IJARST
International Journal For Advanced Research In Science & Technology
A peer reviewed international journal www.ijarst.in
ISSN: 2457-0362

Ethernet Verification Using OVM

Dr.K.Rajender Prasad[1], Nagulanchi Raju[2]
Associate Professor[1], Assistant Professor[2]
Department of Electronics and Communication Engineering
Malla Reddy Engineering

Abstract- The functional verification of a digital design is an expensive step in the design process. As designs become more complex, simulation is challenged throughout the design and verification process, both at the low level (implementation verification), to show that a low level implementation implements a higher-level specification, and at the high level (design verification), to show that a design complies with some abstract specification. In this project we verify a ETHERNET 10/100Mbps, for the verification of this process there is a need of verification plan and according to the plan an environment has to be created in System Verilog language. Every system is automated in order to face new challenges in the present day situation. Automation has less manual operation, so that the flexibility, reliability is high and must be more accurate. In the same way the traditional Test Benches need to be made Automated in terms of Test selection, Run and Validation. There is need to verify designs using these Automated Test Benches which reduces development time through reusable architecture, ease using OOPs based language constructs at higher abstraction like Transaction level instead of hardware signal level, higher depth of code coverage using Constraint Random stimulus generation and verification closure using Checkers and Scoreboard.

Index Terms- ETHERNE, Test selection, System Verilog, Scoreboard

I. INTRODUCTION

Ethernet is a family of frame-based computer networking technologies for local area networks (LAN). The name was inspired by the physical concept of theether. It defines a number of wiring and signaling standards for the Physical Layer of the OSI networking model as well as a common addressing format andMedia Access Control at the Data Link Layer. Ethernet is standardized as IEEE 802.3. The combination of the twisted pair versions of Ethernet for connecting end systems to the network, along with thefiber optic versions for site backbones, is the most widespread wired LAN technology. It has been used from around 1980[1] to the present, largely replacing competing LAN standards such as token ring, FDDI, and ARCNET.Ethernet was developed at Xerox PARC between 1973 and 1975. It was inspired by ALOHAnet, which Robert Metcalfe had studied as part of his Ph.D. dissertation. In 1975, Xerox filed a patent application listing Metcalfe, David Boggs, Chuck Thacker and Butler Lampson as inventors. In 1976, after the system was deployed at PARC, Metcalfe and Boggs published a seminal paper. Metcalfe left Xerox in 1979 to promote the use of personal computers and local area networks (LANs), forming 3Com. He convinced Digital Equipment Corporation (DEC), Intel, and Xerox to work together to promote Ethernet as a standard, the so-called "DIX" standard, for "Digital/Intel/Xerox"; it specified the 10 megabits/second Ethernet, with 48-bit destination and source addresses and a global 16-bit Ethertype-type field and was first published on September 30, 1980 as "The Ethernet, A Local Area Network. Data Link Layer and Physical Layer Specifications". Version 2 of this document was published in November, 1982 and defines what has become known as Ethernet II. The Institute of Electrical and Electronics Engineers (IEEE) first published the 802.3 standard as a draft in 1983[7] and as a standard in 1985. Support of Ethernet's carrier sense multiple access with collision detection (CSMA/CD) in other standardization bodies (i.e., ECMA, IEC, and ISO) was instrumental in getting past delays of the finalization of the Ethernet standard due to the difficult decision processes in the IEEE, and due to the competitive Token Ring proposal strongly supported byIBM. Ethernet initially competed with two largely proprietary systems, Token Ring and Token Bus. These proprietary systems soon found themselves competing in a market inundated by Ethernet products. In the process, 3Com became a major company. 3Com shipped its first 10 Mbit/s Ethernet 3C100 transceiver in March 1981, and that year started selling adapters for DEC/PDP11 and VAXes, as well as Intel Multibus and Sun Microsystems machines.[8] This was followed quickly by DEC's Unibus to Ethernet adapter, which DEC sold and used internally to build itsown corporate network, which reached over 10,000 nodes by 1986; far and away the largest extant computer network in the world at that time.[9] Through the first half of the 1980s, DEC's Ethernet implementation utilized a coaxial cable 0.375 inches (9.5 mm) in diameter, which became known as Thick Ethernet when its successor, ThinnetEthernet was introduced. Thinnet uses a cable similar to cable television cable of the era. The emphasis was on making installation of the cable easier and less costly. The observation that there was plenty of excess capacity in unused unshielded twisted pair (UTP) telephone wiring already installed in commercial buildings provided another opportunity to expand the installed base, and, thus, twisted-pair Ethernet was the next logical development in the mid-1980s, beginning with StarLAN. UTP-based Ethernet became widely known with 10BASE-T standard. This system replaced the coaxial cable systems with a system of hubs linked via UTP. In 1990, Kalpana introduced the first Ethernet switch,[10] which replaced the CSMA/CD scheme in favor of a switched full duplex system offering higher performance and at a lower cost than using routers.Notwithstanding its technical merits, timely standardization was instrumental to the success of Ethernet. It required well-coordinated and partly competitive activities in several standardization bodies such as the IEEE, ECMA, IEC, and finally ISO. In February 1980, IEEE started a project, IEEE 802, for the standardization of local area networks (LAN).[11]The "DIX-group" with Gary Robinson (DEC), Phil Arst (Intel), and Bob Printis (Xerox) submitted the so-called "Blue Book" CSMA/CD specification as a candidate for the LAN specification. Since IEEE membership is open to all professionals, including students, the group received countless comments on this brand-new technology. In addition to CSMA/CD, Token Ring (supported by IBM) and Token Bus (selected and henceforward supported by General Motors) were also considered as candidates for a LAN standard. Due to the goal of IEEE 802 to forward only one standard and due to the strong company support for all three designs, the necessary agreement on a LAN standard was significantly delayed. In the Ethernet camp, it put at risk the market introduction of the Xerox Star workstation and 3Com's Ethernet LAN products. With such business implications in mind, David Liddle (General Manager, Xerox Office Systems) and Metcalfe (3Com) strongly supported a proposal of Fritz Röscheisen (Siemens Private Networks) for an alliance in the emerging office communication market, including Siemens' support for the international standardization of Ethernet (April 10, 1981). Ingrid Fromm, Siemens representative to IEEE 802 quickly achieved broader support for Ethernet beyond IEEE by the establishment of a competing Task Group "Local Networks" within the European standards body ECMA TC24. As early as March 1982 ECMA TC24 with its corporate members reached agreement on a standard for CSMA/CD based on the IEEE 802 draft. The speedy action taken by ECMA decisively contributed to the conciliation of opinions within IEEE and approval of IEEE 802.3 CSMA/CD by the end of 1982. Approval of Ethernet on the international level was achieved by a similar, cross-partisan action with Fromm as liaison officer working to integrate IEC TC83 and ISO TC97SC6, and the ISO/IEEE 802/3 standard was approved in 1984.

II. Existing Work or Literature Survey

The term ASIC stands for Application Specific Integrated Circuit. Is an integrated circuit (IC) customized for a particular use, rather than intended for general-purpose use. Generally an ASIC design will be undertaken for a product that will have a large production run, and the ASIC may contain a very large part of the electronics needed on a single integrated circuit. As feature sizes have shrunk and design tools improved over the years, the complexity in an ASIC has grown from 5,000 over 100 million gates.MRD A Marketing Requirements Document (MRD) outlines the requirements a new product. Engineers use an MRD to create the product. Marketing requirement document covers market needs, the customer value proposition, and product functionality. It is developed by the Marketing team and upper management. Architecture Specification The architect based on the MRD specification, develops the overall architecture of the chip. This is a very high level plan. Architecture Specification includes functional descriptions of each module, Properties and weights. Design Specification The designers and architects sit together to come up with detailed design documents. Design strategies, design partitions, type of memories to use, etc. Verification Plan A Verification specification is called a Verification/test plan. The verification engineer goes through all the above documents and prepares verification plan to verify the design. RTL Design RTL stands for Register Transfer Level. The designer starts implementing the RTL design in HDL like verilog or VHDL. Functional Verification The verification engineers starts developing Test Bench and verifies whether the DUT works according to specification or not.Synthesis Synthesis is the process of taking a design written in a hardware description language, compiling it into a net list of interconnected gates which are selected from a user-provided library of various gates. The design after synthesis is a gate-level design. Physical Design Physical design process includes logic partitioning, floor planning, global routing, detailed routing, compaction, and performance-driven layout. PD team transforms net list representation of a system into layout representation. Timing Analysis Static timing analysis is an important step in analyzing the performance of a design. In the Timing analysis Setup time, hold time ,recovery time ,removal time , Clock latency, clock skew, clock uncertainty etc checks are done. This is the final stage of the design cycle of integrated circuits. Once all the checks are done, the design is ready to be sent to Foundry.Highly covered code isn't necessarily free of defects, although it's certainly less likely to contain them. By definition, code coverage is limited to the design code. It doesn't know anything about what design supposed to do. Even If a feature is not implemented in design, code coverage can report 100% coverage. It is also impossible to determine whether we tested all possible values of a feature using code coverage. For example, randomization may not generate packets with all possible lengths, this cannot be reported by code coverage.. Code coverage is unable to tell much about how well you have covered your logic -- only whether you've executed each line/block etc at least once. Code coverage does not provide information about your test bench randomization quality and it does not report what caused the line execution/state transition etc. Analysis of code coverage require knowledge of design to find which features are not verified which is time consuming and out of scope of verification engineer. If the analysis is done at higher level of abstraction, it would be easier for the test writer to identify the missed serious which is not possible by code coverage. So if the code coverage is less than 100%, it means there is more work to do, if it is 100%, it doesn't mean that the verification is complete.It's getting harder to figure out when to stop testing as the complexity of the protocol is increasing. In directed test environment, for each point mentioned in test plan, there will be a separate test case file. So if there are 100 points in test plan, then the engineer has to write 100 test case files. After writing and executing the 100 test case files, we can say that "all the points in test plan are verified" and we can stop testing.When building a verification environment, the verification engineer often starts by modeling the device input stimulus. In Verilog, the verification engineer is limited in how to model this stimulus because of the lack of high-level data structures. Typically, the verification engineer will create a array/memory to store the stimuli. SystemVerilog provides

high-level data structures and the notion of dynamic data types for modeling stimulus. Using SystemVerilog randomization, stimulus is generated automatically. Stimulus is also processed in other verification components. SystemVerilog high-level data structures helps in storing and processing of stimulus in an efficient way.The generator component generates stimulus which are sent to DUT by driver. Stimulus generation is modeled to generate the stimulus based on the specification. For simple memory stimulus generator generates read, write operations, address and data to be stored in the address if its write operation. Scenarios like generate alternate read/write operations are specified in scenario generator. SystemVerilog provided construct to control the random generation distribution and order. Constraints defined in stimulus are combinatioural in nature where as constraints defined in stimulus generators are sequential in nature.The drivers translate the operations produced by the generator into the actual inputs for the design under verification.
Generators create inputs at a high level of abstraction namely, as transactions like read write operation. The drivers convert this input into actual design inputs, as defined in the specification of the designs interface. If the generator generates read operation, then read task is called, in that, the DUT input pin "read_write" is asserted.Assertions are used to check time based protocols, also known as temporal checks. Assertions are a necessary compliment to transaction based testing as they describe the pin level, cycle by cycle, protocols of the design. Assertions are also used for functional coverage.

WRITE DOWN YOUR STUDIES AND FINDINGS

In test plan, we prepare a road map for how do achieve the goal, it is a living document. Test plan includes, introduction, assumptions, list of test cases, list of features to be tested, approach, deliverables, resources, risks and scheduling, entry and exit criteria. Test plan helps verification engineer to understand how the verification should be done. A test plan could come in many forms, such as a spreadsheet, a document or a simple text file. Sometimes, test plan simply reside in the engineer's head which is dangerous in which the process cannot be properly measured and controlled. Test plan also contains the descriptions of TestBench architecture and description of each component and its functionality.

BUILDING TESTBENCH: In this phase, the verification environment is developed. Each verification component can be developed one by one or if more than one engineer is working it can be developed parallel. Writing the coverage module can be done at any time. It is preferred to write down the coverage module first as it gives some idea of the verification progress.

WRITING TESTS: After the Test Bench is built and integrated to DUT, it's time for validating the DUT. Initially in CDV, the test are ran randomly till some 70 % of coverage is reached or no improvement in the coverage for 1 day simulation. By analyzing the coverage reports, new tests are written to cover the holes. In these tests, randomization is directed to cover the holes. Then finally, the hard to reach scenarios, called as corner cases have to be written in directed verification fashion. Of course, debugging is done in parallel and DUT fixes are done.

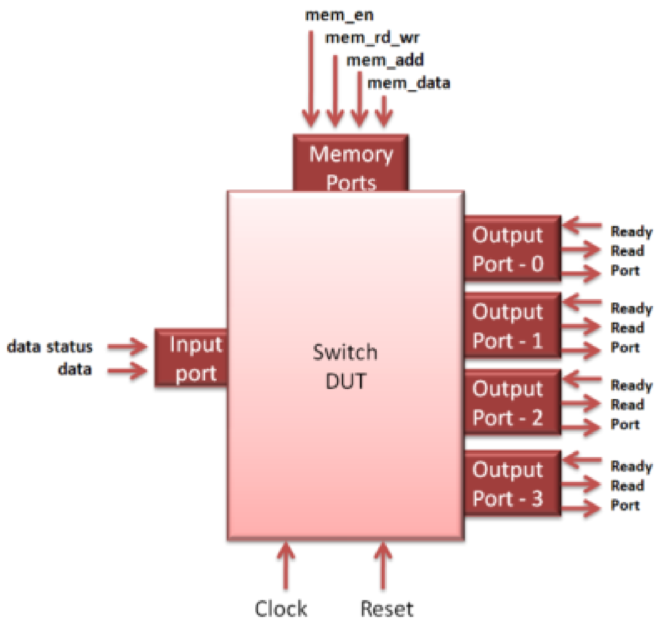
INTEGRATING CODE COVERAGE: Once you have achieved certain level of functional coverage, integrate the code coverage. For doing code coverage, the code coverage tools have option to switch it on. And then do the simulation, the tool will provide the report.

ANALYZE COVERAGE: Finally analyze both functional coverage and code coverage reports and take necessary steps to achieve coverage goals. Run simulation again with a different seed, all the while collecting functional coverage information.

ONES COUNTER EXAMPLE: Following example is Test Bench for ones counter. It has some verification components which are required, but not all the verification components discussed earlier. All the components implementation can be seen in further chapters with another protocol. Description of the language construct is discussed in further chapters, so don't pay attention to them. The intention of showing this example is to make you familiar with some steps required while building verification environment and to help you to understand the flow discussed above. Specification Ones Counter is a Counter which counts the number of one's coming in serial stream. The Minimum value of the count is "0" and count starts by incriminating one till "15". After "15" the counter rolls back to "0". Reset is also provided to reset the counter value to "0". Reset signal is active negedge. Input is 1 bit port for which the serial stream enters. Out bit is 4 bit port from where the count values can be taken. Reset and clock pins also provided.

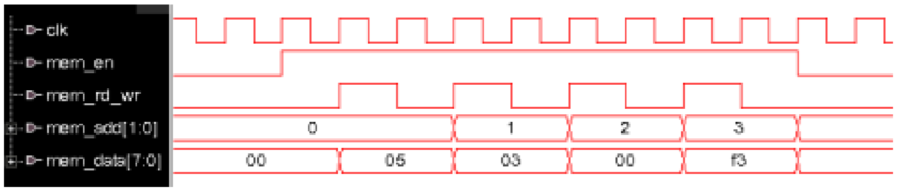
The Verification Plan is the focal point for defining exactly what needs to be tested, and drives the coverage criteria. Success of a verification project relies heavily on the completeness and accurate implementation of a verification plan. A good plan contains detailed goals using measurable metrics, along with optimal resource usage and realistic schedule estimates. Verification plan gives an opportunity to present and review the strategy for functional verification before the verification engineer have gone into detail to implement it. It also establishes proper communication. Just imagine how it would be working with a multisite project and you have a query for which you have to wait till the next day to see the answer in email and they just call you while you are in sleep. It also gives an idea about the areas that are going to be difficult to verify for taking necessary steps. It is used to determine the progress and completion of the verification phase of verification. Verification Planning should start early with system/architecture evaluation phase. Once the functional spec is given to the verification team, they will start its development. A verification plan could come in many forms, such as a spreadsheet, a document or a simple text file. Templates are good if continually used in your company as it makes common interface for information, reviewer know where to look for certain information even in a huge document that he wants to know at this moment, because different reviewers want different infomation in different moments.

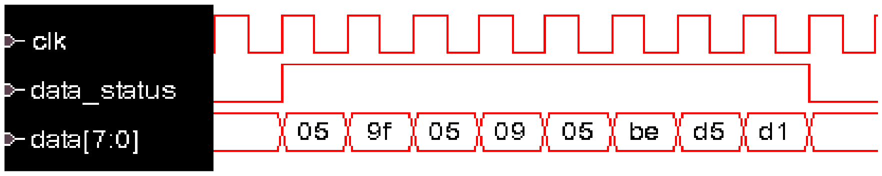
A detailed TestBench architecture is essential for a robust verification environment. In this section describe the topology, about each component of the TestBench, special techniques that are used, IPs, Reused blocks, new blocks, and guidelines on how to reuse the TestBench components. For example if you think upfront about error injection, configuration, component communication, callbacks etc, you can provide hooks to do those.The coverage section explains the functional coverage of the features. A functional coverage plan should be built to help implement coverage points in the verification environment. Genarally it would be better if the coverage block is brocken based on the design logical blocks. It will list the various Coverage groups and assertion.This is a simple switch. Switch is a packet based protocol. Switch drives the incoming packet which comes from the input port to output ports based on the address contained in the packet. The switch has a one input port from which the packet enters. It has four output ports where the packet is driven out.Destination address of the packet is of 8 bits. The switch drives the packet to respective ports based on this destination address of the packets. Each output port has 8-bit unique port address. If the destination address of the packet matches the port address, then switch drives the packet to the output port.Switch has four output ports. These output ports address have to be configured to a unique address. Switch matches the DA field of the packet with this configured port address and sends the packet on to that port. Switch contains a memory. This memory has 4 locations, each can store 8 bits. To configure the switch port address, memory write operation has to be done using memory interface. Memory address (0,1,2,3) contains the address of port(0,1,2,3) respectively.

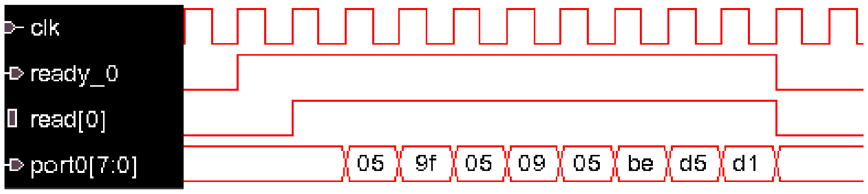


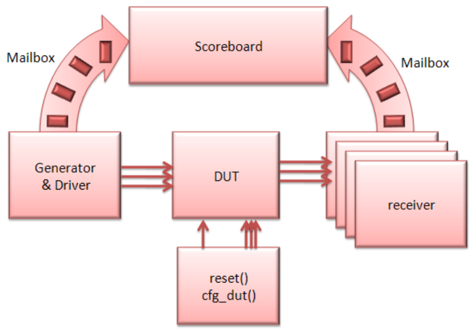
Results and discussion

Through memory interfaced output port address are configured. It accepts 8 bit data to be written to memory. It has 8 bit address inputs. Address 0,1,2,3 contains the address of the port 0,1,2,3 respectively

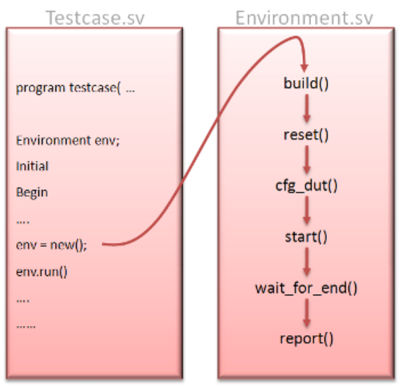


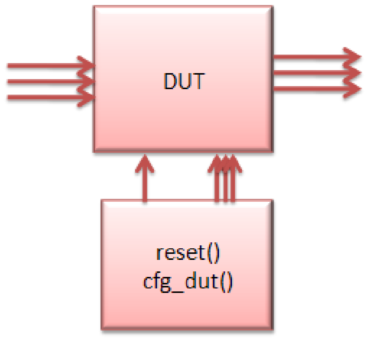


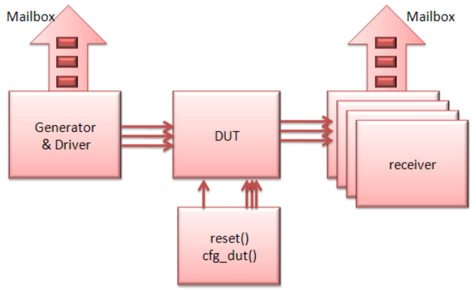




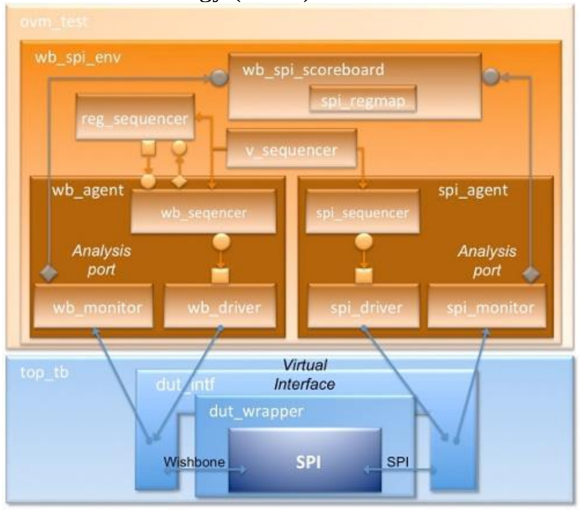
The modules that are included in the source text but are not instantiated are called top modules. This module is the highest scope of modules. Generally this module is named as "top" and referenced as "top module". Module name can be anything.







The Open Verification Methodology (OVM) is a documented methodology with a supporting building-block library for the verification of semiconductor chip designs. The initial version, OVM 1.0, was released in January, 2008, and regular updates have expanded its functionality. The latest version is OVM 2.1.1, released in April, 2010. The current release and all previous releases are available, under the Apache License, on the OVM World site. The OVM has won recognition from Electronic Design Magazine and a DesignVision award from the International Engineering Consortium. The OVM was co-developed by Mentor Graphics and Cadence Design Systems, and they continue to guide its evolution in concert with the nine user companies of the OVM Advisory Group. The OVM is publicly supported by more than 60 partner companies offering tools, training, and services. The OVM was standardized within Accellera, which voted to make it the basis for the Universal Verification Methodology (UVM).



CONCLUSION

Designed and developed reusable verification environment for Ethernet 10/100 Mbps. verified all the features of Ethernet and attained code coverage of 96%.

REFERENCES

Young, G. O. 1964. "Synthetic structure of industrial plastics (Book style with paper title and editor)," in *Plastics*, 2nd ed. vol. 3, J. Peters, Ed. New York: McGraw-Hill, 1964, pp. 15–64.

Chen, W.-K. 1993. *Linear Networks and Systems* (Book style). Belmont, CA: Wadsworth, 1993, pp. 123–135.

Poor, H. . 1985. *An Introduction to Signal Detection and Estimation*. New York: Springer-Verlag, 1985, ch. 4.

Smith, B. "An approach to graphs of linear forms (Unpublished work style)," unpublished.

Miller, E. H. "A note on reflector arrays (Periodical style—Accepted for publication)," *IEEE Trans. Antennas Propagat.*, to be published.

Wang, J. "Fundamentals of erbium-doped fiber amplifiers arrays (Periodical style—Submitted for publication)," *IEEE J. Quantum Electron.*, submitted for publication.