Verification of AXI DMA with UVM

Presenter: Noman Rafiq
Tech Lead: Hassan Ashraf
Date: 04 Feb, 2025

TCP Overview

TCP Description

This project focuses on verifying the AXI DMA IP in Vivado, specifically in **Direct Register Mode** with the Scatter-Gather Engine disabled. The objective is to ensure the DMA functions correctly, adheres to the AXI and AXI-Stream protocols, and handles data transfers reliably.

Features to Verify:

- AXI-Lite Interface - Read/Write Operations
- Interrupt Generation & Handling
- Error Scenarios
- Reset Feature
- MM2S/S2MM Independent Operation
- Automatic Data Re-Alignment
- Run/Stop Control

continued..

Deliverables:

- UVM-based testbench for IP verification.
- Register Abstraction Layer (RAL) model.
- Comprehensive Test Plan
- Verification Architecture Document
- Coverage analysis.

TCP Steps

- Project Initialization
- IP Generation
- Verification Planning
- Testbench Development
- RAL Model
- Scoreboard
- Test Case Development
- Coverage Analysis
- Documentation

TCP Timeline

Total Duration of the Project (1 Month)

- **Start Date:** Dec 12, 2024
- **End Date:** Jan 31, 2024

Week 1:

- Familiarization with Vivado IP and AXI/AXI-Stream protocols.

Week 2:

- Development of Verification Plan & Testbench components (drivers, monitors).

Week 3:

- Development of RAL Model.
- Debugging and fine-tuning testbench.

TCP Timeline

Week 4:

- Scoreboard and Test Case Development

Week 5:

- Test Case Development & Coverage Analysis

Week 6:

- Documentation & Delivery

Week 7:

- Revising and Further Optimizations

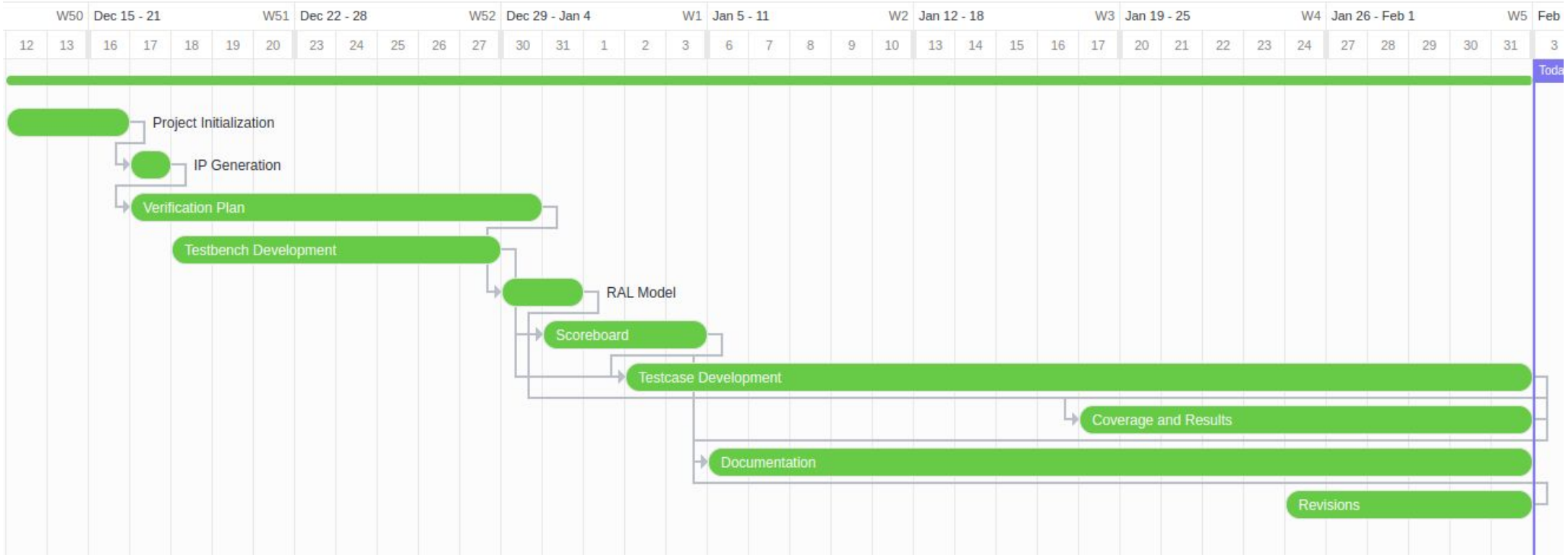
TCP Timeline

Actual vs Expected Timeline

Project Phases	Expected Completion Date	Actual Completion Date	Slack
TCP	**January 12, 2025**	**January 31, 2025**	-19
Project Initialization	December 16, 2024	December 16, 2024	0
IP Generation	December 17, 2024	December 17, 2024	0
Verification Plan	December 31, 2024	December 30, 2024	1
Testbench Development	December 21, 2024	December 28, 2024	-7
RAL Model	December 31, 2024	December 31, 2024	0
Scoreboard	January 2, 2025	January 3, 2025	-1
Testcase Development	January 20, 2025	January 18, 2025	2
Coverage and Results	January 20, 2025	January 24, 2025	-4
Documentation & Delivery	January 20, 2025	January 24, 2025	-4
Revisions - Code Review II	January 29, 2025	January 31, 2025	-2

TCP Timeline

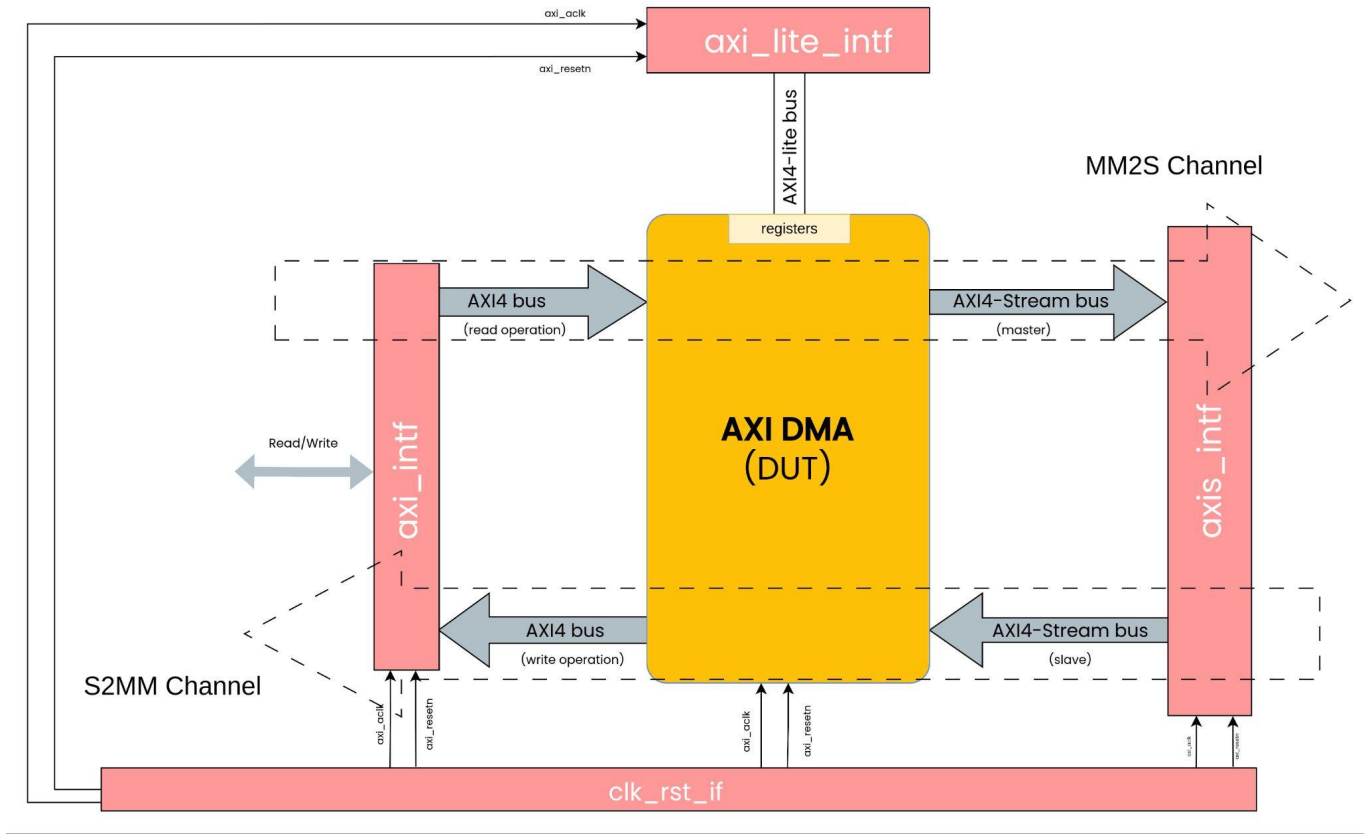
Gantt Chart



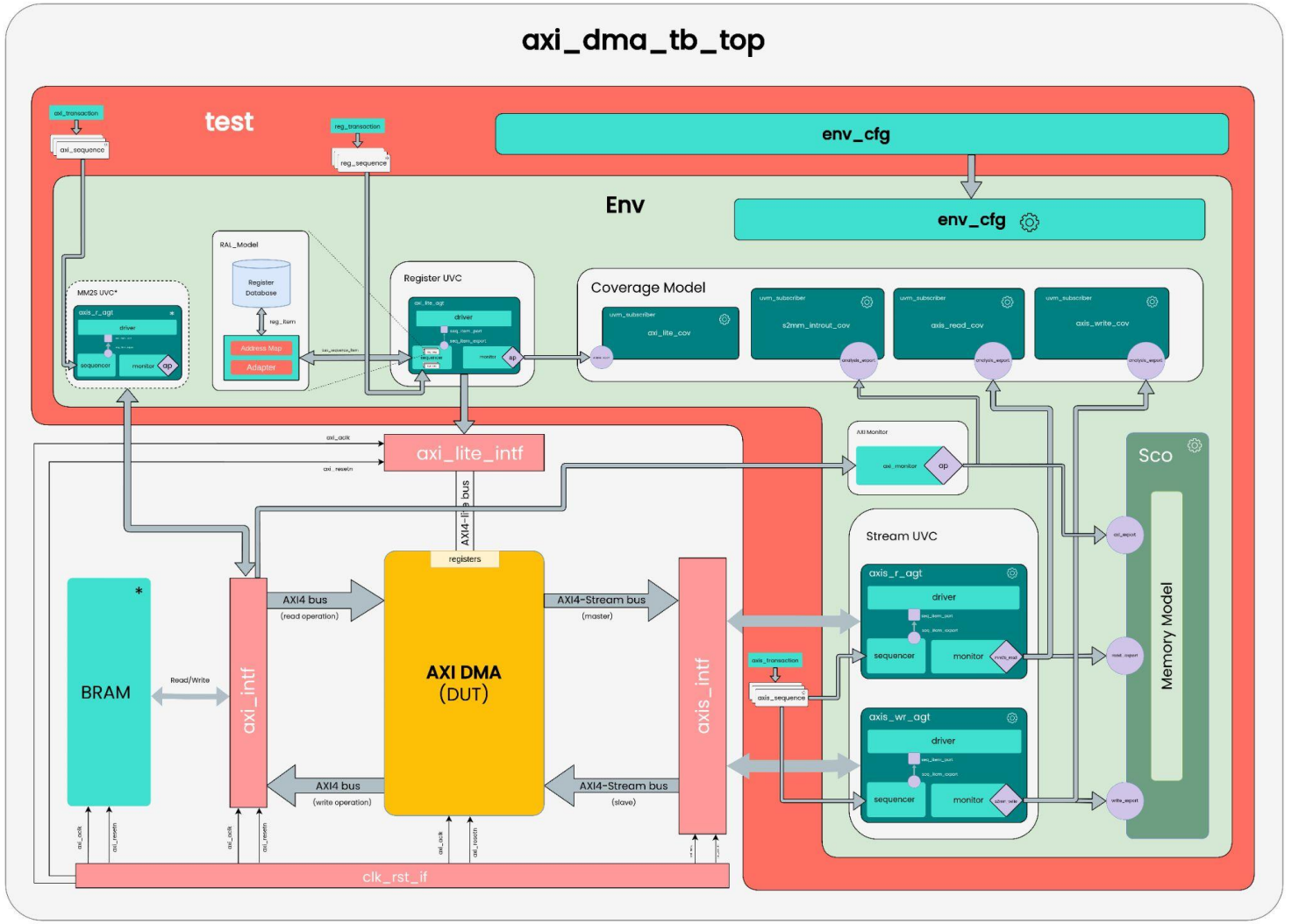
Technical Overview

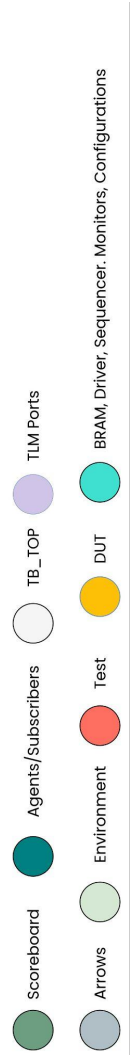
Design Under Test (DUT)

Design Under Test (DUT)



Testbench Structure





Test Workflow

MM2S Read Operation:

- Configure the MM2S Control Registers.
- Start the AXI Read Sequence on the Left Side and hand over data to DMA.
- Start the AXI Stream Read Sequence on the Right Side and get the data from DMA.
- Compare the Received Data with the Expected Data in Scoreboard.
- Check if the Interrupt was Generated after transfer completes.

Test Termination:

- Simulation completes when all transactions are processed.

Verification approach

1. **Architecture**:
   - UVM-based modular environment.
   - Parameterized Components

2. **Methodology**:
   - Constrained Random Testing for robustness.
   - Directed Testing to target specific features like interrupts and error conditions.
   - Random Stimulus Generation for Coverage

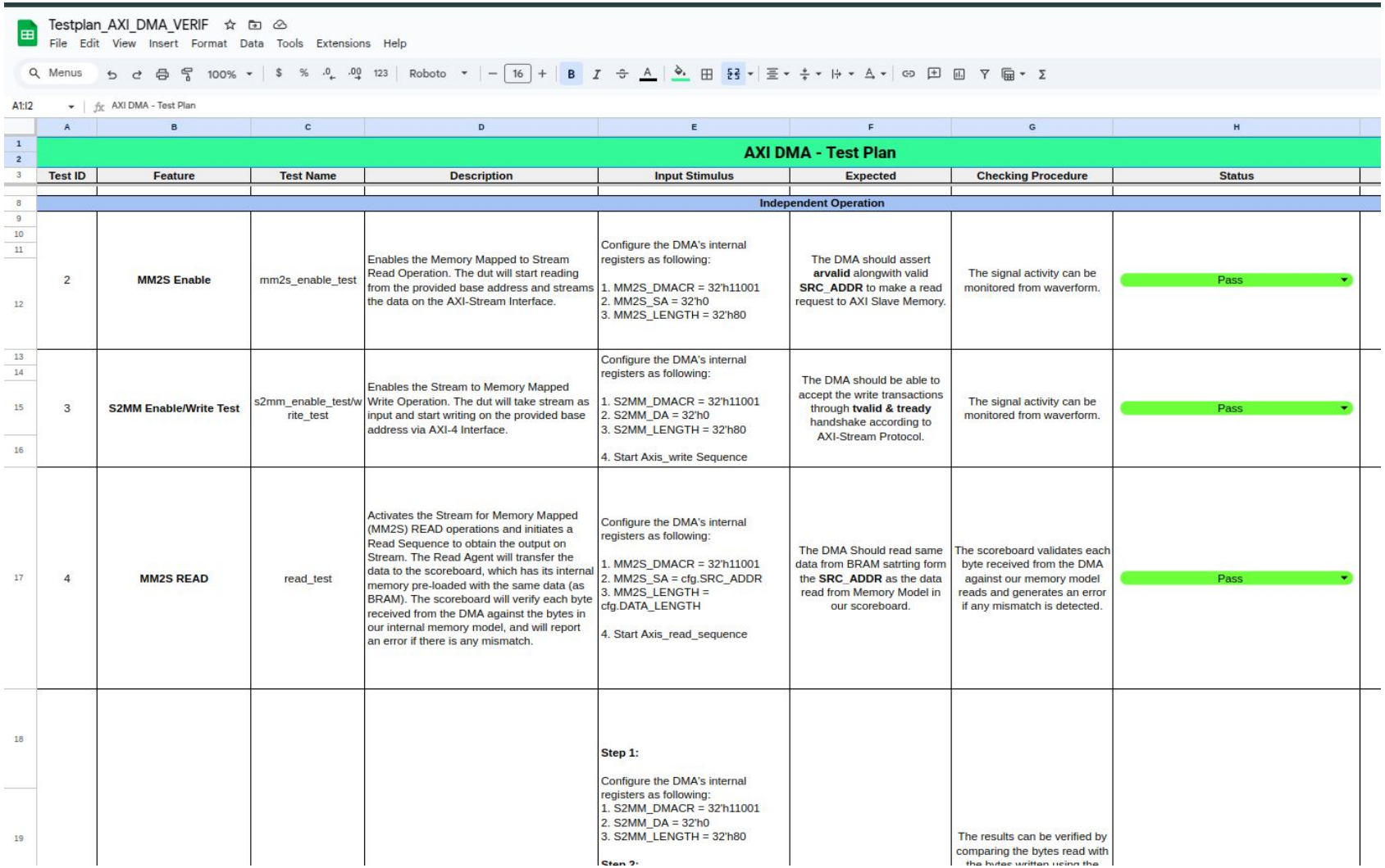
3. **Validation**:
   - Scoreboard-Based Comparison to Validate data integrity for MM2S and S2MM transfers.
   - Assertion-based Verification and Protocol Compliance
   - Coverage Metrics

Test Plan

Key Features to Verify:

- AXI-Lite Interface - Read/Write Operations
- AXI-Stream Interface - Data Transfer
- Handling Packet Boundaries
- Interrupt Generation & Handling
- Error Scenarios
- Reset Feature
- MM2S/S2MM Independent Operation
- 4 KB Boundary Protection
- Automatic Data Re-Alignment
- Halted and Idle State Management
- Run/Stop Control

Test Plan



Testplan_AXI_DMA_VERIF

File Edit View Insert Format Data Tools Extensions Help

A1:I2 | AXI DMA - Test Plan

AXI DMA - Test Plan

Test ID	Feature	Test Name	Description	Input Stimulus	Expected	Checking Procedure	Status
				Independent Operation			
2	**MM2S Enable**	mm2s_enable_test	Enables the Memory Mapped to Stream Read Operation. The dut will start reading from the provided base address and streams the data on the AXI-Stream Interface.	Configure the DMA's internal registers as following: 1. MM2S_DMACR = 32'h11001 2. MM2S_SA = 32'h0 3. MM2S_LENGTH = 32'h80	The DMA should assert **arvalid** alongwith valid **SRC_ADDR** to make a read request to AXI Slave Memory.	The signal activity can be monitored from waverform.	Pass
3	**S2MM Enable/Write Test**	s2mm_enable_test/write_test	Enables the Stream to Memory Mapped Write Operation. The dut will take stream as input and start writing on the provided base address via AXI-4 Interface.	Configure the DMA's internal registers as following: 1. S2MM_DMACR = 32'h11001 2. S2MM_DA = 32'h0 3. S2MM_LENGTH = 32'h80 4. Start Axis_write Sequence	The DMA should be able to accept the write transactions through **tvalid & tready** handshake according to AXI-Stream Protocol.	The signal activity can be monitored from waverform.	Pass
4	**MM2S READ**	read_test	Activates the Stream for Memory Mapped (MM2S) READ operations and initiates a Read Sequence to obtain the output on Stream. The Read Agent will transfer the data to the scoreboard, which has its internal memory pre-loaded with the same data (as BRAM). The scoreboard will verify each byte received from the DMA against the bytes in our internal memory model, and will report an error if there is any mismatch.	Configure the DMA's internal registers as following: 1. MM2S_DMACR = 32'h11001 2. MM2S_SA = cfg.SRC_ADDR 3. MM2S_LENGTH = cfg.DATA_LENGTH 4. Start Axis_read_sequence	The DMA Should read same data from BRAM satrting form the **SRC_ADDR** as the data read from Memory Model in our scoreboard.	The scoreboard validates each byte received from the DMA against our memory model reads and generates an error if any mismatch is detected.	Pass
				Step 1: Configure the DMA's internal registers as following: 1. S2MM_DMACR = 32'h11001 2. S2MM_DA = 32'h0 3. S2MM_LENGTH = 32'h80 **Step 2:**		The results can be verified by comparing the bytes read with the bytes written using the

Statistics

- Total 22 Test Cases
- 100% Functional Coverage for Direct Register Mode

XILINX

Groups Coverage Summary

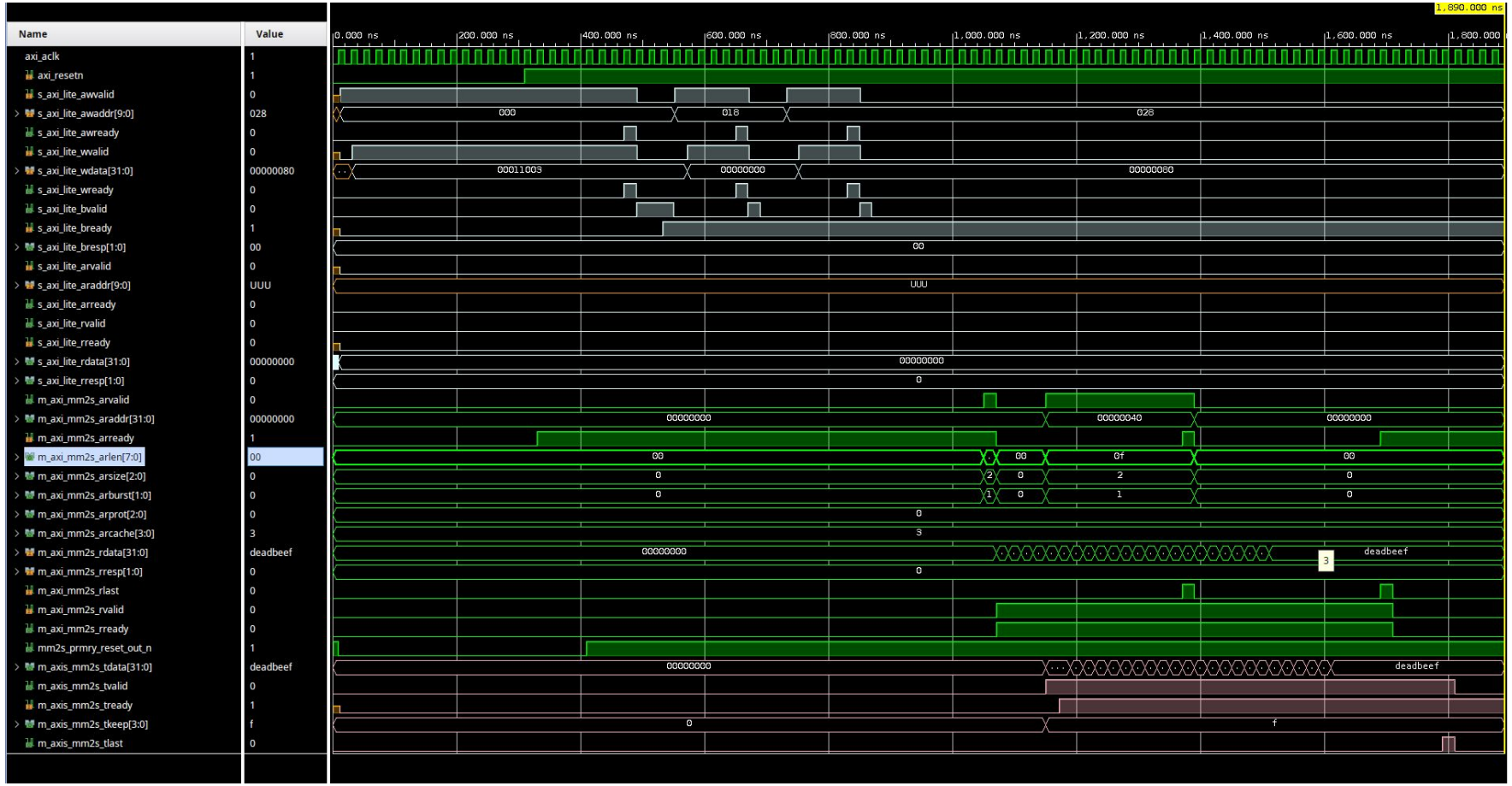
Score	Inst Score
100	100

Total groups in report: 6

Name	Score	Num Instances	Avg Instances Score	Weight	Goal	Merge Instances	Get Inst Coverage	Per Instance	Auto Bin Max	Comment
$unit_params_pkg_sv_891442008::axi_lite_coverage::cg_read_channel	100	1	100	1	100	0	0	0	64	
$unit_params_pkg_sv_891442008::axi_lite_coverage::cg_write_channel	100	1	100	1	100	0	0	0	64	
$unit_params_pkg_sv_891442008::axis_read_coverage::cg_axis_read	100	1	100	1	100	0	0	0	64	
$unit_params_pkg_sv_891442008::axis_read_coverage::cg_mm2s_introut	100	1	100	1	100	0	0	0	64	
$unit_params_pkg_sv_891442008::axis_write_coverage::cg_axis_write	100	1	100	1	100	0	0	0	64	
$unit_params_pkg_sv_891442008::s2mm_introut_coverage::cg_s2mm_introut	100	1	100	1	100	0	0	0	64	

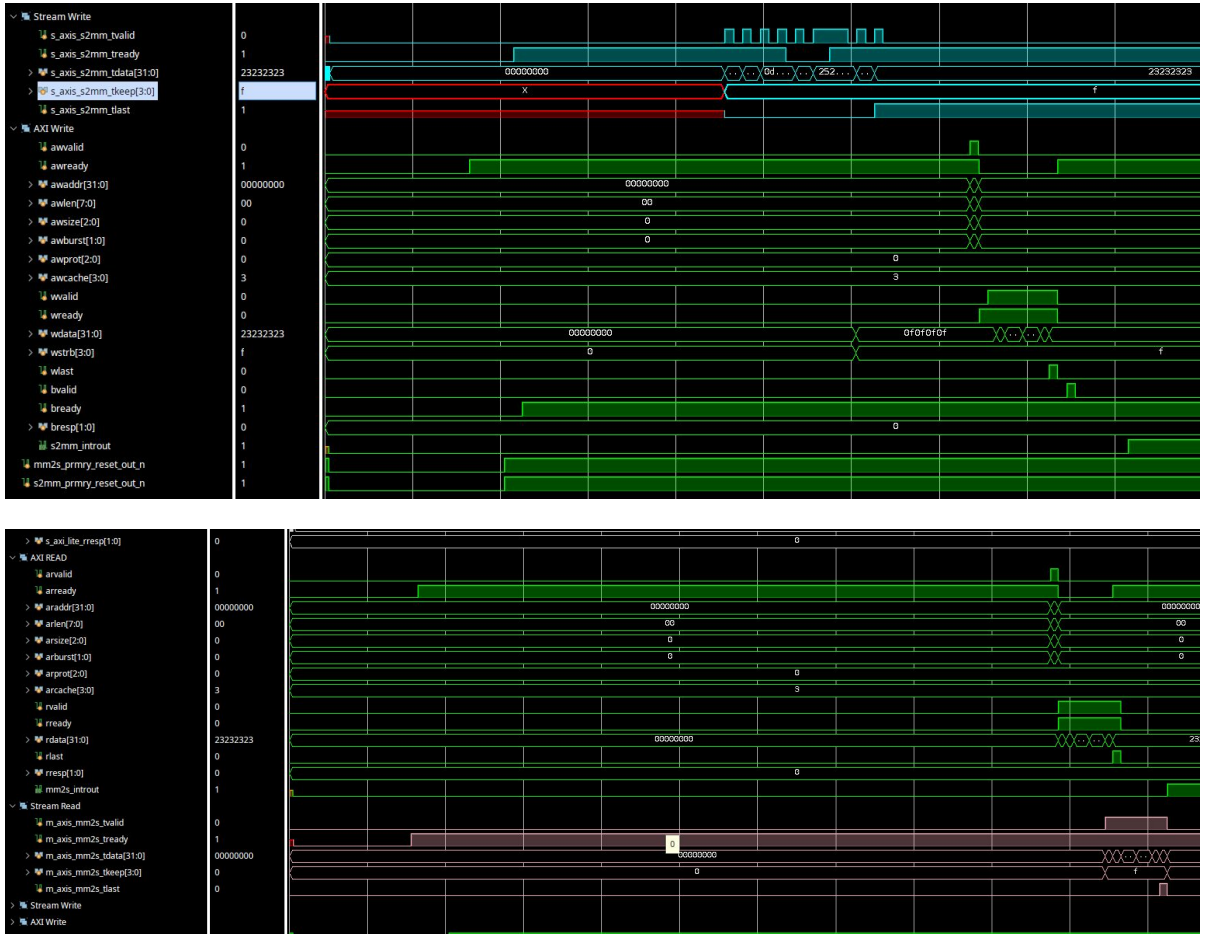
Demo

It's simulation o'clock — let’s see the testbench in action!



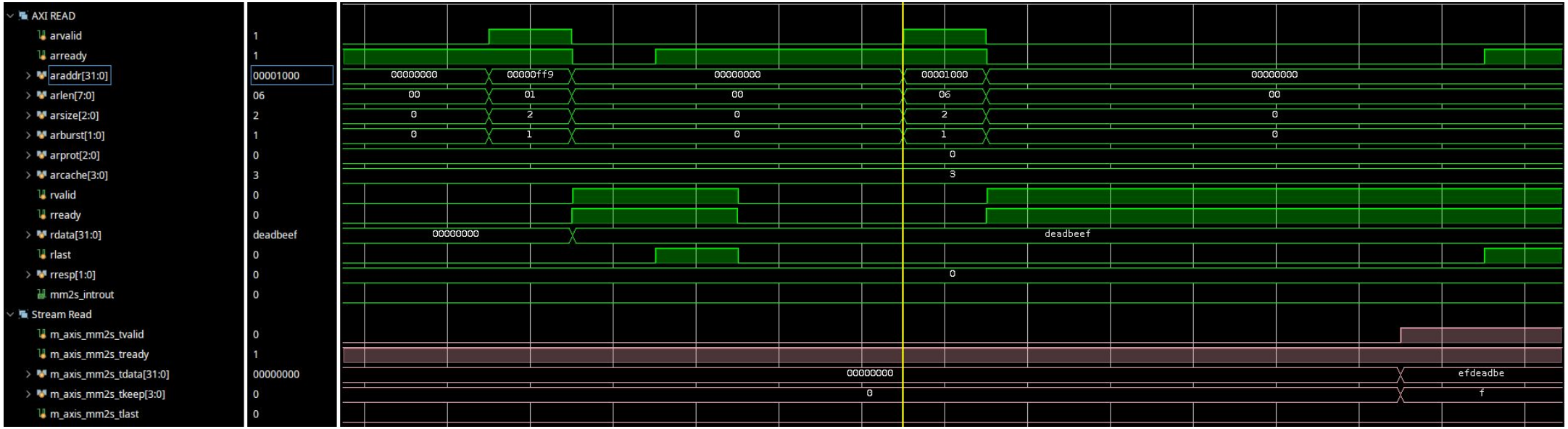
Special Cases

Read After Write Test (RAW)



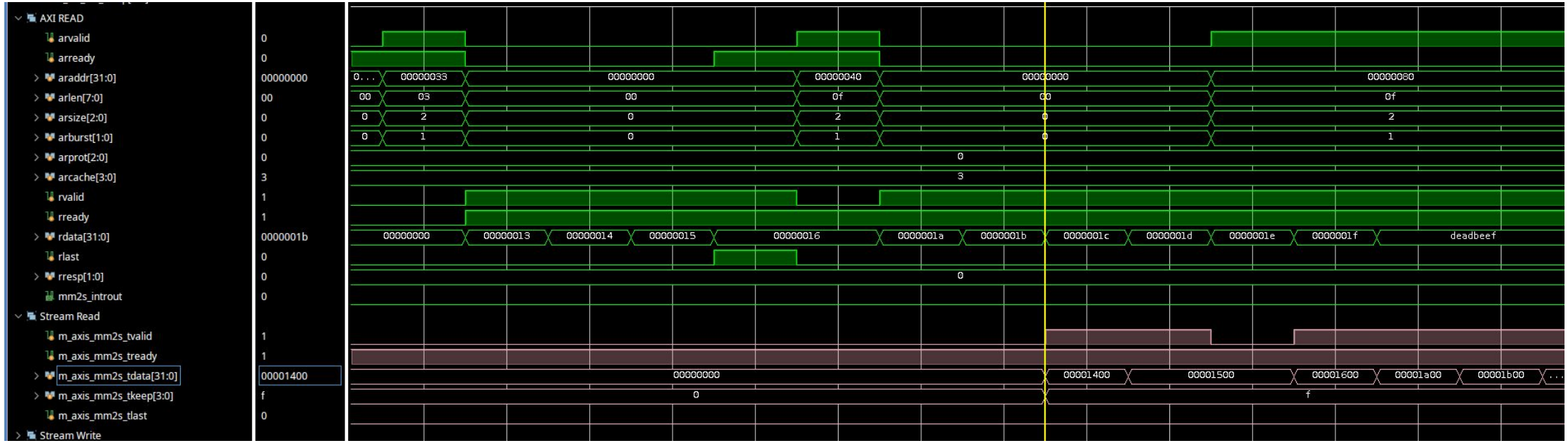
Special Cases

Boundary Test



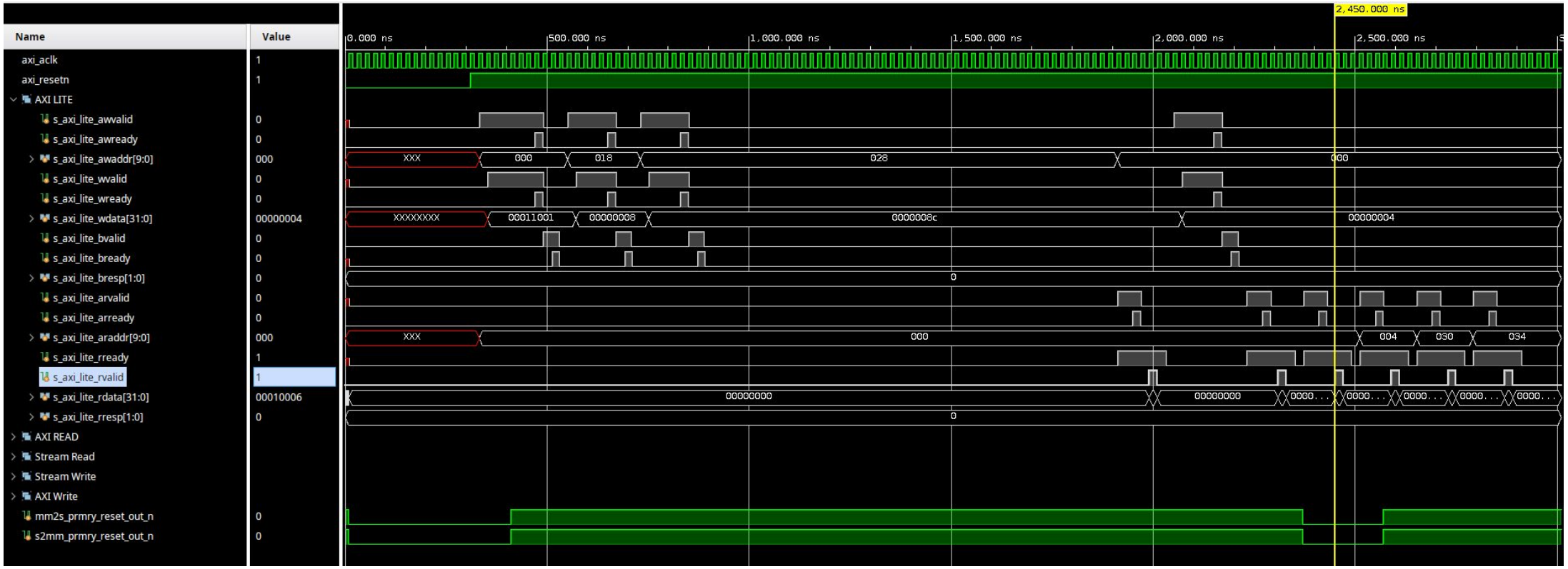
Special Cases

Data Re-Alignment Test



Special Cases

Soft Reset Test



Challenges

- Memory Mapped Read/Write Operation
- AXI Slave Memory Integration
- Read after Write Test (RAW Test)
- Scoreboard Interrupt Checkers
- Random Failures

Achievements

- Fully Developed UVM Testbench
- Functional RAL Model
- Comprehensive Verification Architecture Document
- Github Repository (AXI DMA Verification)
- Test Plan (Verification Testplan)
- Coverage Analysis (Merged Report)
- Makefile Automation

Thankyou.
Questions?