env. The AXI master BFM is implemented by the AXI_m_busBFM class, and the AXI_m_env defines the commands of write, read and getData. The AXI_m_busBFM class has a main task named startBFM which starts one loop for each channel parallelly(AXI has five separated channels). Every loop has the similar function, which is getting the transaction data and calling the task which generates the channel timings. The relationship of these tasks is shown in Fig. 3. And the timing control of the AXI master BFM is generated randomly.

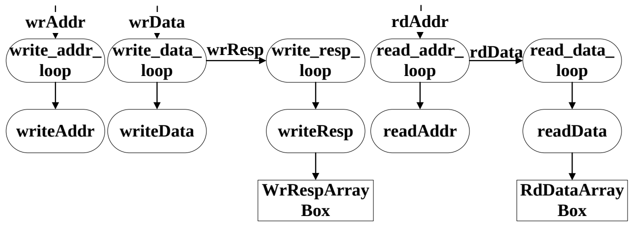

Figure 3. The relationship of tasks in AXI_m_busBFM

The AXI master receives high-level data through the commands AXI_m_env defines and breaks into signals that are sent to the DUV through the interfaces with the help of AXI_m_busBFM.

B. AXI Slave

The AXI slave is built up with an AXI_s_busBFM class and an AXI_s_env class. The AXI slave BFM is implemented by the AXI_s_busBFM class, and the AXI_s_env defines the commands of putData, getData and setWr/RdResp. The AXI_s_busBFM class also has a startBFM task which starts seven loops to simulate the AXI slave behavior. Fig. 4 shows the data flow of the seven loops in AXI_s_busBFM. And the timing control of the AXI slave BFM is generated randomly just the same as that of the master BFM.

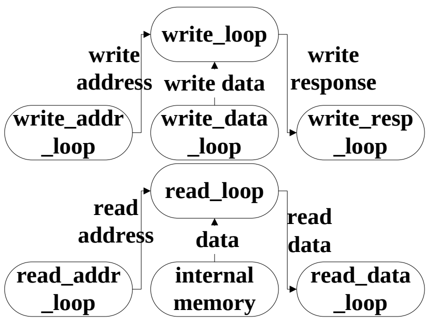

Figure 4. The data flow in AXI_s_busBFM

The AXI slave receives the control/data information from the DUV according to the AXI protocol, and then puts the write data to the internal memory or gets the read data from the memory. The AXI slave internal memory can be set by using the command putData, and the write/read responses can also be set with the command setWr/RdResp.

C. AXI Assertion

SystemVerilog assertion (SVA) is a declaration for the expected behavior. In this paper, assertions are applied to detect whether the signals are correct according to the AXI protocol. For example, a write transaction with burst type WRAP must have an aligned address can be expressed as presented in Fig. 5.

```
 property AXI_ERRM_AWADRR_WRAP_ALIGN;
   @(posedge `AXI_SVA_CLK)
        !($isunknown({AWVALID,AWBURST,AWADDR})) &
        AWVALID & (AWBURST == `AXI_ABURST_WRAP)
     |-> ((AWADDR[6:0] & AlignMaskW) == AWADDR[6:0]);
 endproperty
 axi_errm_awadrr_wrap_align: assert property
(AXI_ERRM_AWADRR_WRAP_ALIGN) else
  $error("AXI_ERRM_AWADDR_WRAP_ALIGN. ");
```

Figure 5. An example of assertion in the testbench

D. AXI Scoreboard

The AXI scoreboard realizes the comparisons between the address and data that an AXI master writes/reads in a burst and the address and data in the slave internal memory. The components in the scoreboard to buffer the comparison data are the write/read mailboxes. Furthermore, the transactions with the same ID must arrive in the same order that they are issued according to the AXI protocol and the transactions with different IDs can arrive in any order, so one mailbox is used for only one AWID or ARID.

When an AXI master transmits a write burst, it will be stored in the write mailbox with the same id as the AWID. When the AXI slave receives the write burst, it will be compared with the one popped from the write mailbox mentioned above. The read mailbox is used in the similar way when AXI master transmits a read burst.

E. Simulation Result

Fig. 6 shows a part of the simulation result when a write burst has been transmitted. A write burst is transmitted from master 0. And then, the 12 bytes data should be put into the internal memory in slave 0 from address 99cc to 99d7. If the 12 bytes data from master 0 and these in the internal memory of slave 0 are the same after master 0 receives the OKAY write response, the write burst transmittance is considered to be successful.

```
Master 0 has transmitted a write burst
 ID       : 3
 ADDR      : 99cc
 burst_len  : 5
 burst_size : 1
 burst_type : INCR
 The write data :
 0 : 6   1 : a5   2 : 9a   3 : 69   4 : 44   5 : ae   6 : f0   7 : dc   8 : cf   9 : 27  10 : 38  11 : 24

Slave 0 has received a write burst
 ID       : 3
 ADDR      : 99cc
 burst_len  : 5
 burst_size : 1
 burst_type : INCR
 The data in the slave from addr 99cc  :
 0 : 6   1 : a5   2 : 9a   3 : 69   4 : 44   5 : ae   6 : f0   7 : dc   8 : cf   9 : 27  10 : 38  11 : 24

Master 0 has received an OKAY write response

The write burst has transmitted successfully!
```

Figure 6. A part of the simulation result

IV. SUMMARY

In this paper, an effective verification environment for AXI bus is developed with SystemVerilog. The proposed multi-layer testbench is comprised of AXI master, AXI slave, assertions, scoreboard and coverage analysis. With the help of the components mentioned above, the verification environment can simulate most cases of the AXI signal, check all the transmitted data automatically and complete coverage analysis during the simulation. So the environment can improve the coverage and reduce the time spending in the verification.

REFERENCES

[1] L. Tao, X. Tong, Z. Yang, L. Huawei, and L. Xiaowei, "Bug analysis and corresponding error models in real designs," in *IEEE International High Level Design Validation and Test Workshop,* 2007, pp. 59-64.

[2] K. Han, Z. Deng, and Q. Shu, "Verification of AMBA bus model using systemverilog," in *Proc. 8th International Conference on Electronic Measurement and Instruments*, *2007*, pp. 1-776-1-780.

[3] AMBA AXI Protocol Version: 2.0 Specification, ARM Ltd, pp. 1-1

[4] A. Hazra, A. Banerjee, S. Mitra, P. Dasgupta, P. P. Chakrabarti, and C. R. Mohan, "Cohesive coverage management for simulation and formal property verification," in *IEEE Computer Society Annual Symposium on VLSI,* 2008, pp. 251-256.

[5] C. Spear, "A Guide to Learning the Testbench Language Features," in *SystemVerilog for Verification,* 2nd ed., Springer Publishing Company, Incorporated, 2008, pp. 11-18.

Xu Chen received the B.S. degree from The School of Electronics Engineering and Computer Science, Peking University, Beijing, in 2010. He is currently pursuing the master degree in school of electronics engineering and computer science, Peking University. He is now engaged in the verification methodology at school of electronic and computer engineering, Shenzhen.

His current research interests include IC design and verification.

Zheng Xie received the B.S. degree from Kunming University of Science and Technology, Kunming, Yunnan, in 2008. He is currently pursuing the Ph.D. degree in school of electronics engineering and computer science, Peking University. He is now engaged in the operator design and verification methodology and verification expert system (VES) at school of electronic and computer engineering, Shenzhen.

His current research interests include IC design and verification, computer-aided design tool development.

Xin-an Wang is the professor of Peking University. He is the vice-president of the school of electronic and computer engineering, Peking University.

His current research interests include IC design, health monitoring and food safety inspection.