Design and Verification of Memory Design

The Design and Verification of Memory Design aims to design, model, and verify a simple memory system using System Verilog advanced verification features.
The project demonstrates the complete **verification methodology** — from stimulus generation to coverage analysis — using a modular, reusable testbench architecture.

The primary goal is to ensure that the memory behaves as expected for all read and write transactions while maintaining data integrity, proper handshaking, and functional correctness.

1. Objectives

The main objectives of this project are:

1. To design a simple **memory model** capable of handling read and write operations.
2. To develop each block of testbench (generator,bfm,moniotor and etc).
3. To develop a **constrained random testbench** that can automatically stimulate the design.
4. To implement **functional coverage** to measure verification completeness.
5. To include **System Verilog Assertions (SVA)** to ensure protocol correctness.
6. To analyze the simulation results and verify 100% functional and code coverage.

2. Design Overview(DUT-memory.sv)

The DUT (Design Under Test) is a simple synchronous memory module with the following ports:

Signal	Direction	Description
clk	Input	System clock
rst	Input	Active-high reset
wr_rd	Input	Write/Read control (1 = Write, 0 = Read)
addr	Input	Memory address bus
wdata	Input	Write data bus
rdata	Output	Read data bus
valid	Output	Indicates valid read/write operation
ready	Output	Indicates memory ready for next transaction

Functionality:

- During **write**, data is stored at the specified address.
- During **read**, the memory outputs data from the given address.
- Signals valid and ready manage synchronization between DUT and testbench.

The design uses clocked always blocks and supports synchronous reset, ensuring that memory and control signals are initialized properly.

3. Verification Environment Overview

The environment is structured based on **UVM-like modular principles**, with individual files representing each component:

File	Function
mem_tx.sv	Transaction class – defines stimulus data items
mem_gen.sv	Generator – creates random or directed transactions
mem_bfm.sv	Bus Functional Model (Driver) – drives signals to DUT
mem_monitor.sv	Monitors DUT interface and collects observed data
mem_scoreboard.sv	Compares expected vs actual results
mem_coverage.sv	Collects functional coverage metrics
mem_assertion.sv	Contains protocol and data assertions
mem_tb.sv	Top-level testbench connecting all components

4. Verification Components in Detail

- **Transaction(mem_tx.sv)**

Defines the basic data structure for stimulus:

```
class mem_tx;
  rand bit wr_rd;
  rand bit [4:0] addr;
  rand bit [7:0] wdata;
  bit [7:0] rdata;
  constraint addr_range { addr inside {[0:15]}; }
endclass
```

Transactions represent a single memory operation.

- **Generator(mem_gen.sv)**

Responsible for generating transactions and sending them to the driver.

- Can operate in **random** or **directed** modes.
- Uses a mailbox to pass transactions.
- Runs continuously until a certain number of operations are completed.

- **Driver/BFM(mem_bfm.sv)**

Implements a Bus Functional Model that drives signals on the DUT interface.

1. Takes transactions from generator.
2. Drives addr, wdata, and wr_rd onto the DUT.
3. Handles valid and waits for ready.
4. Emulates real hardware timing.

- **Monitor(mem_monitor.sv)**

Passively observes DUT signals and creates observed transactions.

- Sends observed read/write data to scoreboard.
- Also updates coverage and assertion triggers.

- **Scoreboard(mem_scoreboard.sv)**

Compares expected and actual results to ensure data integrity.

- Uses queues or associative arrays for storing expected memory values.
- Reports mismatches as errors.

- **Coverage(mem_coverage.sv)**

Defines coverage groups for opcode, address, and read/write operations.

- **Assertions(mem_assertion.sv)**

Contains multiple **System Verilog Assertions (SVA)** to ensure protocol correctness.

Property	Description
Reset Clean	During reset, all signals must be 0
Handshake	If valid is high, ready must go high next cycle
Writes	On write operation, addr and wdata must not be X/Z
Reads	On read operation, addr and rdata must not be X/Z

- **Testbench Top(mem_tb.sv)**

The top-level testbench instantiates:

- DUT (memory)
- Verification components (generator, driver, monitor, scoreboard, etc.)
- Clock and reset generation

5. Simulation result

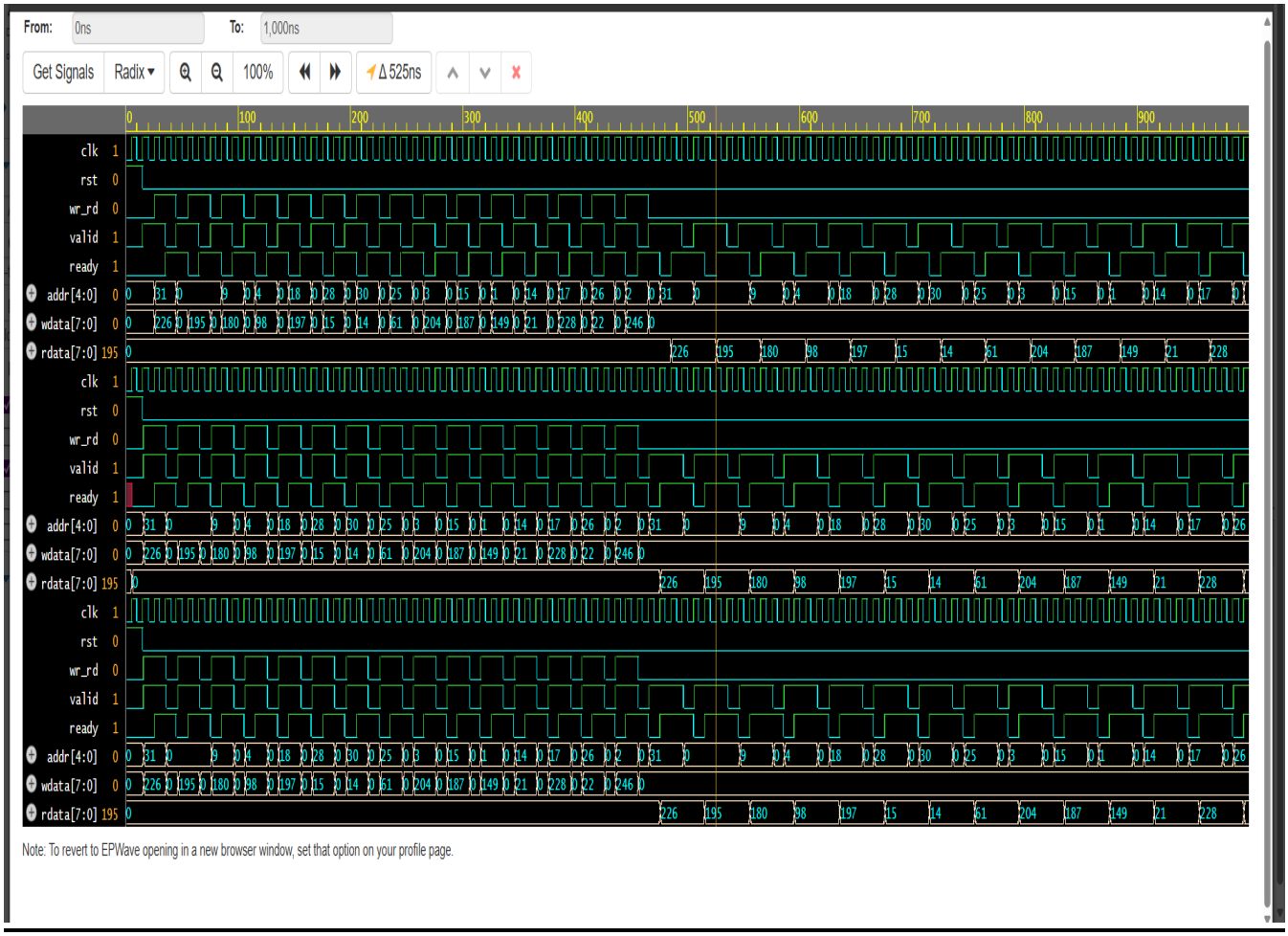
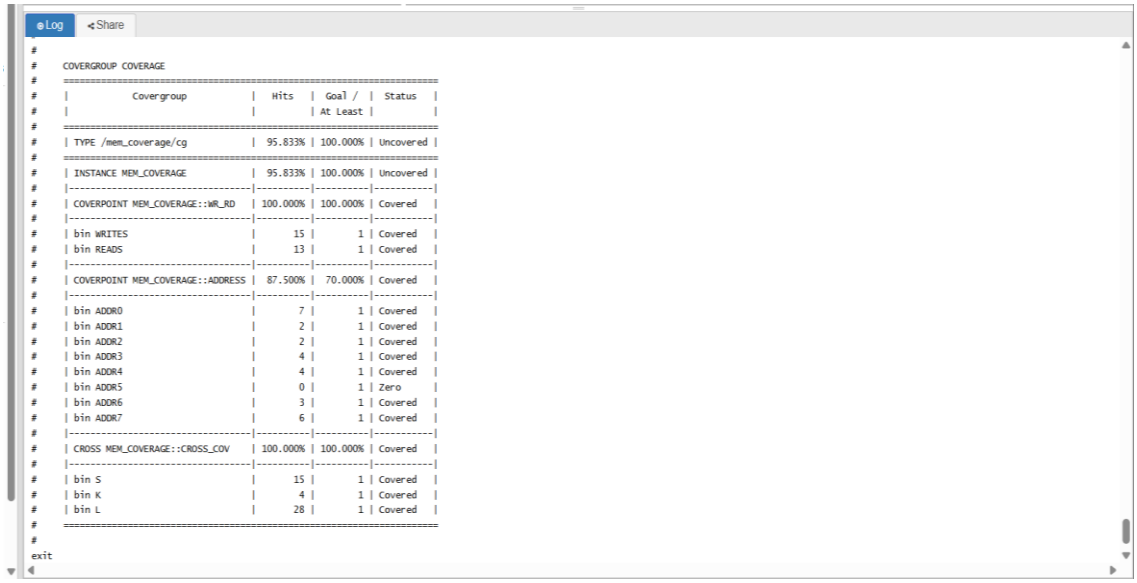


Fig: shows the simulation result of dut, pif and uut signals.



```
#
#    COVERGROUP COVERAGE
#    ============================================================================
#    |           Covergroup            |   Hits   | Goal /  |  Status   |
#    |                                 |          | At Least |           |
#    ============================================================================
#    | TYPE /mem_coverage/cg           |  95.833% | 100.000% | Uncovered |
#    ============================================================================
#    | INSTANCE MEM_COVERAGE           |  95.833% | 100.000% | Uncovered |
#    |---------------------------------|----------|----------|-----------|
#    | COVERPOINT MEM_COVERAGE::WR_RD  | 100.000% | 100.000% | Covered   |
#    |---------------------------------|----------|----------|-----------|
#    | bin WRITES                      |       15 |        1 | Covered   |
#    | bin READS                       |       13 |        1 | Covered   |
#    |---------------------------------|----------|----------|-----------|
#    | COVERPOINT MEM_COVERAGE::ADDRESS |  87.500% |  70.000% | Covered   |
#    |---------------------------------|----------|----------|-----------|
#    | bin ADDR0                       |        7 |        1 | Covered   |
#    | bin ADDR1                       |        2 |        1 | Covered   |
#    | bin ADDR2                       |        2 |        1 | Covered   |
#    | bin ADDR3                       |        4 |        1 | Covered   |
#    | bin ADDR4                       |        4 |        1 | Covered   |
#    | bin ADDR5                       |        0 |        1 | Zero      |
#    | bin ADDR6                       |        3 |        1 | Covered   |
#    | bin ADDR7                       |        6 |        1 | Covered   |
#    |---------------------------------|----------|----------|-----------|
#    | CROSS MEM_COVERAGE::CROSS_COV   | 100.000% | 100.000% | Covered   |
#    |---------------------------------|----------|----------|-----------|
#    | bin S                           |       15 |        1 | Covered   |
#    | bin K                           |        4 |        1 | Covered   |
#    | bin L                           |       28 |        1 | Covered   |
#    ============================================================================
#
exit
```

Fig:coverage report

6. Conclusion

The **Design and Verification of Memory Design** successfully demonstrated a complete SystemVerilog verification flow for a simple memory DUT.

Key accomplishments:

- Full modular testbench with transaction-level communication
- Real-time protocol validation using **SystemVerilog Assertions**
- Functional coverage achievement proving verification completeness
- Robust and reusable testbench suitable for extending to larger memory architectures

This project showcases the effectiveness of **assertion-based verification (ABV)** combined with **constrained random testing**, forming the foundation for industry-grade **UVM verification environments**.