Ethernet MAC Verification with Loopback Mechanism using Efficient Verification Methodology

Sridevi Chitti 1, P Chandrasekhar2, M Asharani3

Department of Electronics & Communication Engg.,
Hyderabad, Telangana, India.

Abstract- *This paper describes Efficient Verification Methodology (EVM) based constrained random verification (CRV) of a SoC. SoC taken into consideration is Ethernet MAC module with loopback mechanism using XGMII interface. The environment of verification, which is created by means of efficient verification methodology for system verilog is scalable, predictable and reusable and can reduce verification time. Using this verification environment, existence of bugs in the design can be found and design errors with corner cases can be easily located by the help of constraints.*

Keywords: Efficient verification methodology, MAC, Verification IP, constrained random verification, Gigabit Ethernet, XGMII.

I. Introduction

Due to advancement in fabrication technology more logic is being placed on a single silicon die. Many components are reused for improving time to market. Overall, more than 70 percent of the time is spent on verification. With the need of constructing a robust and reusable verification environment, efficient verification methodology was introduced to fulfil the goal. Another feature of efficient verification methodology is that, it is supported by all major simulator vendors unlike other methodologies. The goal of this paper is Verification of 10Gbps Ethernet MAC with loopback mechanism which decreases the overall simulation time with performance improvement.

SoC is build by integrating various subsystems which makes a verification task very challenging at SoC level [1]. Various challenges in SoC verification are as follows:
-reusing subsystem level verification environment for minimizing verification effort at SoC level

Connectivity between Intellectual Properties(IP)
1. System Level scenario verification
2.Synchronization of "C" based testcases running on Core with Subsystem/IP level verification environments for enabling maximum reuse

At SoC level, register sequences are used and internal Subsystem/IP level verification environments are configured as Passive agents and Interface IP level verification environments are considered as Active agents. Register sequences are not dependent on BUS protocol, it helps in reusing the register sequences with different BUS at IP/Subsystem/SoC level.

This approach addresses the following aspects of verification at SoC Level [2]

- Control and configuration of verification blocks from embedded software
- Reusability of different components of verification environment from IP to SoC level
- Testcase reusability from IP to SoC level.
- Handing over integration testcases to SoC team, developed by IP verification team.

II. Overview of MAC with XGMII

Table 1 shows IEEE 802.3 data frame which consists of 7 different fields. These fields are set together to form a single data frame which illustrates the 7 fields: Preamble, Start-of-Frame delimiter, Destination Address, Source Address, Length, Data, and Frame Check Sequence.

TABLE I

IEEE 802.3 Ethernet frame

PRE	SOF	DA	SA	Length	Data	FCS
7	1	6	6	2	46-1500	4

Ten Gigabit media-independent interface (XGMII) is a standard which is defined in IEEE 802.3 for connecting full duplex Ten Gigabit Ethernet (10GbE) ports with each other and for other electronic devices on printed circuit board (PCB) [3, 4]. It is comprised of two 32-bit data paths (Tx & Rx) and two four-bit control paths (Txc & Rxc), operating at 156.25 MHz's.

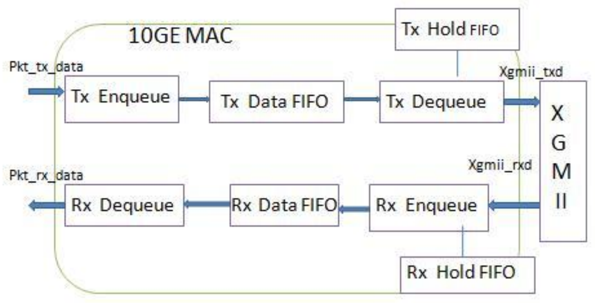

Fig 1: Block Diagram of 10GEMAC

XGMII interface is generally made up of a 32-bit data bus, clocked on rising and falling edges. For simplicity 10GE MAC uses a 64-bit interface (I/F) clocked on the rising edge.

Fig.1 shows block diagram of 10GEMAC has three major modules. Transmit engine, receive engine and XGMII.

A. Transmit Engine

The TX Enqueue Engine receives the frames and stores them in the transmit FIFO in addition to some additional flags like Start of Packet and End of Packet indicators. FIFO fill status is provided to the core by this Tx engine [5].

Fig. 2 shows transmit FIFO is having a depth of 128 bit entry with 64-bit of data and 8-bit of status/entry. As FIFO can store 512 bytes of data (64 x 64-bit), the MAC should operate in flow-through mode means the transmission of a frame on the XGMII interface can start on the enqueue side to the FIFO while the frame is still being written.

The TX Dequeue Engine consists of two state-machines. The first one reads data from data FIFO. After insertion of Ethernet preamble data the Encoding State-Machine reads data from Holding FIFO.

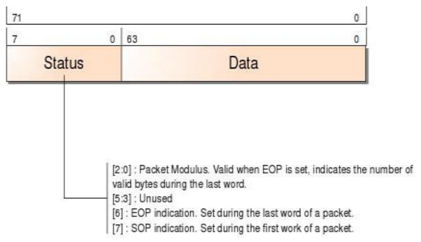
71
0
7
0
63
0
Status
Data
[2:0] : Packet Modulus. Valid when EOP is set, indicates the number of valid bytes during the last word.
[5:3] : Unused
[6] : EOP indication. Set during the last word of a packet.
[7] : SOP indication. Set during the first work of a packet.

Fig 2: Transmit FIFO format

B. Receive Engine

In the RX Enqueue Engine, the Reconciliation layer monitors the XGMII interface for fault condition detection and passes the status for checking [5].

Fig. 3 shows RX FIFO is having a depth of 128 bit entry with 64-bit of data and 8-bit of status/entry. As FIFO can store 512 bytes of data (64 x 64-bit), the MAC should operate in flow-through mode means the transmission of a frame on the packet interface can start on the enqueue side to the FIFO while the frame is still being written.

The RX Dequeue Engine interfaces with user logic. Its primary function is converting the internal status from RX FIFO to meaningful signals on pkt_rx interface.

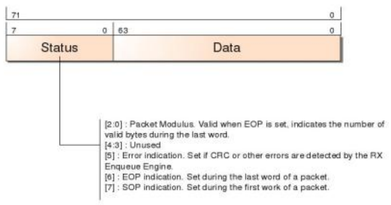
71
0
7
0
63
0
Status
Data
[2:0] : Packet Modulus. Valid when EOP is set, indicates the number of valid bytes during the last word.
[4:3] : Unused
[5] : Error indication. Set if CRC or other errors are detected by the RX Enqueue Engine.
[6] : EOP indication. Set during the last word of a packet.
[7] : SOP indication. Set during the first work of a packet.

Fig 3: Receive FIFO format

C. XGMII

XGMII Tx Control: On 64-bit interface, each bit corresponds to a byte. High status signifies that the byte is a control character and low status indicates that data is carried out by the byte.

XGMII Tx Data: While interfacing with 32-bit devices, xgmii_txd[31:0] is mapped to the positive edge of the clock and xgmii_txd[63:32] is mapped to the negative edge. XGMII Rx Control: On 64-bit interface, each bit corresponds to a byte. High Status Signifies that the byte is a control character and low status indicates that data is carried out by the byte.

XGMII Rx Data: While interfaced with 32-bit devices, xgmii_rxd[31:0] is mapped to the positive edge of the clock and xgmii_rxd[63:32] is mapped to the nagative edge.

Figure 4 shows loopback module provides a mechanism to verify the functionality of the MAC and PHY during simulation. When local loopback is enabled, the Ethernet loopback module takes the transmit frame from the MAC XGMII TX and loops it back to the MAC XGMII RX data path [6].

In accordance with the IEEE 802.3 Clause 46 Ethernet standard, transmitting frames on SDR XGMII interface, MAC Tx ensures the following

- Aligns the first byte of the frame to either lane 0 or lane 4 of the interface.
- Performs endian conversion. Client Transmit the frames to the interface are in big endian format. Frames transmitted on SDR XGMII are in the form of little endian. The MAC Tx transmits frame on this interface from LSB (least significant byte).

The MAC RX receives Ethernet frame from SDR XGMII Interface and forwards the payload with relevant frame information to the client after checking and filtering invalid frames in Rx data path. Data lanes coming through the SDR XGMII are decoded by MAC RX. First byte of the receive frame should received either in lane 0 (most significant byte) or lane 4, as expected by MAC Rx. The Rx frame should be preceded by a column of idle bytes or by an ordered set. Rx frame which does not satisfy this condition is invalid and the corresponding frame is dropped by MAC Rx. After this, the sequence of the frame is checked by MAC Rx. The frame begins with 1-byte START, 6-byte preamble data and 1-byte Start Frame Delimiter (SFD). Otherwise, MAC RX considers the frame as invalid and drops it eventually. For all valid frames, MAC RX removes the START, preamble data, Start Frame Delimiter (SFD) and End Frame Delimiter (EFD) bytes ensuring that the first byte of the frame is aligned to byte 0. End Frame Delimiter is removed by MAC Rx for all valid frames ensuring that the first byte of the frame is aligned to byte 0.

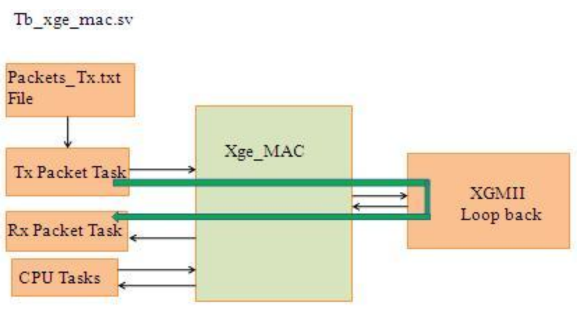

Fig 4: Block Diagram of 10GEMAC Loopback system

III. EFFICIENT VERIFICATION METHODOLOGY VERIFICATION ENVIRONMENT MAC WITH LOOPBACK

In order to verify and evaluate the performance of the MAC RTL and the gate-level net list, Efficient verification methodology verification environment [7], [8] has been set up. Apart from the traditional verification methods such as timing check by assertions [9], a reusable framework to obtain functional and code coverage [10] and constrained random data generation [11] has been implemented here. In MAC Verification Environment of Fig. 3, it integrates several encapsulated, ready-to-use and configurable verification agents. Each of them, the master or slave agent, contains three subcomponents: the sequencer, driver and monitor. As shown in Fig. 5, the signal level or the physical level includes the design for test (DUT) and each agent's driver. The drivers are used to drive (in a master) or respond (in a slave) the bus-signal interface according to the bus protocol. Considering the sequencer and the monitor, all the operations are design-specific and both of them are in the transaction level. The sequencer controls and arranges the flow of sequence items to the driver, and the monitor samples the activities and collects the transactions seen on the signal-level interfaces and sends them into the analyzer.

Assertion Checker: We have implemented an assertion checker. This checker checks whether the requests and responses are protocol specific or not. If any request or response which violates the protocol, it would generate an error response and test would be terminated immediately.

Tests and Sequences: Ethernet VIP has number of tests and a different sequence for each test. Few error tests with erroneous sequences are also added for the users to use to check how the DUT reacts when given an erroneous stimulus. This library of built-in sequencers helps jump-start the task to achieve coverage goals and will enable the user to get to high coverage very rapidly.

In the top module , dut and test are instantiated and connected using the interface. Interface is also passed to the verification environment using configuration.

```
module top;
import mac_test_pkg::*; import uvm_pkg::*;
logic clock_156m25;
logic clock_wb;
logic reset_156m25_n;
logic reset_xgmii_rx_n;
logic reset_xgmii_tx_n;
logic wb_rst_n;
logic [7:0] mgii_txc_reg;
logic [63:0] mgii_txd_reg;
initial
begin
fork wb_reset(); xgmii_rx_reset();
xgmii_tx_reset(); system_reset();
join
end
task wb_reset(); wb_rst_n = 0; repeat(2)
@(posedge clock _wb); wb_rst_n = 1;
endtask
mac_if mif(.clock(clock_156m25),
.reset_156m25_n(reset_156m25_n) );
```

Listing. Top Class

D. *Test Cases*

To check the functionality of the ETHERNET according to the specification the scenarios which have been covered are as follows- Receive Enable, Receive available, Valid data (Tx and Rx), Start of packet (Tx and Rx), End of packet (Tx and Rx), Modulus length (Tx and Rx), Packets data(Tx and Rx), Receive error and Transmit full.

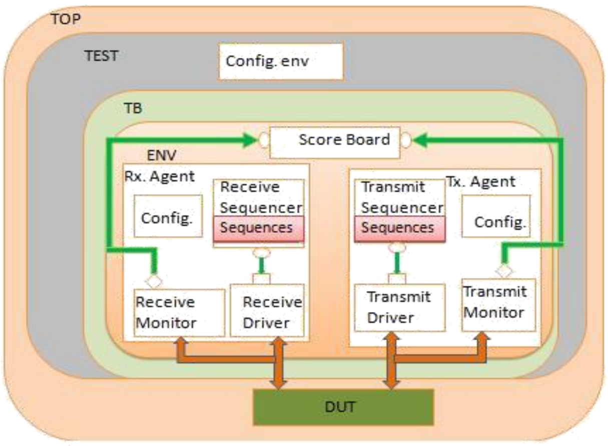

Fig 5: MAC with XGMII VIP Architecture

IV. Coverage Reports and Simulation Results

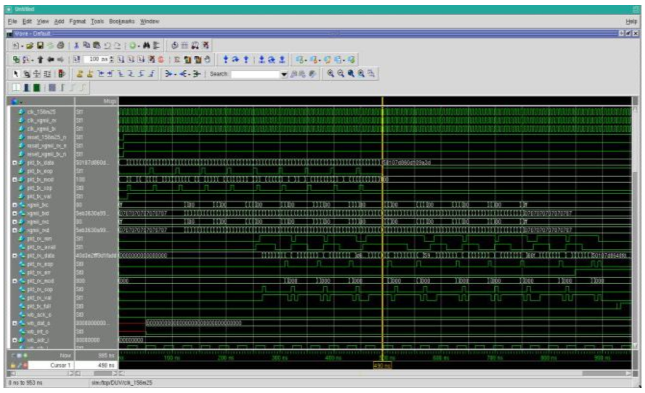

Fig 6: Simulation Results of MAC with XGMII

Fig. 6 represents the simulation result as viewed by using mentor graphics Questa tool. It displays transmitted data from the transmit engine to the XGMII receiver then it retransmit to the MAC through receiver engine while it receives the data from the XGMII transmitter.

TABLE III

Local instance coverage details

Weighted Average:				97.08%
Coverage Type	**Bins**	**Hits**	**Misses**	**Coverage (%)**
Branch	400	365	35	91.25%
Assertion Attempted	9	9	0	100.00%
Assertion Failures	9	0	-	0.00%
Assertion Successes	9	9	0	100.00%

From Simulation waveform we can observe that transmitted packets have been received in the receiver side as per our expectation following the specification.

Table 2 shows code and functional coverage report generated by the Questa-sim simulator for MAC with Loopback mechanism. 94.61% overall coverage has been achieved.

V. Conclusion

Gigabit Ethernet MAC verification has been presented here by using most advanced verification methodology. Verification is very crucial to find bugs which only appear with random stimulus. The constrained random approach is more time-efficient for reaching coverage goal compared to some other simpler methods such as directed test method. The verification flow in this research has not only reduced resources and efforts of SOC team to gather knowledge, develop test bench and test cases and debugging but also minimized the IP team's resources and efforts as well.

Acknowledgement

Authors would like to express sincere thanks to Department of Science and Technology, New Delhi for their financial support to carry out this work under project Grant No. SR/WOS-A/ET- 17 /2012(G). Further our sincere feelings and gratitude to management and principal of Sumathi Reddy Institute of Technology for Women, for their support and encouragement to carry out the research work.

References

[1] Young-Nam Yun;, "Beyond UVM for practical SoC verification", SoC Design Conference (ISOCC), 2011 International, pp158-162, 2011.

[2] Creating a reusable testbench using cadance's testbuilder and AMBA TVM Prakash Rashinkar, Peter Paterson and Leena Singh, SYSTEM-ON-A-CHIP VERIFICATION Boston: Kluwer Academic Publishers, 2001.

[3] MV Lau,, S. Shieh, Pei-Feng Wang, B. Smith, D. Lee, J. Chao, B. Shung, and Cheng-Chung Shih, "Gigabit ethernet switches using a shared buffer architecture,"Communications Magazine, IEEE, vol. 41, no. 12, pp. 76 - 84, dec. 2003.

[4] Assaf, M.H, Arima ; Das, S.R. ; Hernias, W, Petriu, E.M, "Verification of Ethernet IP Core MAC Design Using Deterministic Test Methodology", IEEE International instrumentation and Mesurements Technology Conference, doi.10.1109/IMTC.2008.4547312, victoria, May 2008.

[5] Tonfat, J, Reis, R, "Design and Verification of a layer-2 Ethernet MAC classification Engine for a gGigabit Ethernet Switch", Proc IEEE Electronics, Circuits, and Systems doi. 10.1109/ICECS.2010.5724475, Athens, Dec 2010.

[6] Frazier,H. "The 802.3z gigabit Ethernet Standard", Proc IEEE J, doi10.1109/65.690946, vol-12, May-June 1998.

[7] H. D. Foster, A. Krolnik, and D. Lacey, "Assertion-Based Design," 2nded., Kluwer Academic Publishers, 2004.

[8] J. Bergeron, E. Cerny, A. Hunter, A. Nightingale, "Verification Methodology Manual for SystemVerilog," Springer Publisher, Sep 1, 2005, pp. 260-282.

[9] Accellera, UVM 1.1 Reference Manual, Jun. 2011

[10] Accellera, UVM 1.1 User Guide, May. 2012.

[11] J. Cho, S. Choi, S.-I. Chae, "Constrained-Random Bitstream Generation for H.264/AVC Decoder Conformance Test," IEEE Trans. on Consumer Electronics, vol. 56, no. 2, pp. 848-855, May. 2010.