Reusable Verification Environment for Verification of Ethernet

Sridevi Chitti and Dr. G.Krishnamurthy

Summary:
Design reuse and verification reuse are important to satisfy time to-market requirements. Designer must be able to reuse Intellectual Property in the design as golden model. Reuse of verification environment across different designs of the domain saves time to market further and improves total design verification quality. The Physical Layer is a fundamental layer upon which all higher level functions in a network are based. However, due to the plethora of available hardware technologies with widely varying characteristics, this is perhaps the most complex layer in the OSI architecture. The implementation of this layer is often termed Physical layer device (PHY). The Physical Layer defines the means of transmitting raw bits rather than logical data packets over a physical link connecting network nodes. A PHY chip is commonly found on Ethernet devices. Its purpose is digital access of the modulated link and interface to Ethernet Media Access Control (MAC) using media independent interface (MII) interface. This paper discusses Verification process, issues involved in verification process and Test Methodologies. A broad outline of the comparison of traditional verilog and Universal verification methodologies has been presented here. It also explains verification strategy and reuse of design environment with reference to verifying the Ethernet packet in Ethernet Intellectual Property (IP) Core. Verification environment reuse for different application with different interface is done by developing a wrapper around the Design Under Test (DUT) interface and then interfacing it to the environment. A detailed test plan is made for the complete and exhaustive test for Ethernet MAC Receiver. Coverage goals, coverage obtained and coverage analysis indicate efficiency of the verification methodology.

Key Words: BFM, Coverage, DUT, Ethernet VC, MAC, MDIO, Reuse, Testmethods, Verification.

I. INTRODUCTION

Verification is a methodology used to demonstrate the functional correctness of a design. With automation human errors in process are minimized. Automation takes human intervention completely out of the process However, automation is not always possible, especially in processes that are not well defined and continue to require human ingenuity and creativity, such as hardware design. Another possibility is error due to human intervention by reducing it to simple and foolproof steps. Human intervention is needed only to decide on the particular sequence or steps required to obtain the desired results. It is usually the last step toward complete automation of a process. However, just like automation, it requires a well defined process with standard transformation steps.

Sridevi Chitti and Dr. G.Krishnamurthy are with Department of ECE , SRITW, Warangal. , Emails: Srideviredd y.arram@gmail.com , gkmurthy25@yahoo.com

The verification process remains an art that, to this day, does not yield itself to well-defined steps. Choosing the common origin and reconvergence points determines what is being verified. These origin and reconvergence points are often determined by the tools used to perform the verification. It is important to understand where these points lie to know which transformation is being verified. The main purpose of functional verification is to ensure that a design implements intended functionality. without functional verification, one must not trust that the transformation of a specification document into design and Register Transfer Logic (RTL) code was performed correctly, without misinterpretation of the specifications.

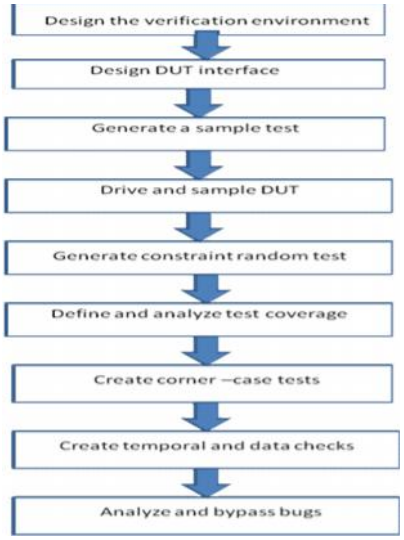

Figure-1 : Generalized Verification flow

Figure 1 shows an overview of the steps involved in verification of a design under test. Verification activity consists of driving vector stream into device and checking the vector stream coming out of the device. Higher level languages are needed, as High-level language eases the creation of an expected value generator. Data check verifies the data correctness while temporal check verifies timing and protocol. Here the shared objects are used for input stimulus. It supports cycle-based behavior, events, and synchronizes with HDL simulator. Figure 1. Generalized Verification Flow Verification planning is an important and integral part of verification, irrespective of the size of the system. About 70% of the design cycle time is spent on verification; with proper verification planning some of the issues faced during the later phases of design can be addressed earlier. For SOC's it is observed that most of the peripherals are reused from the previous design step with some modifications done on the feature set. Use of a

preconfigured and pre-verified suite of code and IP means that adaptation and subsequent re-verification of the code for specific applications is greatly eased . The reuse of verification code and methodology is a major factor providing significant reduction of the overall verification costs One of the fundamental basics of design and verification reuse is the standardization of interfaces . This has resulted in a number of interface and bus protocols that are used to connect different entities together. The logical conclusion for verification is to organize test bench components around those interfaces.

Those components can then be used to verify multiple entities which have a particular interface. VCs are verification components written in the verification language The language is designed specifically for verification. Reuse and extensibility are fundamental e language design principles.

II. VERIFICATION PROCESS

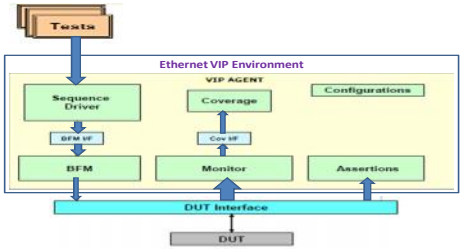

Figure-2: Generalized Verification Environment

Figure 2 show clearly the components used in test environment like test plan, coverage to be achieved, generation of test vectors and its driving to the DUT.

III. ISSUES IN VERIFICATION

Issues to be addressed during any verification activity are
• Capturing of all features of design (functional aspects).
• Compliance to all protocols.
• Coverage for all possible corner cases.
• Checking for locking states in the Finite State Machines (FSMs).
• Working of design in any random state.
• Elimination of redundant tests to save unnecessary simulation cycles and cost.
• Equivalence check with reference design.

• Methodology availability, for verification of each design modification.
• Generation of data patterns for directed or random test.
• Ease of generating flexible and modular test environment
• Reusability and readability of test environment.
• Handling of simulation time in automated test environment.

IV. ISSUES IN VERIFICATION METHODOLOGIES

Lack of effective automation during functional verification due to size and complexity of design, makes development of test environment, scheme for test generation and deterministic tests an intensive manual effort. Checking and debugging test results is also predominantly a manual process. Design Complexity and size makes version control and tracking of design and verification process difficult, both at specification level and functional, which can often lead to architectural-level bugs that require enormous effort to debug. Debugging is always a problem, especially when they occur in unpredictable places. Even the most comprehensive functional test plan can completely miss bugs generated by obscure functional combinations or ambiguous spec interpretations. This is why so many bugs are found in emulation, or after first silicon is produced. Without the ability to make the specification itself executable, there is really no way to ensure comprehensive functional coverage for the entire design intent. The relative inefficiency with which today's verification environments accommodate midstream specification changes also poses a serious problem. Since most verification environments are an ad hoc collection of HDL code, C code, a variety of legacy software, and newly acquired point tools, a single change in the design can force a ripple of required changes throughout the environment, eating up time and adding substantial risk. Perhaps the most important problem faced by design and verification engineers is the lack of effective metrics to measure the progress of verification. Indirect metrics, such as toggle testing or code coverage, indicate if all the flip-flops are toggled or all lines of code were executed, but they do not give any indication of what functionality was verified. For example, they do not indicate if a processor executed all possible combinations of consecutive instructions. There is simply no correspondence between any of these metrics and coverage of the functional test plan. As a result, the verification engineer is never really sure whether a sufficient amount of verification has been performed.

V. TRADITIONAL VERIFICATION METHODOLOGY

Some important points are:
• Productivity & Quality Issues
Verification is more than 50% of an overall project cycle. It May require tens of thousands of lines of verification code. Design spec changes cause major verification delays. Implementing all identified tests in test plan within the project schedule is the Productivity issue.
• Requirement for Productivity Improvement
The verification environment must be created and/or maintained efficiently. Human should spend more time at higher level details providing simulation goals, analyzing errors reported by checkers and providing more direction when goals are not being met.
• Quality Issues

Verification complexity makes it a challenge to think of all possible failure scenarios. It does not provide a way to try scenarios beyond the expected failure scenarios.

• Requirement for Quality Improvement

Confidence about the ratio of identified bugs must increase. An automatic way to know what has been tested must be available.

• Task-based Strategy

To improve test-writing productivity higher level of abstraction is used for specifying the vector stream, and higher-level tasks are created in HDL or C. Task-based strategy limitations are, high test writing effort, many parameters values must be selected manually and high - level intent is not readily apparent. There is no need to buy new tools or licenses and it provides homogeneous environment.

VI. UNIVERSAL VERIFICATION METHODOLOGY

The Universal Verification Methodology (UVM) standard, developed by Accellera's Verification IP (VIP) Technical Subcommittee (TSC), is available as a Class Reference Manual accompanied by an open-source SystemVerilog base class library implementation and a User Guide. The UVM standard establishes a methodology to improve design and verification efficiency, verification data portability and tool, and VIP interoperability.

VIP Design and Development:

This included development of different modules of Verification IP (i.e. transactions, sequence driver, bfm, monitor, functional coverage etc.)

Test bench Development

This stage included development of PHY module (Collision detection and collision sensing module) and stitching the VIP with reference module.

Random Test Generation

This stage included development of different test scenarios for GbE testing. Generated scenarios are properly constrained so that they won't generate any invalid scenarios. Users can configure to generate either random packets or directed packets.

Assertions Development

Development of assertions for GMII (Gigabit Media Independent Interface) using System Verilog Assertions (SVA) were used to check this interface.

Befits :

1.Pipe-cleaning advanced verification flows

- Allow feasibility study on a shareable test case
- Decouple complexity of flows and complexity of designs
- Deployment of verification methodologies

2. Standard tools and methods ramp-up

- Using a common design has major benefit on training efficiency
- Customer tools integrated into a standardized flow

3. Easily demonstrating interoperability

- Open source is key to prove interoperability across tools

VII. WHY SYSTEMVERILOG AS VERIFICATION LANGUAGE?

SystemVerilog (IEEE 1800TM) is the industry's first unified Hardware Description and Verification Language (HDVL) standard. It was developed originally by Accellera to dramatically improve productivity in the design of large gate-count, IP-based, bus-intensive chips. SystemVerilog has been adopted by 100's of semiconductor design companies and supported by many EDA, IP and training solutions worldwide.

This language is chosen because of following key advantages:

• One language for Design and Verification.

• SystemVerilog provides a complete verification environment with the help of features like:

Constraint Random Generation

Coverage Driven Verification

Assertion Based Verification.

These features improve dramatically the verification process.

• SystemVerilog is an integrated part of the Simulation tool like Modelsim. There is no need for any external tool, GUI or interface (such as PLI) in order to run it. Hence, the adoption process of SystemVerilog is very fast causing designers and verification engineers to find it friendly and easy to use.

• The performance of doing verification using SystemVerilog is about 2 times faster than with any other verification languages, which requires external interface to the simulation tool.

• SystemVerilog is an extension of the popular Verilog language, the adoption process of SystemVerilog by engineers is extremely easy and straightforward. SystemVerilog enables engineers to adopt a modular approach for integrating new modules into any existing code. As a result, the risks and costs of adopting a new verification language are reduced.

VIII. CONCLUSION

With increasing the demand for the Ethernet Verification, it has faster and complete verification with lower rate of failure/ re_spin .

This Verification can be widely used in Soc verification Requirements which require higher coverage and improve simulation runtime.

REFERENCES

[1] Janick Bergeon, Writing test benches – functional Verification of HDL models

[2] Faisal I Haque. Et.al. The art of verification with VERA

[3] Samir Palnitkar, Design verification with e

[4] Brendan Mullane and Ciaran MacNamee, Circuits and System Research Centre (CSRC), University of Limerick, Limerick, Ireland, Developing a Reusable IP Platform within a System-on-Chip Design Framework targeted towards an Academic R&D Environment www.design-reuse.com/.

[5] Ben Chen, , Cisco Systems, Shankar Hemmady, Rebecca Lipon of Synopsys, Verification IP reuse for complex networking ASICs- eetindia.com
[6] 0. Petlin, *A.* Genusov, ASIC Alliance Corporation, L.Wakeman, Lucent Technologies, Methodology and Code Reuse in the verification of telecommunication SOCs, Proceedings of 13th Annual IEEE International ASIC/SOC Conference, 2000 (Cat. No.00TH8541) Verification Planning for Core based Designs
[7] Anjali Vishwanath, Ranga Kadambi, Infineon Technologies Asia Pacific Pte Ltd Singapore
[8] Kambiz Khalilian, Stephen Brain, Richard Tuck, Glenn Farrall, Infineon Technologies Reusable Verification Infrastructure for A Processor Platform to deliver fast SOC development**,** www.design-reuse.com/.
[9] Managing Functional Verification Projects Meeting the challenges of high-level verification in today's SoCs-Synopsys white paper-Oct 2007
[10] Hannes Froehlich, Verisity Design, Increased Verification Productivity through extensive Reuse, Design and Reuse, Industry articles.
[11] Verification Reuse Methodology, Essential Elements for Verification Productivity Gains- Verisity white papers-2002.