JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

IP Level Verification of Ethernet Protocol for OpenPOWER Processor Core based Fabless System on Chip (SoC)

[1]T. Poojitha, [2]K.Mamatha, M.Tech.,(Ph.D), [3]Dr.Gannera Mamatha M.Tech., Ph.D

[1]PG Scholar, [2]Assistant Professor (Adhoc), [3]Assistant Professor
Department of ECE,
JNTUACEA, Ananthapuramu 515002, Andhra Pradesh, India.

Abstract: The modern mainstream of IC innovation is SoC (System-on-a-Chip), which is the modern international VLSI latest generation. According to Moore's Law, as transistor dimensions get smaller, IC complexity on a chip has been growing rapidly. The primary problem in SoC design is to effectively integrate various IP cores and keeping their capacity to carry out specified tasks and interact with one another; thus, it is necessary to confirming the quality of integration at the SoC level. Ethernet is a bi – directional networking protocol which governs and specifies way data interacts through a data transmission in both directions. The hardware enables 10/100/1000Mbps using its Ethernet MAC module. Incorporating Media Independent Interface (MII) enables time-sensitive applications over bridged Local area network (LAN) also with AXI4 Master interface and AXI4 Slave interface, Ethernet MAC is compliant with IEEE 802.3 standard. It supports CSMA/CD Protocol. In this paper, the design and validation of an AXI bus-based MAC controller are discussed.

Keywords- AXI bus; Media Access Control; Verification Methodology Manual; System Verilog; Verilog

I. INTRODUCTION

Today's hardware designs have a very large proportion, with tens of millions of transistors or more integrated on a single chip. In fact, this technological breakthrough has reached the point where it is difficult to design a complete system from scratch. The industry has already started designing this SoC's from a large repertoire of intellectual property components or IP cores sold by many vendors. System-on-chip design usually involves the integration of different components on standard buses. For example, a vendor wants to protect his IP core in-house by just providing interface specifications. As a result, verification of such designs is becoming increasingly difficult. This paper outlines a new method for formal verification of IP core-based system-on-chip designs. Verification is the most expensive design process nowadays and counts for around 70% to 80% of all design effort, as is universally acknowledged. A typical SoC consists of a processor or processor subsystem, a processor bus, peripheral buses, a bridge between two buses, numerous peripherals including such as data conversion engines, etc. The complete and partial duplex operations of 10/100/1000 Mbps are supported by gigabit Ethernet media access controllers. A separate one is used for encoding and decoding. When linked to other PHYs, the Ethernet interface allows MII and GMII interfaces.

Additionally, it gives access towards the MII management interface, which is utilized to set up the PHY. The primary goal of SoC verification is to examine how well different components are integrated. The fundamental premise is that every component has previously undergone independent testing. This unique orientation calls for unique methods. As a result, we want a method for defining intricate test scenarios and evaluating how effectively we handle their unique implementation requirements. Reusing verification components is also suggested by combining and reusing many of hardware IP blocks. Acquiring reusable verification components is made simple and enjoyable by first distinguishing reusable verification components (such as external interface stimuli, checkers, and coverage). Considering verification for reuse, there's really much more to say. The method we model the verification system is essential to having useful reusable components.

II. STRUCTURAL OVERVIEW

There are several little components that make up the system. A block diagram showing the direct connections in between modules is shown here under. Every interaction between the chip and external gear starts at the advanced level Ethernet interface. We describe the transmission of information in more context here. Our internal module controls access to the ports and chip data flow by seeking approval from the arbiter. Once accessibility to the source module has been approved, information is transmitted to the Ethernet controller module, that includes state machines to interact well with chip.

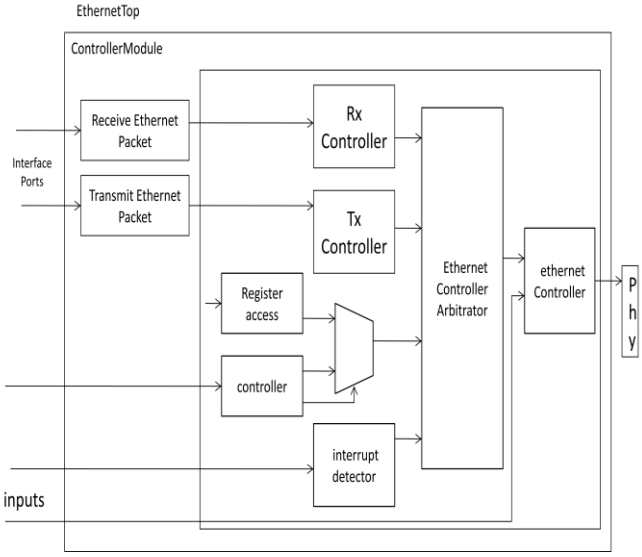

Fig.1: Ethernet MAC module

2.1. Ethernet Top and Interface Description:

The Ethernet top module, that creates as well as joins those required construction blocks, implements the link layer protocol. This step involves the instantiation of each of the transmit and the receive Ethernet modules. Interface ports make it easy to create black box interfaces with integrated Ethernet controllers. Our interface uses handshaking techniques to send and receive packets. Interfaces must have four ports including both incoming and outgoing traffic. It also emits an interrupt signal for packets that are delivered. In its most basic form, an outside module starts Ethernet tx/rx by pushing the relevant request line high as a controller interface.

2.2. Ethernet frame standards:

As shown in Fig.2 Typical Ethernet frames contain frames that contain Ethernet data. Fields are conveyed from left to right in a frame. Bytes are shifted from a field's left to right (least significant bit to most significant bit unless otherwise noted). The Ethernet MAC can manage large Ethernet frames including data fields that are significantly larger beyond 1500 bytes. The following paragraphs include descriptions for every Ethernet frame field as well as additional key MAC features.

2.2.1. Preamble:

This information is added instantly for communication by the Ethernet MAC. The preamble field, that has historically been employed for synchronization, is transmitted from left to right and is made up of 7 bytes. This field is indeed erased from incoming frames after reception before any data delivery. An Ethernet MAC will accept Ethernet frames devoid of a preamble provided that a proper start-of-frame delimiter is provided.

2.2.2. Start of Frame Delimiter:

The beginning of the frame is indicated by the Start of Frame Delimiter field. This field is added back by the Ethernet MAC whenever information is sent over the direct connection. This field is indeed erased from incoming frames after reception before transferring any information.

2.2.3. MAC Address Fields:

2.2.3.1. MAC Address:

Depending upon this least significant bit of the first byte, a MAC address can be either single/unicast (0) or group/multicast (1). Multicast addressing is used to send information to several recipients. A broadcast address is a multicast location which targets every node on a local area network (the destination address field usually includes one). Ethernet MAC may transmit and receive broadcast, multicast, and unicast packets. Because addresses are conveyed in the Ethernet frame least significant bit first, the bit referring to the individual's or group's address is the very first bit to show up there.

2.2.3.2. Destination Address:

Both packet information during transmitting and packet information for reception usually contain the MAC address field, which is the primary field in an Ethernet frame. It offers the recipient's MAC address for your connection.

2.2.3.3. Source Address:

Both packet information intended for transmitting and packet information intended for reception usually contain the second field in Ethernet frames, known as the MAC address field. The network from which frame arrived is transmitted along with its MAC address. Although TEMAC does not change source addresses, source addresses for Ethernet transmissions should constantly be provided.

2.2.3.4. Length/Type:

In accordance with IEEE Std. 802.3-2008, the content of this field indicates whether it should be treated as a length or type field. Variables with far more than 1,536 decimal places are interpreted into type fields by the Ethernet MAC. When utilized as a length field, the content of this field reflects the number of bytes in the subsequent data field. Whatever bytes that could be

inserted to the padding field after the data field are not included in this amount. Before transmitting, the length/type information is not processed by the Ethernet MAC. Whether this field is the receiving length field, the Ethernet MAC reception engine examines this value and deducts padding from the padding field (if necessary). If the field is a type field, the Ethernet MAC disregard the data and sends it together with the packet contents with no further analysis. The length/type field is always kept in incoming packet information.

2.2.3.5. Data:

During normal frames, the information field size varies from 0 to 1500 bytes. Ethernet MAC is capable of supporting jumbo frames of any size. Each field is always kept in entering packet data and is always considered part departing packet data.

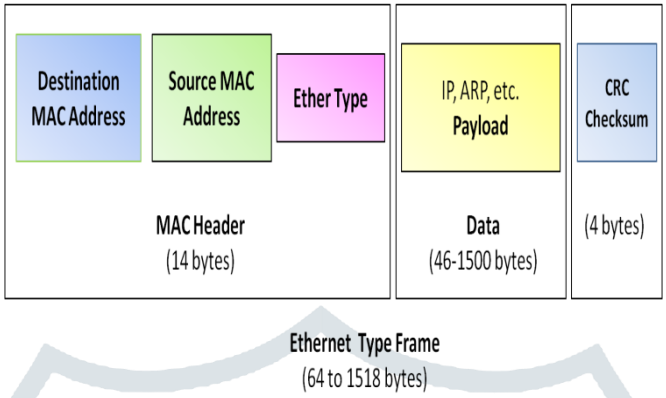

Fig.2: Ethernet Frame Format

2.2.3.6. Pad:

The padding field has a size range of 0 to 46 bytes. Ensure if the frame is at least 64 bytes length using this field (the preamble and this SFD fields are not included in part of the frame during the calculation). Reliable CSMA/CD performance depends upon it. The length parameter does not contain the content of this field, which is utilized to define the frame check sequence. This field as well as the data field must be at least 46 bytes long together. The padding field is 46 bytes if the data field is empty. The padding field is 0 bytes if the data field is 46 bytes or greater. Either the user or the Ethernet MAC simply generates that field for communication. The Ethernet MAC analyzes and introduces the FCS field after the Pad field is placed by the Ethernet MAC. If indeed the slack field is user-supplied, the FCS is either inserted by the Ethernet MAC or given by the user in conformance with configuration register bits.

III. AXI INTERFACE

AXI Bus Protocol is now a part of the current Advanced High Performance Bus (AHB). The core of AXI is designing high-performance, high-frequency systems. To transfer address/control and data, the AXI protocol comprises five unique unidirectional channels. The valid and ready two-way handshake protocol is used by each channel. Address Read (AR), Address Write (AW), Write Data (WD), Data Read (RD), and Write Response (WR) are five distinct channels (B). Write and read transactions are addressed and managed via the AW and AR channels. The read and write transaction structures are described by the control signals of certain channels. A burst is mainly composed of many identically lengthened broadcasts.

Through WD and RD channels, data is sent from the master port to the slave port. The slave can notify the successful or unsuccessful completion of a write operation via the Write Response Channel (B). Burst transactions, in which just the beginning address is provided, are one of the characteristics of the AXI bus. Each transaction has a transaction ID field because to the Split Transaction AXI protocol, which also permits transactions to finish in any order. Order constraints do not apply to transactions with various IDs with the same master port, but they do apply to transactions with the same ID, which must finish in the proper sequence. AXI enables numerous pending transactions, enables transactions to finish out of order, and permits address information to be broadcast prior to the actual data transfer. Interleaving and out-of-order execution are the two main characteristics of the AXI bus that offer maximum speed. The mechanism of sending information from one block to a different is called AXI4 streaming. The purpose of streaming devices is to deliver a constant stream of fast data, often providing a fresh chunk of information with each clock pulse.

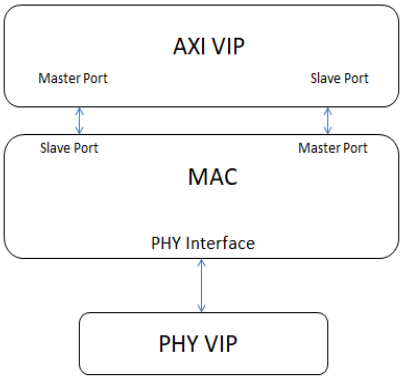

Fig.3: Ethernet module Verification System Architecture

The purpose of streaming devices is to deliver a constant stream of fast data, often providing a fresh chunk of information with each clock pulse. The streaming bus has no addressing in order to decrease overhead. There is no need for addressing because streaming connections are point-to-point. There are several necessary and alternative signals in AXI4 Stream I/F, making them quite adaptable. We utilise Tvalid and Tready to manage the flow of data.

IV. MAC CONTROLLER MODULE: (MEDIAACCESSCONTROLMODULE)

4.1. Send Ethernet Packet:

The ability to deliver Ethernet frames with MAC addresses is provided by this component. The top-level Ethernet top module describes the input and output ports for the interface. This module's goal is to construct an Ethernet frame in the manner indicated in the standard section. Most state machines begin in a wait state. An external component that intends to send a packet immediately begins to wait for approval from the internal arbiter. The first byte delivered towards the transmitting buffer is the MAC address. Due to the fact that this is a broadcasting communication, all devices may see it, set the target MAC address to 0xFFFF. Coding makes it simple to fix this. The source MAC is then taken out of the register and sent to the TX buffer. When transmitting the MAC, the module specifies the protocol type. Then, information is taken from the input and sent to the transmit buffer. The component recognizes that the transmitter has concluded transmitting data once the requested line is confirmed. The module notifies the interface module of a transmission when the transfer has been completed.

4.2. Receive Ethernet Packet:

An interface to external hardware is provided by the entering Ethernet packet component, which is utilized to construct Ethernet frames. The receiving component has four branches and is much more straightforward than the broadcast module. It begins out by going into standby. The module watches for the availability of the data before responding to something like an Ethernet receive request. The header data is read again when the data is prepared. Since CRC validation is not applied, this information is saved instead of being used. The data section of the packet is transmitted to the data output port once the header has been read. It maintains the processing complete line for one round after it has finished reading all the data, then goes back to the wait state.

4.3. Ethernet Controller:

In combination to a directive port, writable port, readable port for address or data, and 16-bit data, the Ethernet module offers several connection points for interaction. The Ethernet controller must choose whether to read or write data since the data port is utilized for both reading and writing. The location of the register users wish to write is provided on the data line. After dropping below, write data is maintained for at least one cycle. Before beginning to write data to a specific register, wait at least two cycles. To read a register from a controller, you should first write the register's address to the controller. The read line is pulled down and information is first obtained from the controller rather than the address being rewritten after writing. To enable successive readings of registers without writing the address each time, the controller has an auto-increment read register address.

4.4. Interrupt Detector:

To manage interruptions and alert the appropriate module when one happens, interrupt detectors are employed. When an interrupt handler request comes in, the function awaits. The kind of interrupt is therefore decided based on the permission request made by the interrupt module. The on-chip interrupt register is given a read. The interrupt module decides whether the interrupt was brought on during the end of TX or RX according to the data obtained from the register. After one cycle, the associated interrupt flag is removed.

4.5. RX Controller:

The information delivered across the Ethernet Network is analyzed by a receiving controller. This module simply accepts bits delivered across the Ethernet connection just because it works under the MAC sub layer. It begins in a waiting state unless an incoming packet is asked, similar to any other component. The information size in bits is verified inside this instance. Assuming this has been established, it figures out how many bits were delivered. It also contains an index that is read by every cycle until all of the bits have been read. The request engine receives each section and processes it. We have now attained the package's conclusion.

4.6. TX Controller:

Transmission of data over the Ethernet cable is started using the send packet module. This is the case since this module only sends the bits it received and operates just under the MAC sub layer. The Transmit Packet Module tries to connect to the Ethernet controller through an arbitration permit when it receives information. Upon receiving permission, the module will begin to await for the proper TX data to approach the module. Data is continually transmitted as as long as the client recognizes their tx request line. The module sends a message transfer after receiving all the information and awaits for an interrupt to show that the text transmission was accomplished. After then, the module resumes its waiting state.

V. VERIFICATION STRATEGY

There are two aspects to this design, system and network, each with a sending block and a receiving block. Each side of the design has a chain of PHY MII MAC blocks in TX and a reverse chain in RX. The chain connects the other side with the same pattern PHY MAC MII PHY. Active monitors read data into the DUT and passive monitors read data out of the DUT. Once synchronized, the data from these two blocks are evaluated and validated. The theme supports multiple rates and paths. Therefore, based on the UVM phase, each basic test is created separately where basic verification is performed. The specific scenario under test is generated by another test that derives from the above and performs the additional functions necessary to generate the scenario and validate the results. The concept of UVM phases takes into account the interdependencies of blocks. It can therefore

be used to create any number of new tests using the concept of inheritance without disturbing the basic structure. All basic configurations are done in the base test, so it appears in all tests derived from them.

VI. SIMULATION RESULTS

Results In this Ethernet MAC IP, the provided input data is taken by the WDATA channel of the AXI interface, frames are constructed according to the selected mode speed, transmitted and received accordingly, and finally observed. Output using the RDATA channel of the AXI interface.

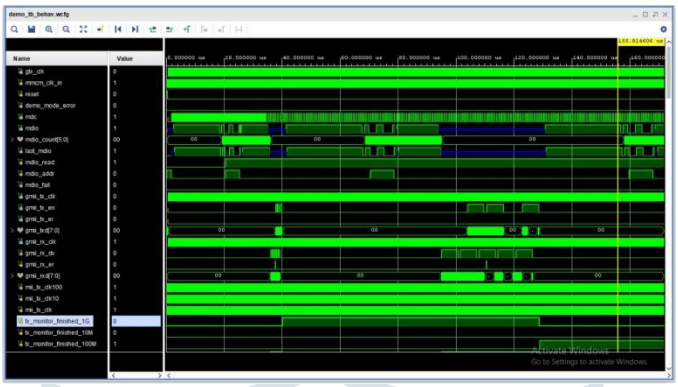

Fig.1: Ethernet Control Signals

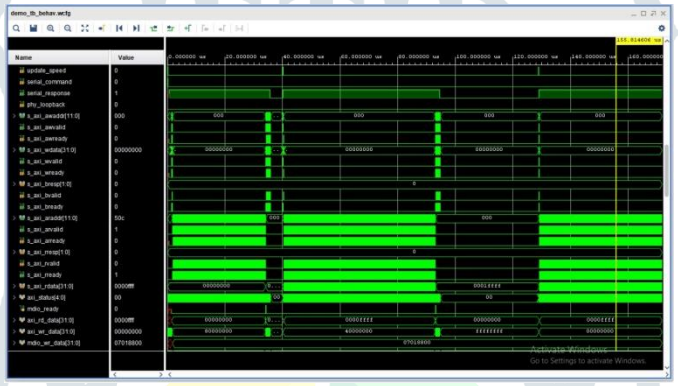

Fig.2: AXI configuration signals

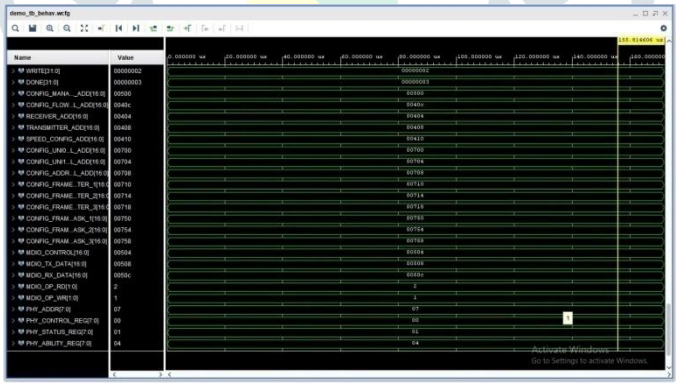

Fig.3: Data transmission signals

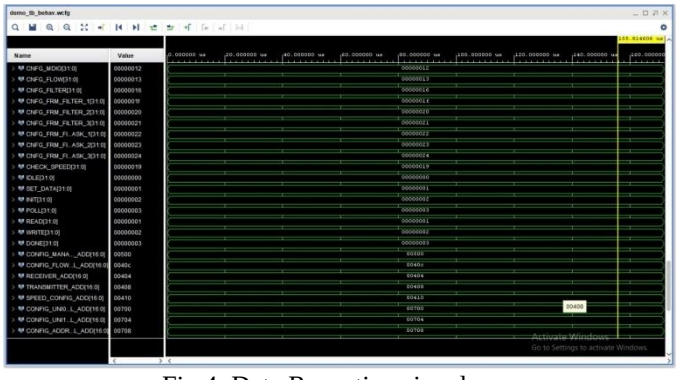

Fig.4: Data Reception signals

VII. CONCLUSION

Verification is the most eventful part of SoC and VIP proposals. Universal Verification Methodology (UVM) is one of the most notorious verification methods for verifying complex IPs and SoC's. This document examines Ethernet interfaced with AXI bus and checks the data transmission successfully from AXI read and write channels and also verifies its protocol based on a common test form using test benches. The Ethernet MAC specification has been successfully verified using the Vivado tool. This document provides an overview of the Verilog system and universal verification methods used to verify Ethernet protocols. By applying this strategy, we can meet all the requirements for validation of this complex protocol.

REFERENCES

[1] Mentor Graphics, Verification Academy. Uvm Cookbook. Mentor Graphics; 2012. P. 1-569.
[2] Samir Palnitkar's Verilog Hdl: A Guide To Digital Design And Synthesis, 2nd Ed. 2nd Edition Is A Comprehensive Book For Electronics & Communication Engineering.
[3] Spears C. System Verilog For Verification: A Guide To Learning The Testbench Language Features. 2nd Ed. Springer:Business Media Llc; 2007.
[4] P. Chauhan, E.M. Clarke, Y.Lu And Dong Wang, Verifying IPcore Based System-On-Chip Designs, Carnegie Melon University.
[5] M.H Assaf, Arima, S.R. Das, W Hernias And Petriu, "Verification Of Ethernet Ip Core Mac Design Using Deterministic Test Methodology", Ieee International Instrumentation And Measurements Technology Conference, May 2008