**TDP RAM UVC**

**“Verification of True Dual Port RAM using SV and UVM”**

Functional Verification Plan

Revision 0.1

EITRA

Crafting Preferred Engineers

Contents

[1 Document Details 4](#_Toc419795954)

[1.1 Revision History 4](#_Toc419795955)

[1.2 Definition, Acronyms, and Abbreviations 5](#_Toc419795956)

[1.3 References 6](#_Toc419795957)

[1.4 Open Items 6](#_Toc419795958)

[2 Overview 7](#_Toc419795959)

[2.1 Scope 7](#_Toc419795960)

[2.2 Verification Flow 7](#_Toc419795961)

[2.3 Verification Objectives 7](#_Toc419795962)

[2.4 Verification Methodology 7](#_Toc419795963)

[2.5 Verification Strategy 7](#_Toc419795964)

[3 Verification Environment Architecture 8](#_Toc419795965)

[3.1 Overview of TDP RAM UVC Verification Architecture 8](#_Toc419795966)

[4 DUV Considerations Relevant To Verification 9](#_Toc419795967)

[5 Verification component 10](#_Toc419795968)

[6 Verification setup 11](#_Toc419795969)

[6.1 Tool Versions 11](#_Toc419795970)

[6.2 VIP Versions 11](#_Toc419795971)

[6.3 Directory Structure 11](#_Toc419795972)

[6.4 Project Setup 11](#_Toc419795973)

[6.5 Simulation Script Usage 11](#_Toc419795974)

[6.5.1 Running Single Testcase 11](#_Toc419795975)

[6.5.1 Running Regressions 12](#_Toc419795976)

[6.5.2 Compilation define and Simulation Argument Switch 12](#_Toc419795977)

[7 Coverage & Regression 12](#_Toc419795978)

[7.1 Running Regression 12](#_Toc419795979)

[7.2 Code coverage 12](#_Toc419795980)

[7.3 Assertion coverage 12](#_Toc419795981)

[7.4 Functional coverage 12](#_Toc419795982)

[7.4.1 Directed-Random Testing 12](#_Toc419795983)

[7.4.1.1 Objectives of Directed Random Testing 12](#_Toc419795984)

[7.4.1.2 Why Functional Coverage? 12](#_Toc419795985)

[7.4.2 How do we decided functional coverage bins / What to randomize ? 12](#_Toc419795986)

[7.4.2.1 When / Where /How to sample data to be covered? 12](#_Toc419795987)

[7.4.3 Regression with Functional Coverage 13](#_Toc419795988)

[7.4.3.1 To achieve functional coverage 13](#_Toc419795989)

[7.4.4 Directed-Random Testing : Self Checking 13](#_Toc419795990)

[7.4.4.1 How to check correctness of test with directed-random testing 13](#_Toc419795991)

[8 Assumptions and Limitation 13](#_Toc419795992)

[9 Final Release checklist 13](#_Toc419795993)

[10 Appendix A: References 13](#_Toc419795994)

[10.1 Revision History 13](#_Toc419795995)

[10.2 Items Out Of Scope of Verification 13](#_Toc419795996)

**Figures**

**No table of figures entries found.**

**Tables**

[Table 1‑1 Revision History 4](#_Toc419796106)

[Table 1-2 Definition, Acronyms and Abbreviations 5](#_Toc419796107)

[Table 1-3 References 6](#_Toc419796108)

[Table 10‑1 References 13](#_Toc419796109)

[Table 10‑2 Revision History 13](#_Toc419796110)

# Document Details

## Revision History

Table 1‑1 Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Revision | Author | Description |
| 09/09/2023 | Rev 0.1 | Mahammadshahid | Initial Draft |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

## Definition, Acronyms, and Abbreviations

Table 1-2 Definition, Acronyms and Abbreviations

|  |  |
| --- | --- |
| Term | Description |
| DS | **D**esign **S**pecification document |
| DUV | **D**esign **U**nder **V**erification |
| DUT | **D**esign Under **T**est |
| UVM | **U**niversal **V**erification **M**ethodology |
| VE | **V**erification **E**nvironment |
| TB | **T**est **B**ench |
| Expected | The word “Expected” is used to indicate that the transaction value under consideration is prepared by VE or is reference value. The transaction received from RTL is then compared with this expected transaction for data integrity checks. |
| Actual | The word “Actual” is used to indicate that transaction value under consideration is driven by DUV. This driven transaction is sampled by VE and then compared with the expected transaction for data integrity checks. |

## References

Table 1-3 References

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Document | Version Date | Remarks |
| 1 |  |  |  |
| 2 |  |  |  |
| 3 |  |  |  |
| 4 |  |  |  |
| 5 |  |  |  |
| 6 |  |  |  |
| 7 |  |  |  |
| 10 |  |  |  |

## Open Items

# Overview

## Scope

This document specifies the plan for verifying the front-end design of True Dual Port RAM for functional accuracy using simulations. It describes the flow, objectives, methodology, and strategy of execution for functional verification.

## Verification Flow

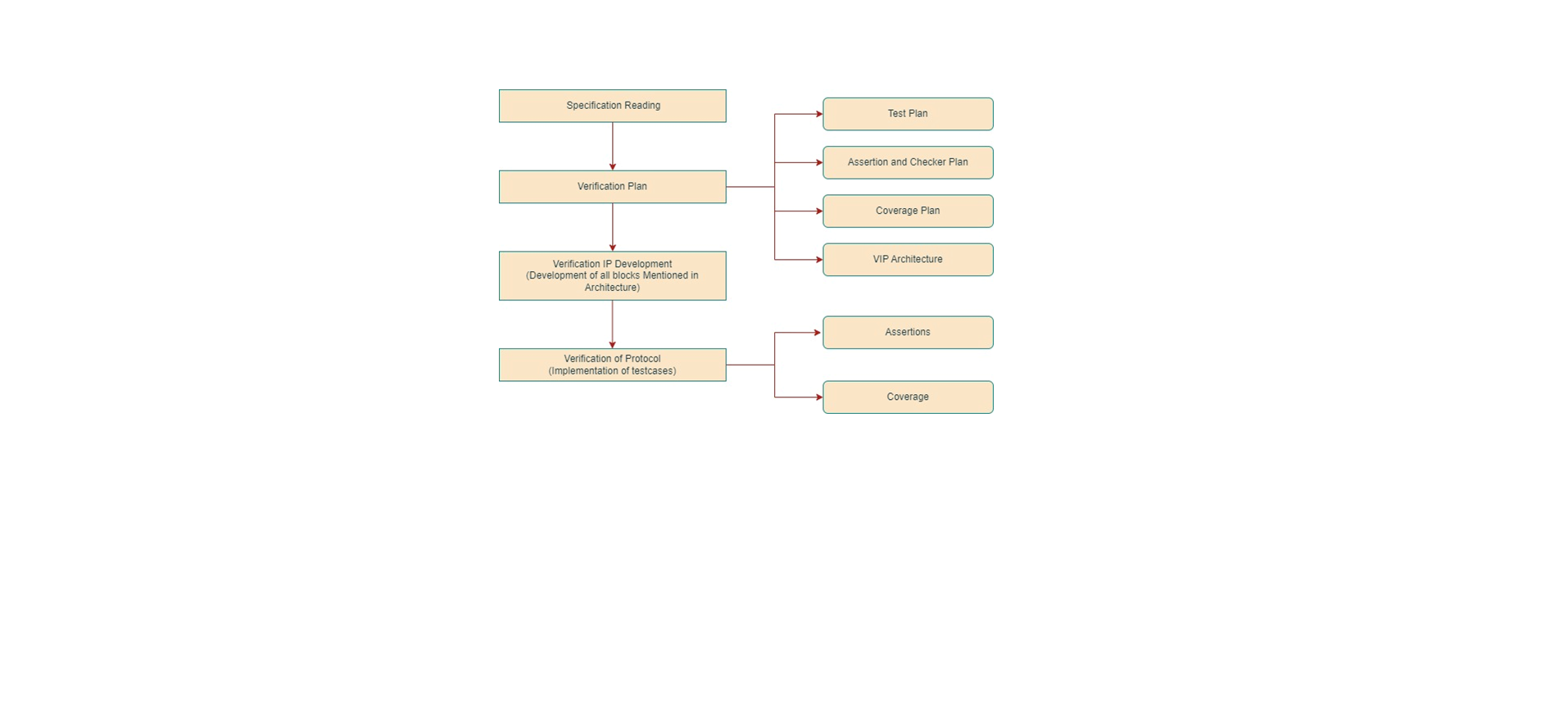


Fig. 2.1 Verification Flow of Project

Below is the general flow for the verification of a communication protocol using the Universal Verification Methodology (UVM), along with a diagram illustrating the key components and interactions:

* **Protocol Specification Reading:** In the first phase, the verification team needs to thoroughly understand the specification. This includes studying its features, functionalities, and any associated advantages or limitations. The goal is to extract typical scenarios and corner cases that will be essential for comprehensive protocol verification.
* **Verification Plan:** During this phase, a detailed verification plan is created. It should encompass the test plan, assertion and checkers plan, and coverage plan. The test plan outlines the test cases and scenarios that will be used to verify the protocol's functionality. The assertion and checkers plan includes the System Verilog Assertions (SVAs) and checker modules that will be used to monitor and validate the protocol behavior during simulation. The coverage plan defines the coverage goals and metrics that will be used to measure the progress and completeness of the verification process.
* **Verification Plan:** During this phase, a detailed verification plan is created. It should encompass the test plan, assertion and checkers plan, and coverage plan. The test plan outlines the test cases and scenarios that will be used to verify the protocol's functionality. The assertion and checkers plan includes the System Verilog Assertions (SVAs) and checker modules that will be used to monitor and validate the protocol behavior during simulation. The coverage plan defines the coverage goals and metrics that will be used to measure the progress and completeness of the verification process.
* **Verification Environment Development:** This is the coding phase of the verification process where the verification environment is developed using the UVM methodology. The environment includes the implementation of the protocol's functionality, the development of transaction-level models (TLMs), creation of sequences, and the integration of any required monitor, scoreboard, and functional coverage components.
* **DUT Verification:** After the development phase, the actual protocol verification begins. Stimulus is generated using sequences to execute various test scenarios that cover the features and functionalities outlined in the protocol specification. The verification environment applies these stimuli to the design under test (DUT) and monitors its behavior. The UVM components, such as the monitor and scoreboard, analyze the responses from the DUT and check for protocol compliance. System Verilog Assertions (SVAs) are used to perform formal checks on the protocol's behavior. The verification process aims to identify and report any protocol violations or corner-case issues.
* **Coverage and Closure:** Throughout the verification process, functional coverage data is collected to measure the completeness of the verification. The coverage plan defined in the verification plan guides this process. The goal is to achieve sufficient coverage to ensure that critical scenarios and functionalities of the protocol have been thoroughly tested. Once the coverage goals are met, and any reported issues are resolved, the verification process is considered complete, and the protocol is deemed ready for integration into the larger system.

## Verification Objectives

The objective of this project is to learn and implement the UVC Testbench for verification of True Dual Port RAM. The goal is to create a modular and reusable test bench with directed and/or randomized stimulus to cover various data transactions, different transfer widths.

## Verification Methodology

Universal Verification Methodology (UVM) is used to verify DUT Design of True Dual Port RAM.

## Verification Strategy

**1. Testbench Components Development:**  
In this phase, I will build all testbench components including top, environment, agent, driver, monitor, sequencer, scoreboard, etc.

**2. Verification using Sanity Test:**  
In this phase, I will build components related to test and sequences to run the basic sanity test and will test basic functionality of DUT.

**3. Development and Verification of Advance Testcases:**  
In this phase, I will implement advance testcases and will check functionality of DUT with each and every corner cases.

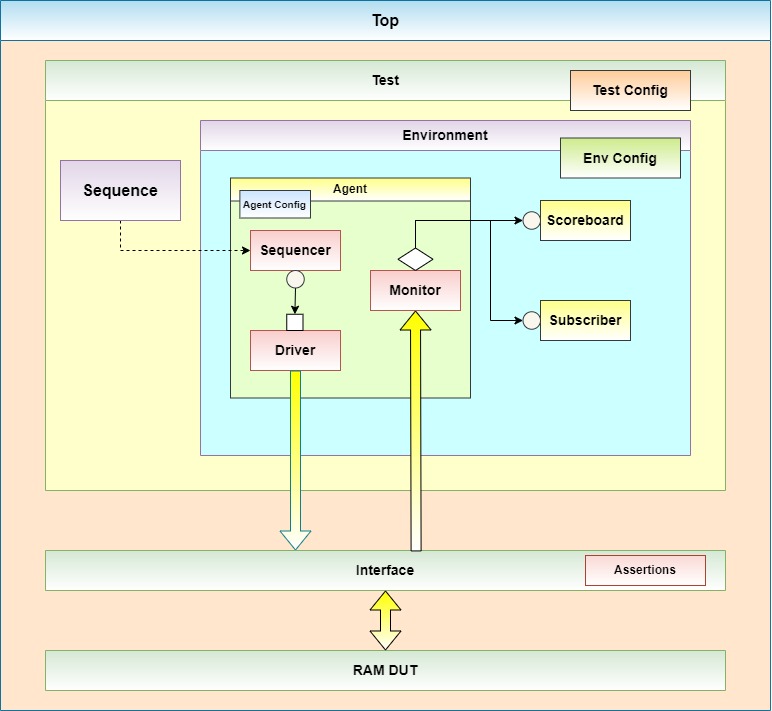
**4. Regression Setup:**  
In this phase, I will create a script to execute each and every testcase directly. The script will be based on python which will also provide support for regression.

**5. Coverage Development:**  
In this phase, I will implement the coverage plan and will implement the coverage to check functional coverage of our UVC.

**6. Assertion Development:**  
In this phase, I will implement all the assertions related to verification of RAM based transaction.

# Verification Environment Architecture

## Overview of TDP RAM UVC Architecture

**Fig. 3.1 Architecture of TDP RAM UVC**

Cover following points per block/IP:

* Verification architecture diagram and description
* Verification component list and description
* Test Plan link
* Test Flow
* Features (To Be) Verified
* Functional Coverage Plan and Details
* Test case execution

# DUV Considerations Relevant To Verification

Brief description of design

List down block/feature list to be verified

# Verification component

Multiple tables containing information per class:

* Instance Name
* Verification Component Name & Description
* Method/Member Name & Description

Also add brief information about each component/object of environment.

Coverage, Assertions, Checkers, Scoreboard strategy/plan to be described in detail.

Wherever applicable, add brief/detailed description, diagram, flow chart, tables, etc.

# Verification setup

This section shall provide details of about the verification environment requirements. It describes the requirements on tools and their versions, the directory structure used, steps to set up the project independently, and how to execute the tests.

## Tool Versions

Table to list details of all tools required/used

* Tool name
* Source
* Tool Revision Used
* Comment

## VIP Versions

Table to list details of all VIPs required/used

* VIP name
* Source
* VIP Revision Used
* Comment

## Directory Structure

For directory structure along with file details

Also mention details about files/directories created after compiling/running tests

## Project Setup

The setup step for the above tool configuration will be as follows.

The following are the steps required to setup the verification environment.

* Set Environment Variables
* Source various files to setup tools

## Simulation Script Usage

Simulation will be run in simulation run directory already setup. Details of the simulation run script can be obtained with the help option of the run script. The simulation script is available at $PRJ\_ROOT/<path\_to\_simulation\_script>.

### Running Single Testcase

Provide use model and details of each switch used while sourcing simulation script.

### Running Regressions

The regression command for a specific test list file.

Provide use model and details of each switch used while sourcing simulation script.

### Compilation define and Simulation Argument Switch

Provide details of each compilation define and simulation argument

# Coverage & Regression

This section identifies different strategies for running regression, regression analysis and coverage collection and closure.

## Running Regression

Following strategy will be used for regression ……………

## Code coverage

## Assertion coverage

## Functional coverage

### Directed-Random Testing

#### Objectives of Directed Random Testing

#### Why Functional Coverage?

### How do we decided functional coverage bins / What to randomize ?

#### When / Where /How to sample data to be covered?

### Regression with Functional Coverage

#### To achieve functional coverage

### Directed-Random Testing : Self Checking

#### How to check correctness of test with directed-random testing

# Assumptions and Limitation

# Final Release checklist

# Appendix A: References

|  |  |  |  |
| --- | --- | --- | --- |
| Sr. No. | Document | Version | Remarks |
| 1 |  |  |  |
| 2 |  |  |  |
| 3 |  |  |  |
| 4 |  |  |  |
| 5 |  |  |  |

Table 10‑4 References

## Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Version No. | Date | Owner | Changes |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table 10‑5 Revision History

## Items Out Of Scope of Verification