Verification Plan for Single Port Memory

==================================================================

Document #: 01

Release Date: 26/12/2024

Revision #:

Revision Date:

==================================================================

Originator

Name: Hythem Ahmed

Phone: +201145985850

email: hythemahmed29@gmail.com

==================================================================

Approved

Name:

Phone:

email:

==================================================================

==================================================================

Revisions History

Date:

Version:

Author:

Description:

# Introduction

This document establishes the verification plan for the **single port memory** design specified in the requirement specification. It identifies the features to be tested, the test cases, the expected responses, and the methods of test case application and verification.

The verification plan provides a definition of the testbench and verification environment, test sequences, application of test cases, and verification approaches for the single port memory design as specified in the requirement specification, and in the implementation specification.

# IP Design Details

A **single-port memory** is a type of random-access memory (RAM) that has one access port. This means that it can perform either a read or write operation at any given time, but not both simultaneously.

The block diagram of the **single-port memory** is shown in figure 1.

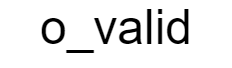
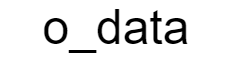
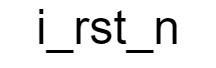
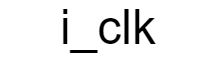
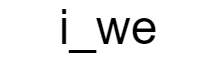
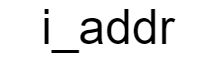
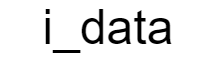
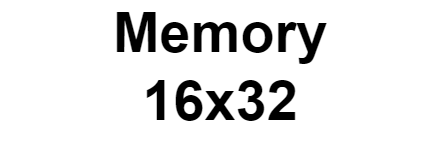


Figure 1. Block Diagram of Single-Port Memory

The Input/Output description of the **single-port memory** is shown in table 1.

|  |  |  |
| --- | --- | --- |
| Port | Width | Description |
| i\_data | 32 | Input data to be stored in the memory. |
| i\_addr | 4 | Input address that specifies location to be written/read to/from memory. |
| i\_we | 1 | Input write enable which enables the data to be written to the memory. |
| i\_clk | 1 | Input clock. |
| i\_rst\_n | 1 | Input active low reset. |
| o\_data | 32 | Output data to be read from memory. |
| o\_valid | 1 | Output flag which indicates that the read data is valid. |

Table 1. Input/Output Description of Single-Port Memory.

# Verification Strategy

Our verification strategy for verifying the single-port memory is a class-based environment using SystemVerilog. The architecture of the environment is shown in figure 2.

Sequencer

Driver

Monitor

Scoreboard

Subscriber

*Environment*

Virtual Interface

Interface

Memory

Transaction

Transaction

Transaction

Transaction

Transaction

Figure 2. Architecture of Class-Based Environment.

# Exit Criteria

# Test Items

The register shown below contains 33 bits

* Bit 32 → specifies whether memory can accept data or not.
* Bit 31:0 → data input to memory

|  |  |
| --- | --- |
| Memory Write Enable  reg[32] | Memory Output Data  reg[31:0] |
|

* Memory Write Enable:
  + Set to 1:
    - Memory accepts data.
    - Data is stored at memory location specified by address.
    - Write operation is done @ posedge of clock.
  + Set to 0:
    - Memory does not accept data.
    - Data is read from memory location specified by address.
    - Read operation is done @ posedge of clock.
* Memory Output Data:
  + Valid Data.
  + Invalid Data.

# Test Cases Table

|  |  |  |  |
| --- | --- | --- | --- |
| Tst # | Feature | Test Sequence | Description |
| 1 | RESET | i\_rst\_n = 0 | * All memory locations store a default value which is 0 * Output data is 0 * Valid flag is 0 |
| 2 | WRITE | i\_rst\_n = 1  i\_we = 1  i\_data = <data>  i\_addr = <address> | * The input data is stored in the memory location specified by the address. * Output data is last read data * Valid flag is 0 |
| 3 | READ | i\_rst\_n = 1  i\_we = 0  i\_addr = <address> | * The output data is read from the memory location specified by the address. * Valid flag is 1 |
| 4 | WRITE WITH RESET | i\_rst\_n = 0  i\_we = 1  i\_data = <data>  i\_addr = <address> | * All memory locations store a default value which is 0 * Output data is 0 * Valid flag is 0 |
| 5 | READ WITH RESET | i\_rst\_n = 0  i\_we = 0  i\_addr = <address> | * All memory locations store a default value which is 0 * Output data is 0 * Valid flag is 0 |

# Traceability Matrix

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Tst #** | **Reset** | | **Write Enable** | | **Output Data** | |
| Asserted | Deasserted | Enable | Disable | Valid | Invalid |
| 1 | ✅ |  |  | ✅ |  | ✅ |
| 2 |  | ✅ | ✅ |  |  | ✅ |
| 3 |  | ✅ |  | ✅ | ✅ |  |
| 4 | ✅ |  | ✅ |  |  | ✅ |
| 5 | ✅ |  |  | ✅ |  | ✅ |

# Coverage Results

## Code Coverage

* Statement Coverage: 100%

# ===============================Statement Details===============================

#

# Statement Coverage for instance /top/dut --

#

#     Line         Item                      Count     Source

#     ----         ----                      -----     ------

#   File memory.sv

#     18              1                         57

#     20              1                         12

#     20              2                        192

#     21              1                        192

#     23              1                         12

#     24              1                         12

#     27              1                         30

#     28              1                         30

#     31              1                         15

#     32              1                         15

# Toggle Coverage:

#     Enabled Coverage              Bins      Hits    Misses  Coverage

#     ----------------              ----      ----    ------  --------

#     Toggles                         68        68         0   100.00%

A screen shot of a computer

Description automatically generated

* Branch Coverage: 100%

# ================================Branch Details================================

#

# Branch Coverage for instance /top/dut

#

#     Line         Item                      Count     Source

#     ----         ----                      -----     ------

#   File memory.sv

# ------------------------------------IF Branch------------------------------------

#     19                                        57     Count coming in to IF

#     19              1                         12

#     26              1                         30

#     30              1                         15

# Branch totals: 3 hits of 3 branches = 100.00%

#

#

# Statement Coverage:

#     Enabled Coverage              Bins      Hits    Misses  Coverage

#     ----------------              ----      ----    ------  --------

#     Statements                      10        10         0   100.00%

A screenshot of a computer

Description automatically generated

* Condition Coverage: NA
* Expression Coverage: NA
* FSM Coverage: NA

## Functional Coverage

A screenshot of a computer

Description automatically generated

# Opened Issues

# Feature Assessment