Ethernet MAC Verification by Efficient Verification Methodology for SOC Performance Improvement

Sridevi Chitti, P. Chandrasekhar, M. Asharani and G. Krishnamurthy

Abstract Verification of Gigabit Ethernet Media Access Control (MAC), part of most of the networking SOC is accomplished by using the most advanced verification methodology i.e. Universal Verification Methodology (UVM) has been presented in this paper. The main function of MAC is to forward Ethernet frames to PHY through interface and vice versa. With the use of UVM factory and configuration mechanism, coverage driven verification of MAC Characteristics such as frame transmission, frame reception etc. is achieved in best possible way. Coverage metrics and self-checking which reduces the time spent on verifying design. By using UVM methodology, a reusable test bench is developed which has been used to run different test scenarios on same TB environment.

Keywords UVM · MAC · Verification IP · Gigabit ethernet · TB environment

1 Introduction

In general, for verifying a SoC firstly, we need to verify the standard bus interconnecting IP Cores present in the system [1]. The whole verification process of SoC consumes approximately 70 % of total design time. In this research work, the problems taken care of are as follows:

S. Chitti (✉) · P. Chandrasekhar · M. Asharani · G. Krishnamurthy
Department of Electronics and Communication Engineering, SRITW, Warangal, Telangana, India
e-mail: Sridevireddy.arram@gmail.com

P. Chandrasekhar
e-mail: sekharpaidimarry@gmail.com

M. Asharani
e-mail: ashajntu1@yahoo.com

A. Nagar et al. (eds.), *Proceedings of 3rd International Conference on Advanced Computing, Networking and Informatics*, Smart Innovation, Systems and Technologies 44, DOI 10.1007/978-81-322-2529-4_12

1. Verification of Ethernet MAC which is an essential part of Ethernet SoC verification.
2. Development of VIP for MAC unit.
3. Using that MAC VIP, Ethernet MAC has been verified and coverage analysis has been performed [2].

2 Universal Verification Methodology (UVM)

UVM uses system Verilog as its base language. UVM methodology is vendor independent which is not the case for rest of the verification methodologies [3]. Need of a common verification methodology which provides the base classes and framework for constructing scalable and reusable verification environment has been achieved by the introduction of UVM [3]. UVM Improves productivity and ensures re-usability. Maintenance of the verification components is much easier because the components are standardized.

3 Proposed System

10 Gigabit Ethernet MAC implements a MAC controller conforming to IEEE 802.3 specification. This proposed system consists of two modules namely transmit module and receive module. Table 1 shows IEEE 802.3 data frame which consists of seven different fields. These fields are put together to form a single data frame which illustrates the seven fields: Preamble, Start-of-Frame delimiter, Destination Address, Source Address, Length, Data, and Frame Check Sequence [4–6].

3.1 Transmit and Receive Module

The transmit and receive engine provides the interface between the client and physical layer. Figure 1 shows a block diagram of the transmit and receive engine with the interfaces to the client and physical layer and vice versa [7].

The Fig. 2 describes the components of ETHERNET MAC verification Architecture, which consists of verification components like agent, driver etc.

Table 1 IEEE 802.3 ethernet frame

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

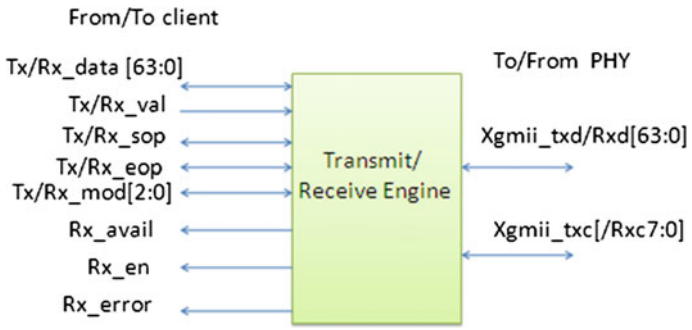

Fig. 1 Block diagram of ETHERNET transmit module

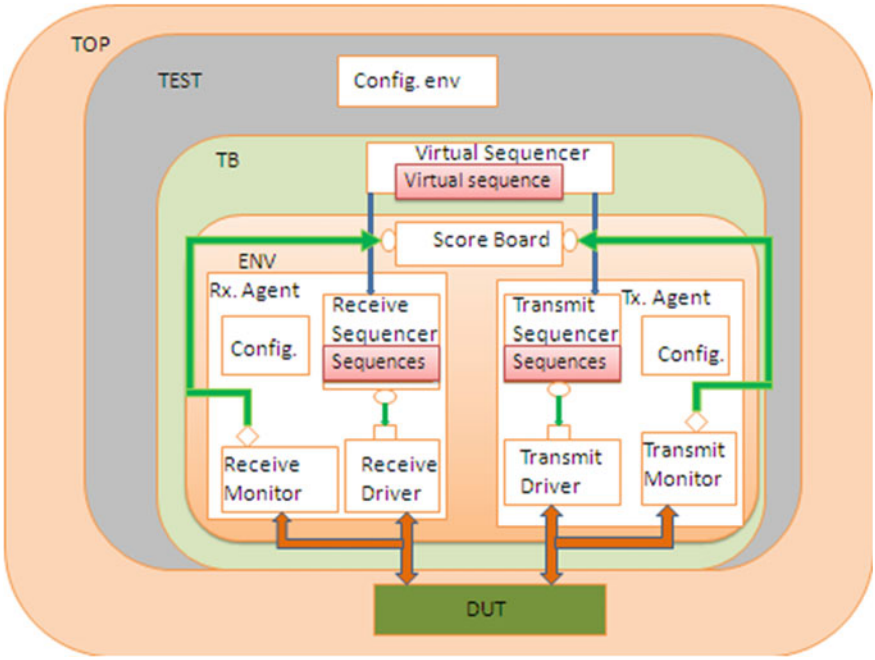

Fig. 2 ETHERNET VIP architecture

Generally data items are generated and transmitted to the DUV in a typical test. A large number of meaningful tests can be created by randomizing data item fields using System Verilog constraints thus maximizing coverage [8].

A driver fetches data repeatedly from sequencer, drives the DUT based on the protocol using the virtual interface.

```
task mac_tx_driver::run_phase(uvm_phase phase);
                forever begin
            seq_item_port.get_next_item(req);
            send_to_dut(req);
            seq_item_port.item_done();
            end
        endtask
```

The sequences cannot directly access test bench resources, which are available in the component hierarchy. Using a sequencer, sequences can access test bench resources as a key into the component hierarchy [9].

```
class mac_tbase_seq extends uvm_sequence #(transmit_xtn);
        task mac_tx_xtns::body();
    begin
     req=transmit_xtn::type_id::create("req");
     strat_item(req);
assert(req.randomize()  with  {data[2]  inside  {[20:0]};
en==1'b1; val==1'b1; avail==1'b1;});
          finish_item(req);
    end
    endtask
```

The monitor extracts signal information from the bus and translates it into transactions. Monitor is connected to other components via standard TLM interfaces like Analysis port and export.

```
task mac_tx_monitor::run_phase(uvm_phase phase);
         forever
         collect_data();
        endtask
    // add logic here
```

In an agent there are three specific components viz: sequencer, driver, and monitor. They can be reused independently. ETHERNET MAC has two agents: transmit and receive agents.

A scoreboard is a crucial element which verifies the proper operation of the design at functional level by comparing the predicted output from reference model with the actual output from receiver [8, 9].

```
function  void  mac_scoreboard::mac_transmit(transmit_xtn
tr);
if(txd_fifo_status ==0 && tr.full==1)
`uvm_info("MAC             transmit             function",
$psprintf("fifo_data=%b",fifo_data), UVM_LOW)
     begin
    fifo_en=tr.val;
     fifo_data = tr.data;
end
// add logic here
// similarly add mac_receive() method
 $cast(ref_xtn , re.clone());
    if(mac_receive(ref_xtn))
    begin
//compare
`uvm_info(get_type_name(),  $sformatf("Scoreboard  -  Data
Match successful"), UVM_MEDIUM)
xtns_compared++ ;
```

The environment acts as the top-level component for all the verification components. Interface is a static component that encapsulates communication between the hardware blocks. It provides a mechanism to group together multiple signals into a single unit that can be passed around the design hierarchy thus reducing the amount of code and promotes reuse [8].

Based on the declared directions, Modport restricts the interface access within a module. The function of the clocking block is to identify the clock signals and to capture the synchronization and timing requirements of the modeled blocks. With the help of the clocking block, test bench drives the signals on time. Interface can contain more than one clocking block depending on the environment. Set up and hold time of the DUV can also be modeled [8].

```
interface mac_if(input bit clock);
     //  transmiter driver CB
   clocking tdr_cb @ (posedge clock);
          default input #1  output #1;
          output tx_data, tx_val,tx_sop, tx_eop,tx_mod;
          input tx_full;
    endclocking
modport tdr_mp(clocking tdr_cb);
```

Virtual interface instance is created by using keyword “virtual”. By which drivers and monitors can be created and deleted dynamically during run time [9].

A sequencer that is not attached to any driver and does not process items by itself can be used for high-level control of multiple sequencers from a single sequencer. This kind of sequencer is referred to as a virtual sequencer [10].

Table 2 Local instance coverage details

Weighted average				94.61 %
Coverage type	Bins	Hits	Misses	Coverage (%)
Branch	400	357	43	89.23
Assertion attempted	10	10	0	100.00
Assertion failures	10	0	–	0.00
Assertion successes	10	10	0	100.00

Table 3 Coverage details for transmit and receive packet cover group

Coverage group	Goal (%)	% of Goal
Cover group type mac_fcov1	100.00	89.58
Cover group type mac_fcov2	100.00	88.88

3.2 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.

4 Coverage Reports and Results

According to Test Plan, the test cases are verified by developing the Verification IP for Ethernet Protocol. The Test Cases are written in the form of sequences in the Sequencer using System Verilog UVM methodology.

By using Questa simulation software, the Verification of Ethernet components such as transmit Agent and receive agent are done and the log files for the test cases are generated with Coverage report.

Table 2 shows the coverage of the whole environment for code and functional coverage. 94.61 % overall coverage has been obtained. Table 3 shows Tx coverage is 89.58 % and Rx coverage is 88.88 %. The cover group coverage is not 100 % as all the registered address is not required to be checked which results in 89.23 % coverage.

5 Conclusion

The specifications of ETHERNET are verified successfully using UVM methodology on QuestaSim simulator. Functional coverage i.e. measure of implementation of design is carried out and 94.61 % of coverage is extracted. The coverage can

be improved by modifying the code according to the need. The scoreboard successfully compares the result of every transaction generated.

Acknowledgments 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. Chauhan, P., Clarke, E.M., Lu, Y., Wang, D.: Verifying IP core based system-on-chip designs. Carnegie Mellon University Research Showcase
2. Samanta, P., Chauhan, D., Deb, S., Gupta, P.K.: UVM based STBUS Verification IP for Verifying SOC Architectures. In: Proceedings of IEEE VLSI Design and Test, 18th International Symposium, doi:10.1109/ISVDAT.2014.6881037, Coimbatore, July 2014
3. Vaidya, B., Pithadiya, N.: An introduction to universal verification methodology. J. Inf. Knowl. Res. Electron. Commun. Eng. **2**, Nov-12 to Oct-13
4. Assaf, M.H., Arima; Das, S.R., Hernias, W., Petriu, E.M.: Verification of ethernet IP core MAC design using deterministic test methodology. In: IEEE International Instrumentation and Mesurements Technology Conference, victoria, May 2008. doi:10.1109/IMTC.2008.4547312
5. Tonfat, J., Reis, R.: Design and verification of a layer-2 ethernet MAC classification engine for a gGigabit ethernet switch. In: Proceedings of IEEE Electronics, Circuits, and Systems, Athens, Dec 2010. doi:10.1109/ICECS.2010.5724475
6. Frazier, H.: The 802.3z gigabit ethernet standard. In: Proceedings of IEEE Journal, vol. 12, May–June 1998. doi:10.1109/65.690946
7. Lau, M.V., Shieh, S., Wang, P.-F., Smith, B., Lee, D., Chao, J., Shung, B., Shih, C.-C.: Gigabit ethernet switches using a shared buffer architecture. Communications Magazine, IEEE, **41**(12), 76–84 (2003)
8. Bergeron, J.: Writing Test Benches using SystemVerilog. Springer, ISBN-10: 0-387-29221-7, Business Media (2006)
9. www.accellera.org/
10. www.testbench.in/ [online]