TESTPLAN DOCUMENT FOR 10GE MAC CORE VERIFICATION PROJECT

Prepared By:
Mrinal Mathur

Contents

1. Project Overview

This project is intended as a hands-on opportunity to build a UVM-based verification environment, on a 10Gb Ethernet MAC Core design. The source code for the 10GE Mac Core is provided as Verilog RTL. Furthermore, a rudimentary testbench and simulation environment is also provided.

The first step in the project is to get the understanding of the specification of the design. Next step will be to understand existing verification environments. Once the basic sanity checks are performed and testbench is working as expected, building of the UVM testbench is started.

The main purpose of the project is to build a UVM infrastructure where each component has a specialized job. This document explains all the components involved in creating the testbench by giving a brief description and providing reference to the actual blocks in the design.

In addition to the provided testcase, five other testcases were added to the project. These test cases tests the different scenarios of the design and the randomization technique is used to explore the corner cases.

The master script is also added to the project. The script is designed to execute all the test cases, provides log files and displays summary.

2. Unit Under Test

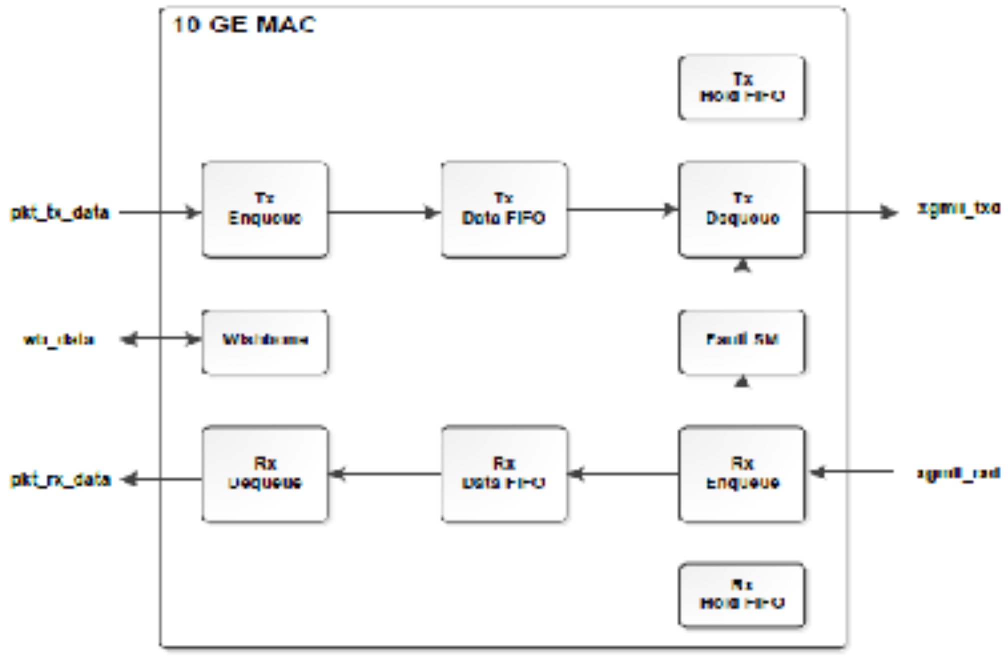
The 10GE Mac design is an Open Source design. The RTL is available freely for anyone who wishes to use it

There are 3 different interfaces associated with the 10gEthernetMAC design:
1. PKT_TX and PKT_RX interface
2. XGMII_RX and XGMII_TX interface
3. WISHBONE interface

In this project, we will be dealing only with the TX and RX interface (loopback). This is to say that the XGMII interface will be configured in loopback mode.

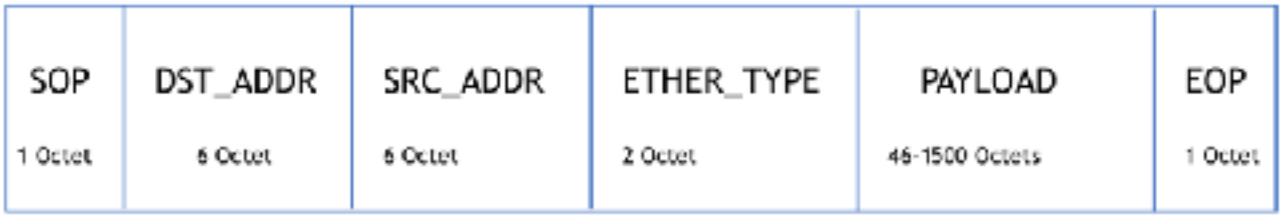
The wishbone interface present is only initialized and is not been used to verify.

The top level diagram of the design is as shown below:



3. Ethernet Frame

The structure of ethernet frame according to the IEEE standards are as shown above:



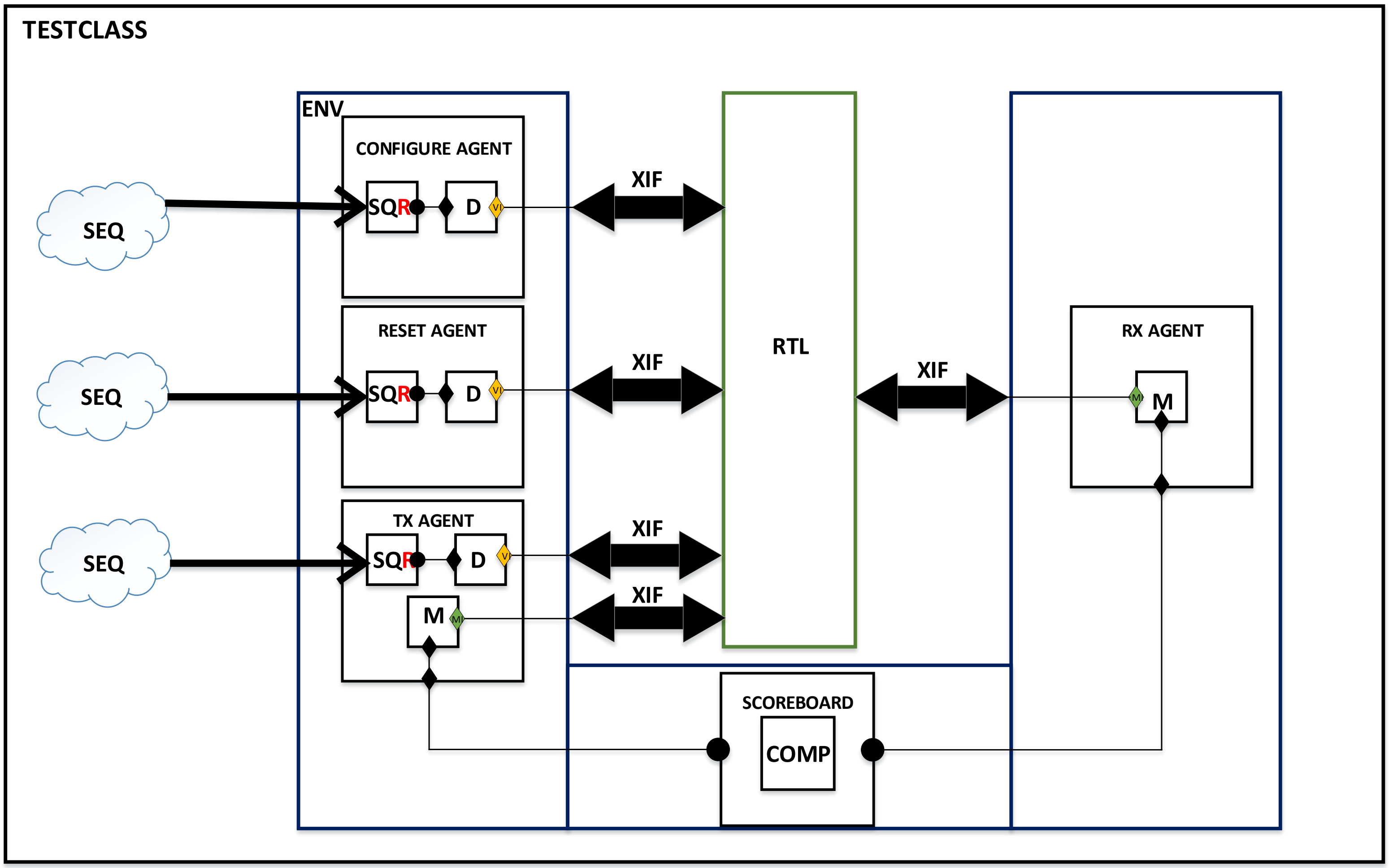
SOP	DST_ADDR	SRC_ADDR	ETHER_TYPE	PAYLOAD	EOP
1 Octet	6 Octet	6 Octet	2 Octet	46-1500 Octets	1 Octet

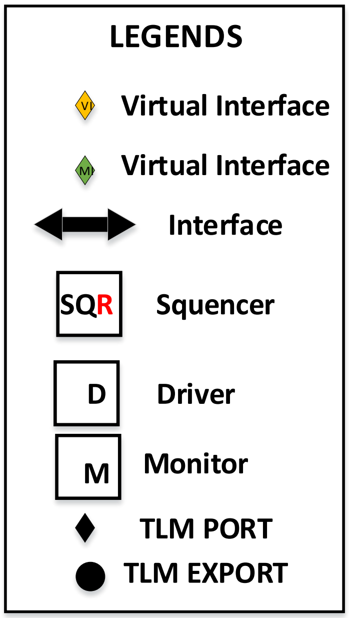
Fig. Ethernet Frame Structure

SOP (Start of Packet): The ethernet packet starts with SOP bit which is set high when a new packet transfer starts.
DST_ADDR (Destination Address): This is the Destination MAC address.
SRC_ADDR (Source Address): This is the Source MAC address.
Ether_Type (Ether Type): This denotes the ethernet protocol used in the payload.
Payload : The minimum payload is 46 octets and maximum is 1500 octets. When the actual payload is less, padding bytes are added by the design. Jumbo frames are the ones with payload greater then 1500 octets.
IPG (Inter Packet Gap): This is the idle time between packets. This is ideally between 10-50 octets of idle time.

4. Testbench Architecture

Below depicts the architecture of the testbench used in this project.





5. Testbench Components

Verification Environment or **Testbench** is used to check the functional correctness of the Design Under Test (**DUT**) by generating and driving a predefined input sequence to a design, capturing the design output and comparing with-respect-to expected output.

Verification environment is formed by grouping the several components performing specific task/operation.

In verification environment, separate class's will be written to perform specific operation i.e, generating stimulus, driving, monitoring etc. and those class will be named based on the operation.

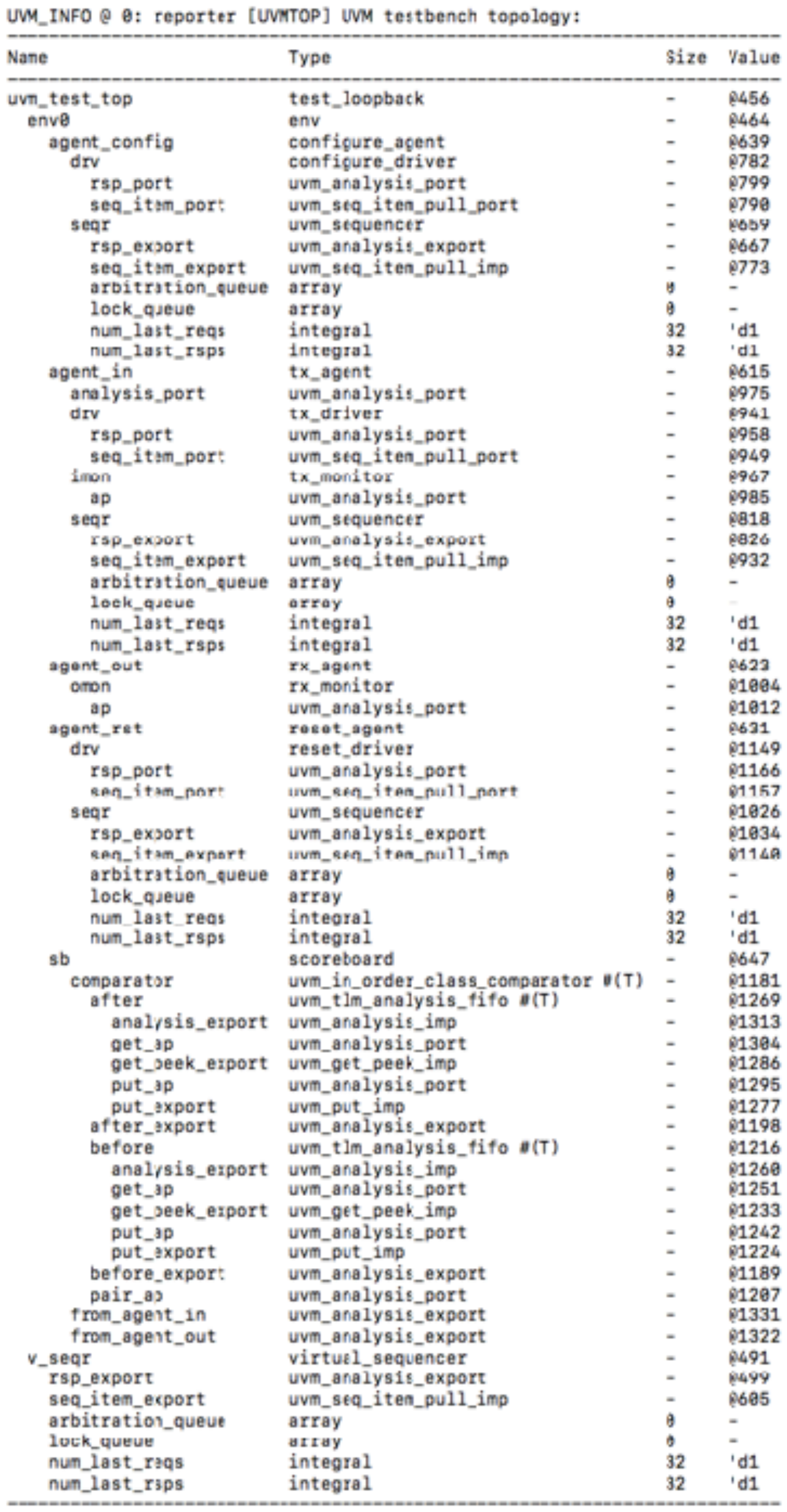
Below is the list of class's that the verification environment have,

Name	Type	Description
Configure Agent	class	This agent contains driver class to send the configure/initialization sequence packets to the RTL
Reset Agent	class	This agent contains driver class to send the reset sequence packets to the RTL
TX Agent	class	This active agent contains driver and monitor class to send/receive packets to the RTL
RX Agent	class	This passive agent contains monitor class to receive packets from the RTL
Sequencer	class	This is UVM built-in class. It runs on the connected sequence and provides the packets to the Driver.
driver	class	Generates the stimulus and drives the packet level data inside transaction into pin level (to DUT). It request packets from Sequencer running on sequence.
monitor	class	Observes pin level activity on interface signals and converts into packet level which is sent to the components such as scoreboard using TLM Port.
scoreboard	class	Receives data items from monitors and compares with expected values using TLM export. Expected values can be either golden reference values or generated from reference model.
environment	class	The environment is a container class for grouping higher level components like agents and scoreboard.

Testclass	class	This is a container class for grouping components like environment and sequences.
Testcase	program	Test is responsible for, • Configuring the testbench. • Initiate the testbench components construction process. • Initiate the stimulus driving.
Test bench	module	This is the top most file, which connects the DUT and TestBench. It consists of DUT, Test and interface instances, interface connects the DUT and TestBench.
xge_Interfac e	interface	This is the interface module, which contains bundled wires connecting testcase, driver, monitor and UUT.
xge_mac	module	This is the top level DUT
packet	class	Creates packet with all its attributes.

6. Factory Registered Components Topology

Below diagram depicts the component class hierarchy. The top most entity is UVM_Test_Top which is created by UVM. All the child components are created by the Verification engineer and registered with the factory. Below you can see the Factory Components Topology as printed out by running simulation:



```
UVM_INFO @ 0: reporter [UVMTOP] UVM testbench topology:
--------------------------------------------------------------------------
Name                       Type                                Size  Value
--------------------------------------------------------------------------
uvm_test_top               test_loopback                       -     @456
  env0                     env                                 -     @464
    agent_config           configure_agent                     -     @639
      drv                  configure_driver                    -     @782
        rsp_port           uvm_analysis_port                   -     @799
        seq_item_port      uvm_seq_item_pull_port              -     @790
      seqr                 uvm_sequencer                       -     @659
        rsp_export         uvm_analysis_export                 -     @667
        seq_item_export    uvm_seq_item_pull_imp               -     @773
        arbitration_queue  array                               0     -
        lock_queue         array                               0     -
        num_last_reqs      integral                            32    'd1
        num_last_rsps      integral                            32    'd1
    agent_in               tx_agent                            -     @615
      analysis_port        uvm_analysis_port                   -     @975
      drv                  tx_driver                           -     @941
        rsp_port           uvm_analysis_port                   -     @958
        seq_item_port      uvm_seq_item_pull_port              -     @949
      imon                 tx_monitor                          -     @967
        ap                 uvm_analysis_port                   -     @985
      seqr                 uvm_sequencer                       -     @818
        rsp_export         uvm_analysis_export                 -     @826
        seq_item_export    uvm_seq_item_pull_imp               -     @932
        arbitration_queue  array                               0     -
        lock_queue         array                               0     -
        num_last_reqs      integral                            32    'd1
        num_last_rsps      integral                            32    'd1
    agent_out              rx_agent                            -     @623
      omon                 rx_monitor                          -     @1004
        ap                 uvm_analysis_port                   -     @1012
    agent_rst              reset_agent                         -     @631
      drv                  reset_driver                        -     @1149
        rsp_port           uvm_analysis_port                   -     @1166
        seq_item_port      uvm_seq_item_pull_port              -     @1157
      seqr                 uvm_sequencer                       -     @1026
        rsp_export         uvm_analysis_export                 -     @1034
        seq_item_export    uvm_seq_item_pull_imp               -     @1140
        arbitration_queue  array                               0     -
        lock_queue         array                               0     -
        num_last_reqs      integral                            32    'd1
        num_last_rsps      integral                            32    'd1
    sb                     scoreboard                          -     @647
      comparator           uvm_in_order_class_comparator #(T)  -     @1181
        after              uvm_tlm_analysis_fifo #(T)          -     @1269
          analysis_export  uvm_analysis_imp                    -     @1313
          get_ap           uvm_analysis_port                   -     @1304
          get_peek_export  uvm_get_peek_imp                    -     @1286
          put_ap           uvm_analysis_port                   -     @1295
          put_export       uvm_put_imp                         -     @1277
        after_export       uvm_analysis_export                 -     @1198
        before             uvm_tlm_analysis_fifo #(T)          -     @1216
          analysis_export  uvm_analysis_imp                    -     @1260
          get_ap           uvm_analysis_port                   -     @1251
          get_peek_export  uvm_get_peek_imp                    -     @1233
          put_ap           uvm_analysis_port                   -     @1242
          put_export       uvm_put_imp                         -     @1224
        before_export      uvm_analysis_export                 -     @1189
        pair_ap            uvm_analysis_port                   -     @1207
      from_agent_in        uvm_analysis_export                 -     @1331
      from_agent_out       uvm_analysis_export                 -     @1322
  v_seqr                   virtual_sequencer                   -     @491
    rsp_export             uvm_analysis_export                 -     @499
    seq_item_export        uvm_seq_item_pull_imp               -     @605
    arbitration_queue      array                               0     -
    lock_queue             array                               0     -
    num_last_reqs          integral                            32    'd1
    num_last_rsps          integral                            32    'd1
--------------------------------------------------------------------------
```

7. Testcases

1) loopback/testcase.sv
   The first testcase is the “loopback” scenario. This is the exact same testcase as the one that comes with this rudimentary testbench. The main purpose of this particular testcase is to bring up the testbench environment.

2) missing_eop/testcase.sv
   This testcase is a scenario where EOP is missing in the packet. The design is expected to handle it gracefully and receive the next packet normally.

3) missing_sop/testcase.sv
   This testcase is a scenario where SOP is missing in the packet. The design is expected to handle it gracefully by skipping the current packet and receive the next packet normally.

4) undersize_packet/testcase.sv
   This testcase is a scenario where packet size is smaller then the design specs. The design is expected to handle it gracefully by appending 0’s in the end to make it of minimum specification.

5) oversize_packet/testcase.sv
   This testcase is a scenario where packet size is bigger then normal. The design is expected to handle it gracefully by receiving it normally.

6) zero_ipg_packet/testcase.sv
   This testcase is a scenario where inter packet gap is zero. The design is expected to function normally by receiving all the packets.

8. OOP Testbench Directory Tree

The following directories contain all design and testbench related files and documents:
lab-project-10gEthernetMac

- doc/
  - o This is where specifications doc is located.
- rtl/
  - o include/ (has verilog defines parameters and other utilities)
  - o verilog/ (contains RTL source code for the 10gEthernetMAC design)

- scripts/
  - o This is where regression scripts i.e. regression.py is present. In this directory, type “python regression.py -h” for information relating to running the script. This script executes the individual runsim scripts, creates log files in respective directory and provides summary of all tests performed.

- sim/
  - o verilog/(simulationdirectory)
    It contains the runsim script of all the testcases. These runsim script can be manually run here as well.

- testbench/
    - o verilog/ The top-level testbench, “testbeanch.sv” is located here. It also contains all the component as well as data classes.

- testcases/ This directory houses all your testcases
    - o loopback/testcase.sv
    - o missing_eop/testcase.sv
    - o missing_sop/testcase.sv
    - o undersize/testcase.sv
    - o oversize/testcase.sv
    - o zero_ipg/testcase.sv

9. Summary and Conclusion

The design was verified by building the Constrained-Random UVM based testbench environment. The component classes were created as part of Testbench infrastructure. The data variable such as packets were used during runtime to drive and monitor the design. Scoreboarding technique was used to verify the actual packets data versus the expected data. The course and the project provided the necessary skills required for building the UVM-based testbench. The professor went in to details of all the testbench components as well as specified the Do’s and Don’ts of the programming techniques. The assignments after each class helped in eventually building a complete infrastructure for the project.