International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395-0056
Volume: 06 Issue: 05 | May 2019 www.irjet.net p-ISSN: 2395-0072

Verification of AXI IP Core(Protocol) using System Verilog

Nagendrareddy S[1], Jagadeesha kolour K[2], Basarkod P I[3]

Dept. of ECE Engineering, Reva University, Karnataka, Bangalore-India, 560064

Abstract: **The word Design Verification itself tells that this paper does not involve Designing of AXI VIP Core, for the verification one needs its Design Specification Sheet to understand the working of the design so that it can be simulated in the Advanced Verification tools. Advance Extensible Interface (AXI) is using more in todays industry due to frequency and performance operations in the absence of complicated bridges. Actual AHB and APB interfaces are additionally good with AXI. We use AXI for multiple tasks it is not possible in the other protocol but it is possible in AXI because it consists of five different channels write address and information channels and dependent on the transaction ID. AXI supports burst based transfer. The AXI protocol has been designed using the system Verilog and Universal Verification Methodology (UVM) we verified it .The five channels of AXI as write address, write data, write response, read address, read data channels are observed in verification. With respect to AXI convention verification environment is created and distinct test cases will be passed by randomization. The AXI protocol design can be verified by using Questa sim tool and assertions and functional coverage has been obtained.**

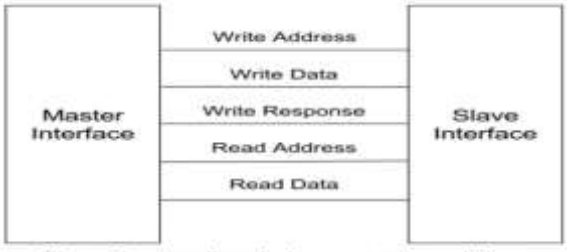
Keywords: **channel, system Verilog. AMBA ,AXI UVM,DUT**

1. INTRODUCTION:

Protocols are of two types on chip and off chip nothing but peripheral protocols, on chip protocol consists of AMBA it has AHB APB and AXI. The AXI has advantage of high frequency system design and high performance.it is depends on a point to point interconnect to maintain strategic distance from bus sharing and it permits low latency and high transfer speed. The idea of the AXI protocol is that it gives a system for how unique squares inside a chip will speak with one another. The communication is clear and continuous when it gives methodology before transmitting anything .this is the means by which distinct squares can communicate to one another without descend on each other. The procedure for AXI protocol is as follows

- Master and slave must handshake to confirm a valid signals.
- Control signal transmission must be in separate phases
- Signal transmission occurs in separate channels
- Continuous transfer may be accomplished through burst-type communication

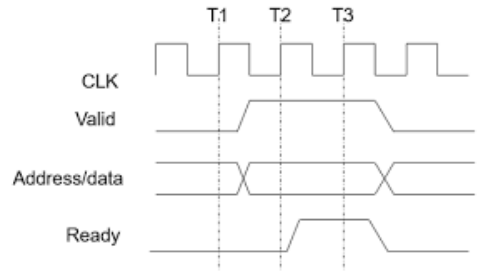
Working of AXI protocol master and slave devices, it work on five channels that incorporates read and write address Write and read data and write response. Every channels incorporates its own particular signals.it can send handshake reaction without any interrupt so that it can get and put in a order. Along this channel that has priority will be responded to fist and forth. The master will give a valid flag and one that gets a proper reaction from receiver. The transmission happing in a different stages, it enables the exchange of data to be performed in a one by one manner. This implies that a handshake is received fist, at that point the information is transform from master to slave and that's how the AXI protocol works to move data between sources without obstruction protocol is utilized in so many gadgets, it makes it simpler to connect distinct devices into the similar hub since priority is established.



Channel connections between master and slave interfaces

1.1AXI handshake process:

Protocol bolsters five distinct channels utilize the equivalent VALID/READY handshake to exchange information and control data. The data and control information rate can be handle by master and slave respectively. The master produces VALID flag to demonstrate when information or control data points are open . The slave gives READY flag to show that it can now accepts data or control information



2. CHANNEL ARCHITECTURE:

Figure 1-1 Channel architecture of reads

In the above fig master initiates address and control to slave through the bus channel after receiving the information the slave must respond back to the master with response signal.

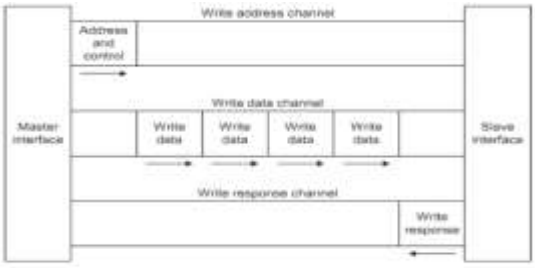


Figure 1-2 Channel architecture of writes

2.1AXI Transaction:

AXI assist distinct variety of transactions

Burst type:

Burst of three kinds fixed increment and wrap :in fixed for every transfer the address is same for transfers. In incrementing transfer the address for every transfer is the addition of the past transfer. The estimation of augmentation relies upon exchange measure. Wrap is like augmentation for each exchange address of burst is addition of past exchange. However it wraps the address around the boundary when it reaches. Aligned and unaligned Use below equations to find out addresses of transfers within a burst:

Start Address = **ADDR**

Number Bytes = 2**SIZE**

Burst Length = **LEN** + 1

Aligned Address = (INT(Start Address / Number Bytes)) x Number Bytes.

3. VERIFICATION STEPS:

(i) Features listing down.

(ii) Scenario listing down.

(iii) Test plan development.

(iv) Functional Coverage Point listing down.

(v) Test-bench architecture definition.

(vi) Test-bench component coding.

(vii) Sanity test case development.

(viii) Sanity test case bring up.

(ix) Other test cases.

(x) Setting up regression.

(xi) Running regression and debugging regression results.

(xii) Generating coverage results.

(xiii) Analyze coverage results.

(xiv) Closing functional Coverage.

4. VERIFICATION ENVIRONMENT:

i. Agent: agent is designed as active or passive agent is derived from UVM component .the active agent consists of a driver, monitor and sequencer an active agents generates the stimulus these generated stimulus will be drives to DUT. passive agents samples the generated DUT signals but doesn't drive them.so that it consists of single monitor.

ii. Sequencer: A sequencer produces an data transactions in the form of class objects and sends it for driver execution. The sequencer controls the progression of request and response to sequence things among the sequences and driver.

iii. Sequence: The real pushed information is irregular in the progression. From the succession, here irregular information is inclined to driver by means of sequencer.

iv. Driver: Driver as a functioning element that as information of how to drive signs to DUT through interface.

Monitor: It receives the signals from interface and

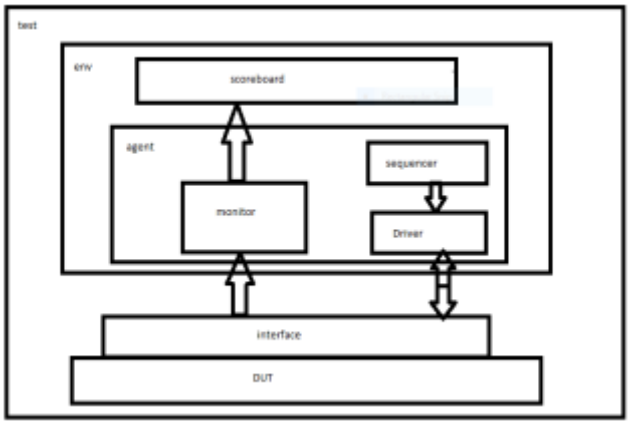


fig: verification environment using UVM methodology

samples the signals these sampled signals will be send to coverage block .

v. Scoreboard: It comprises of checker and reference model. Its gets exchanges from output monitor and reference model and check for rightness of output by looking at both the transactions.

vi. Environment: The object of this class is made in test and is a most important component of UVM test bench. Environment makes agents, scoreboard, virtual sequencer etc. The environment makes association among monitors and scoreboard. On the off chance that it has virtual sequencer, here in environment we need to interface physical sequencers with the handles of physical sequencers in virtual sequencer

vii. Test: sequencer is having the successions ,where test will established. In base test we will set all of the parameters of arrangement database class according to prerequisites. Here in addition get the interface upon top and reinforced to the interface handles in neighborhood design database classes. The base test additionally makes condition. all of these things happens in build phase of base test. The tyke tests are gotten from the base test class. In tyke test we essentially make the handle of virtual sequence and start it on virtual sequencer, before beginning the arrangement a protest is hoisted and this protest is released again subsequent to starting the succession. By chance that we don't raise the protest the test system will believe that there is no run stage to execute, so the test system can bounce legitimately to remove stage. The all out number of raised protests ought to be equivalent to the quantity of dropped complaints.

Coverage:

Coverage is used to measure the tested and untested portion of design. In this paper we will measure two types of coverages

i. Functional coverage: Code report does not know anything about what the structure ought do. There is no real wat to discover what is absent in the code by the code report, yet a functional report can get the missing functionality in the Design code. The fundamental objective of verification is to ensure that a design acts accurately in its genuine environment. Functional report is client specified to attach the verification environment to the design aim or functionality. Functional coverage enlightens us regarding whether every one of the functionalities given in particular are incorporated into the RTL or not. Functional report is additionally called as "detail Coverage".

ii. Assertion coverage: Assertions are essentially used to evaluate the conduct of a design.an assertion is a check embedded in to a design unit during simulation. warnings or errors are produced on failure of particular condition or sequence of events. these are additionally used to give functional coverage. There are two types of assertions

Immediate assertions: immediate assertions check for condition at the present simulation time. They must be put inside procedural squares

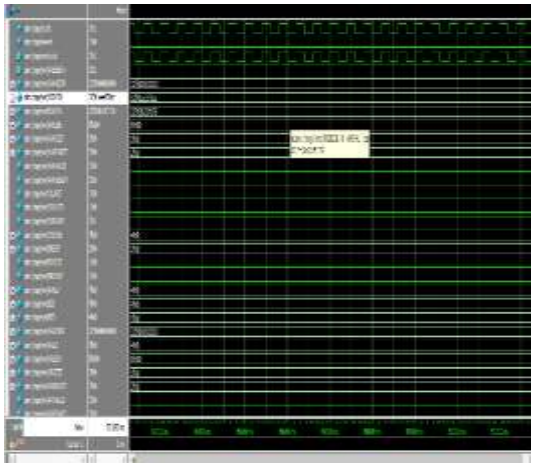
Concurrent assertions: concurrent assertions check the group of events spread over different clock cycles. These are evaluated only at the event of the clock.it can be place in a module an interface and in procedural squares.

Test cases:

Different test cases like write address channel matching, read address channel matching , id matching burst transactions are verified successfully.

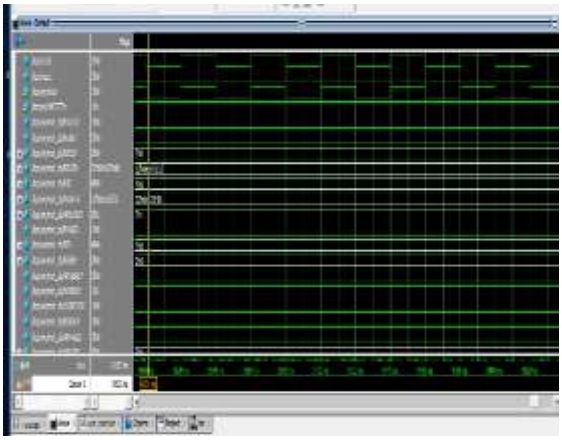
5. RESULTS:

In this fig a shows the transactions happening from both master to slave. The signals related to write address ,data, response and read related signals are displayed.

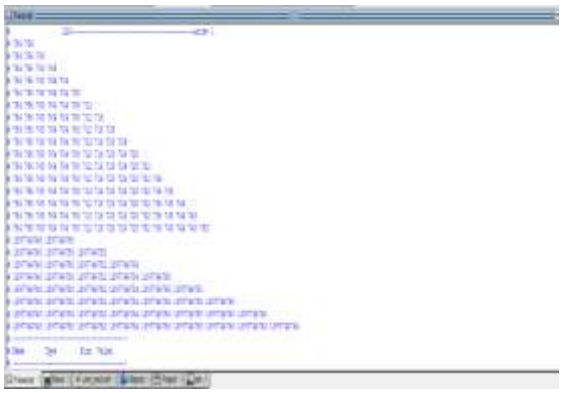


Fig(a):Output wave form for basic write and read transaction obtained from Questa sim

Here the transaction of type increment the initial address will be provided data will be written to the address in the increment manner with specified size of the address and data.

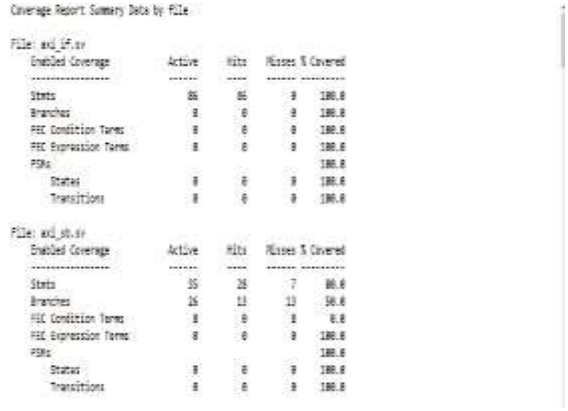


Fig(b):output wave from for increment transaction obtained from Questa sim



Fig(c) out put of burst type transaction with increment and wrap.

Here along with increment burst type transaction can be done the wrapping can be of specified length. warp happening for particular boundary.



Coverage Report Summary Data by File

File: axi_if.sv

Enabled Coverage	Active	Hits	Misses	% Covered
Stmts	86	86	0	100.0
Branches	0	0	0	100.0
FEC Condition Terms	0	0	0	100.0
FEC Expression Terms	0	0	0	100.0
FSMs				100.0
States	0	0	0	100.0
Transitions	0	0	0	100.0

File: axi_sb.sv

Enabled Coverage	Active	Hits	Misses	% Covered
Stmts	35	28	7	80.0
Branches	26	13	13	50.0
FEC Condition Terms	0	0	0	0.0
FEC Expression Terms	0	0	0	100.0
FSMs				100.0
States	0	0	0	100.0
Transitions	0	0	0	100.0

Fig(d):coverage report generated by Questa sim

Here we written all the required cover bins and also cover points to get the coverage and assertion coverage also obtained.

6. CONCLUSION:

In this, paper AXI protocol is structured and all specification details has been implemented and functional confirmation is done of effectively. analyzes by surveying individual test cases. These experiments are actualized with respect to sanity test . The functional coverage and assertion report is gotten .

REFERENCES:

1. Mahesh G, Shaktivel SM. Functional Verification of the Axi2Ocp Bridge using System Verilog and effective bus utilization calculation for AMBA AXI 3.0 Protocol. IEEE Sponsored 2nd International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS). 2015.
2. Mahesh G, Shaktivel SM. Verification of Memory Transactions in AXI Protocol using System Verilog approach. IEEE ICCSP Conference. 2015.
3. Naidu KJ, Srikanth M. Design and Verification of Slave block in Ethernet Management Interface using UVM. Indian Journal of Science and Technology. 2016 Feb; 9(5):1-7.
4. Sebastian R , Mary SR, Gayathri M, Thomas A. Assertion Based Verification of SGMII IP Core incorporating AXI Transaction Verification Model. International Conference on Control,

Communication and Computing India (ICCC). 2015; p. 585-88.
5. Chen X, Xie Z and Wang XA. Development of Verification Environment for AXI Bus Using System Verilog. International Journal of Electronics and Electrical Engineering. 2013; 2(1):112-14.
6. Chen CH, Iu JC, Huang II. A Synthesizable AXI Protocol Checker for SoC Integration. IEEE Transl, ISOCC. 2010; 8:103-6.
7. Ranga A, Venkatesh LH, Venkanna V Design and Implementation of AMBA-AXI Protocol Using VHDL for SOC Integration. International Journal of Engineering Research and General Science. 2012; 2(4):1-5.
8. [8] Universal Verification Methodology (UVM) 1.1 User'Guide, May,2011.
9. [9] UVM Cookbook, Mentor Graphics, Sept, 20