**I2C\_Controller Verification Environment Specification**

Table of Contents

[Document history 3](#_Toc192583005)

[1.Introduction 4](#_Toc192583006)

[1.1 Scope 4](#_Toc192583007)

[1.2 Abbreviations 4](#_Toc192583008)

[1.3 References 4](#_Toc192583009)

[2. I2C\_Controller Overview 5](#_Toc192583010)

[2.1. I2C\_Controller Features 5](#_Toc192583011)

[2.1.1. Master mode features 5](#_Toc192583012)

[2.1.2. Slave mode features 6](#_Toc192583013)

[2.2. I2C\_Controller Interfaces 6](#_Toc192583014)

[3.Verification Environment 7](#_Toc192583015)

[3.1 Environment Overview 7](#_Toc192583016)

[3.2 Components description 8](#_Toc192583017)

[3.4 Checkers 10](#_Toc192583018)

[3.6 Tests 11](#_Toc192583019)

[4. Simulation flow 12](#_Toc192583020)

[5. How to run a test 13](#_Toc192583021)

[6. Limitations and assumptions 13](#_Toc192583022)

# Document history

| Version | Date | Approved by | Created by | Description |
| --- | --- | --- | --- | --- |
| 0.1 | 03/03/2025 | Dorian Stan | Zsolt Laszlo | Document created |
|  |  |  |  |  |

# 1.Introduction

## 1.1 Scope

This document describes the verification environment for the I2C\_Controller using UVM

## 1.2 Abbreviations

|  |  |
| --- | --- |
| **DUT** | Device Under Test |
| **APB** | Advanced Peripheral Bus |
| **I2C** | Inter-Integrated Circuit |

## 1.3 References

|  |  |  |  |
| --- | --- | --- | --- |
| **No** | **Name** | **Revision** | **Description** |
| 1 | I2C\_Controller Specification.pdf | 1.0 | DUT RTL specification |
| 2 | I2C\_Controller Metric plan.xlsx | 1.0 | Coverage, Checkers and Test plans |

# 2. I2C\_Controller Overview

* 1. A diagram of a bridge

     AI-generated content may be incorrect.

Figure 1. The I2C\_controller input/output interfaces & architecture diagram

## 2.1. I2C\_Controller Features

* Master/Slave functionality
* Variable clock divider
* Variable transfer length up to 256 bytes
* Large data transfer with minimal software access
* Configurable interrupt request interface

### 2.1.1. Master mode features

* Clock stretching: Slave devices can slow down the transfer rate by driving the SCL low for the required period and then releasing it
* Clock synchronization and arbitration: When two master devices try to control SCL, the line will be held low by the device with the longest low period. The first master that will detect SDA being held low while trying to drive it high loses arbitration. Large data transfer with minimal software access

### 2.1.2. Slave mode features

* While in slave mode, the device will stop processing a WRITE operation when it detects a STOP condition. If the protocol is violated and the START/STOP condition is generated in the middle of a transfer, the data byte for that particular transfer is discarded and the transfer stopped
* SR ( Repeated Start ) is supported while in Slave mode but with certain limitations. Every SR condition is treated as a STOP and the previous transfer is ended, giving registers access from the APB side until the address following the SR condition is matched. This interval can be used either to process the data or stop future transfers by setting ENABLE\_ACK to 0.
* When in slave mode, SCL and SDA are sampled using func\_clock and as a requirement, its frequency must always be at least 6 times higher than the SCL frequency. This also applies to START and STOP conditions as the time difference between the SDA toggle and the SCL toggle needs to be at least equal to the HIGH period of the SCL.

## 2.2. I2C\_Controller Interfaces

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **interface** | **signal** | **witdh** | **direction** | **note** |
| **APB** | psel | 1 | I | active high, indicates start of APB transaction |
| penable | 1 | I | active high, device enable, indicates acces pahse of transaction |
| pwrite | 1 | I | APB read -> 0 | APB write -> 1 |
| paddr | 6 | I | address of register to be accessed |
| pwdata | 32 | I | data to be written in the regster |
| prdata | 32 | O | data read from the register |
| pready | 1 | O | slvave ready |
| pslverr | 1 | O | slave error |
| **I2C** | scl | 1 | I/O | Serieal clock |
| sda | 1 | I/O | Serial data |
| **INTERRUPT** | irq | 1 | O | interrup request |

# Verification Environment

## 3.1 Environment Overview

* 1. The I2C\_Controller verification environment diagram is presented in Figure 2.

1. A diagram of a software

   AI-generated content may be incorrect.

Figure 2. The I2C\_Controller Verification Environment Diagram

## 3.2 Components description

|  |  |
| --- | --- |
| DUT | I2C controller |
| Virtual Interface | A bridge between the physical interface of the device and the high-level components |
| Driver | Puts the transaction on the interface according to the specific protocol |
| Monitor | Gets the relevant data from the interface and sends it to the scoreboard for checking and coverage |
| Sequencer | Initiates a sequence to the driver |
| Coverage | Notes the trigger of specific events or states that the device is/was in |
| Agent | A configurable bundle of high-level components specific to a protocol |
| Scoreboard | A global monitor that checks the data integrity and DUT specific operations, it is not bound by any protocol |
| REG model | A high-level register bank that mirrors the behavior of the registers found in DUT |
| Virtual sequencer | A bridge that links the test to the env. Launches the test specific sequences to the corresponding components |

## 3.3 Register model

A diagram of a software system

AI-generated content may be incorrect.

Figure 3. Register model functionality

* All the legal read and write APB sequences are initiated via the register models adapter.
* APB sequences which aim for PSLVERR to be asserted are launched outside of the register model.
* The reading sequence also includes and automated checking mechanism utilizing the mirror method.
* The read only registers are not updated in the model, the mirror value is predicted before reading.

## 3.4 Checkers

A diagram of a process flow

AI-generated content may be incorrect.

Figure 3. Main checking (located in scoreboard) flowchart

The protocol specific checkers are in their corresponding agents’ monitor or in the assertions located in the protocols interface file.

The main checking (device and registers) is in the scoreboard

All the checkers and coverage implementations can be found in the checkers/coverage sections of the attached metric plan.

## 3.6 Tests

The tests’ description can be found in I2C\_Controller metric driven plan.xlsx, the “Tests” sheet.

# A diagram of a flowchart AI-generated content may be incorrect.4. Simulation flow

Figure 4. Simulation flow of a standard test

# 5. How to run a test

Running a scrip through Makefile:

* make run\_\* TESTNAME=<test\_name> SEED=<seed> VERBOSITY=<verbosity> => **for singular test**
* make run\_regression => **for running all tests**

# 6. Limitations and assumptions