International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 04 Issue: 09 | Sept -2020 ISSN: 2582-3930

Validation of Advanced eXtensible Interface Verification IP (VIP)

Varsha Dilip Patil

Dept of E&TC DIEMS, Aurangabad, India.

---***---

Abstract -In the current era of increasing IP (Intellectual Property) based SOC (System on Chip) designs, it is highly likely that we have heard about AMBA, AHB, APB, AXI, AXI-lite, ACE etc. somewhere or other. Silicon densities, both for ASICs and FPGAs, can now support true systems-on-chip (SoCs). This level of design requires busing systems to connect various components, including one or more microprocessors, memory, peripherals, and special logic. This project focuses onhow the Advanced Extensible interface (AXI) is useful for high bandwidth and low latency interconnects. This is a point to point interconnect and overcomes the limitations of a shared bus protocol in terms of number of agents that can be connected. The protocol also was an enhancement from AHB in terms of supporting multiple outstanding data transfers (pipe-lined), burst data transfers, separate read and write paths and supporting different bus widths. Design and Verification Environmentcoding of AXI Protocol is done in Mentor Graphics tool QuestaSim- 10.4e. Editor used for a project is gVim 7.4. In this dissertation System Verilog is used for verification purpose of the AXI Protocol.

***Key Words*:**AXI, VIP, SOC,HDL/HVL,SV

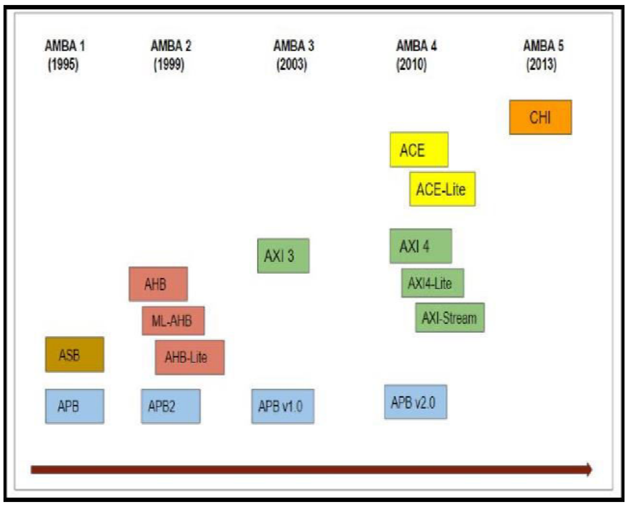
1.INTRODUCTION

The progress in the semiconductor industry and technology leads to complex digital designs which brings a large number of intellectual property cores for large usage. With these IPs, In present evolution these SoC designs are becoming more popular and effectively used for more number of applications. For large SoC design subsystems are connected by communication bus protocol. This makes function coverage so crucial as reusability of large number of IP cores in SoC is increased. Functional coverage takes 70invested in design progress. Expenses of verification takes 28cycle expenses. In this 28short time investment goals, verification engineershasintroduced methodology which verifies the functionality of chips. This inbuild verification environment is known as Verification IP. This paper is set up to investigate various features of AXI interconnect. To realize that I build a SV Environmentwhich consist of AXI master and AXI slave. To build AXI architecture, need to connect master VIP to master interconnect and slave VIP to slave interconnect. The Masters and Slaves are used mainly for testing purposes. The verification environment is a standard method of verifying the DUV in code coverage mode. The following components are present in the verification environment;

AXI Environment
AXI Master VIP:
AXI Generator
AXI BFM
AXI Monitor
AXI Functional Coverage
AXI Slave VIP:
AXI Transaction

2. LITERATURE REVIEW

SoC and SoC Bus protocols: Overview Nowadays the designs are with high level of integration of complex design components known as Intellectual property IP. These IPs are possible with increasing number of transistors and decreasing process technologies. Soc become IC which now can implement all the functionality of the electronic system. Some Particular Soc designs have different programmable components with are application specific. These application specific IPs includes cores, some application specific circuits and some on chip memory. The biggest challenge in these complex Socs is connecting the different components. And to connect these components of different requirements communication buses are used. Which have a big impact on the performance of the design. In this present era, standard onchipbusprotocolsweredevelopedtoavoidthenumber



ofinterfacesandprotocolsoncomplexSoCs.

Fig – 1: ARM AMBA Evolution

3.VERIFICATIONENVIRONMENT

Verification environment is group of classes performing specific operations. The new verification constructs canbe easily reused for the objected-oriented feature of SystemVerilog.The main reason of verification Environment is to analyze the comparison response of DUT (Design Under Test) and the response of the reference model or register model,check the function coverage and apply the different scenarios.The testbench is aimed to generate stimulus, apply stimulus,note down the response,check for functional correctness and make conclusion against aimed verificationtarget.

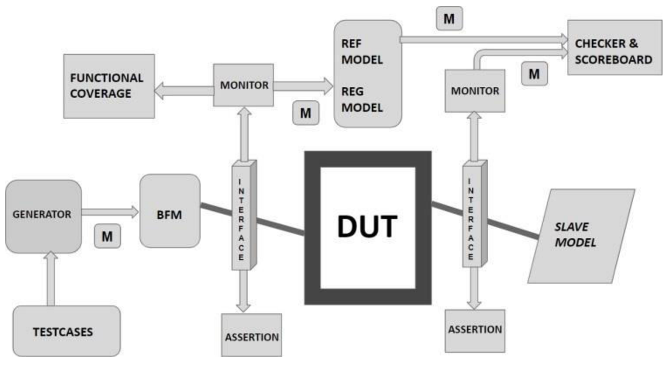


Fig – 2: Generic Verification Environment

4.AXI VERIFICATIONENVIRONMENT

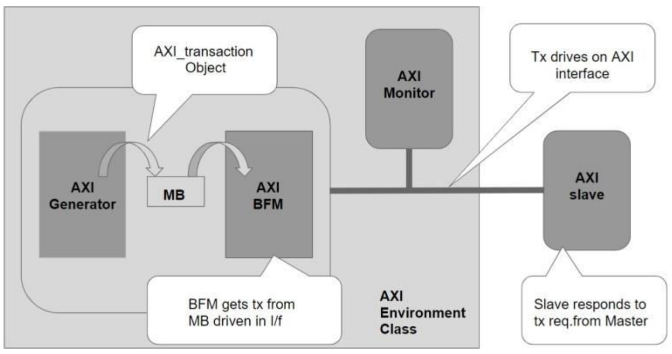

AXI_transaction Object
AXI Monitor
Tx drives on AXI interface
AXI Generator
MB
AXI BFM
AXI slave
BFM gets tx from MB driven in I/f
AXI Environment Class
Slave responds to tx req.from Master

Fig - 3: AXI Verification Environment

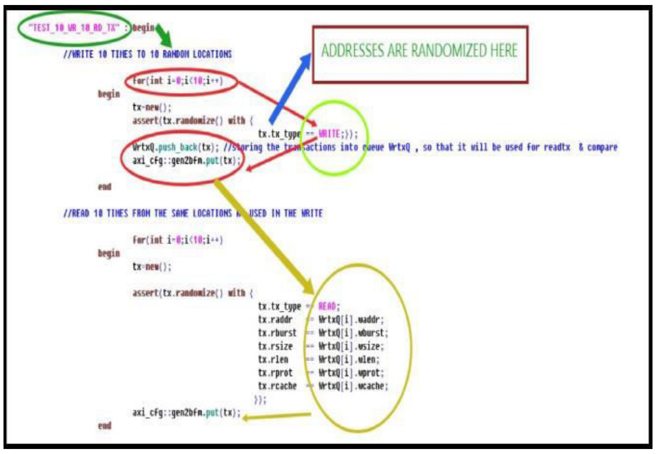
This paper, transaction class consists of all the attributes which includes properties, methods and constraints. Properties includes the read and write transaction signals not the handshaking signals. The methods implemented are print(), copy(), and compare().which are used for the displaying transaction by checking transfer type and copying transaction so that the queue of objects can be saved for comparison purpose. And according to features constraints are applied. These transactions are given to generator class through mailbox. These mail-boxesarestaticsothatcanbeusedinwholetestbench.

In generator class,testcases are created which will generate all possible scenario.Scenarios may be error or includes features.We can run these testcases by passing the testcase name in the command line arguments in simulation tool. For example: +testname=TESTNAME ThesescenariosarepasstotheBFMusingmailboxwhich is static.As we cannot pass transaction object to the slave,BFM will convert object level signal to pin level signals. Interface consists of all signals of design.These signals are declared as separate interface,is declared in thetopmoduleasphysicalinterface.axi-intfpif(clk,rst);// This is an instantiation of physical interface. To usetheinterface in every class, interface is declared as thevirtual interface.The physical interface is assigned to the virtual interface. axi-cfg::vif = pif;Clock and Reset are passed as an argument. The pin level activities are all observed by axi monitor.And the same signals are applied to the axi slave.Slave will generate the response.All the changes are observedandcollectedbythemonitor.Coverageclasswill collect all the changes it will check which covergroup is got hit,it will analyze and create reports showing how many signals are hit by the testcase. All the classes(axi-tx,axi-gen,axi-bfm,axi-mon,axi-cov) are included in the environment class(axi-env) ,which is under the most module In the environment class,all run task methods of class under it are included.This verification environment isunderthetopmostmodule.Intopmodule,includes

- clk,rstdeclaration
- designinstantiation
- interfaceinstantiation
- TBinstantiation
- assertioninstantiation
- clockgeneration
- logic to read thetestcase
- logic to decide when to end the simulation (when to call$finish)

5.IMPLEMENTATIONRESULTS

- In this testcase, transactions are generated in for loop.



```
"TEST_10_WR_10_RD_TX" : begin
   //WRITE 10 TIMES TO 10 RANDOM LOCATIONS
         for(int i=0;i<10;i++)
      begin
         tx=new();
         assert(tx.randomize() with {
                        tx.tx_type == WRITE;});
         WrtxQ.push_back(tx); //Storing the transactions into queue WrtxQ , so that it will be used for readtx  & compare
         axi_cfg::gen2bfm.put(tx);
      end
   //READ 10 TIMES FROM THE SAME LOCATIONS W USED IN THE WRITE
         for(int i=0;i<10;i++)
      begin
         tx=new();
         assert(tx.randomize() with {
                        tx.tx_type == READ;
                        tx.raddr   == WrtxQ[i].waddr;
                        tx.rburst  == WrtxQ[i].wburst;
                        tx.rsize   == WrtxQ[i].wsize;
                        tx.rlen    == WrtxQ[i].wlen;
                        tx.rprot   == WrtxQ[i].wprot;
                        tx.rcache  == WrtxQ[i].wcache;
                       });
         axi_cfg::gen2bfm.put(tx);
      end
```

Memory is created to all the transactions with new method. All the transactions of write are randomized. In case of transactions are randomize with constraint. With constraint is useful when particular signal to be constraint.

Fig - 4: Multiple Write and read

- Below waveform shows 2 transactions with different IDs. First have AWLEN=6 which means first transaction includes 7 transfers and WLAST gets asserted.In second transaction,AWLEN=b which means 13 transactions are happeningandWLASTwillgetasserted.

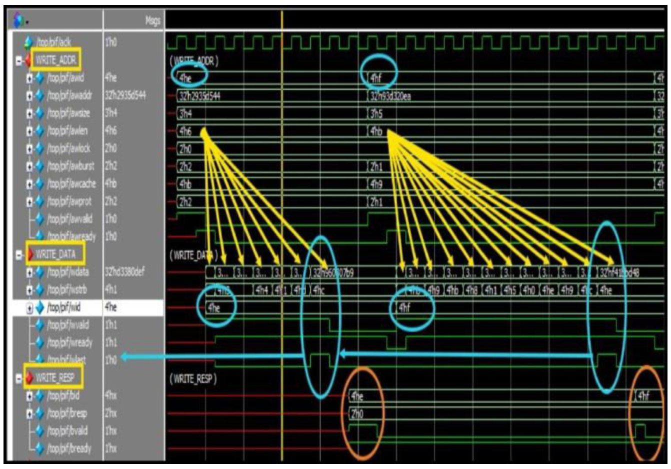
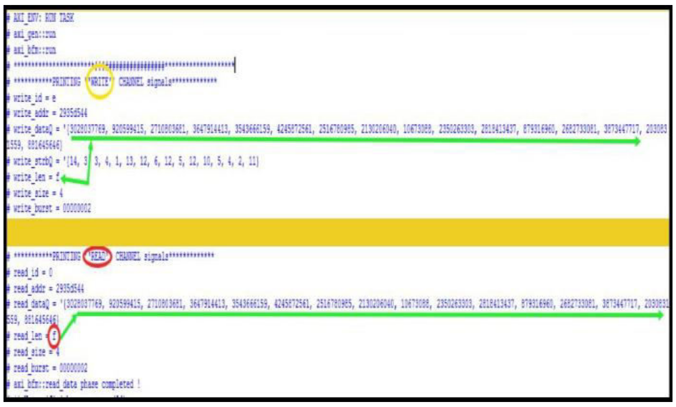


Fig - 4: Burst Transaction

- Below transcript shows write channel signal and read channel signal.Write and Read happening on same ad- dress location.write-dataQ and read-dataQ includes all writeandreaddatainqueuerespectively.



```
# AXI_ENV: RUN TASK
# axi_gen::run
# axi_bfm::run
# ****************************************************************
# **********PRINTING "WRITE" CHANNEL signals*************
# write_id = e
# write_addr = 2935d544
# write_dataQ = '{3028037769, 920599415, 2710803681, 3647914413, 3543666159, 4245872561, 2516780985, 2130206040, 10673088, 2350263303, 2818413437, 879316960, 2682733081, 3873447717, 203083
1559, 881645646}
# write_strbQ = '{14, 3, 3, 4, 1, 13, 12, 6, 12, 5, 12, 10, 5, 4, 2, 11}
# write_len = f
# write_size = 4
# write_burst = 00000002

# **********PRINTING "READ" CHANNEL signals*************
# read_id = 0
# read_addr = 2935d544
# read_dataQ = '{3028037769, 920599415, 2710803681, 3647914413, 3543666159, 4245872561, 2516780985, 2130206040, 10673088, 2350263303, 2818413437, 879316960, 2682733081, 3873447717, 2030831
559, 881645646}
# read_len = f
# read_size = 4
# read_burst = 00000002
# axi_bfm::read_data phase completed !
```

Fig - 5: Transcript window for write read

- In this coverage report, covergroup is created which includes write and read length and size. In this report, the percentage of these four coverpoints shows howmany bins got hit. As length is constraint to 15 and bin HIGH- LENconsistof[9:15].

CONCLUSIONS

This dissertation work starts from reading specifications to verification closure using regression, code coverage, functional coverage and assertion coverage as closure criteria. This proposed verification environment was able to target some set of transactions towards one specific feature of protocol. Some scenarios in testcases are as 10 write transactions back to back ,10 read transactions back to back, Generating 16 different transactions with unique IDs.

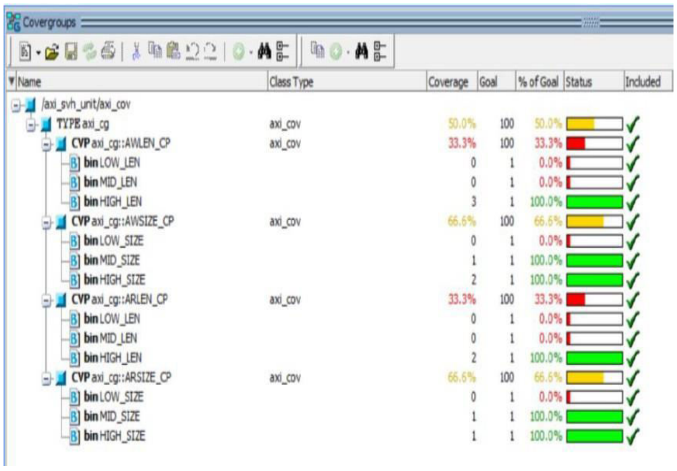
While developing verification IP, it involves various steps like

- Listing down features
- Developing test plan
- Developing TB architecture
- Coding TB components
- TB component integration
- Developing sanity testcases and functional testcases
- Setting up regression and verification closure

In future,Itis expected to verify other features like overlapping and out of order transaction, which will target all signals in coverage. This project is based on SV testbench environment,in future AXI 3.0 protocol environment could be implemented in UVM (Universal Verification methodology)environment which is nothing but developing UVC(Universal Verification Component).The VIP (Verification IP) development of AXI4.0 is expected as well as UVC of protocol could be done.As ARM recently introduced AXI 5.0, development of UVC and VIP can be done. As there are so many AMBA generations includes many protocols which can be the future development projects.

ACKNOWLEDGEMENT

This is by far the most significant accomplishment in my life and it would be impossible without people who supported me and believed in me. I would like to express my deep sense of respect and gratitude towards my advisor and guide Prof. Ms. S.K.Kapde, for giving me the opportunity to work under her and lending every support at every stage of this project work. I truly appreciate and value his esteemed guidance and encouragement from the beginning to the end of this dissertation. I am in debt to him for providing insightstowards the solution. I want to express my sincere respect and gratitude to Dr. Ulhas Shivurkar , Dr. R.M.Autee, Prof.A.R.Wadhekar Prof.S.K.Kapde for teaching me and helping me how to learn. On a personal note I would like to thank my family and all my M. Tech friends and special thanks to colleagues as well as friends for their support and all the thoughtful and mind stimulating discussions we had, which prompted us to think beyond the obvious. I have enjoyed their companionship thoroughly during my project. None of this would have been possiblewithouttheconstantandmoralsupportfromthem.



Name	Class Type	Coverage	Goal	% of Goal	Status	Included
/axi_svh_unit/axi_cov						
TYPE axi_cg	axi_cov	50.0%	100	50.0%		✓
CVP axi_cg::AWLEN_CP	axi_cov	33.3%	100	33.3%		✓
bin LOW_LEN		0	1	0.0%		✓
bin MID_LEN		0	1	0.0%		✓
bin HIGH_LEN		3	1	100.0%		✓
CVP axi_cg::AWSIZE_CP	axi_cov	66.6%	100	66.6%		✓
bin LOW_SIZE		0	1	0.0%		✓
bin MID_SIZE		1	1	100.0%		✓
bin HIGH_SIZE		2	1	100.0%		✓
CVP axi_cg::ARLEN_CP	axi_cov	33.3%	100	33.3%		✓
bin LOW_LEN		0	1	0.0%		✓
bin MID_LEN		0	1	0.0%		✓
bin HIGH_LEN		2	1	100.0%		✓
CVP axi_cg::ARSIZE_CP	axi_cov	66.6%	100	66.6%		✓
bin LOW_SIZE		0	1	0.0%		✓
bin MID_SIZE		1	1	100.0%		✓
bin HIGH_SIZE		1	1	100.0%		✓

Fig - 6: Coverage Report

REFERENCES

1. Murali .M. ,Umadevi. S,Sakthivel.S.M. Verification IP for AMBA AXI Protocol using System Verilog International Journal of Applied Engi- neering Research ISSN 0973-4562 Volume 12, Number 17 (2017) pp. 6534-6541 Research IndiaPublications.

2. AMBAAXIandACEProtocolSpecificationAXI3,AXI4,and AXI4-Lite, ACE and ACE-Lite

3. P. Naveen Kalyan K. Jaya Swaroop AMBA-AXI PROTOCOL VERI- FICATION BY USING UVM International Journal of Electronics and Communication Engineering and Technology (IJECET) Volume 7, Issue 4,July-August2016,pp.7684,ArticleID:IJECET0704009
4. IEEE standard for system Verilog -Unified Hardware Design, Specification and Verification Language

5. Book: Chris Spear Greg Tumbush SystemVerilog for Verification:A GuidetoLearningtheTestbenchLanguageFeatures

– – –