

Verification of I²C

Subhadip Sen

27th February, 2025

Contents

1. Introduction
2. Objective
3. Block Diagrams
4. FSM of different blocks
5. Testbench
6. Simulation Results

1. Introduction:

I²C is a widely used, synchronous, multi-master, multi-slave, two-wire serial communication protocol developed by Philips (now NXP). It enables communication between integrated circuits, typically on the same PCB, using just two bidirectional lines:

- **SCL (Serial Clock Line)**: Carries the clock signal, generated by the master.
- **SDA (Serial Data Line)**: Carries the data bits.

Key features:

- Two-wire interface (**SCL & SDA**) → minimal pin count.
- Supports **multiple masters and multiple slaves**. Number of nodes limited by address space & total capacitance of **400pF**.
- **Master controls the clock** and initiates communication.
- **7, 10, 16 bit** addressing modes.
- Supports **standard (100 kbps)**, **fast (400 kbps)**, **fast+ (1 Mbps)**, and **high-speed (3.4 Mbps)** modes.
- **ACK/NACK**-based handshaking for data integrity.

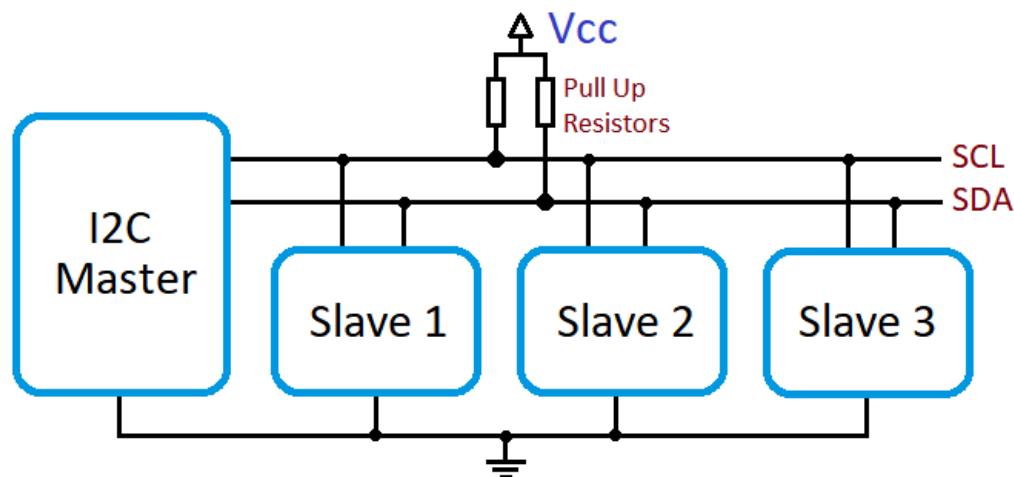


Fig 1: Master & Slave

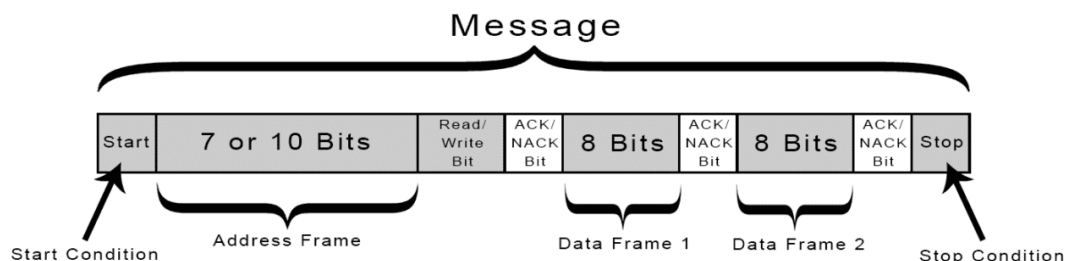


Fig 2: Frame Structure

2. Objective:

The objective of this project is to **design and verify a robust I²C (Inter-Integrated Circuit) protocol controller** that adheres to the standard I²C specification. The design will support:

- **7-bit addressing**
- **Single master, single slave support**
- **Standard and fast mode operations (100 kbps & 400 kbps)**
- **Configurable read/write operations**
- **Start, repeated start, stop condition handling**
- **ACK/NACK detection and generation**

The verification will be done using **System Verilog UVM methodology**, covering functional and corner cases with comprehensive test scenarios to ensure correctness, reliability, and protocol compliance.

3. Block Diagram:

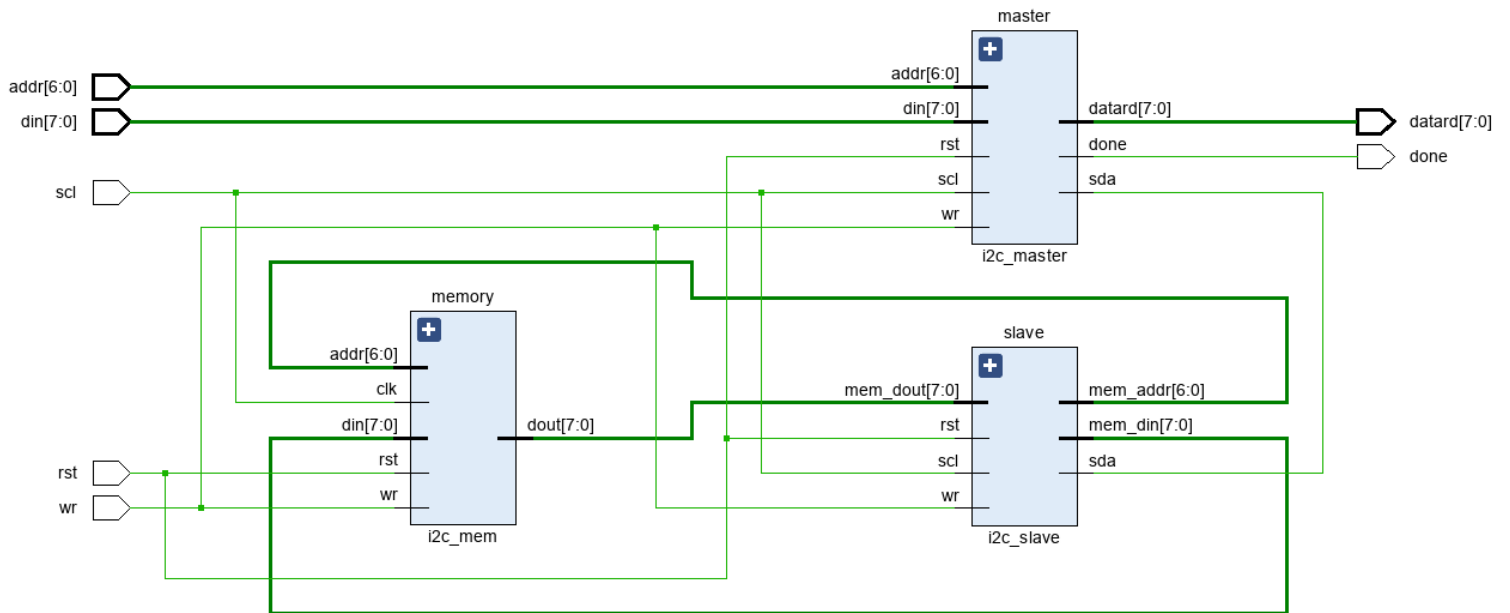


Fig 3: DUT block diagram

a. master:

Designed and implemented a custom I²C Master controller in System Verilog as part of an integrated I²C master-slave memory communication system. The I²C Master

initiates and manages communication with a slave memory module based on the I²C protocol.

I2C Master – Read Operation Summary

The I2C Master takes inputs: addr [6:0], din [7:0], wr, rst, and scl. For a read (wr = 0), it performs the following steps:

- **Start Condition** – Pulls SDA low while SCL is high.
- **Send Address + Read Bit** – Sends {addr, 1'b0} on SDA.
- **Wait for ACK** – Monitors slave response.
- **Read Data** – Receives 8-bit data from slave.
- **Complete** – Asserts done and outputs data on datard.

The operation is managed by an FSM with states: IDLE, START, SEND_ADDR, GET_ACK1, READ_DATA, COMPLETE. The module ensures protocol-compliant I2C read functionality.

I2C Master – Write Operation Summary

The I2C Master takes inputs: addr [6:0], din [7:0], wr, rst, and scl. For a write (wr = 1), it performs the following steps:

- **Start Condition** – Pulls SDA low while SCL is high.
- **Send Address + Write Bit** – Sends {addr, 1'b1} on SDA.
- **Wait for ACK** – Waits for ACK from slave.
- **Send Data** – Shifts out din [7:0] to the slave.
- **Wait for ACK** – Confirms slave received the data.
- **Complete** – Asserts done and returns to idle.

Controlled by an FSM with states: IDLE, START, SEND_ADDR, GET_ACK1, SEND_DATA, GET_ACK2, COMPLETE. Ensures proper I2C protocol handling for data writes.

b.slave:

The I2C Slave module handles communication with the master and interfaces with an external memory module.

Key Features:

- **Protocol Handling** – Detects start condition, receives 7-bit address + R/W bit, and manages ACK/NACK signaling.
- **Read Operation**
 - On a read request, fetches data from the external memory and sends it to the master bit-by-bit on SDA.

- **Write Operation**
 - Receives 8-bit data from the master and writes it to the external memory at the specified address.
- **External Memory Interface** – Interacts with a separate memory module for read/write access, based on the decoded address and data.

FSM States:

IDLE, START, GET_ADDR, SEND_ACK1, GET_DATA, SEND_ACK2, SEND_DATA, COMPLETE. The slave ensures I2C protocol compliance and acts as a communication bridge between the I2C master and the memory module.

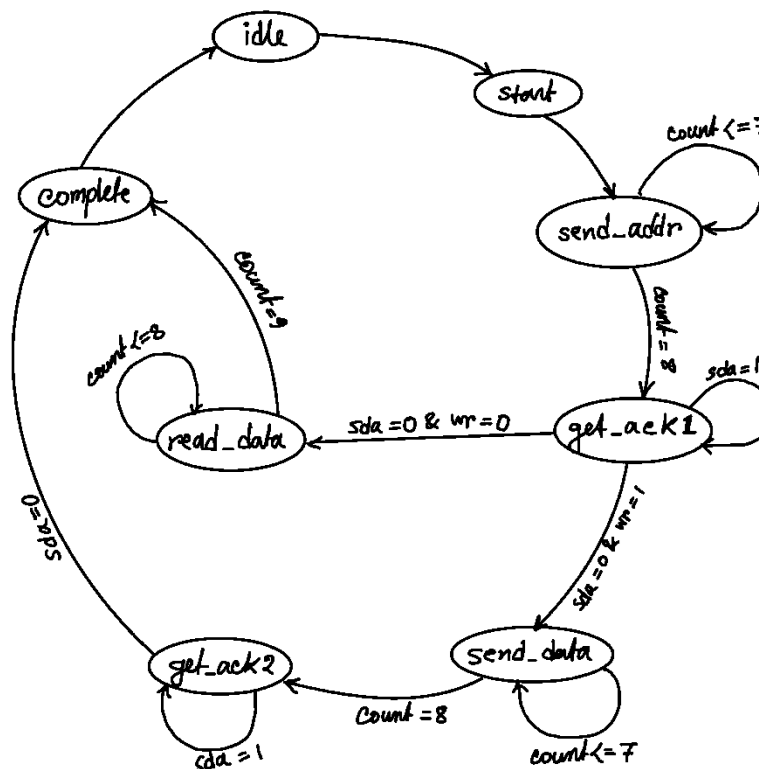
c. memory:

The memory module is a simple 128-byte register array designed to store and retrieve 8-bit data as directed by the I2C Slave Controller.

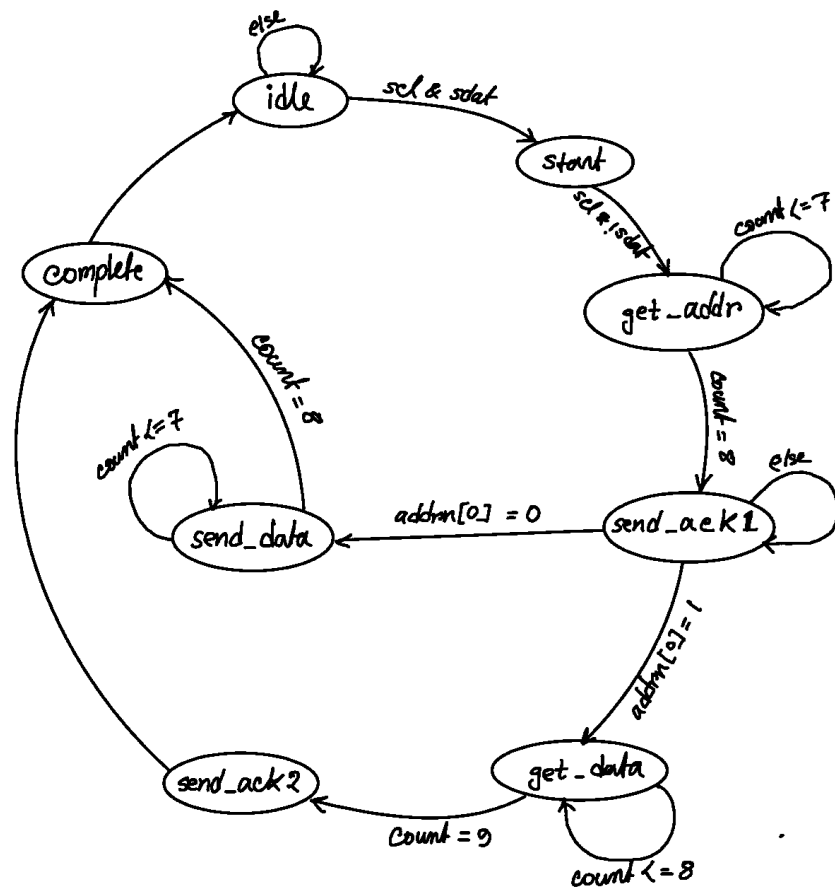
Key Features:

- **Size:** 128×8 -bit memory array.
- **Read/Write Interface:**
 - **Write:** Accepts write_enable, address, and data_in signals from the slave to store data.
 - **Read:** Provides data_out based on the given address when read_enable is asserted.
- **Synchronous Operation:** Write operations are triggered on the rising edge of the clock. Read is done through continuous assignment.

4. FSM of Master:



FSM of Slave:



5. Testbench:

The testbench verifies the functionality of the I2C Master-Slave system, including reset, write, and read sequences. It stimulates the DUT (Device Under Test) with realistic I2C transactions and checks correctness using random input patterns.

Key Features:

- **Reset Sequence:**

Initializes all DUT signals and waits for a few clock cycles before starting transactions.

- **Write Sequence:**

- Generates a **random 7-bit address** and **8-bit data**.
- Sets $wr = 1$ and triggers a write transaction via the I2C master.
- Waits for done to be asserted, indicating transaction completion.

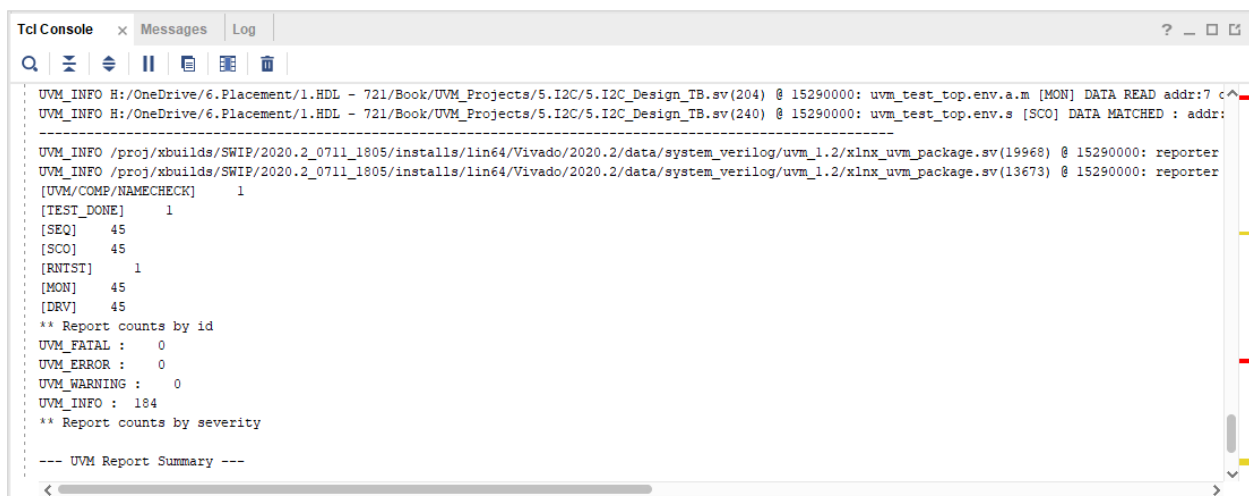
- **Read Sequence:**

- Uses the same address to initiate a read ($wr = 0$) from the slave.
- Waits for done, then compares datard (data read) with expected data.

- **Randomization:**

Uses \$random to generate varied address and data values for robustness.

Monitor captures the send address and data during write and read operation, broadcast the packet through analysis port. **Scoreboard** has reference memory which also store the data during write operation and compares the data with DUT during read operation, reports pass or fail status.



```
Tcl Console x Messages Log ? _ □ □
UVM_INFO H:/OneDrive/6.Placement/1.HDL - 721/Book/UVM_Projects/5.I2C/5.I2C_Design_TB.sv(204) @ 15290000: uvm_test_top.env.a.m [MON] DATA READ addr:7 c^
UVM_INFO H:/OneDrive/6.Placement/1.HDL - 721/Book/UVM_Projects/5.I2C/5.I2C_Design_TB.sv(240) @ 15290000: uvm_test_top.env.s [SCO] DATA MATCHED : addr:
-----
UVM_INFO /proj/xbuilds/SWIP/2020.2_0711_1805/installs/lin64/Vivado/2020.2/data/system_verilog/uvm_1.2/xlnx_uvm_package.sv(19968) @ 15290000: reporter
UVM_INFO /proj/xbuilds/SWIP/2020.2_0711_1805/installs/lin64/Vivado/2020.2/data/system_verilog/uvm_1.2/xlnx_uvm_package.sv(13673) @ 15290000: reporter
[UVM/COMP/NAMECHECK] 1
[TEST_DONE] 1
[SEQ] 45
[SCO] 45
[RNTST] 1
[MON] 45
[DRV] 45
** Report counts by id
UVM_FATAL : 0
UVM_ERROR : 0
UVM_WARNING : 0
UVM_INFO : 184
** Report counts by severity
--- UVM Report Summary ---
```

Fig 4: Simulator Output

Simulation results:

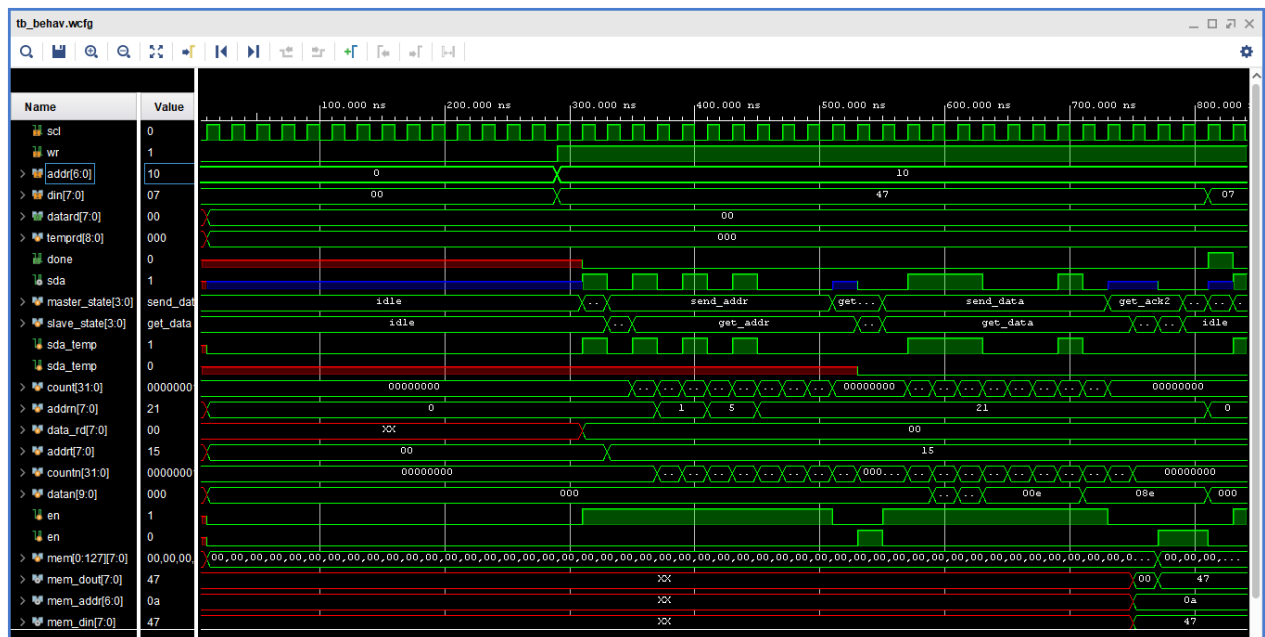


Fig 5: Single write to memory

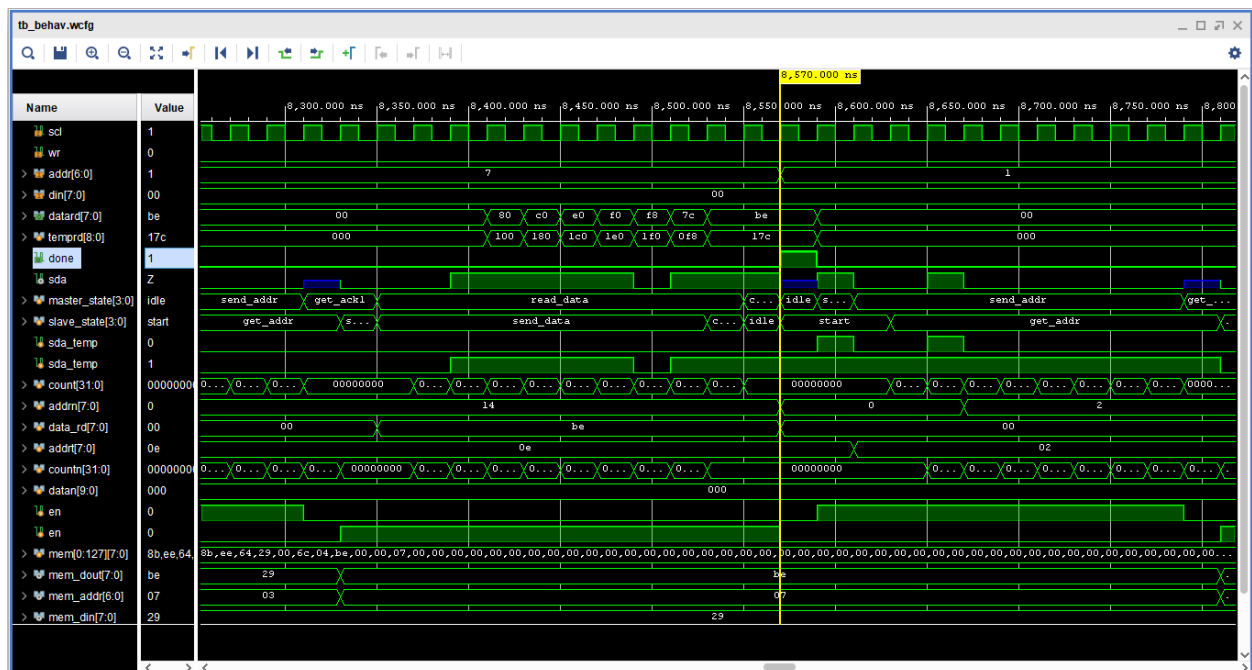


Fig 6: Single read from memory

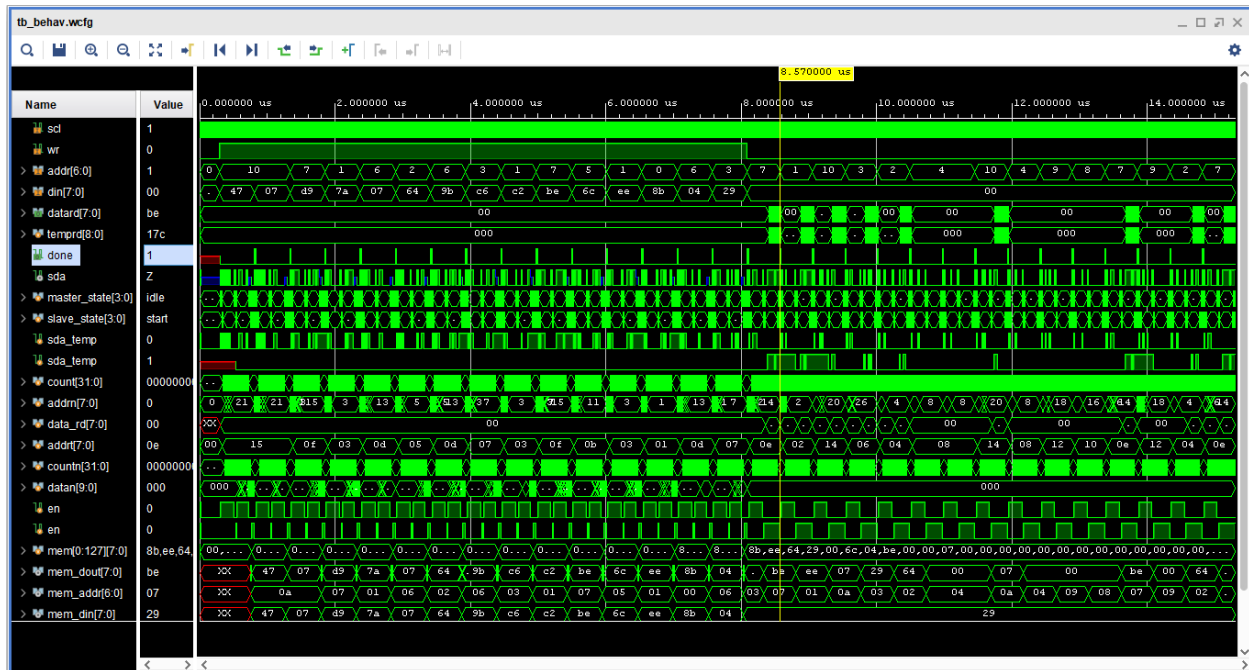


Fig 7: All sequence simulation result

It can be seen that all sequences are passing, read and write data matching for all memory locations.

- System verilog for Design & UVM verification environment written & simulated in Vivado.