Deployment of Formal Verification Methodology for AXI based Protocol for a Memory Model Verification

Arpit Awasthi
Holy Mary Institute of Technolgy and Science, Hyderabad 501301 (INDIA)
Email: **arpit.a@hmgi.com**

Received 11.02.2021 received in revised form 13.12.2021, accepted 18.01.2022

DOI: 10.47904/IJSKIT.13.3.2023.12-16

Abstract- **AXI Protocol is an On-Chip Communication Protocol used for communication between different Intellectual Property blocks inside a System-On-Chip. With increasing logic density and a plethora of blocks interacting with each other, it becomes pertinent to not only verify the functional correctness of individual IP's but also to evaluate the integrity of transactions amongst each other. Simulation-based techniques fail to exhaustively assess all the kinds of transactions possible and primarily focus only on a few critical areas for AXI Protocol Verification. This paper proposes the integration of SystemVerilog Assertions coupled with Coverage model to verify the AXI Transactions from Master to Slave. This would aid in developing Verification Intellectual Property (VIP) helping in developing modular reusable components which can be leveraged across different verifications cycles. This paper demonstrates it empirically harvesting results relevant to building Assertion based verification, calculation of bus utilization factor and Coverage closure for RTL Signoff.**

Keywords- Constrained Random Verification (CRV), Formal Verification, Deadlock, Livelock, RTL Signoff

1. INTRODUCTION

AXI Protocol [1] by ARM Corporation is a de facto industry standard communication protocol for data transfer between different IP's. The usual approach is to build a SystemVerilog TestBench Environment to verify different AXI Transactions possible with varying different parameters like Burst Length, Data Size and Mode of Data Transfer Type by randomising the stimulus. To handle the data transfer between Master and Slave an AXI Interconnect exists for proper mapping of paths between Master and Slave, detecting false slave address and for arbitration between different Master and Slave Components. The objective of this paper is to build a modular and reusable Verification IP which can be used to authenticate the veracity of AXI Transactions rules as in specifications framed by ARM.

2. LITERATURE SURVEY

The verification task primarily deals with evaluating compliance with specifications. Traditional approach has been generating stimulus to activate the Design Under Test (DUT) and observe output behaviour. The major drawbacks in such a process are poor quality of stimulus which fails to exhaustively evaluate the design. 2020 Wilson Research Group Functional Verification Study [2] did an elaborative study and concluded that statistically 68 percent of projects are running behind schedule and multiple respins happening owing to logical functional faults. A paradigm shift is needed for overcoming these showstoppers bugs which hinder forward progress transitions in complex FSM models propelled need for development of novel analytical frameworks with a combination of semantic analysis and formal methods to develop RTL Signoff methodology [3]. This paper adopts a Formal Verification based Model Checking approach for a AXI module to exhaustively evaluate the different transaction modes and overcome the pitfalls of simulation-based approach.
Formal Verification (FV) is a rigorous mathematical algorithmic approach in which the different temporal activities in design are captured as properties and then fed to a Formal Verification tool to test those scenarios exhaustively by applying all the possible combinations [4]. The adoption of FV was very limited for RTL Signoff but with recent advancements in SMT and SAT based solvers have enabled enhanced adoption and development of RTL Signoff Techniques [5] [6] [7]. Siegal [8] laid the novel foundational work of developing an empirical driven property development approach for exhaustive verification using formal techniques. There is currently a lack of systemic methodology and techniques for deploying for End-to-End RTL Signoff. Yalin [9] outlines an effective strategy for combating issues encountered in deploying FV aiding in seamlessly mitigating issues encountered. Nicole et al. [10] [11] demonstrate a SystemVerilog Assertions (SVA) [12] based approach for detecting verification blindspots and Hardware Trojans vulnerabilities for robust assessment. Ronak et al. [13] exhibit successful integration of FV to shrink verification signoff at subsystem level. N.

Bombieri et al. [14] [15] presents an Assertion Based Verification (ABV) environment solution to build assertion reusable libraries and plug the gap with respect to bug escapes. B. Alizadeh et al. [16][17] introduced a formal debugging approach coupled with mutation analysis to detect multiple functional specification mismatch in a shorter run time. P. Aggarwal et al. [18] illustrate a robust coverage driven formal methodology for determining the effectiveness of different abstraction models and enhance coverage metric. Mahesh et al. [19] propose a verification methodology for AXI2OCP Bridge which helps in translating signals in two different protocols. The developed VIP would help in mitigating SoC Verification complexity issues. They have developed SVTB components and exciting different cases like Read, Write and Read-Write. The limitation is only 3 scenarios were triggered and thus low bus utilization factor of 77, 81,95 per cent respectively. The scenarios are generated using pseudo-random generators so limited in coverage scope. The primary responsibility of AXI2OCP bridge is to map AXI signals to OCP format and vice versa. The limitation is only 3 scenarios are considered out of multiple transaction combinations possible. Also, coverage analysis is missing. N. Gaikwad et al. [20] have developed a verification environment for AMBA AXI achieving successful write and read operations for incrementing test bench architecture with scalable test bench features. The authors present a brief overview of the AXI protocol and build a randomised testbed for evaluating the integrity of Read and Write operations by fixating certain parameters and randomising others. 3 cases are considered. In the first case, AWLEN is incremented and AWSIZE is fixed and all the remaining parameters are randomised. After the read and write case the valid ID is matched indicating that it was a successful operation without loss of data. One limitation was that only simple linear transactions were considered and remaining complex interleaving could have been incorporated for exhaustive testing to check out of order transactions. The locked and exclusive transfer is also not verified.

Chen et al. [21] in order to combat the growing complexity of increasing bus transactions propose a rule-based verification methodology in which they try to encapsulate the 44 rules to establish on-chip accuracy. The benefits of using rule-based design include improving observability, reducing debug time, improving integration through correct usage checking, and improving communication through documentation. They have also developed an informative Error Reference Table which populates all the violating assertions in the monitor transaction and aids in faster debugging by avoiding going through lengthy log files. The rules developed are exhaustive and consistent with respect to verification goals and would aid in RTL Signoff confidence. The only limitation is that the functional coverage features are missing which are critical to assess the kind of stimulus being generated and identify blind spots.

C. Prasad et al. [22] present their findings on developing an SV based modular verification methodology with high functional coverage metric for AXI2APB Bridge. In this work, the verification of different modes like fixed, wrapping and incremental modes for reading and write transaction and functional verification also performed using Synopsys VCMX and VERDI simulator. The bridge is modelled using 3 FIFO's .to connect the AXI and APB, using synchronous FIFOS. The bridge consists of 3 FIFO's namely request FIFO, write FIFO and READ FIFO. For reading and writing the packets usual SVTB model is generated with components like Generator, Monitor, Driver and BFM to drive transactions to DUT. Covergroups are scripted capturing different scenarios for burst, length and transaction type. They were able to achieve 100 percent coverage. The approach used builds modular components aiding in shrinking verification time and catching bugs. AXI Interconnect can be used for detecting slave address decode error. This scenario can be verified in future. Siddhhan et al. [23] analyse the internal parameters of the AMBA Advanced eXtensible Interface (AXI) protocol. The Bus Monitor (BM) plays an important role in the measurement of the performance metrics of the AXI protocol. This paper gives the design of a bus monitor and comes up with the parameter values of total transfer count, total transfer size, valid cycle count, busy cycle count and read latency count and write latency count. The advantage of this proposed Bus Monitor is that write and read operations are dealt with separately and stored in individual registers so that the designer's final analysis becomes simpler.

3. RESEARCH METHODOLOGY

The SystemVerilog Testbench environment is crafted to verify AXI Protocol. The AXI Transaction class has all the variables that are to be randomised and AXI Generator class uses this randomised stimulus to generate different types of Transaction Mode possible. User-defined constraints can be defined to fine-tune randomised stimulus to invoke specific scenarios and target verification hotspots which are more error-prone and. After generating constrained randomised stimulus these are written onto a Mailbox from which these transaction packets are sampled by AXI Bus Functional Model (BFM)/Driver class. All the components are encapsulated into an Environment class which sits inside Top Class.

Having these modular class-based components aids in reusability and rapid prototyping of verification components depending on slave requirements. The BFM then drives these transactions to a Memory Slave Model via AXI Interface. The AXI Interface controls the flow of transactions packet between master and slave. The BFM is the main component for driving transactions to the slave DUT via AXI Interface Model. All these are depicted in Figure 1 about different AXI components. The types of transactions type being driven are following:

- Burst Write Transfer
- Burst Read Transfer
- Write_Read Transfer
- Out of Order Transfer Transaction
- Fixed, Increment, Wrap Transaction

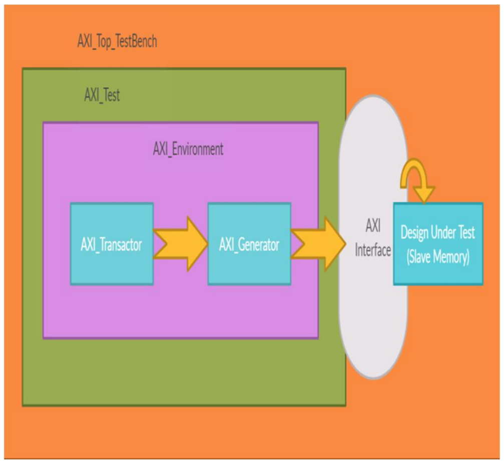


Fig. 1: AXI Verification TestBench Components

The following Verification hotspots are being targeted:

- Validation of ID Mapping for Write and Read Transaction and Handshaking Mechanism
- Checking the integrity of 3 Types of Burst Transfer:
    - Fixed Address Transfer
    - Increment Address Transfer
    - Wrap Address Transfer
- Evaluating the data matching for combination of out of order transactions.
- Verification of 3 modes of security in Bus Function Model:
    - Normal Transaction
    - Exclusive Transaction
    - Lock Transaction
- AXI Protocol Checkers at Interface class between Master and Slave DUT.

4. RESULTS ANALYSIS

The simulation wavefroms are shown in Figure 2 and Figure 3. In the AXI Transaction Address Phase the the contents related to Address Data Phase are loaded into the Address Data Channel.AXI Channel has 5 channels for data transfer between Master and Slave

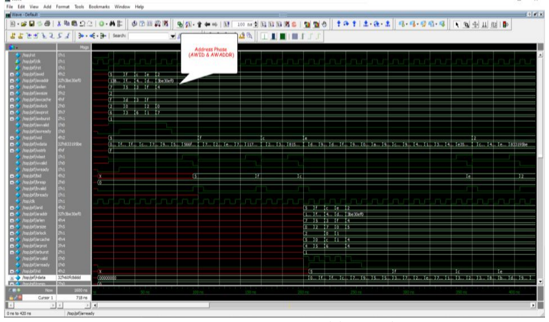


Fig. 2: AXI Address Phase

- AW Address Channel
- W Data Channel
- B Data Response Channel
- AR Read Address Channel
- R Read Data Channel

Each of the individual channels have their own handshaking mechanism. After matching of VALID and READY signals only data transfers are deemed to be logically correct transactions.

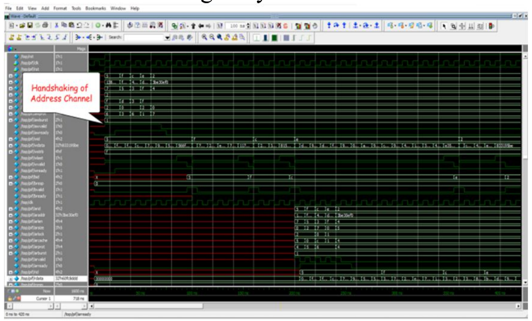


Fig. 3: Handshaking of Address Channel

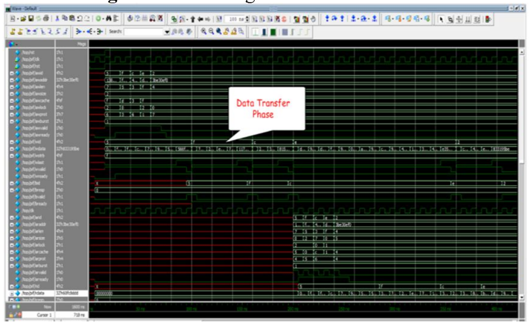


Fig. 4: Data Transfer Phase

After the transaction of the Address Phase, Data Phase starts. The number of data bytes transferred is dependent on the Burst Length and Strobe value which will determine the number of valid bytes in a 32-bit data transfer. The Data Transfer can have a maximum of 4KB address boundary. WLAST signal will be asserted at the end of data transfer indicating to the slave to stop address calculation. The simulation wavefroms are shown in Fig. 3 and 4.

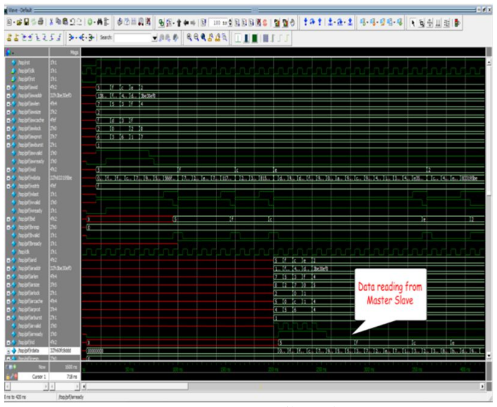
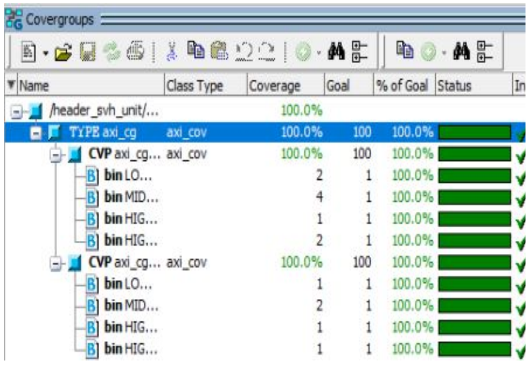


Fig. 5: Data Reading Phase

The data fed into the Slave DUT is now read by the master by mapping the Transaction ID and Slave address. The integrity of the transaction is checked by matching Write and Read Data operations for each Transaction ID. The bus width is 32 bits wide and for the values of transaction in which AWSIZE= 2 implies 22 bytes which is equivalent to 32 bits. Thus, 100 percent bus utilisation was captuted for transactions reflected in the coverage results section.

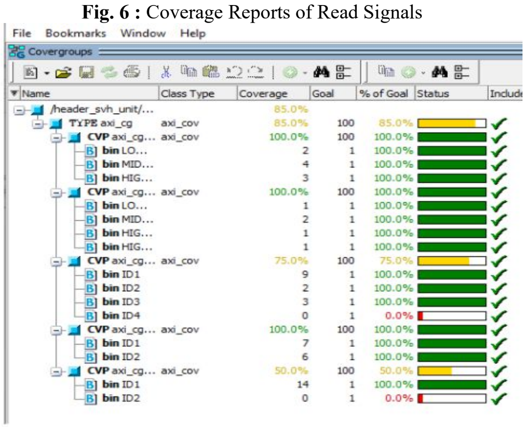
COVERAGE RESULTS:

The coverage harvested is 100 percent with respect to AWLEN and ARLEN lengths for both Read and Write Transactions. These reports are shown in fig 6 and 7.



Name	Class Type	Coverage	Goal	% of Goal	Status	In
/header_svh_unit/...		100.0%				
TYPE axi_cg	axi_cov	100.0%	100	100.0%	🟩	✓
CVP axi_cg...	axi_cov	100.0%	100	100.0%	🟩	✓
bin LO...		2	1	100.0%	🟩	✓
bin MID...		4	1	100.0%	🟩	✓
bin HIG...		1	1	100.0%	🟩	✓
bin HIG...		2	1	100.0%	🟩	✓
CVP axi_cg...	axi_cov	100.0%	100	100.0%	🟩	✓
bin LO...		1	1	100.0%	🟩	✓
bin MID...		2	1	100.0%	🟩	✓
bin HIG...		1	1	100.0%	🟩	✓
bin HIG...		1	1	100.0%	🟩	✓

Fig. 6 : Coverage Reports of Read Signals



Name	Class Type	Coverage	Goal	% of Goal	Status	Include
/header_svh_unit/...		85.0%				
TYPE axi_cg	axi_cov	85.0%	100	85.0%	🟨	✓
CVP axi_cg...	axi_cov	100.0%	100	100.0%	🟩	✓
bin LO...		2	1	100.0%	🟩	✓
bin MID...		4	1	100.0%	🟩	✓
bin HIG...		3	1	100.0%	🟩	✓
CVP axi_cg...	axi_cov	100.0%	100	100.0%	🟩	✓
bin LO...		1	1	100.0%	🟩	✓
bin MID...		2	1	100.0%	🟩	✓
bin HIG...		1	1	100.0%	🟩	✓
bin HIG...		1	1	100.0%	🟩	✓
CVP axi_cg...	axi_cov	75.0%	100	75.0%	🟨	✓
bin ID1		9	1	100.0%	🟩	✓
bin ID2		2	1	100.0%	🟩	✓
bin ID3		3	1	100.0%	🟩	✓
bin ID4		0	1	0.0%	🟥	✓
CVP axi_cg...	axi_cov	100.0%	100	100.0%	🟩	✓
bin ID1		7	1	100.0%	🟩	✓
bin ID2		6	1	100.0%	🟩	✓
CVP axi_cg...	axi_cov	50.0%	100	50.0%	🟨	✓
bin ID1		14	1	100.0%	🟩	✓
bin ID2		0	1	0.0%	🟥	✓

Fig. 7 : Coverage Reports of Write Signals

5. CONCLUSION

The primary conclusion derived in this paper is that the stimulus driven dynamic verification is inefficient and impotent approach to tackle the deadlock and livelock issues present inside an RTL Design. The Constrained Random Verification Stimulus generated is non-exhaustive in nature leading to coverage holes and bug escapes and design specification violations. In stark contrast, the deployment of Formal Verification ensures robust validation of all design features and in thwarting potential RTL violations aiding in early verification signoff and delivering bug free designs with faster time to market.

6. REFERENCES

[1]. AMBA AXI and ACE Protocol Specification AXI3, AXI4, and AXI4-Lite ACE and ACE-Lite https://developer.arm.com/documentation/ihi0022/e/

[2]. M.Graphics,"https://blogs.sw.siemens.com/verificationhorizons/2020/10/27/prologue-the-2020-wilson-research-group-functional-verification-study/",[Online: accessed 11-2-2022]

[3]. P. Ashar, "A paradigm shift in verification methodology," 2016 Formal Methods in Computer-Aided Design (FMCAD), Mountain View, CA, USA, 2016, pp. 6-6, doi: 10.1109/FMCAD.2016.7886652.

[4]. Carl Seger, An Introduction to Formal Hardware Verification, University of British Columbia, Vancouver, BC, Canada, 1992

[5]. N. Een, A. Mishchenko and R. Brayton, "Efficient implementation of property directed reachability," 2011 Formal Methods in Computer-Aided Design (FMCAD), Austin, TX, USA, 2011, pp. 125-134.

[6]. Aaron R. Bradley. 2011. SAT-based model checking without unrolling. In Proceedings of the 12th international conference on Verification, model checking, and abstract interpretation. Springer-Verlag, Berlin, Heidelberg, 70–87.

[7]. B. Ustaoglu, S. Huhn, F. Sill Torres, D. Große and R. Drechsler, "SAT-Hard: A Learning-Based Hardware SAT-Solver," 2019 22nd Euromicro Conference on Digital System Design (DSD), Kallithea, Greece, 2019, pp. 74-81, doi: 10.1109/DSD.2019.00021.

[8]. M. Siegel, "Achieving earlier verification closure using advanced formal verification," Formal Methods in Computer Aided Design, Lugano, Switzerland, 2010, pp. 275-275.

[9]. Hu, Yalin. "Exploring formal verification methodology for FPGA-based digital systems." Sandia National Laboratories, New Mexico, California (2012).

[10]. N. Fern and K. Cheng, "Evaluating Assertion Set Completeness to Expose Hardware Trojans and Verification Blindspots," 2019 Design, Automation & Test in Europe Conference & Exhibition (DATE), Florence, Italy, 2019, pp. 402-407, doi: 10.23919/DATE.2019.8714883.

[11]. N. Fern, I. San, Ç. K. Koç and K. Cheng, "Hardware Trojans in incompletely specified on-chip bus systems," 2016 Design, Automation & Test in Europe Conference & Exhibition (DATE), Dresden, Germany, 2016, pp. 527-530.

[12]. "IEEE Standard for SystemVerilog--Unified Hardware Design, Specification, and Verification Language," in IEEE Std 1800-2017 (Revision of IEEE Std 1800-2012) , vol., no., pp.1-1315, 22 Feb. 2018,

doi: 10.1109/IEEESTD.2018.8299595

[13]. R. M. Sarikhada and P. K Shah, "Speed up the validation process by formal verification method," 2020 IEEE International Conference for Innovation in Technology (INOCON), Bengaluru, India, 2020, pp. 1-4, doi: 10.1109/INOCON50539.2020.9298384.

[14]. N. Bombieri, R. Filippozzi, G. Pravadelli and F. Stefanni, "RTL property abstraction for TLM assertion-based verification," 2015 Design, Automation & Test in Europe Conference & Exhibition (DATE), Grenoble, France, 2015, pp. 85-90, doi: 10.7873/DATE.2015.0121.

[15]. T. Ghasempouri, A. Danese, G. Pravadelli, N. Bombieri and J. Raik, "RTL Assertion Mining with Automated RTL-to-TLM Abstraction," 2019 Forum for Specification and Design Languages (FDL), Southampton, UK, 2019, pp. 1-8, doi: 10.1109/FDL.2019.8876941.

[16]. B. Alizadeh, P. Behnam and S. Sadeghi-Kohan, "A Scalable Formal Debugging Approach with Auto-Correction Capability Based on Static Slicing and Dynamic Ranking for RTL Datapath Designs," in IEEE Transactions on Computers, vol. 64, no. 6, pp. 1564-1578, 1 June 2015, doi: 10.1109/TC.2014.2329687.

[17]. B. Alizadeh , R. Sharafinejad, and T. Nikoubin, "Formal Verification of Non-Functional Strategies of System-Level Power Management Architecture in Modern Processors," 2020 IEEE 14th Dallas Circuits and Systems Conference (DCAS), Dallas, TX, USA, 2020, pp. 1-6, doi: 10.1109/DCAS51144.2020.9330633.

[18]. P. Aggarwal, D. Chu, V. Kadamby and V. Singhal, "Planning for end-to-end formal using simulation-based coverage," 2011 Formal Methods in Computer-Aided Design (FMCAD), Austin, TX, USA, 2011, pp. 9-16.

[19]. G. Mahesh and S. M. Sakthivel, "Functional verification of the Axi2OCP bridge using system Verilog and effective bus utilization calculation for AMBA AXI 3.0 protocol," 2015 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), Coimbatore, 2015, pp. 1-5, doi: 10.1109/ICIIECS.2015.7193091.

[20]. N. Gaikwad and V. N. Patil, "Verification of AMBA AXI On-Chip Communication Protocol," 2018 Fourth International Conference on Computing Communication Control and Automation (ICCUBEA), Pune, India, 2018, pp. 1-5, doi: 10.1109/ICCUBEA.2018.8697587.

[21]. C. Chen, J. Ju and I. Huang, "A synthesizable AXI protocol checker for SoC integration," International SoC Design Conference, Seoul,, pp. 103-106, doi: 10.1109/SOCDC.2010.5682961.

[22]. Prasad, Rama & Paradhasaradhi, Damarla & Madan, G & Reddy, Sankar & Karumuri, Srinivasa Rao & Prabhakar, V & Student, P. (2018). Design and verification of axi apb bridge using system Verilog. Journal of Advanced Research in Dynamical and Control Systems. 10.

[23]. Siddhan, Ravi & K, Ezra & Kittur, H. (2014). Design of a Bus Monitor for Performance Analysis of AXI Protocol based SoC Systems. International Journal of Applied Engineering Research. 9. 6313-6324.