COVERAGE DRIVEN FUNCTIONAL TESTING ARCHITECTURE FOR PROTOTYPING SYSTEM USING SYNTHESIZABLE ACTIVE AGENT

Dipakkumar Modi and Usha Mehta

EC Department, Institute of Technology, Nirma University, Ahmedabad, India

ABSTRACT

Time and efforts for functional testing of digital logic is big chunk of overall project cycle in VLSI industry. Progress of functional testing is measured by functional coverage where test-plan defines what needs to be covered, and test-results indicates quality of stimulus. Claiming closer of functional testing requires that functional coverage hits 100% of original test-plan. Depending on the complexity of the design, availability of resources and budget, various methods are used for functional testing. Software simulations using various logic simulators, available from Electronic Design Automation (EDA) companies, is primary method for functional testing. The next level in functional testing is pre-silicon verification using Field Programmable Gate Array (FPGA) prototype and/or emulation platforms for stress testing the Design Under Test (DUT). With all the efforts, the purpose is to gain confidence on maturity of DUT to ensures first time silicon success that meets time to market needs of the industry. For any test-environment the bottleneck, in achieving verification closer, is controllability and observability that is quality of stimulus to unearth issues at early stage and coverage calculation. Software simulation, FPGA prototype, or emulation, each method has its own limitations, be it test-time, ease of use, or cost of software, tools and hardware-platform. Compared to software simulation, FPGA prototyping and emulation methods pose greater challenges in quality stimulus generation and coverage calculation. Many researchers have identified the problems of bug-detection / localization, but very few have touched the concept of quality stimulus generation that leads to better functional coverage and thereby uncover hidden bugs in FPGA prototype verification setup. This paper presents a novel approach to address above-mentioned issues by embedding synthesizable active-agent and coverage collector into FPGA prototype. The proposed architecture has been experimented for functional and stress testing of Universal Serial Bus (USB) Link Training and Status State Machine (LTSSM) logic module as DUT in FPGA prototype. The proposed solution is fully synthesizable and hence can be used in both software simulation as well as in prototype system. The biggest advantage is plug and play nature of this active-agent component, that allows its reusability in any USB3.0 LTSSM digital core.

KEYWORDS

Testing, Functional Coverage, Synthesizable Active Agent, Universal Serial Bus (USB), Link Training and Status State Machine (LTSSM)

1. INTRODUCTION

In recent times the complexity of the VLSI designs has increased rapidly with advancements in nanotechnology and gate capacity from million gates to billion gates in a chip. High-speed serial communication interfaces such as USB and PCIe provides high data transfer capabilities which is a need of modern era applications. In cycle accurate and time-based software simulations, the testing-time becomes bottleneck while running iterative simulations to achieve functional coverage and detect bugs in time, and overall it affects project schedule

DOI : 10.5121/vlsic.2018.9304

The next level of verification with prototyping and emulation systems provide accelerated verification platforms to run time-sensitive scenarios. For high-speed serial protocols, there are two time-sensitive test areas; a) link-training and low power phase b) data-transfer phase. Verification in FPGA prototyping system greatly help stressing out the DUT with heavy data traffic stimulus and reduces the test-time for long and iterative simulations. The DUT can be simulated by injecting heavy data-traffic with different types of data, and number of data-packets with varying length to uncover more corner scenarios. The data transfer test-scenarios are driven from higher-level in the prototyping system, where user has better controllability and observability for closed-loop testing and hence proper functional coverage can be realized. However, for link-training related test scenarios there is very limited access where generating right stimulus and achieving functional coverage is a big challenge.

The USB3.0 LTSSM, many other standard protocols, has timers ranging from 10 µsec to 300 msec for link-training or low power state transitions. Stress testing and coverage for such huge time sensitive scenarios is herculean task in software simulation, and its unreachable in standard prototype system. This paper proposes architecture where time-sensitive link-training and low power states logic is targeted with an embedded active-agent in prototype system that uses advanced error injection. In a proposed architecture, the active agent controls the stimulus and coverage collector provides observability for closed-loop functional testing of such time-sensitive scenarios. The technique used in this architecture focuses on stress testing time sensitive LTSSM transition arcs and mutation coverage where injected error is observed for predictable behaviour of DUT. This approach is feasible for many such prototyping systems where lower layers of DUT can be stress tested using synthesizable active-agent such as proposed in this paper.

2. BACKGROUND AND MOTIVATION OF RESEARCH

In a typical test-environment of prototyping system, traffic injection to DUT is done in two ways; a) through a test-component b) using a standard real device.

For USB designs when the DUT is USB target device, the test-component at other end could be a proprietary hardware from Electronic Design Automation (EDA) company, that mimic real USB-host component. The DUT in such emulated system may run at slower operating clock and sees the test-component as protocol compliant device at the other end. Such test-components provide greater flexibility in generating desired stimulus. However, these high-end proprietary test-components are costly, and interface requirements with DUT are complex. Moreover, if multiple USB components exist in a system, multiple such test-components would be required to stress test the system concurrently.

Prototyping system with a real device against a DUT is comparatively easy to build and cost effective. Such test setup also helps early software development and testing with real components. The advantage is that all the testing happens in real world and at-speed. The drawback is controllability of stimulus as the other end of the DUT is actual device where it behaves as per the protocol defined. In such prototyping system heavy data-traffic can be injected with upper layer application control. For USB link, when the DUT is USB target device and other end of the link is real USB-Host, host side application can inject various types of transfers and data traffic. However, there is very minimum control over how the lower layer of the design is stimulated. Lower level module for USB-target device is link layer logic with LTSSM. Also, negative testing options are very limited. Now, here is where the opportunity lies for improvement in the prototyping system that became motivation of our research work.

The architecture presented here provides controllability of stimulus with synthesizable active-agent and observability with coverage collector for link-layer LTSSM of the USB3.0 target

device DUT. The prototype of the proposed architecture is operational and presents promising results with better stimulus generation. The scope of improvement and future work is to reduce resource overhead in the prototyping system to provide competitive edge to this approach. The same approach is feasible for many such prototyping systems where lower layers of design can be stress tested using synthesizable active-agent such as proposed in this paper.

3. RELATED WORK

Authors from academia and industry have researched in area of pre-silicon verification, synthesizable test-components, controlled stimulus, issue detection, and coverage collection. All the work relates to improving effectiveness of testing and its suitability to specific application.

F. Moraes et al. in the paper [1], "A generic FPGA emulation framework" perfectly highlights challenges associated with software-simulation based testing and proposes generic emulation framework to improve controllability and observability of design under verification (DUV). The presented structure is combination of software and hardware components. It uses host-PC for higher level control of stimuli and the tests are driven to DUV via standard interfaces like PCIe or Ethernet.

S. S. Shankar et al. in paper [2], "Synthesizable Verification IP to Stress Test System On-Chip Emulation and Prototyping Platforms", authors have highlighted the challenges associated with pre-silicon verification and proposed synthesizable verification architecture compliant with Standard Co-Emulation Modelling Interface (SCE-MI) infrastructure through which the protocol specific traffic is injected at industry standard interfaces. This approach is based on combination of high-level test sequences and protocol specific transactors implemented with synthesizable models. The paper presents the approach with USB2 logic with Transceiver Macrocell Interface (UTMI) interface as design under test. This approach highly relies on directed testing and the stimulus is controlled from higher layer test-sequences.

H. Krrikyan et al. in paper [3], "Prototyping system for USB3.0 link layer using synthesizable assertions and partial reconfiguration", presents problems related to bug detection/localization and coverage calculation. The proposed solution is based on embedding synthesizable assertions into prototype and the experiment was done on USB3.0 link layer FPGA prototype. The paper focuses on observability aspect of verification with assertion-based testing in prototype system. Synthesizable assertions allow its usability in both software-based simulation test-environment and FPGA prototype /emulation platforms. The experiment utilizes FPGA partial configuration feature for Xilinx architecture-based FPGA and tool set.

O. Amin et al. proposes an architecture in paper [4], "System Verilog Assertions Synthesis Based Compiler", helps assertion-based verification idea for prototyping systems. The non-synthesizable code written as SystemVerilog Assertions (SVA) in software-simulation based test-environment can be converted to synthesizable code that can be ported on FPGA. Again, this approach focuses on observability part of DUT verification and is used to watch how the design performs in testing setup.

A. Yehia et al. in paper [5], "Faster coverage closure: Runtime guidance of Constrained Random stimuli by collected coverage" proposes closed-loop verification concept to improve the coverage and make verification more efficient. This approach is targeted for software simulation based functional testing where stimuli based on constrained random verification uses data from coverage collector to optimize the testing run time. It shows how collected coverage can dynamically guide random stimuli generators during simulation to generate uncovered scenarios

and avoid generating redundant ones. Similar concept is applied in our architecture for the prototyping system.

4. PROPOSED ARCHITECTURE

Presented in Figure 1. is use case showing active agent placement in overall FPGA prototype system. The prototyping system on Xilinx Kintex®-7 FPGA, constitutes USB3.0 Device controller as DUT, the application layer of DUT is supported by standard mass storage device driver running on local processor with Advance Microcontroller Bus Architecture (AMBA) - Advanced eXtensible Interface (AXI) based system-bus. The link layer of DUT connects to USB3.0 PHY via intel PHY Interface for PCIe and USB (PIPE). The other side of the USB link is standard host with USB3.0 host port that generates stimulus while running real bulk-transfer application. Building such prototyping system is easy and cost effective compared to protocol specific high-end emulators or test-components. The objective of our prototyping system is to address time sensitive stimulus generation and stress test the DUT. Here data intensive traffic is injected from standard USB host and the application can inject desired stimulus. To address stimulus generation for lower layer logic area, LTSSM in this case, the synthesizable active agent plays a role.

The synthesizable active-agent design is based on Universal Serial Bus (USB) specification [6] and PIPE [7] specification. The configuration space access of the USB device IP core is accessed through AXI-Lite interface that follows AMBA-AXI specification [8]. The active agent intercepts the PIPE interface signalling of USB3.0 link layer and injects error to create specific test scenario that otherwise wouldn't have occurred in normal prototype test-environment setup.

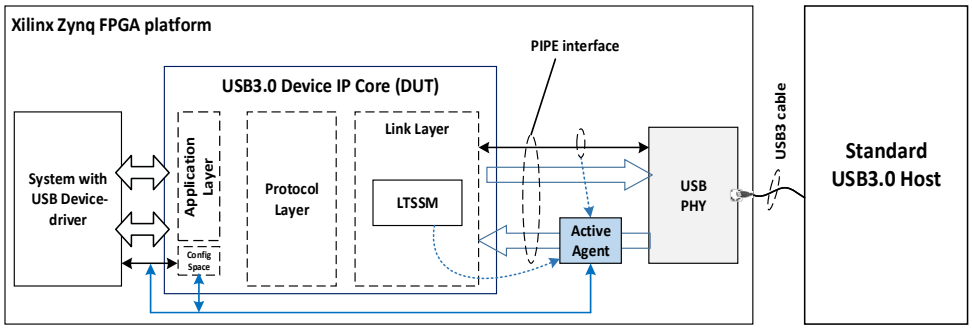

Figure 1. Active-agent in prototyping system for USB3 LTSSM

The link Layer interface with PHY is Intel PIPE interface having unidirectional data-flow that allows adding pipe-line stages to insert our active-agent, without affecting specification timing requirements. The active-agent is placed on the receive path at upstream port (USP) PIPE interface where USB3.0 Device IP core connects with standard USB3.0-PHY.

This active agent is a plug-and-play component and work with any USB3.0 controller DUT with just knowledge about the core's LTSSM current state value that is available as part of debug port in typical FPGA prototype system. The active-agent maintains scenario coverage information and accordingly injects anomaly at available opportunity in an ongoing test. For simplicity This paper focuses on specific portion of LTSSM and the same would be extended to cover more scenarios for stress testing of complete LTSSM in an automated way.

In traditional simulation-based verification architecture there are passive agents acting as protocol checker, scoreboard data collector, or functionality monitors that are vital components in overall

closed-loop verification. Also, call-back mechanism is used to inject error in the transfer by manipulating the information and thus, creating specific directed scenario. Similarly, active-agent manipulates information, within specification boundaries, at standard interface thereby deviating from standard state-transition of DUT. It is obvious that this active-agent needs some extra information, to be more effective, rather than purely taking the DUT as black-box. This information helps find appropriate opportunity to inject variation in the received information from other end of the USB-link and execute certain directed scenarios.

Active-agent architecture incorporates most part of decode logic in USB link layer that includes descrambler, CRC, LFPS detection, and de-framing. It adds pipe-line stages to the receive datapath to cover for decode latency and allow time to inject variation at right time. The active-agent also implements configuration space with information on masking specific scenario injection and collecting coverage statistics.

Major functional blocks are shown in the architectural diagram Figure 2. of active agent.

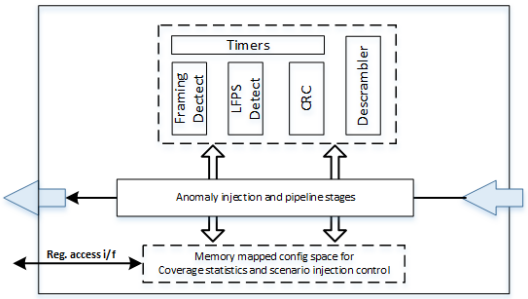

Figure 2. USB3.0-LTSSM Active Agent Architecture

4.1. DESCRAMBLER

Before the information can be identified to act upon by the agent it must be decoded at the receive path of PIPE interface, and the first stage is descrambling. It's based on free running linear feedback shift register (LFSR) and if enabled the logic follows USB3.0 specification with polynomial;

$$G(x) = x^16 + x^5 + x^4 + x^3 + 1$$

Control (K codes) symbols are bypassed and the LFSR is reset to FFFF at every COM symbol. If the PHY contains this logic, it can be easily bypassed from the agent logic.

4.2. FRAMING DETECTION

After descrambling the decode logic performs detection of framing pattern and special symbols. The sliding pattern match mechanism is used to detect symbol position in aligned or unaligned packet. Framing detection is performed for Link Commands (LC), Header Packet (HP), Data Packet (DPP, END/ENDB), and Ordered Sets (OS). The packets are checked for CRC and after anomaly injection the CRC is recalculated to re-validate the packet.

4.3. LFPS DETECT

Low Frequency Periodic Signalling (LFPS) detection is performed as per USB3.0 specification and is primarily used for low power mode exit and warm-reset. The details of normative LFPS

implementation can be found in the USB specification. The latency increases by few clock cycles while the decode logic is active before it acts for scenario injection based on constraints. This does not impact the functionality of the DUT neither it violates USB3.0 or PIPE specification.

4.4. CONFIGURATION SPACE

The active agent also has configuration space that is accessible through simple register space access interface that is memory mapped in local processor. The configuration spaces access is used to fetch coverage statistic and to enable or disable specific scenario injection logic and to configure and modify constraints of scenario injection driver.

4.5. ERROR INJECTION AND PIPELINE STAGES

This block is part of pipeline stage and performs error injection in receive path with manipulated information. The required information for manipulation and content update is received from other modules. Scenario injection logic acts based on the information from coverage statistics that help update the constraints of scenario injection logic for uncovered state transitions during future test iterations. Coverage constraints applicability is restricted to state-transitions where occurrence of transition conditions can be varied in time. Manipulating receive path information is tricky and requires extra care while exercising this approach staying within boundaries of USB3.0 specification. The implementation is simple once the concept is understood and the efforts put in developing such active-agent are worth as the results seen are promising for coverage closer on prototype system.

Table 1. Xilinx-Kintex7 FPGA series resource utilization

Resource	Slice LUTs	Slice Registers	RAMB36
USB-Device	21764	12766	16
Active agent	2840	810	0
% of DUT	*13%*	*6%*	*0%*

5. VERIFICATION AND RESULTS

As per USB3.0 specification there are total of 20 LTSSM state and sub-states for USP with more than 85 possible transition arcs where 20 transitions are associated with ‘directed’ from upper layer through warm-reset. From rest of the transition arcs, less than 20 transition arcs are usually covered in normal prototype environment. There is scope of error injection, within specification boundaries, to stress test the DUT and the active agent plays effective role. All the LTSSM states and sub-state transitions are tracked and coverage information is collected at state entry event. During the research work specific set of test conditions were targeted for this experiment.

A simple event of cable connect-disconnect can exercise more test-condition and increase the DUT coverage. However, performing this connect-disconnect test manually and expecting it to occur at specific point in time at all possible opportunities, is practically impossible in typical FPGA prototype platform. With active agent this is made possible. USB device can be connected or disconnected from host port at any time and cover all reachable scenarios on existing prototype platform. At PIPE interface the connect-disconnect event is stimulated using relevant input signals as listed in table 2.

By overriding inputs signals to DUT at PIPE interface device connect-disconnect event is tested for each all LTSSM states and sub-states. The active-agent has helped significantly in uncovering multiple corner conditions that did not get caught during normal testing on the prototype system.

Table 2. PIPE interface signals for active-agent

Signal	Direction	Description
RxDataValid	Input	Instruct the Link layer DUT logic to ignore the data interface for one clock cycle. A value of one indicates the upper layer logic will use the data, a value of zero indicates it will not use the data.
RxValid	Input	Indicates symbol lock and valid data on RxData and RxDataK
PhyStatus	Input	Used to communicate completion of several PHY functions including stable PCLKafter Reset# de-assertion, power management state transitions, rate change, and receiver detection. When this signal transitions during entry and exit from any PHY state where PCLK is not provided, then the signalling is asynchronous.
RxElecIdle	Input	Indicates receiver detection of an electrical idle. While de-asserted with the PHY in P0, P1, P2, or P3, indicates detection of LFPS
RxStatus[2:0]	Input	Encodes receiver status and error codes for the received data stream when receiving data. 3'b000 : Received data OK 3'b001 : 1 skip OS added 3'b010 : 1 skip OS removed 3'b011 : Receive detected 3'b100 : 8b10b decode error 3'b101 : EB overflow 3'b110 : EB underflow 3'b111 : Receiver disparity error
PowerPresent	Input	Indicates the presence of VBUS

Scenario injection is simpler and more effective while exercising negative conditions for state transition. Let's understand this with one of the core state of the 'LTSSM: Polling' state and sub-states. Polling.LFPS is a sub-state designed to establish the PHY's DC operating point for LFPS operation, and to synchronize the operation between the two link partners after exiting from Rx.Detect. This is also a sub-state for a port to identify itself based on various Polling.LFPS signatures.

Figure 3. Shows polling sub-states of USB LTSSM. Let's understand how the active agent generates desired stimulus and help generate quality stimulus. In normal operation, with standard prototype system, the Polling state moves straight forward through Polling.LFPS -> Polling.RxEQ -> Polling.Active-> Polling.Configuration-> Polling.Idle-> U0. There are multiple other functionally possible test conditions that are not reachable in standard prototype system.
When the scenario injection is enabled the active agent finds opportunity of injecting specific condition when the LTSSM enters Polling.LFPS. So apart from its normal route that is Polling.LFPS → Polling.RxEQ, the LTSSM is stress tested for error conditions in Polling.LFPS state and for transitions to SS.Disabled or Compliance. There is a provision to enable or disable scenario injection by masking condition via configuration space access interface.

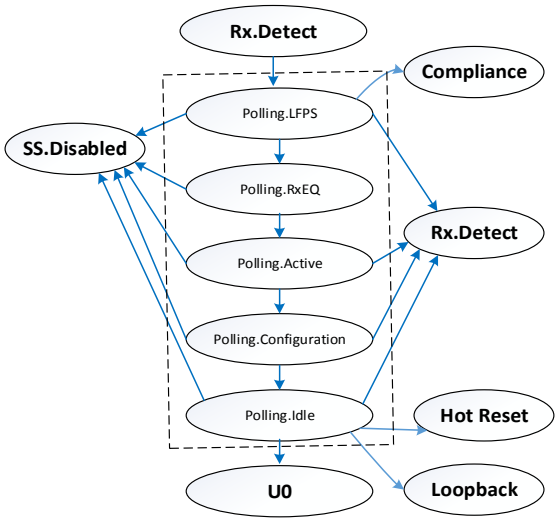

Figure 3. USB3.0-LTSSM – Polling sub-states

Table 3. lists multiple functional testing scenarios for Polling.LFPS sub state in UpStream Port component. It is evident from the list of possible scenarios that just state transition coverage is not sufficient to gain confidence before claiming closer in pre-silicon verification with FPGA prototype systems.

Table 3. Polling.LFPS Test scenarios and Active agent

#	Next state	Test Condition and role of Active agent
1	Polling.RxEQ	• 16 consecutive Polling.LFPS bursts sent • two consecutive Polling.LFPS bursts received and four consecutive Polling.LFPS bursts sent after receiving one Polling.LFPS burst
2	Polling.RxEQ	• Corrupt received LPFS at every alternate burst and allow handshake completion just before the tPollingLFPSTimeout
3	Polling.RxEQ	• Continue corruption of LFPS at various stages of LFPS burst, for multiple iterations of this test, allow successful handshake at different time before tPollingLFPStimeout
4	eSS.Disabled	• 360ms tPollingLFPSTimeout after having trained once since PowerOn Reset • Conditions to transition to Polling.RxEQ are not met
5	Compliance	• 360ms tPollingLFPSTimeout • port has never successfully completed Polling.LFPS after PowerOn Reset • Conditions to transition to Polling.RxEQ are not met

The real benefit of active agent is not just covering all the state transition of DUT on FPGA prototype, rather its ability to inject error without violating the specification to stress test specific state of LTSSM.

Table 4. presents improved results with active-agent generated stimulus for the DUT.

Table 4. Polling.LFPS coverage

Transition arc		Test	Coverage	
Current State	Exit State	#	Default	With Active agent
Polling.LFPS	Polling.RxEQ	1	1	1
Polling.LFPS	Polling.RxEQ	1a	0	1
Polling.LFPS	Polling.RxEQ	1b	0	1
Polling.LFPS	Compliance	2	0	1
Polling.LFPS	SS.Disabled	3	0	1

Irrespective of what method is chosen for offline simulation, pre-silicon verification, or post-silicon testing, important aspects of any verification idea are controllability, and observability. In proposed method, this is covered to some extent with control of scenario injection and coverage collector. However major concern with such non-trivial testing methodology for FPGA prototype is trouble shooting and bug localization. To manage this concern, a simple debug mechanism was created. Explained in brief; local memory was added, with configurable depth, that logs state transitions and the contents rollover when the test is run for longer period beyond its storage capacity. Interrupt is asserted to local processor for initiating the memory-read when entry to Rx.Detect occurs for warm-reset and was not intended as per scenario injection. The content of memory gives transaction log that helps understand state transitions. In our experiment we've debug port from DUT that provides all important error flags due to which the state transitions deviate from its expected route. This information helps in bug-localization.

6. CONCLUSION

The stress testing and coverage improvement using proposed architecture is demonstrated in this paper with sample experiment. The proposed approach is very efficient in uncovering hidden bugs without inviting much in hardware or software infrastructure resources. The experiment was done on limited scenarios of LTSSM, however more scenarios would be added in future to exercise further stress testing with varying state exit and error injection. This active agent is reusable component and can be safely used with any USB3.0 USP DUT. The same methodology can be applied to USB3.0 DSP, and for that matter, any standard protocol interface for the purpose of stress testing and finding corner scenarios at earliest with less effort on FPGA prototype systems.

ACKNOWLEDGEMENTS

The authors would like to thank Softnautics LLP. for their supports and providing FPGA prototype platform along with all the associated infrastructure to test active-agent. The authors would also like to thank Dr. D. J. Shah for his guidance.

REFERENCES

[1] F. Moraes et al., (2012) "A generic FPGA emulation framework", 19th IEEE International Conference on Electronics, Circuits, and Systems, pp. 233-236.

[2] S. S. Shankar & J. S. Shankar, (2011) "Synthesizable verification IP to stress test system-on-chip emulation and prototyping platforms", International Symposium on Integrated Circuits, Singapore. pp. 609-612.

[3] H. Krrikyan, T. Hovhannisyan and S. Manukyan, (2015) "Prototyping system for USB3.0 link layer using synthesizable assertions and partial reconfiguration", Computer Science and Information Technologies (CSIT), pp. 19-22.

[4] O. Amin, Y. Ramzy, O. Ibrahem, A. Fouad, K. Mohamed and M. Abdelsalam, (2016) "System Verilog Assertions Synthesis Based Compiler," 17th International Workshop on Microprocessor and SOC Test and Verification (MTV), pp. 65-70.
[5] A. Yehia, (2013) "Faster coverage closure: Runtime guidance of Constrained Random stimuli by collected" Saudi International Electronics, Communications and Photonics Conference, pp. 1-6.
[6] Universal Serial Bus Rev3.2 Specification, September 2017. Available: https://www.usb.org
[7] PHY Interface For the PCI Express, SATA, and USB3.1 Architectures Specification v5.0. Available: https://www.intel.com
[8] AMBA AXI-4 Specification. Available: https://www.arm.com
[9] Abhishek Jain, Piyushkumar Gupta, Hima Gupta, Sachin Dhar, (2013) "Accelerating SystemVerilog UVM based VIP to improve Methodology for Verification of Image Processing Designs using HW emulator" International Journal of VLSI design & Communication Systems (VLSICS), Vol. 4, No.6, pp. 13-25.
[10] Sateesh Kumar H.C., Sayantam Sankar, Satish S Bhairannawar, Raja K.B. Venugopal K.R., (2015) "FPGA Implementation of Moving Object and Face Detection using Adaptive Threshold" International Journal of VLSI design & Communication Systems (VLSICS), Vol. 6, No.5, pp. 15-35.
[11] N. Mohamad, C. Y. Ooi, N. Ismail and J. Teh, (2016) "SVA checker generator for FPGA-based verification platform" IEEE International Symposium on Circuits and Systems (ISCAS), pp. 1750-1753.
[12] B. Vermeulen and S. K. Goel, (2002) "Design for debug: catching design errors in digital chips" IEEE Design & Test of Computers, vol. 19, no. 3, pp. 35-43.
[13] A. M. Gharehbaghi, M. Fujita, (2011) "Transaction-based debugging of system-on- chips with patterns", Proc.of IEEE International Symposium on Quality Electronic Design, pp. 1 -6.
[14] C. H. Chen, J. C. Ju and I. J. Huang, (2010) "A synthesizable AXI protocol checker for SoC integration," International SoC Design Conference, pp. 103-106.
[15] Tan J, Fresse V, Rousseau F, (2011) "Generation of emulation platforms for NoC exploration on FPGA" 22nd IEEE International Symposium on Rapid System Prototyping (RSP), pp. 186-192.

AUTHORS

Dipakkumar Modi received the BE degree in instrumentation and control engineering from L. D. College of Engineering, Ahmedabad, Gujarat in 2001. He is currently working with Softnautics LLP as technical director IP group. He has worked in VLSI industry for more than 16 years, his research interests are in field of SoC design, hardware accelerated verification, and emulation.

Prof. Usha Mehta has done her Ph. D. in the area "Testing of VLSI Design" and is currently working as Professor at Nirma University, Ahmedabad. She has more than 22 years of academic and industrial experience. She has one patent, has published a book and more than 50 research papers in international journals and conferences. She is a senior member of IEEE and currently holds the position of chair, IEEE WIE Affinity group, Gujarat section. Her research interests are in testing VLSI Design, Digital VLSI Design, and programmable Logic.