

Maseeh College of Electrical and Computer Engineering

Verification of UDP Protocol using UVM

ECE 593 – Spring 2018  
Final Project Report

Team members

Vadiraja Mysore Nagendra   
Sai Teja Bodanki   
Manisha Balakrishna Garade

Index

|  |  |  |
| --- | --- | --- |
| No. |  | Page no. |
| 1. | Objective | 3 |
| 2. | What was accomplished | 3 |
| 3. | Introduction to UDP Protocol | 3 |
| 4. | Specification of UDP Protocol | 3 |
| 5. | Workflow | 4 |
| 6. | Block diagram of the verification environment | 5 |
| 7. | Sequences & Stimuli | 5 |
| 8. | Various tests | 6 |
| 9. | Driver | 7 |
| 10. | Monitor | 7 |
| 11. | Analysis Port block diagram | 7 |
| 11. | Block diagram of the Analysis Port | 7 |
| 12. | Coverage | 7 |
| 13. | Checker | 10 |
| 14. | Introduction/Injection of bugs through RTL | 11 |
| 15. | Challenges encountered | 11 |
| 16. | Pre-existing ideas/codes borrowed | 11 |
| 17. | What if we had more time? | 11 |
| 18. | Future scope | 11 |
| 19. | Output waveform | 11 |
| 20. | Sources referred | 12 |

**Objective:** To verify the *User Datagram Protocol* (UDP) using UVM.

**What was accomplished:** Verified UDP core functionality as proposed in the verification plan.

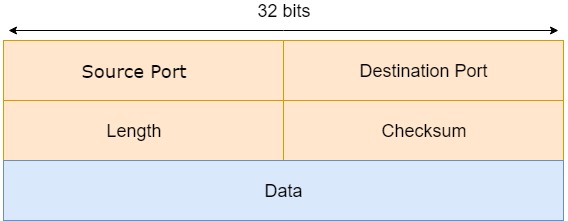
**Introduction to UDP Protocol**

The Internet protocol suite is the conceptual model and set of communications protocols used on the Internet and similar computer networks. It is commonly known as UDP/IP because the foundational protocols in the suite are the User Datagram Protocol (UDP) and the Internet Protocol(IP). It is a simple transmission model.

* It has no handshaking dialogues
* No guarantee in delivery or duplicate protection

**Specification of UDP Protocol**

The UDP Protocol is a packet-based protocol. The packet consists of UDP header and UDP data.



The header consists of 4 fields. The fields are 2 bytes each.   
  
Source port: The source port identifies the sender’s port.  
Destination port: The destination port identifies the receiver’s port.  
Length: The length field specifies the length of the UDP header and the UDP data.  
Checksum: This field is used for error checking for header and data.

**Work Flow**  
The Verification was divided into three parts between each team member to meet 2 Phases.

**Phase 1 - Test UDP TX Path (40 Hours / Person)**

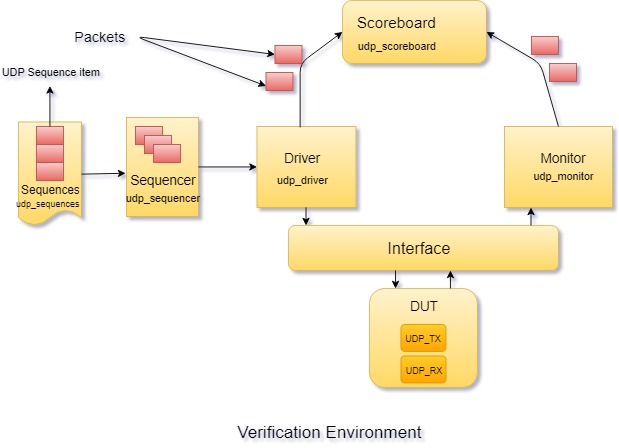
* Constrained Random Stimuli Generation by Vadi
  + Various Types of Payload patterns (More Weights to PRBS, RAMP)
    - RAMP (0,1,2…)
    - TRIANGLE (0,1,2…2,1,0)
    - All Zeros, Ones
    - Stream of Low and High (0,0,0,255,255,255)
    - PRBS – RANDOM
  + Various SRC and DST ports (Constrained to a range of 1 - 2^12)
  + Variable Length Packets (1-1572)
* Interface with Function call to do transaction that follows UDP TX Protocol, UVM Framework with one UDP Agent (Monitor, Driver, Sequencer) by Sai
* Functional Coverage to analyze the I/O by Manisha
  + Type of packets
  + SRC, DST (Various Ranges)
  + Length (Various Ranges)
* Scoreboard to check the functionality.

**Phase 2 – Test Full TX-RX with a Loopback (30 Hours/Person)**

* Add UDP TX module onto the design and update Interface module with extra functions to support RX protocol and Add Loopback connection form TX to RX by Sai
* UVM Framework additional Checks for RX Protocol and Monitor changes by Vadi
* Added Coverage bins to support RX protocol and Added more cross coverage to TX and Scoreboard changes for RX module by Manisha

In the end of Phase 2 we have complete framework to send a packet form TX Input and Could receive the same from RX via the Loopback.

**Block diagram of the Verification Environment:**



The figure shows the block diagram of the verification environment. How each block was used in our project is explained as follows.

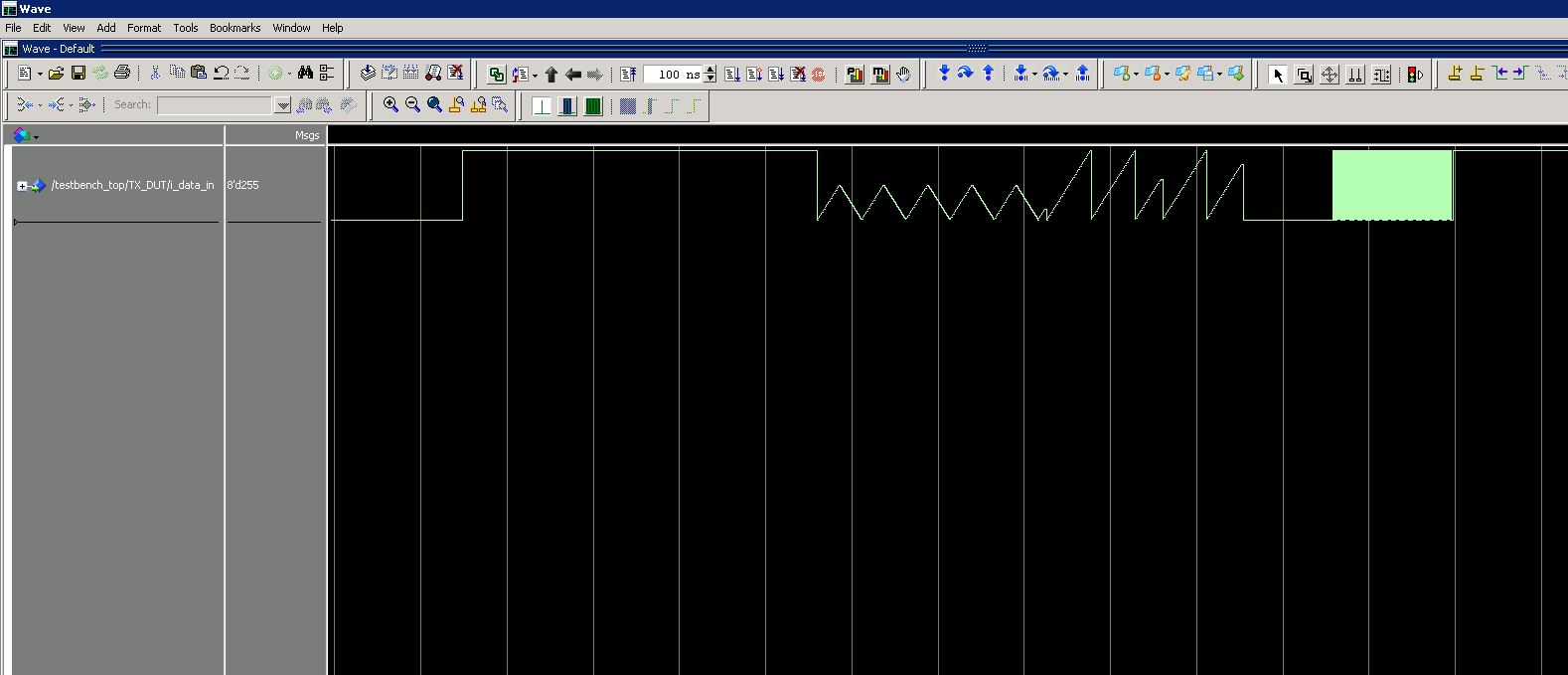
**Sequences:**

Different types of test cases were sent through sequences (test cases discussed in the next section).

**Stimulus:**

The Idea is to generate all kinds of patterns, cases, range of values which the module can support and not support, monitor them, check for any protocol violations.

The various patterns that were sent is as shown in the diagram below



The list explains all kinds of tests the we performed, including the results

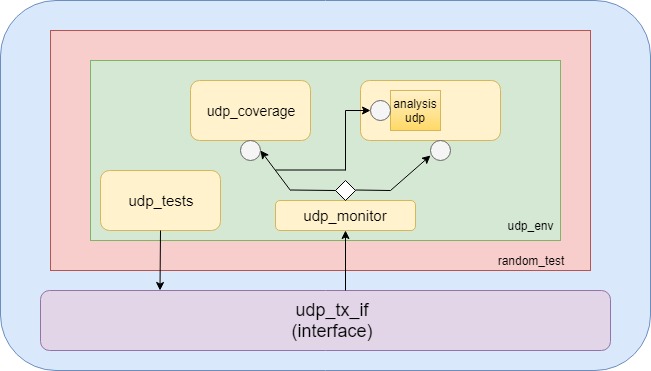
|  |  |  |  |
| --- | --- | --- | --- |
| Module | Test | Pass/Fail | Note |
| TX | Send Variable Length Packets | Pass | Basic |
| TX | Send Variable DST, SRC ports | Pass | Basic |
| TX | Terminate Packet after giving Header | Pass | Corner Case |
| TX | Terminate Packet before Sending data equivalent to Length in Header | Fail | Corner Case |
| TX | 0 Latency Between TX and RX | Pass | Basic Functionality |
| RX | Drop Non-UDP Packets | Pass | Basic Functionality |
| RX | Back to Back UDP packets | Pass | Basic Functionality |
| RX | UDP after Non UDP packets | Pass | Basic Functionality |
| RX | Non UDP after UDP Packets | Pass | Basic Functionality |
| RX/TX | By Pass Error Packets | Pass | Corner Case |
| RX/TX | AXI stream for data | Pass | Basic Functionality |
| RX/TX | Exceed Maximum Length | Pass | Corner Case |
| RX/TX | Checksum | Fail | Not Supported |
| TX | Real time packets logged from wireshark | Pass | Basic Functionality |

**Driver:**

The driver drives the randomized stimulus on to the DUT via the interface. The driver unpacks the sequence item from the sequencer and sends the signals at the low-level pin interface.   
  
**Monitor:**

The monitor receives the output from the DUT through low level pin interface and packs the signals and sends it to the scoreboard through the analysis port and the scoreboard being a subscriber receives the packet.

**Analysis Port:**



**Coverage:**

The coverage determines the quality of the testbench. The coverage includes input and output coverage which internally include cross coverage.

The below table is a description of the input coverage.

|  |  |  |
| --- | --- | --- |
| Signals | Number of bins | Description |
| Udp\_tx\_start | 2 - Each bin for low and high. | Sent to start header transfer |
| Udp\_tx\_data\_length | 3 bins – 1 bin each for lowest value and highest value and one bin for anything in between | Indicates the data length of the stimulus packet. |
| Udp\_tx\_src\_port | Array bin for most of the destination ports that lie in between 1 and 8000. | Source port address |
| Udp\_tx\_dest\_port | Array bin for most of the destination ports that lie in between 1 and 8000. | Destination port address |
| Udp\_tx\_dst\_ip\_addr | 3- One bin each for minimum and maximum value and one bin for the remaining values. | Destination IP address of the UDP packet. |
| Udp\_tx\_data\_valid | 2-One bin each for low and high value. | Indicates valid data on the data bus |
| Udp\_tx\_data\_last | 2- One bin each for low and high value | Indicates Last byte on the data bus |

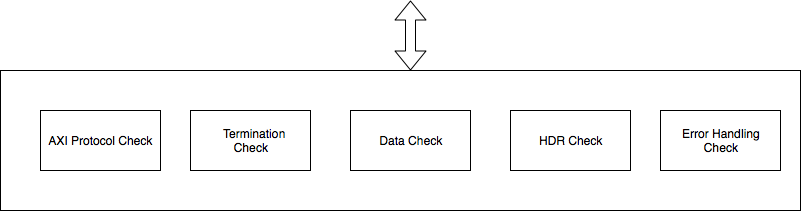
The below table is a description of the output coverage.

|  |  |  |
| --- | --- | --- |
| Signals | Number of bins | Description |
| Udp\_rx\_start | 2 - Each bin for low and high. | Indicates receive header from RX |
| Udp\_rx\_data\_length | 3 bins – 1 bin each for lowest value and highest value and one bin for anything in between | Indicates the data length of the packet received from RX. |
| Udp\_rx\_src\_port | Array bin for most of the destination ports that lie in between 1 and 8000. | Source port address |
| Udp\_rx\_dest\_port | Array bin for most of the destination ports that lie in between 1 and 8000. | Destination port address |
| Udp\_rx\_src\_ip\_addr | 3- One bin each for minimum and maximum value and one bin for the remaining values. | Source IP address of the UDP packet given by the RX |
| Udp\_rx\_data\_valid | 2- One bin each for low and high value. | Indicates valid data given the RX |
| Udp\_rx\_data\_last | 2- One bin each for low and high value | Indicates Last byte on the data bus by RX |

Cross coverage of the source and destination port is considered to check various combinations of the source and destination ports.



**Checker:**



The Checker Check for any Protocol Violations

* + Out of range Length Packets Terminations
  + Calculates the expected Header at TX/RX output and compare with it once the output is ready.
  + Collects the data from TX and Compares with RX Received data
  + Checks of the terminations of the error packets happen or not
  + Checks for early terminations of unfinished packets
  + AXI Stream Protocol checker

1. AXI Protocol:

* Checks for Protocol Violations in the control signals.

1. Termination:

* Check for out of range length termination success
* Check for early termination success

1. Data Check:

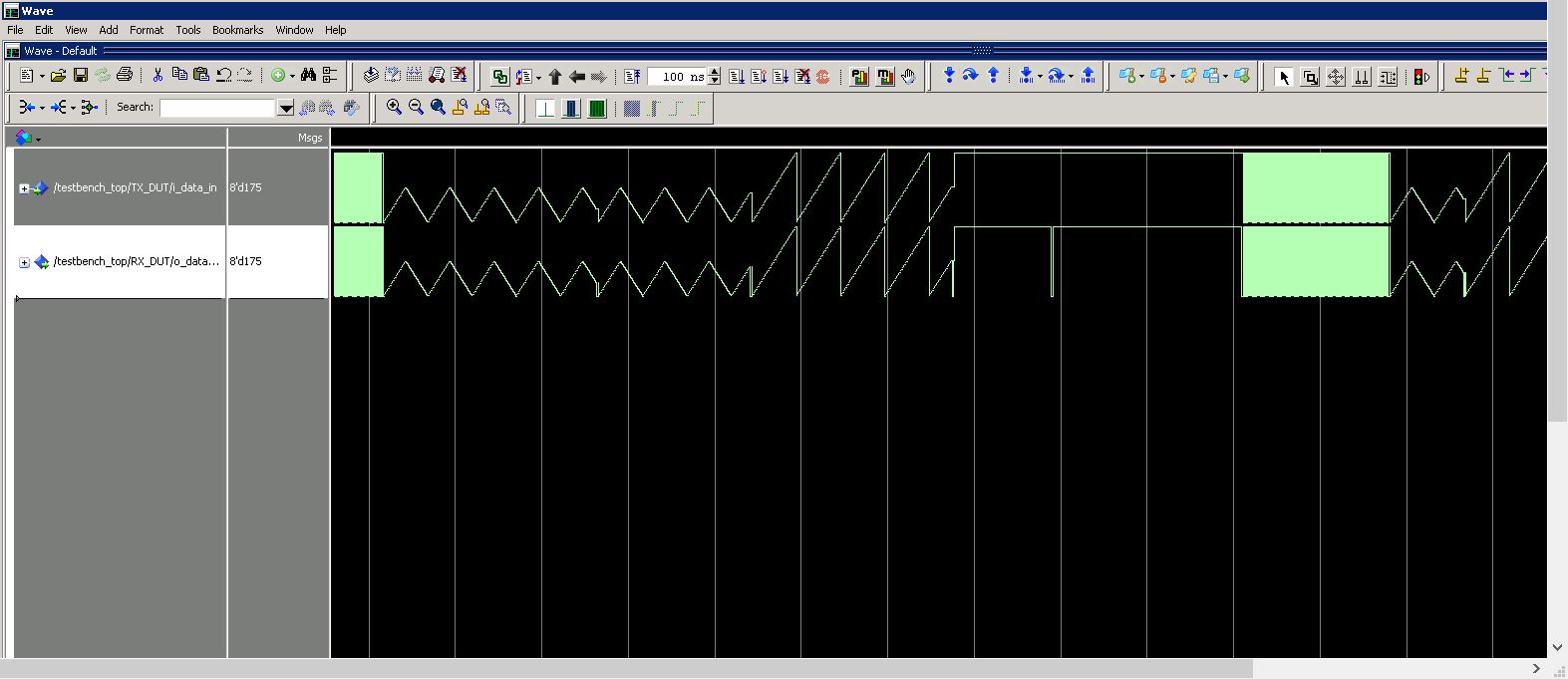
* For a successfully received the data is buffered and compared with the expected value (Obtained by buffered data while sending to TX)

1. HDR:

* Checks for received header to expected header
  + Length
  + IP address (Source in TX, becomes Destination in RX)
  + DST, SRC Port
  + Protocol

1. Error Handling:
2. Check of error packets terminations.

**Output waveform:**



The data that is sent by TX is received by the RX

**Introduction of bugs through RTL:**

* We corrupted the RTL design for length logic and noticed that the length isn’t matching in the checker.

**Challenges encountered during the project:**

* The DUT being in VHDL it was difficult to interface it (Records to Structure connection) – We had to re-write the TOP of TX and RX.
* Deciding coverpoints for various signals.

**Pre-existing ideas/codes borrowed:**

* The full design is taken form opensource, it is a full Ethernet Stack, Proven on FPGA. <https://opencores.org/project/udp_ip_stack>

**What would we do if we had more time?**

* We would first achieve all the tests that we failed to achieve now. Then, we’d extend the project to the IPV4 layer.

**Future Scope:**

* The project can be extended to an additional layer of the OSI model. That is, we could include the IPV4 TX and RX which receives data from MAC layers. Furthermore, the full Ethernet stack itself can be verified.

**Resources:**

**Tools:** Questasim, Wireshark

**Sources referred:**

* Slides from the class
* UVM Primer
* Verification Academy
* [www.TESTBENCH.in](http://www.TESTBENCH.in)
* YouTube videos
* Wikipedia
* <https://opencores.org/project/udp_ip_stack>