OS Buffer IP Verification Plan

1. Introduction

1.1. Design Description

The OS Buffer is the interface between the LTSSM and the Tx block. The LTSSM inserts the ordered set packets like TS1, TS2 and SKP in this buffer whenever needed. The Tx controller decides when to transmit the data from the OS Buffer across the PIPE.

1.2. Block Diagram and Architecture

Figure 1 shows the I/O Signals for the OS Buffer module.

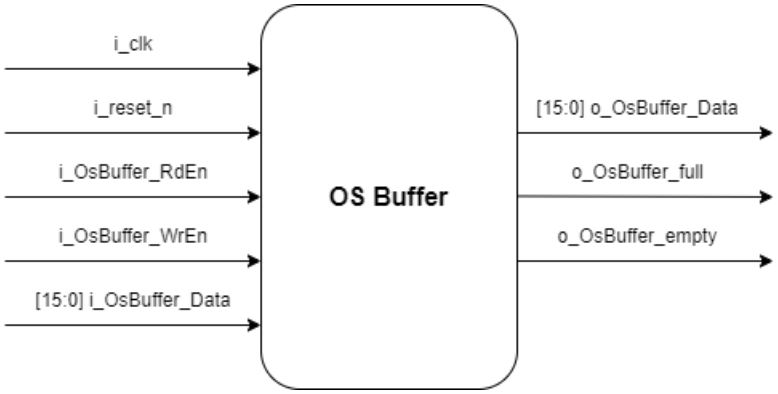

Figure 1: OS Buffer block diagram.

Figure 2 illustrates the location of the OS Buffer in the Tx architecture.

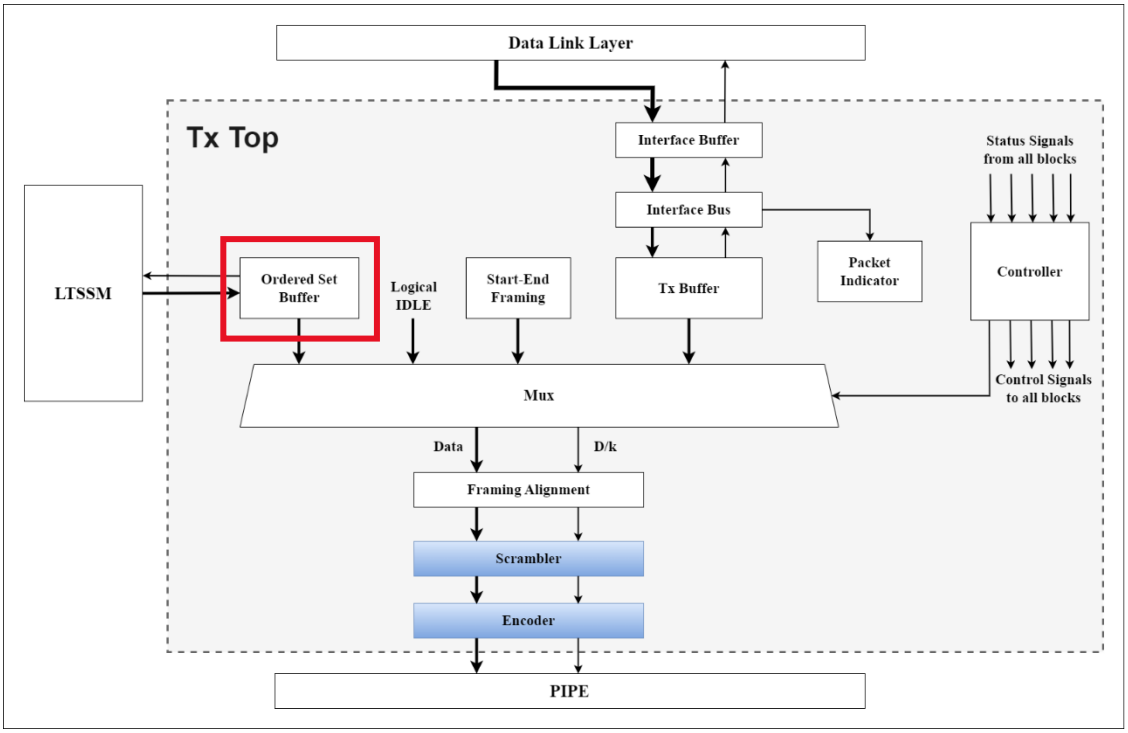

Figure 2: Location of the OS Buffer in the Tx architecture.

2. Verification Approach

2.1. Testbench Architecture and Hierarchy

The testbench architecture for the OS Buffer module is shown in figure 2 below. Two agents are needed as the OS Buffer communicates with both the LTSSM and the Tx.

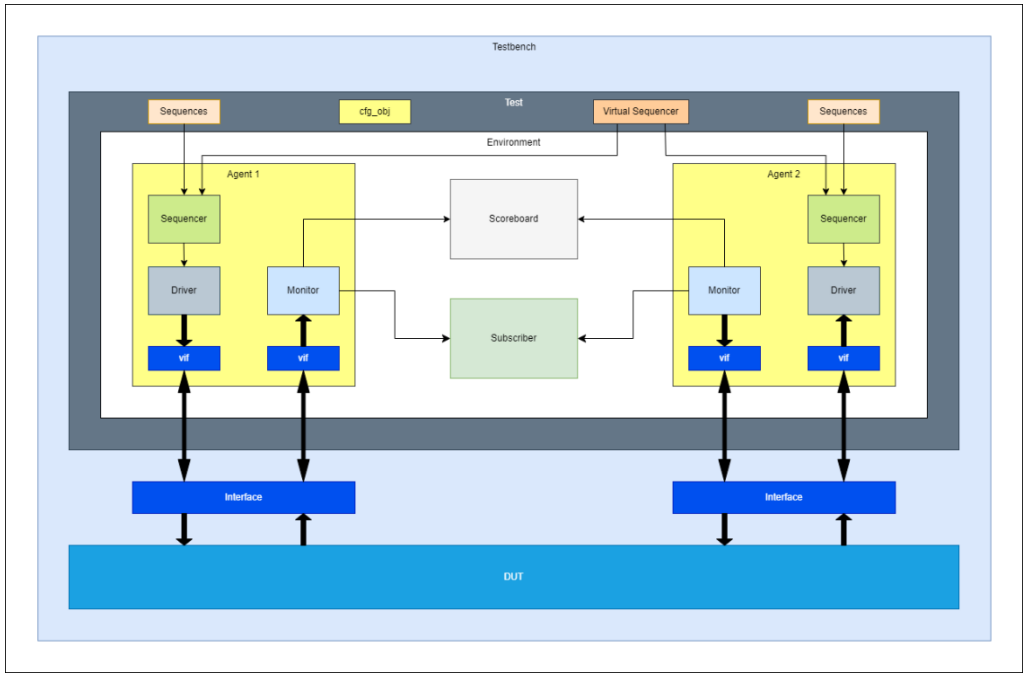

Figure 3: UVM TB Architecture.

2.2. Exit Criteria

To ensure that the module operation and functionalities are mostly covered, functional coverage of 90% and code coverage of 95% are required as exit criteria.

3. Test Items and Test Cases

3.1. Test Items

- i_OsBuffer_RdEn:
  - 0: The buffer read operation is disabled.
  - 1: The buffer read operation is enabled.
- i_OsBuffer_WrEn:
  - 0: The buffer write operation is disabled.
  - 1: The buffer write operation is enabled.
- i_OsBuffer_Data:
  - Invalid value
  - Valid value

3.2. Implemented Sequences

The table below shows the sequences that will be implemented and used in constructing the test cases.

Sequence Name	Description
GEN	General read after write or write after read sequence with variable read and write lengths.

3.3. Test Cases

Test Name	i_OsBuffer_RdEn	i_OsBuffer_WrEn	i_OsBuffer_Data
Test 01	Random	Random	Random

3.4. Implemented Tests

The table below shows the tests that will be applied to the design, the sequences used, and the description of each test.

Test Name	Description	Sequences
GEN	General read after write or write after read test with variable read and write lengths. The GEN sequence is randomized and repeated a number of times until a high coverage value is achieved.	GEN

4. Constraints

The table below shows the constraints applied to the random variables/signals.

Signal/Variable	Constraints Description
i_OsBuffer_Data	The allowed values for this signal are combinations of two symbols as shown: {COM, PAD}, {COM, Link}, {PAD, NFTS}, {Lane, NFTS}, {DR_ID, LIDLE}, {TS1_ID, TS1_ID}, {TS2_ID, TS2_ID}, {COM, SKP}, {SKP, SKP}, {COM, FTS}, {FTS, FTS}, {LIDLE, LIDLE}
W_cnt	This is a random variable that defines the write length. The allowed range for this variable is between 1 and 2049.
R_cnt	This is a random variable that defines the read length. The allowed range for this variable is between 1 and 2049.
OP	This is a random variable that defines the performed operation. Allowed operations are: 1. Multiple writes (partial). 2. Multiple reads (partial). 3. Single write. 4. Single read. 5. Multiple writes (full). 6. Multiple reads (full). 7. Multiple neither read nor write (no operation). 8. Multiple simultaneous read and write. 9. Random. The values of this signal should have a distribution such that the most probable sequences should have a higher probability of occurrence.

5. Checking Mechanism

The design will be compared against a golden reference model implemented in the scoreboard class using SystemVerilog. Any discrepancies will be recorded and reported.

The scoreboard implementation is shown in figure 4 below. Two functions are implemented. The reference model function takes the DUT input signals and generates the expected outputs that are compared against the actual outputs in the checking errors function.

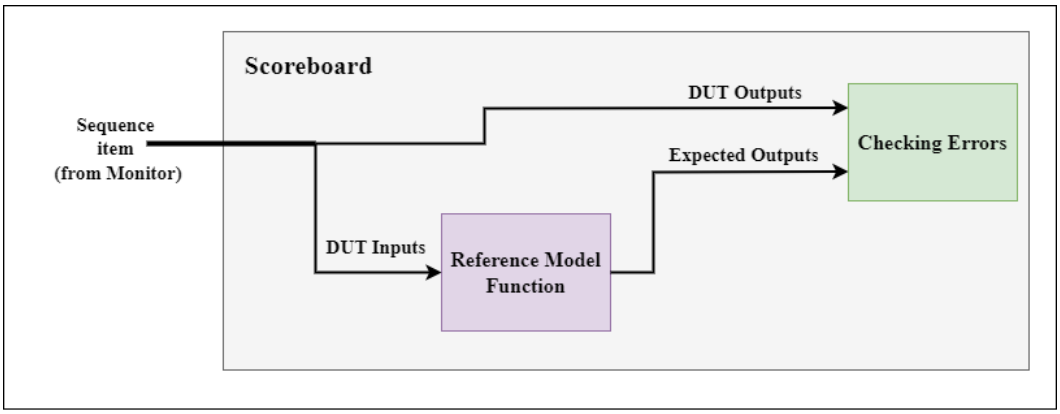

Figure 4: Scoreboard implementation.

6. Coverage Collection

Functional coverage collection will be performed using the subscriber class where one covergroup containing six coverpoints is created. The table below shows the coverage bins needed for coverage collection. Code coverage will be performed automatically by the tool.

Signal	Coverage Bins Description
i_OsBuffer_RdEn	single read - multiple reads
i_OsBuffer_WrEn	single write - multiple writes
i_OsBuffer_Data	The signal takes one of the following values at least once: {COM, PAD}, {COM, Link}, {PAD, NFTS}, {Lane, NFTS}, {DR_ID, LIDLE}, {TS1_ID, TS1_ID}, {TS2_ID, TS2_ID}, {COM, SKP}, {SKP, SKP}, {COM, FTS}, {FTS, FTS}, {LIDLE, LIDLE}
o_OsBuffer_Data	The signal takes one of the following values at least once: {COM, PAD}, {COM, Link}, {PAD, NFTS}, {Lane, NFTS}, {DR_ID, LIDLE}, {TS1_ID, TS1_ID}, {TS2_ID, TS2_ID}, {COM, SKP}, {SKP, SKP}, {COM, FTS}, {FTS, FTS}, {LIDLE, LIDLE}
o_OsBuffer_full	full
o_OsBuffer_empty	empty

Functional coverage of 100% and code coverage of 100% were achieved.

7. Traceability Matrix

Test Name	i_OsBuffer_RdEn		i_OsBuffer_WrEn		i_OsBuffer_Data	
	Enabled	Disabled	Enabled	Disabled	Valid	Invalid
Test 01	☑			☑	☑	
Test 02		☑	☑		☑	
Test 03	☑		☑		☑	
Test 04		☑		☑	☑	
Test 05	☑			☑		☑
Test 06		☑	☑			☑
Test 07	☑		☑			☑
Test 08		☑		☑		☑

8. Evaluation and Detected Bugs

The table below shows the detected bugs.

Bug Name	Signal	Severity	Description
Bug 01	o_OsBuffer_full	LOW	The o_OsBuffer_full signal is asserted one cycle earlier than it should. At the time it is asserted, the buffer is not full yet and has a size of 2047 not 2048.

This module is NOT good to go.