www.ijecs.in

International Journal Of Engineering And Computer Science ISSN:2319-7242

Volume 2 Issue 6 June , 2013 Page No. 2037-2040

COVERAGE DRIVEN VERIFICATION IP AMBA AXI4-LITE

Manjunatha A S[1], Narayana Swamy J C[2]

[1]Electronics and Communication Department, Bangalore Institute of Technology,
Bangalore, India
manju.as50@gmail.com

[2]Electronics and Communication Department, Bangalore Institute of Technology,
Bangalore, India
jcnswamy@gmail.com

Abstract: ***As the technology improving, the system-on-chip designs become more and more complex. For the verification of these complex SoC designs, coverage metrics and the responses to them guide the entire verification flow. As like the designs are moving towards reusable and portable environments, the verification components and verification environments should also supports reusability. Hence there is a need for standalone, pre-verified and built-in verification infrastructure, which can be easily plugged in the verification environment. The Verification Intellectual Property (Verification IP) is an integral and important component of such infrastructure, which aids designers and verification engineers in the task of validating the functionality of their design. The developed Verification IP, in this paper provides the ability to verify a design against the requirements of a standard AMBA AXI4-Lite protocol. It includes the functional coverage models to track back the verification progress. This verification IP will help engineers to quickly create verification environment and test their AXI4-Lite master and slave devices.***

Keywords: AXI4-Lite, Coverage Driven Verification, Verification IP, Functional Coverage**.**

1. Introduction

Verification is the biggest challenge in the design of system-on-chip (SoC) devices and reusable IP blocks. It is very difficult to use the traditional verification methods for these complex designs. The only way to address this problem is to adopt a reuse-oriented, coverage-driven verification methodology built on the standard verification language and by using Verification IP's. The Verification Intellectual Property (Verification IP) is the verification model used in all levels of simulation-based verification. These are based on standard protocols used in computer, networking and system designs, such as PCI/PCIx, AMBA, and Ethernet. These components are pre-verified to the standard protocols and contain the necessary infrastructure for testbench generation, checking mechanisms and all the appropriate routines to create individual protocols, commonly known as Bus Functional Models (BFM).

These Verification IP components provide enhanced productivity to the system and ASIC designers by reducing the time to create the verification infrastructure and testbench environment including the required models. The verification IP based on system verilog Language allows both verification and design teams to quickly and easily create random scenarios. They also allow users to easily create directed test scenarios and test sequences for their designs. These test cases greatly aid users in finding functional bugs early in design cycle, hence reducing the overall verification time. Every design verification team requires coverage metrics to track the verification progress. VIP should include functional coverage metrics to ensure that all corner cases of the protocol have been verified.

Functional coverage is the measure of how much functionality of the design has been exercised by the verification environment. The AXI4-Lite Verification IP implemented here solves the time to market issue and reduces the efforts required by the verification engineers.

2. AMBA AXI4 Architecture

AXI4 is the advanced generation of the AMBA protocol. It is the latest revision of the AXI protocol. Functionality has been added and several known issues in AXI3 have been addressed to ensure that AMBA busses remain the dominant standard in SoC connectivity. Several key points of AXI4 protocol are as follows,

- Unaligned data transfers and up-dated write response requirements.
- Variable-length bursts, from 1 to 16 data transfers per burst.
- A burst with a transfer size of 8, 16, 32, 64, 128, 256, 512 or 1024 bits wide is supported.
- Updated AWCACHE and ARCACHE signaling details.
- Maximum burst length has been increased from 16 to 256 transfers for certain types of bursts.
- Additional quality-of-service signal has been added.

The channels defined by the AXI4 protocol between master and slave devices are as shown in below figure.

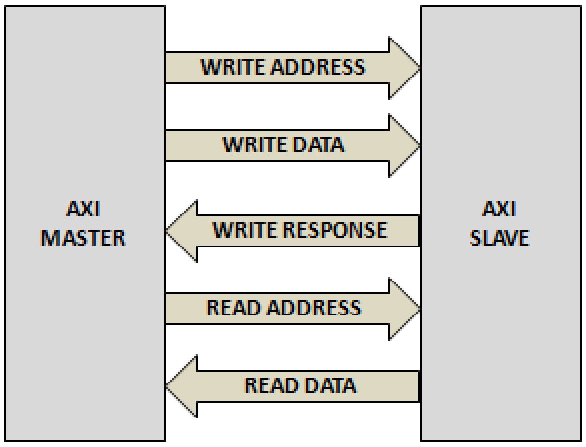


Figure 1: Various channels between master and slave components

The protocol includes five different channels between master and slave components. All channels define the several set of signals which facilitates the transfer between the devices.

Each transaction is burst-based which has addressed and control information on the address channel that describes the nature of the data to be transferred. The data is transferred between the master and slave using a write data channel to the slave or a read data channel to the master.

2.1 AXI4-Lite

The AXI4-Lite protocol is a subset of the AXI4 protocol intended for communication with simpler, smaller control register-style interfaces in components. The key features of the AXI4-Lite interface are:

- All transactions are of burst length one.
- All data accesses are the same size as the width of the data bus. AXI4-Lite supports a data bus width of 32-bit or 64-bit.
- All accesses are Non-modifiable, Non-buffer able.
- Exclusive accesses are not supported.

Below Table 1 shows the various signals in the AXI4lite protocol.

Table 1: Signals used in AMBA AXI4-Lite protocol

GLOBAL	WRITE ADDRESS CHANNEL	WRITE DATA CHANNEL	WRITE RESPONSE CHANNEL	READ ADDRESS CHANNEL	READ CHANNEL
ACLK	AWVALID	WVALID	BVALID	ARVALID	RVALID
ARESETn	AWREADY	WREADY	BREADY	ARREADY	RREADY
-	AWADDR	WDATA	BRESP	ARADDR	RDATA
-	AWPROT	WSTRB	-	ARPROT	RRESP

3. AXI4-Lite Verification IP

The AXI4-Lite Verification IP described here in this paper is a solution for verification of AXI4-Lite master and slave devices. Below Fig2 shows the verification environment of AXI4-Lite Verification IP. A Verification IP defines the environment for simulating and testing the logical functionality of a given protocol and connects at the physical (logic) levels to the design under test. These models may handle timing checks for input/output signals.

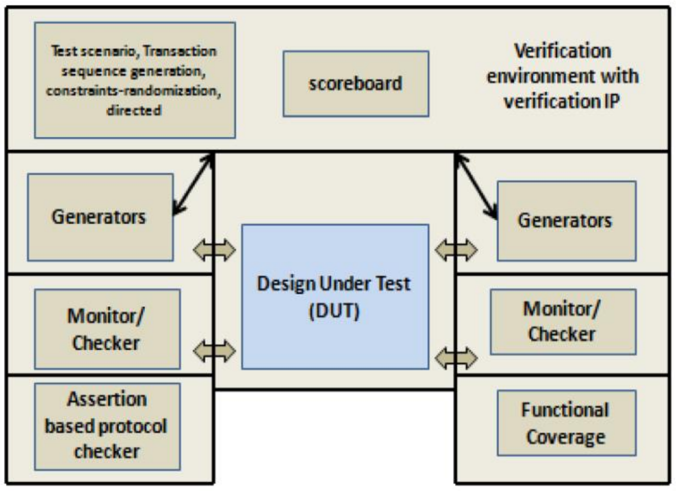

Test scenario, Transaction sequence generation, constraints-randomization, directed
scoreboard
Verification environment with verification IP
Generators
Monitor/ Checker
Assertion based protocol checker
Design Under Test (DUT)
Generators
Monitor/ Checker
Functional Coverage

Figure 2: Verification IP with Verification Environment

The modeling architecture is based on the layered testbench methodology. Each Verification IP primarily consists of four layers as follows.

- Layer 0, signal layer or physical layer, allows direct access to Design-Under-Test (DUT) signals at the physical boundary.
- The command layer implements the BFMs, protocol generation, checking routines, and monitor methods at the most primitive levels.
- The transaction layer implements a high-level transaction generation. At this level, sequences of BFM commands can be generated in a directed and automatic randomized fashion and checked for correctness.
- The application layer is the highest-level abstraction in the Verification IP, where application specific test scenarios can be generated in pseudo-random and full-directed environments.

The functional coverage routines and methods will allow users to gather information about tests, most importantly about standard protocols functionality coverage. As such, these coverage methods should be accessible from higher layers and can act as templates for users to add their own in the higher layers. BFMs generate the input stimuli to the DUT. Protocol checks and monitors are used to ensure the standard compliancy and monitor the correct bus activities. Monitors are developed as integral part of verification IP and should be dynamic and include specific and descriptive error messages that are user controllable. The Verification IP embedded Checkers and Monitors are usually included in Layer1 and can be accessed by all the layers. The testbench and Verification IP configuration is to be provided at higher layers, for examples at layers 2 and 3.

The various features of this AXI4-Lite verification IP are as follows,

- Operates as a master or slave
- Supports 1, 2, 4, 8 and 16 bytes data block size
- Supports multiple outstanding transactions
- Programmable response type
- Supports all protocol burst types, burst lengths and response types
- Read/write response check
- Supports wait states injection
- Supports full random timings
- Supports misaligned transfers
- Provides functional coverage reports
- Provides the total amount of checks has been made during simulation, total number of checks which are passed and total number of checks which are failed.

3.1 AXI4-Lite Master

The AXI4-Lite Master Verification IP (VIP) initiates transfers on the AXI4-Lite bus. The important tasks or functions carried out by the master verification IP are as follows,

- It specifies minimum and maximum value of burst length.
- Enable/disable bus random timings.
- Provides write data buffer. Returns pointer where will be locate write response buffer.
- Generates read transactions. Returns pointer where will be located read data buffer.
- Returns write response, read data and response buffers from specified location.
- Provides read data buffer starting from address and compares it with polled Data buffer. Repeat until buffers are equal or until time out occurred.
- It holds the write address, write data and read address channels in the idle state for specified clock cycles.
- Wait until all transactions in the write address, write data, write response, read address and read data channel buffer are finished.
- Prints all the errors occurred during simulation time and returns error count.

3.2 AXI4-Lite Slave

The AXI4-Lite Slave Verification IP models AXI4-Lite slave device. It has an internal memory which is accessible by master as well as by corresponding commands. The important tasks or functions carried out by the slave verification IP are as follows,

- Enables/disables slave response random delays.
- Sets slave write response and read response type for the specified address.
- It puts data buffer to the internal memory.
- It reads data from the internal memory.
- Read data buffer from internal memory and compare it with poll data buffer. Repeat until buffers are equal or until time out occurred.
- Prints all the errors occurred during simulation time and returns error count.

4. Results

The functional coverage models are developed by using system verilog language. These models are integrated into the verification IP. Mentor graphics Questa simulator tool is used to run the cycles of simulation. The coverage results are collected for various signals of different channels in the AXI4-Lite protocol.

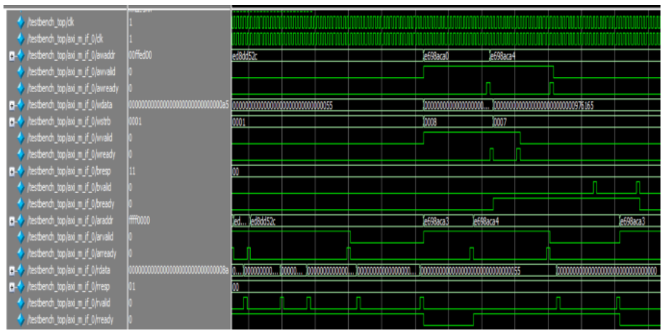


Figure 3: Simulation results showing master interface signals

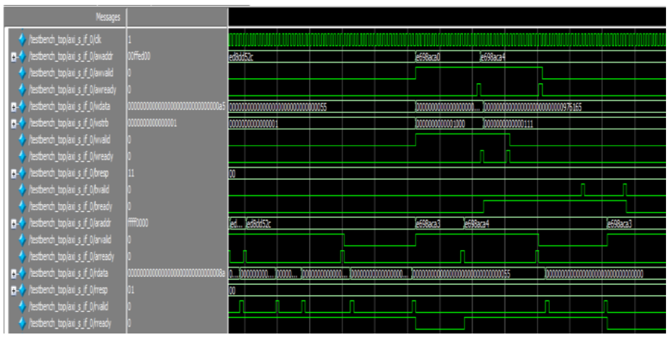
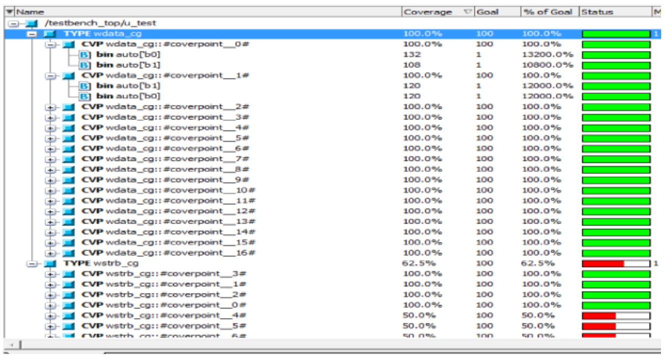


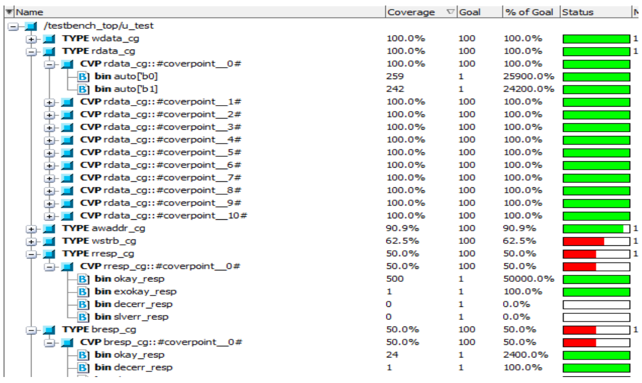
Figure 4: Simulation results showing slave interface signals

Figure 3 and Figure 4 Shows the simulation results for both master and slave interface signals.



Name	Coverage	Goal	% of Goal	Status
/testbench_top/u_test				
TYPE wdata_cg	100.0%	100	100.0%	🟩
CVP wdata_cg::#coverpoint__0#	100.0%	100	100.0%	🟩
bin auto['b0]	132	1	13200.0%	🟩
bin auto['b1]	108	1	10800.0%	🟩
CVP wdata_cg::#coverpoint__1#	100.0%	100	100.0%	🟩
bin auto['b1]	120	1	12000.0%	🟩
bin auto['b0]	120	1	12000.0%	🟩
CVP wdata_cg::#coverpoint__2#	100.0%	100	100.0%	🟩
CVP wdata_cg::#coverpoint__3#	100.0%	100	100.0%	🟩
CVP wdata_cg::#coverpoint__4#	100.0%	100	100.0%	🟩
CVP wdata_cg::#coverpoint__5#	100.0%	100	100.0%	🟩
CVP wdata_cg::#coverpoint__6#	100.0%	100	100.0%	🟩
CVP wdata_cg::#coverpoint__7#	100.0%	100	100.0%	🟩
CVP wdata_cg::#coverpoint__8#	100.0%	100	100.0%	🟩
CVP wdata_cg::#coverpoint__9#	100.0%	100	100.0%	🟩
CVP wdata_cg::#coverpoint__10#	100.0%	100	100.0%	🟩
CVP wdata_cg::#coverpoint__11#	100.0%	100	100.0%	🟩
CVP wdata_cg::#coverpoint__12#	100.0%	100	100.0%	🟩
CVP wdata_cg::#coverpoint__13#	100.0%	100	100.0%	🟩
CVP wdata_cg::#coverpoint__14#	100.0%	100	100.0%	🟩
CVP wdata_cg::#coverpoint__15#	100.0%	100	100.0%	🟩
CVP wdata_cg::#coverpoint__16#	100.0%	100	100.0%	🟩
TYPE wstrb_cg	62.5%	100	62.5%	🟥
CVP wstrb_cg::#coverpoint__3#	100.0%	100	100.0%	🟩
CVP wstrb_cg::#coverpoint__1#	100.0%	100	100.0%	🟩
CVP wstrb_cg::#coverpoint__2#	100.0%	100	100.0%	🟩
CVP wstrb_cg::#coverpoint__0#	100.0%	100	100.0%	🟩
CVP wstrb_cg::#coverpoint__4#	50.0%	100	50.0%	🟥
CVP wstrb_cg::#coverpoint__5#	50.0%	100	50.0%	🟥
CVP wstrb_cg::#coverpoint__6#	50.0%	100	50.0%	🟥

Figure 5: Functional coverage Results showing write data channel signals



Name	Coverage	Goal	% of Goal	Status	N
/testbench_top/u_test					
TYPE wdata_cg	100.0%	100	100.0%	🟩	1
TYPE rdata_cg	100.0%	100	100.0%	🟩	1
CVP rdata_cg::#coverpoint__0#	100.0%	100	100.0%	🟩	
bin auto['b0]	259	1	25900.0%	🟩	
bin auto['b1]	242	1	24200.0%	🟩	
CVP rdata_cg::#coverpoint__1#	100.0%	100	100.0%	🟩	
CVP rdata_cg::#coverpoint__2#	100.0%	100	100.0%	🟩	
CVP rdata_cg::#coverpoint__3#	100.0%	100	100.0%	🟩	
CVP rdata_cg::#coverpoint__4#	100.0%	100	100.0%	🟩	
CVP rdata_cg::#coverpoint__5#	100.0%	100	100.0%	🟩	
CVP rdata_cg::#coverpoint__6#	100.0%	100	100.0%	🟩	
CVP rdata_cg::#coverpoint__7#	100.0%	100	100.0%	🟩	
CVP rdata_cg::#coverpoint__8#	100.0%	100	100.0%	🟩	
CVP rdata_cg::#coverpoint__9#	100.0%	100	100.0%	🟩	
CVP rdata_cg::#coverpoint__10#	100.0%	100	100.0%	🟩	
TYPE awaddr_cg	90.9%	100	90.9%	🟩	1
TYPE wstrb_cg	62.5%	100	62.5%	🟥	1
TYPE rresp_cg	50.0%	100	50.0%	🟥	1
CVP rresp_cg::#coverpoint__0#	50.0%	100	50.0%	🟥	
bin okay_resp	500	1	50000.0%	🟩	
bin exokay_resp	1	1	100.0%	🟩	
bin decerr_resp	0	1	0.0%		
bin slverr_resp	0	1	0.0%		
TYPE bresp_cg	50.0%	100	50.0%	🟥	1
CVP bresp_cg::#coverpoint__0#	50.0%	100	50.0%	🟥	
bin okay_resp	24	1	2400.0%	🟩	
bin decerr_resp	1	1	100.0%	🟩	

Figure 6: Functional coverage Results showing read data channel signals

5. Conclusion

This verification IP provides a better solution for the verification of AXI4 master and slave devices. This VIP includes functional coverage models developed by using system verilog language. it requires a detailed verification plan and much time creating the cover groups, analyzing the results, and modifying tests to create the proper stimulus. This may seem like a lot of work, but is less effort than would be required to write the equivalent directed tests. Additionally, the time spent in gathering coverage helps us better track our progress in verifying design. One can easily develop the verification environment by using this verification IP to verify master and slave AXI4-Lite devices.

ACKNOWLEDGMENT

I would like to thank my guide Narayana Swamy J.C and coordinator Dr.Vijaya Prakash A.M for their encouragement, guidance and support. Also I would like to thank my classmates and Department of Electronics and Communication, Bangalore institute of technology for their support.

References

[1] AMBA® AXI™ and ACE™ Protocol Specification by ARM.

[2] ARM, AMBA Specification (Rev4.0) available at www.arm.com, 2010.

[3] Anurag Shrivastava, G.S. Tomar, Ashutosh Singh, “Performance Comaparision of AMBA Bus-Based System On-Chip Communication Protocol”, International Conference on Communication System and Network.

[4] Ying-Ze Liao, "System Design and Implementation of AXI Bus",National Chiao Tung University, October 2007.

[5] Mehdi Mohtashemi, Azita Mofidian “Verification Intellectual Property (IP) Modeling Architecture”.

[6] Priyanka Gandhani, Charu Patel “Moving from AMBA AHB to AXI Bus in SoC Designs: A Comparative Study”,august,2011.

[7] Chien-Hung Chen, Jiun-Cheng Ju, and Ing-Jer Huang, “A Synthesizable AXI Protocol Checker for SOC Integration,”IEEE,,ISOCC 2010.

[8] Questa Simulator User Manual from Mentor Graphics.

[9] Benini and D. Bertozzi, “Network-on-chip architectures and design methods,” IEEE Proc.-Comput. Digit. Tech., Vol. 152, No. 2, March 2005

[10] Ing. Dries Driessens and. Tom Tierens, “Overview of Embedded Buses,” 2003 by Patrick Pelgrims

[11] Marcus Harnisch ,”Migrating from AHB to AXI based SoC Designs”,from Doulos, 2010.

[12] Akilesh kumar,richa sinha “Design and verification analysis of APB3protocol with coverage”,international journal of advanced in engineering& technology, nov2011.

[13] Deepak Shankar. “Comparing AMBA AHB To AXI Bus Using System Modeling”, February 2010