Design & Verification of a Dual-port Memory as an APB subordinate (slave)

Nader Alnatsheh

December 16, 2025

Contents

1 Introduction

The purpose of this project is to build a reusable verification environments to be used for my personal projects in the future. The verification enviroments are based on the SystemVerilog (SV) Object-Oriented Programming (OOP) methodology, and the Universal Verification Mehtodology (UVM).

Since the main focus is verification, the design used in the project is a simple dual-port memory

2 Design Specs of the Dual-port Memory with an AMBA APB interfaces

2.1 Dual-port Memory Specs

2.1.1 Read Operation

After the rising edge of the PENABLE signal the PRDATA provides the data after 2 clock cycles.

2.1.2 Write Operation

After the rising edge of the PENABLE signal the PWDATA copies the the data after 2 clock cycles.

1. Addresses ranges from (0x0 - 0xf) are READ ONLY, and the manager (master) should not write to them, when the manager tries to write to a READ ONLY address, the subordinate (slave) raises the PSLVERR signal after 1 clock cycle to indicates an error.

2. For all other Addresses the write operation takes 4 clock cycles. And the memory contents updates on the 5th clock cycle.

2.2 AMBA APB Specs

2.2.1 Read Operation

2.2.2 Write Operation

3 Verification Specs of the Dual-port Memory with an AMBA APB interface

3.1 Test-plan

3.2 OOP SV Environment

3.2.1 Transaction (Transaction.sv)

The Transaction class contains:

1. The fields required to generate the randomized stimulus
2. Any constraints needed to constraint the randomized stimulus
3. Any functions that might be used in other classes. Since we will declare the Transaction as a handle in other classes

3.2.2 Generator

The Generator class is responsible for:

1. Generating the stimulus by randomizing the Transaction class
2. Sending the randomized stimulus to the Driver class via a Mailbox. (Note: that the Mailbox handle will be coming from the Environment class, since the same Mailbox will be shared between the Generator and Driver classes)
3. An event is declared and will be used to indicate when the Generator class has finished generating the transactions

3.2.3 Driver

The Driver class is responsible for:

1. Recieves the randomzied stimulus generated by the Generator class via the Mailbox
2. Drives the DUT by assigining the Transaction class values to the interface signals

3.2.4 Monitor

3.2.5 Scoreboard

3.2.6 Enviromnent

3.3 UVM Environment

3.3.1 Sequence

3.3.2 Sequencer

3.3.3 Driver

3.3.4 Monitor

3.3.5

3.3.6

4 References