Development of JTAG Verification IP in UVM Methodology

Milna M. J.[1] **Deepa N. R.**[2]
[1]M. Tech(Student) [2]Asst. Professor
[1, 2]Electronics & Communication Engineering Department
[1, 2] FISAT

Abstract—IEEE 1149.1/1149.6 (JTAG) Verification IP provides a smart way to verify the IEEE 1149.1/1149.6 (JTAG) component of a SOC or an ASIC. The SmartDV's JTAG Verification IP works in a highly randomized manner to generate wide range of scenarios for effective verification of DUT(device under test).JTAG VIP includes an extensive test suite covering most of the possible scenarios and detection of protocol violation using an effective protocol-checker. IEEE 1149.1/1149.6 (JTAG) VIP is supported natively in System Verilog, VMM, RVM, AVM, OVM, UVM, Verilog, SystemC, VERA, Specman E and non-standard verification environment.

I. INTRODUCTION

In today's complex systems, testability is an increasing concern in almost every application and in every area of application development. Manufacturers that thoroughly address the issue of testability at the device, board, and system levels deliver more consistently reliable and cost-effective products to the marketplace. This means building in test capabilities in every phase of development and deployment, including design verification, hardware and software integration, manufacturing, and in the field.

II. JTAG HISTORY

In the 1980s, the Joint Test-Action Group (JTAG) formed by representatives from makers and users of components and boards, recognized that only a cooperative effort could address the mounting testability problems in a coordinated way. Its mandate was to propose design structures that semiconductor makers would incorporate into device designs to aid in testing boards and systems. In 1990 the IEEE adopted the proposal as IEEE Standard 1149.1. Its stated purpose was to test interconnections between Integrated Circuits (ICs) installed on boards, modules, hybrids, and other substrates. Manufacturers adopting the standard could also test the IC itself.

III. JTAG ARCHITECTURE

Architecture IEEE Standard 1149.1 is a testing standard. However it is described as a collection of design rules applied principally at the IC level that allow software to alleviate the growing cost of designing and producing digital systems. The primary benefit of the standard is its ability to transform extremely difficult printed circuit board testing problems that could be attacked with ad-hoc testing methods into well-structured problems that software can easily and swiftly deal with. To conform to the boundary-scan standard IEEE 1149.1, a device must contain the following:

A) Test Access Port Controller (TAP)
B) Scan Instruction Register
C) Scan Data Registers

A. THE TAP CONTROLLER

The TAP controller is a finite state machine with a state diagram containing sixteen (16) states. A transition between states only occurs on a rising edge of the Test Clock (TCK) or asynchronously with the assertion of Test Reset (TRST*) if it exists.

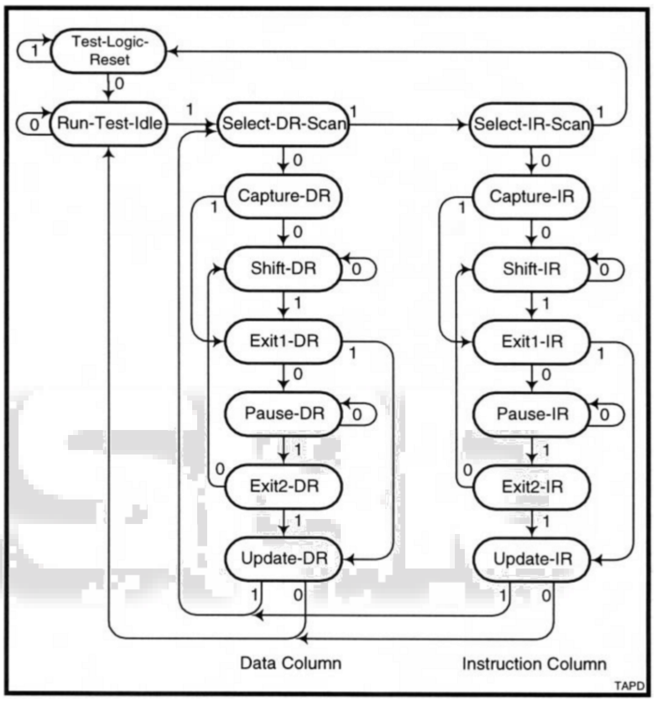

Fig. 1: JTAG TAP Controller

IV. EXISTING TECHNIQUES

Existing design are verified either through their FPGA prototypes or when the real silicon comes out. For JTAG interfaces which will be part of a very complex SOC, it takes long time to find out a bug when it is verified in FPGA or in the real silicon as the design cycle itself consumes more than six months. Finding the issue after this time may even kill the product since the average life cycle of a product is now reduced to around 6 months. Therefore the emphasis now is to verify the design during the RTL development phase itself. In verification, to mimic the actual environment around the design under test, Bus Functional Models (BFM) of different interfaces is required. JTAG interface is a very common interface which will be present in almost 90% of todays ASIC/ SOC's. Existing JTAG BFM's which are present in the market will be either developed without any methodologies i.e. Pure Verilog or with methodologies like VMM, OVM, RVM etc.

JTAG BFM's developed with UVM methodology overcomes the tool dependencies. It also incorporates the best practices from other methodologies like OVM, VMM etc. It also has other features like the factory mechanism which helps us to easily override the various classes used in the environment. Interaction between driver and the sequencer are taken care which reduces the coding effort. It has inbuilt print () functions which reduces the coding required for debug.

The development of JTAG BFM in UVM methodology helps the verification engineer of a complex SOC to develop a JTAG block which can be used in any design/ SOC's at a faster pace and much importantly with more reliability. The UVM class library brings much automation to the System Verilog language such as copy, compare etc. Also this BFM is supported by multiple Tool vendors like Aldec, Cadence, Mentor, and Synopsys.

V. IMPORTANCE OF METHODOLOGY

Methodology shrinks verification efforts with its predefined libraries. It will make our life easier down the road by preventing us from making mistakes or poor decisions whose outcome may not be obvious at the time you made them. It also helps make sure that whatever you do will mesh nicely with what others do (re-usability) .Methodology is basically set of base class library which we can use to build our test benches. The verification giants have already defined such common classes and functions which we will mostly require to build test benches and have formed one library. That's it; they call this base class library a methodology.

VI. GUIDELINES FOR SUCCESSFUL SOC VERIFICATION IN UVM

With the increasing adoption of OVM/UVM, there is a growing demand for guidelines and best practices to ensure successful SoC verification. It is true that the verification problems did not change but the way the problems are approached and the structuring of the solutions, i.e. verification environments, depends much on the methodology. There are two key categories for SoC verification guidelines: process, enabled by tools, and methodology. The process guidelines are about what you need to do and in what order, while the methodology guidelines are about how to do it. This paper will first describe the basic tenets of OVM/UVM, and then it tries to summarize key guidelines to maximize the benefits of using state of the art verification methodology such as OVM/UVM.

A. QUALIFY YOUR VIPS

Qualify your VIP during development and before releasing them. First, several tools can conduct static checking on your VIP components for common errors and conformance to coding styles. They can also provide statistics about the size of your code, checks and cover-groups. Second, typically a simulator can provide statistics about memory consumption and performance bottlenecks of your VIP. Although System Verilog has automatic garbage collection, you can still have memory leaks because you keep a reference to dynamically allocated objects somewhere and forget about them. Third, your VIPs should be robust to user mistakes whether in connections or proper use. You need to have sanity checks that can flag early a user error. Finally, peer review is still beneficial to point-out issues that are missed in the other steps.

VII. VERIFICATION COMPONENTS DEVELOPED

The major components developed are TOP ENV, TESTBENCH, JTAG sequence library and random test case. TOP ENV consists of JTAG ENV, which contains JTAG agent. JTAG agent comprises a JTAG Monitor, JTAG Sequencer and also JTAG driver. The JTAG driver receives the packet from the sequencer and drives it to the JTAG interface.

JTAG sequencer sends the packet to the JTAG driver and waits until the driver sends the packet to the interface. JTAG sequence library contains the various combinations of JTAG packet which can be driven to the DUT. DUT register database has the replica of DUT registers. JTAG driver updates these register whenever DUT register is updated. JTAG monitor reads the register value from these register whenever a DUT register is read and compares the values.

Test bench instantiates the DUT and connects the DUT signals to the JTAG verification interface. Test case selects the sequences from the sequence library which needs to be driven to the Design under test.

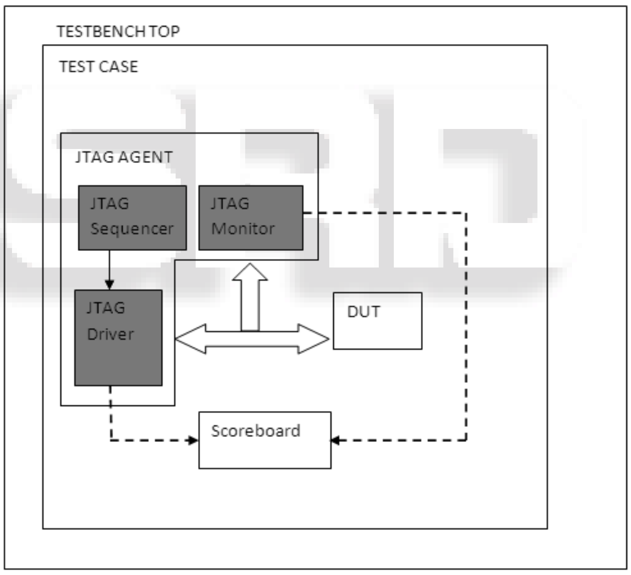

Fig. 2: Test bench architecture

VIII. FLOW OF TESTBENCH

Test case calls one of the random sequences inside the jtag_seq_lib (JTAG sequence library). jtag_seq_lib generates a sequence and sends to the sequencer and waits for the driver to send the transfer to the JTAG interface. Driver acknowledges the sequencer that data was sent correctly to the JTAG interface. The driver acknowledges the sequence library sends the next sequence to the sequencer and the above steps are repeated.

JTAG monitor verifies the following protocols in the IEEE 1149.1 in addition to verifying the TAP controller state machine transitions. Instruction register LSB bits are assigned the values “01”. During Shift IR state, the design drives serially TDO from the instruction register value on the rising edge of TCK. During Shift IR state, the design also shifts the new instruction bits into the Instruction Register from TDI. During Shift DR state, the design drives serially TDO from the instruction register value on the rising edge of TCK. During Shift DR state, the design also shifts the new instruction bits into the Instruction Register from TDI. During Capture DR state, data can be parallel-loaded into the shift portion of the data register selected by the current instruction on the rising edge of TCK. During Capture IR state, data can be parallel-loaded into the shift portion of the instruction register. During UPDATE-DR state for Read commands the shifted data from the SHIFT DR state won't be updated i.e. it will hold the previous data before the read command.

IX. RESULTS

The simulation of this project without implementing UVM methodology has been done using the software modelsim altera starter edition 6.4a. It supports multiple languages including Verilog, SystemVerilog, VHDL and SystemC.

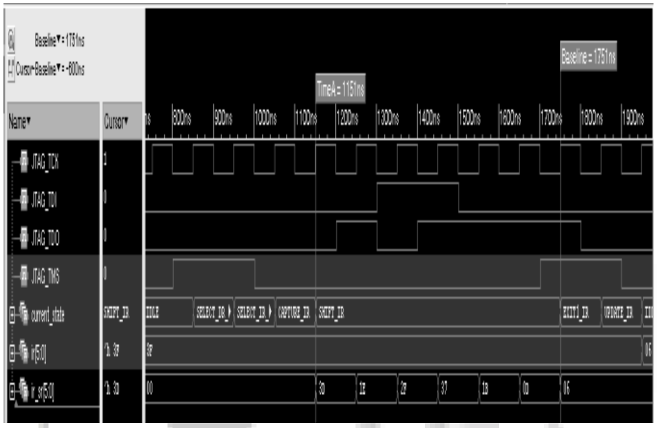

Fig. 3: simulation in UVM environment using cadence tool for Instruction OP_WR_TESTMODES (0x06)

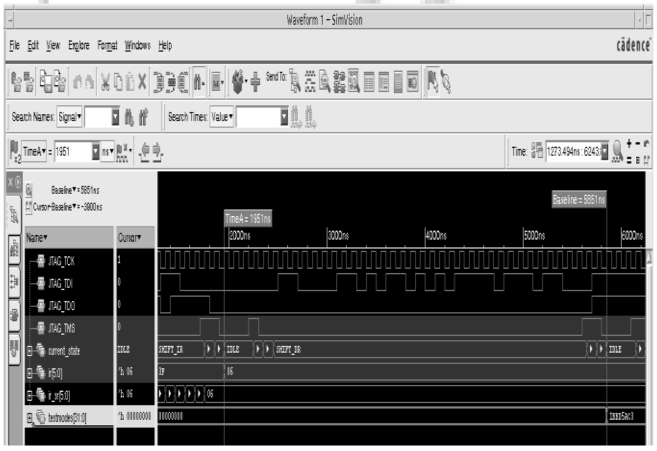

Fig. 4: simulation in UVM environment using cadence tool for test mode register DR update

X. CONCLUSION AND FUTURE SCOPE

This paper describes the development of JTAG functional verification IP in UVM methodology. The developed verification IP can be verified with a silicon proved JTAG interface RTL used inside a USB hub. The JTAG specific registers and also the USB hub design specific registers can be verified using the developed JTAG interface. The developed verification IP can be used to verify any JTAG RTL. Also this verification IP does not have any tool dependency as this is developed in UVM, which can be run on Synopsys, Cadence, Mentor Graphics, Modelsim etc The Verification IP can be used to verify the JTAG parallel interface protocol. It can be updated to support the serial wire JTAG interface protocol as well. The latest chip's/SOC's are targeted to reduce the number of pins; Serial wire JTAG interface will be playing a bigger role in the coming years as it requires only two pins, one for the serial clock and the another for the serial data when compared with the five pin parallel JTAG interface. The modules which require update will be only the JTAG driver and JTAG monitor. The remaining modules can be reused from existing environment.

REFERENCES

[1] The boundary-scan handbook, analog and digital, second edition, Kenneth P Parker
[2] www.ieee.org
[3] www.amontec.com
[4] www.testbench.in