VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY

HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY

FACULTY OF COMPUTER SCIENCE AND ENGINEERING

LOGIC DESIGN ASSIGNMENT REPORT

CLASS CC01 – GROUP 2

SUBJECT:

8X8X8 MEMORY

Instructor: Nguyễn Thành Lộc

Student	ID
Lê Võ Nghĩa Hiệp	2452347
Phan Thế Thông	2453210
Nguyễn Văn Minh Huy	2452401
Ngô Đức Anh	2452054

Table of Contents

1. Introduction

The report focuses on the design and implementation of an 8x8x8 memory system using Verilog HDL. The scope includes practicing hierarchical design approaches, improving skills in digital circuit design, and applying knowledge of memory circuits used in computer engineering. The report also emphasizes teamwork, research, and documentation. The project simulates and demonstrates how memory modules can be structured and accessed, offering both theoretical understanding and practical application.

2. Background and applications

2.1 Concept

In digital electronics, memory is a hardware circuit capable of storing and retrieving binary data (0s and 1s). Unlike combinational logic circuits, which produce outputs based only on the current inputs, memory circuits belong to sequential logic because they can retain information over time, even when inputs change. This retention makes memory one of the most fundamental components of any computing system.

A basic memory cell stores one bit of information. By grouping cells together, we can form a word (multiple bits). In this project, each word is 8 bits wide (1 byte). These words are arranged in a matrix of rows and columns, allowing data to be accessed using addresses.

2.2 Classification of Memory

From a functional perspective, digital memory can be broadly classified into two categories:

1. Volatile Memory
   Volatile memory requires a continuous supply of electrical power to preserve stored information. Once the power is removed, the stored contents are lost. The most widely used example is Random Access Memory (RAM), which includes:
   - Static RAM (SRAM): utilizes flip-flops to store data, offers high speed and low latency, but is expensive and consumes more silicon area. It is often employed in cache memory.

  - Dynamic RAM (DRAM): employs capacitors for storage, enabling higher density and lower cost, but requires periodic refresh cycles. It is commonly used as main system memory.

2. Non-Volatile Memory

   Non-volatile memory retains data even in the absence of electrical power. Examples include Read-Only Memory (ROM), Electrically Erasable Programmable Read-Only Memory (EEPROM), and Flash memory. Such devices are typically employed for firmware storage, system boot programs, and long-term data retention.

The design investigated in this assignment corresponds conceptually to a synchronous RAM, wherein both read and write operations are triggered on the rising edge of a clock signal. This ensures deterministic timing behavior, which is essential in synchronous digital systems.

2.3 Organization of 8×8x8 Memory

The memory module considered in this project is organized as a **two-dimensional array** comprising **8 rows and 8 columns**, resulting in a total of **64 addressable locations**. Each location is capable of storing an **8-bit word**, yielding a total storage capacity of **64 bytes**.

The memory is addressed through two 3-bit signals:

- `row_addr[2:0]` selects one of the eight rows.
- `col_addr[2:0]` selects one of the eight columns.

Thus, the pair `(row_addr, col_addr)` uniquely specifies a storage location. The control signal `wrt_read` governs the functional mode:

- `wrt_read = 0` initiates a **write operation**, in which the value present on `data_in` is stored into the selected location.

- `wrt_read = 1` initiates a **read operation**, in which the value from the selected location is transferred to `data_out`.

The system is characterized as a **single-port memory**, implying that, within one clock cycle, only one operation—either read or write—can be executed.

2.4 Applications

Memory plays a pivotal role in the hierarchical organization of modern computer systems:

- Registers serve as the smallest and fastest memory elements embedded within the Central Processing Unit (CPU).
- Cache memory, typically implemented with SRAM, provides rapid access to frequently used instructions and data.
- Main memory (RAM) holds both programs and data required during execution.

Although the memory module designed here is limited in capacity (8×8), it embodies the fundamental operational principles employed in large-scale memory systems utilized in contemporary computing architectures.

In Field-Programmable Gate Arrays (FPGAs), memory constructs are pervasive. The Look-Up Tables (LUTs) used to implement logic functions are essentially small memory blocks. In addition, modern FPGAs integrate dedicated Block RAM (BRAM) resources for larger storage requirements. Designing and simulating an 8×8 memory using Verilog HDL allows students to:

- Comprehend the organization of digital memory.
- Gain practical experience with address decoding and synchronous data access.
- Understand how synthesis tools infer memory primitives from HDL descriptions.

From a pedagogical standpoint, the design of an 8×8 memory block serves as an instructive case study. It enables students to:

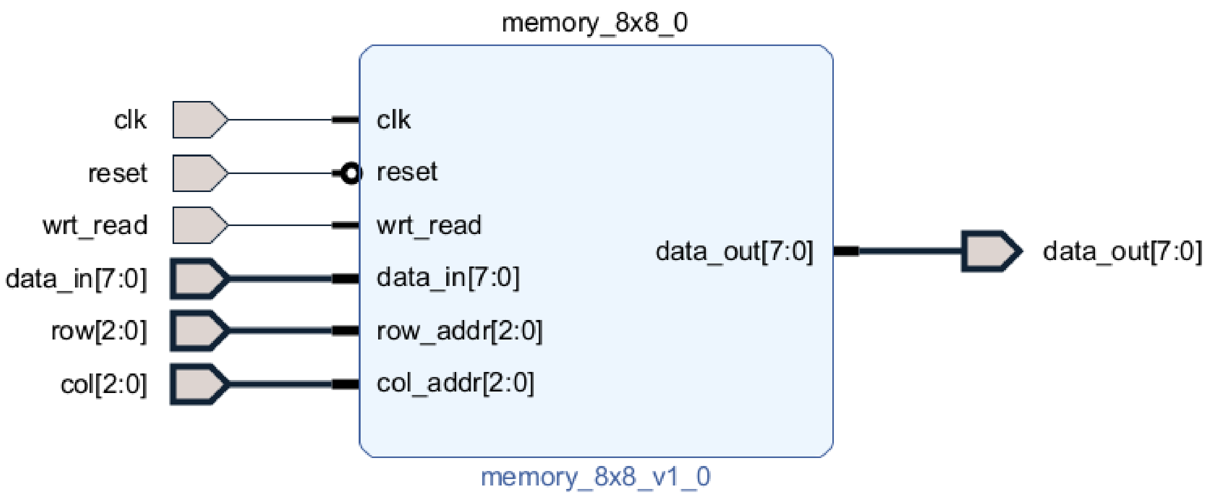
- Acquire proficiency in Hardware Description Languages (HDL) such as Verilog.

- Simulate and observe the timing behavior of read and write operations.
- Establish the conceptual connection between **basic sequential elements** (e.g., flip-flops, latches) and **complex functional units** (e.g., RAM, cache, and ROM).

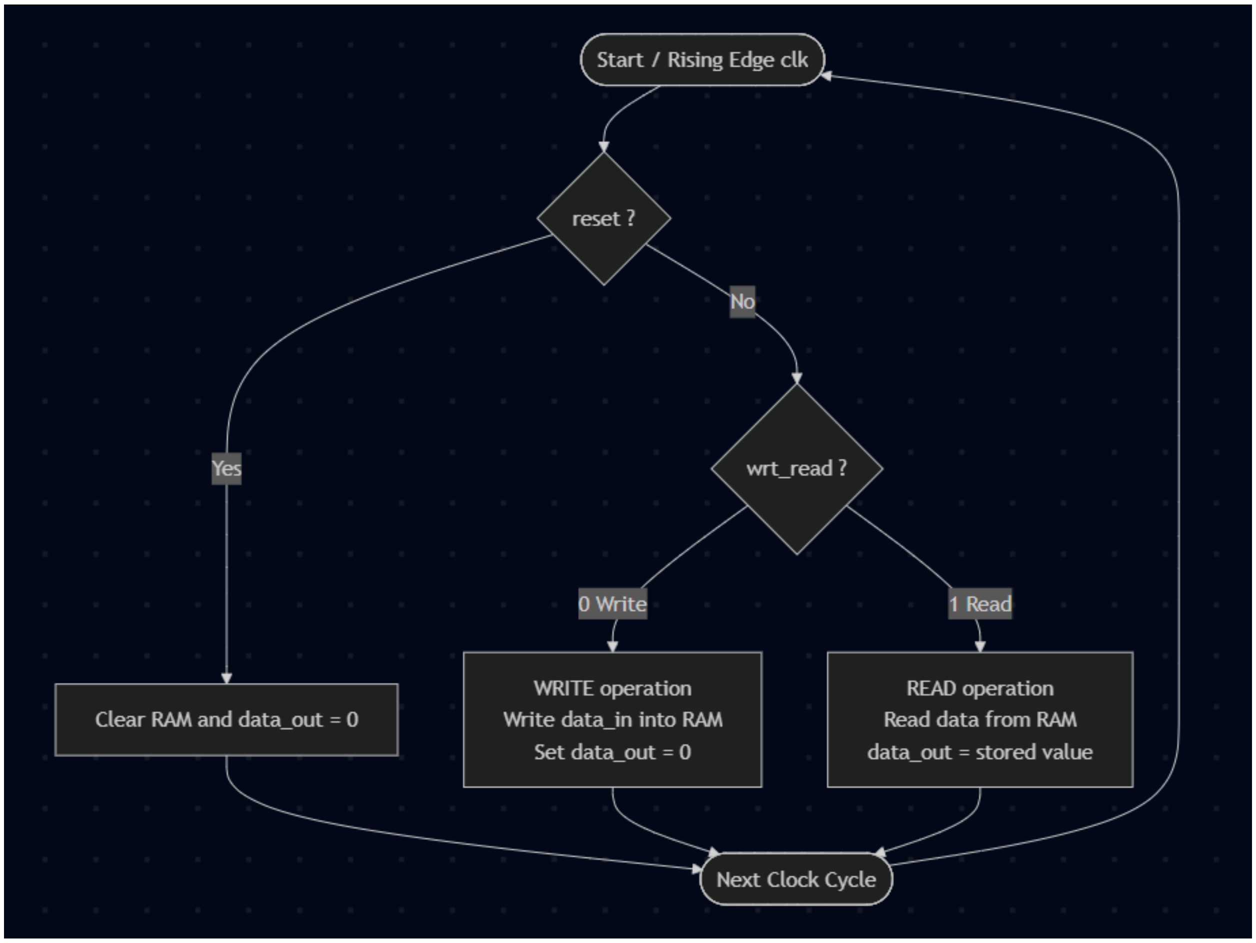
Consequently, the project functions not only as a practical exercise in digital design but also as a conceptual model of real-world memory architectures.

3. Design

Block diagram

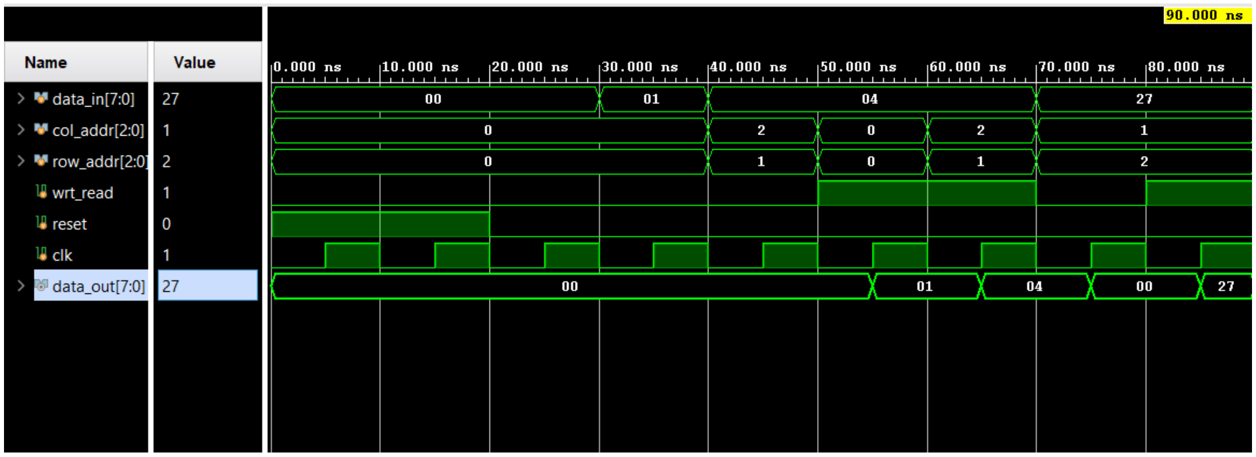


Flowchart


Start / Rising Edge clk
reset ?
No
Yes
wrt_read ?
0 Write
1 Read
Clear RAM and data_out = 0
WRITE operation
Write data_in into RAM
Set data_out = 0
READ operation
Read data from RAM
data_out = stored value
Next Clock Cycle

4. Result

After implementing design, the testbench result will be as shown in Figure.



5. Conclusion

a. Working Achievements

- Implemented a synchronous 8×8×8 memory (64 locations, 8-bit word) with row and column addressing.

- Defined a clean interface and protocol for write/read operations with deterministic timing.

- Single-cycle write and registered read path (one-cycle latency) with data_out forced to 0 when not reading.

- Behavior verified conceptually by write/read sequences, reset initialization, and address sweep in simulation testbenches.

- Synthesizable RTL that maps efficiently to LUTRAM/BRAM on mainstream FPGA toolchains.

b. Advantages

- Simple and modular architecture that is easy to explain, implement, and expand.

- Small resource footprint (512 bits of storage) suitable for buffers, scratchpads, and lookup tables.
- Deterministic, glitch-free outputs thanks to synchronous reset and registered read path.
- Portable coding style enabling inference of vendor memory primitives.
- Scalable to other array sizes and word widths with minor code changes.

c. Disadvantages

- Single-port access limits throughput; no concurrent read/write to different addresses in the same cycle.
- Limited capacity relative to data-intensive tasks; larger designs incur bigger decoders and multiplexers.
- Registered read adds a one-cycle latency compared with purely combinational reads.
- Baseline design lacks parity/ECC, byte-enable, and low-power features.

d. Future Works

- Implement a dual-port (or true dual-port) version to enable concurrent read/write on different addresses.
- Add byte-write enables and optional parity/ECC for reliability and fine-grained updates.
- Wrap the memory with a memory-mapped bus interface (e.g., APB/AXI-Lite) and support HEX/MEM initialization.
- Integrate Built-In Self-Test (BIST) and formal properties (assert/cover) for verification coverage.

• Parameterize rows, columns, word width, and read-mode; provide synthesis scripts and timing/power reports.

• Demonstrate a full application (e.g., 8×8 image block buffer or LED matrix controller) with measured throughput.

6. Reference

[1] M. Morris Mano and C. R. Kime, *Logic and Computer Design Fundamentals*, 5th ed. Pearson, 2015.

[2] David A. Patterson and John L. Hennessy, *Computer Organization and Design: The Hardware/Software Interface*, 5th ed. Morgan Kaufmann, 2013.

[3] William Stallings, *Computer Organization and Architecture: Designing for Performance*, 10th ed. Pearson, 2016.

[4] Pong P. Chu, *FPGA Prototyping by Verilog Examples: Xilinx Spartan-3 Version*. Wiley, 2008.

[5] Alan V. Oppenheim and Ronald W. Schafer, *Discrete-Time Signal Processing*, 3rd ed. Pearson, 2009.