Lab 6: Memory Interfaces

8.1 Lab 6A: Synchronous SRAM Controller

Objective

The objective of this lab is to design and verify a **Synchronous SRAM Controller** for a 32K × 16-bit memory module. The controller should support single-cycle **read** and **write** operations, ensure proper timing for address/data buses, and implement chip enable and output enable controls.

Specification

- Interface to **32K × 16 synchronous SRAM**
- Support **single-cycle read/write operations**
- Correct handling of **address and data bus timing**
- Control of **chip enable (CE), output enable (OE), and write enable (WE)** signals

Interface Timing Analysis

1. **Study of SRAM datasheet timing requirements**
    - Setup and hold times for address and data relative to the clock
    - Read and write cycle timing specifications
2. **Timing diagrams for read and write cycles**
    - **Write Cycle**: Address and data stable before write enable goes active
    - **Read Cycle**: Address stable before output enable goes active

3. **Setup/Hold Calculations**
    - Setup time (tSU) ensures data/address is stable before clock edge
    - Hold time (tH) ensures data/address remains stable after clock edge
4. **Address/Data Multiplexing**
    - Address lines directly drive SRAM
    - Data lines are **bidirectional (tri-state)**, driven only during writes

System Design Description

- A **SRAM model** was created to represent the 32K × 16-bit memory array.
- A **controller** was designed using a **Finite State Machine (FSM)** with three states:
    - **IDLE** – waiting for request
    - **READ** – performing read cycle
    - **WRITE** – performing write cycle
- **Tri-state buffers** were used to manage the shared data bus between controller and memory.
- A **Top-Level module** integrated both the controller and the SRAM model.
- A **testbench** verified correct operation for write and read requests.

Results and Discussion

- The **SRAM controller** successfully interfaces with a 32K × 16 SRAM model.
- Simulation confirmed correct **read and write operations**.

- **Tri-state bus handling** was correctly implemented to allow bidirectional data transfer.
- The controller immediately returns to **IDLE** state after each transaction, enabling multiple consecutive accesses.
- The design ensures timing requirements are satisfied based on synchronous operation with the clock.
- This design can be further extended to support **burst operations, wait states, or integration with AXI/AHB bus protocols**.

Conclusion

This lab demonstrated the design of a **Synchronous SRAM Controller** interfacing with a 32K × 16 SRAM. The exercise reinforced key concepts of memory interfacing, timing analysis, and tri-state bus control. The final design was successfully verified in simulation, providing a strong foundation for more advanced **memory subsystem designs** in digital systems.