4*4 MEMORY READ/WRITE

Special Assignment and Lab Project Report

Course: 2EC201CC23: Digital Logic Design

Submitted by:

23bec171
SAMARTH MODH

23bec153
RAHUL KUMAR AGARWAL

Department of Electronics and Communication Engineering,
Institute of Technology,
Nirma University,
Ahmedabad 382 481

[November-2024]

Abstract

All the computing systems consist of the Control Unit (CU), Arithmetic Logic Unit (ALU), Registers, Memory, and buses. The control unit is the brain of the computer architecture which decides what function need to be implemented on which part of information. The buses help in making a bridge between the desired memory locations and the ALU (where the desired operation is performed).

Read operation refers to that mode when the n-bit memory stored in any memory location is accessed to be observed by the person. It simply means fetching that information and presenting it the raw form to the user. Write operation means that the given piece of information is embedded or entered to any specific memory location. The old memory is erased and new information is updated.

Here, we intend to make 3 memory locations, each can store information 4-bit wide. Either the read or write operation is chosen by the user according to his will. The user can write the information to the memory location and can check whether the information is stored or not by the read operation.

Contents

1. Introduction

The 4*4 memory read/write operation is just updating the binary information stored in the memory location by checking what previously existed at that location.

Application

It has widespread application as it is the heart of the computer architecture. In any computer system, we need three things- one is which memory location is accessed, other is what information is stored over there and last one is which operation needs to be performed. The project aims to just store the information along with a provision to update it.

2. Literature Survey/State of the Art Technology Available

The evolution of the computer architecture has brought many changes in the scope of computing world but the very foundation has never changed. The dependence on the memory unit, ALU and CU has always been there and with technological advancements, the capability of all these parts have increased exponentially. Earlier, the Turing machines[3-4] (developed in 1936) were taking the strip of information in the sequence and updating the information by comparing it with the algorithm already fed. The inputs were given by the switches in the massive computers that came next.

Currently, the von Neumann architecture being used today is also the same at the fundamentals.

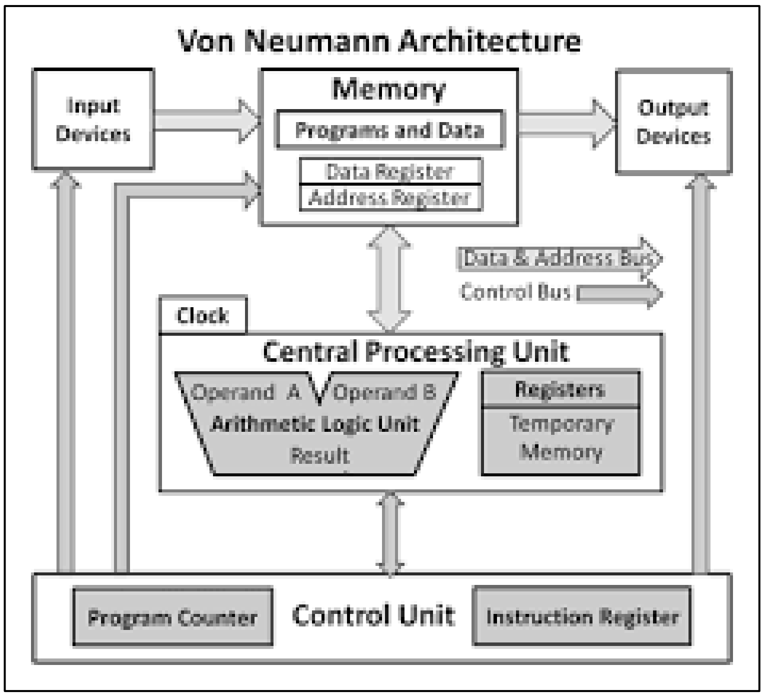


Figure 1: Von Neumann Architecture

3. Proposed Solution/Methodology

The project is based on the implementation of the same read/write operation that occurs in the computer system deep within at around 3 TOPS (Trillion operations per second). Here, we have used manual pulse generation for fulfilling the purpose of the clock.

4. Block Diagram & Circuit Diagram

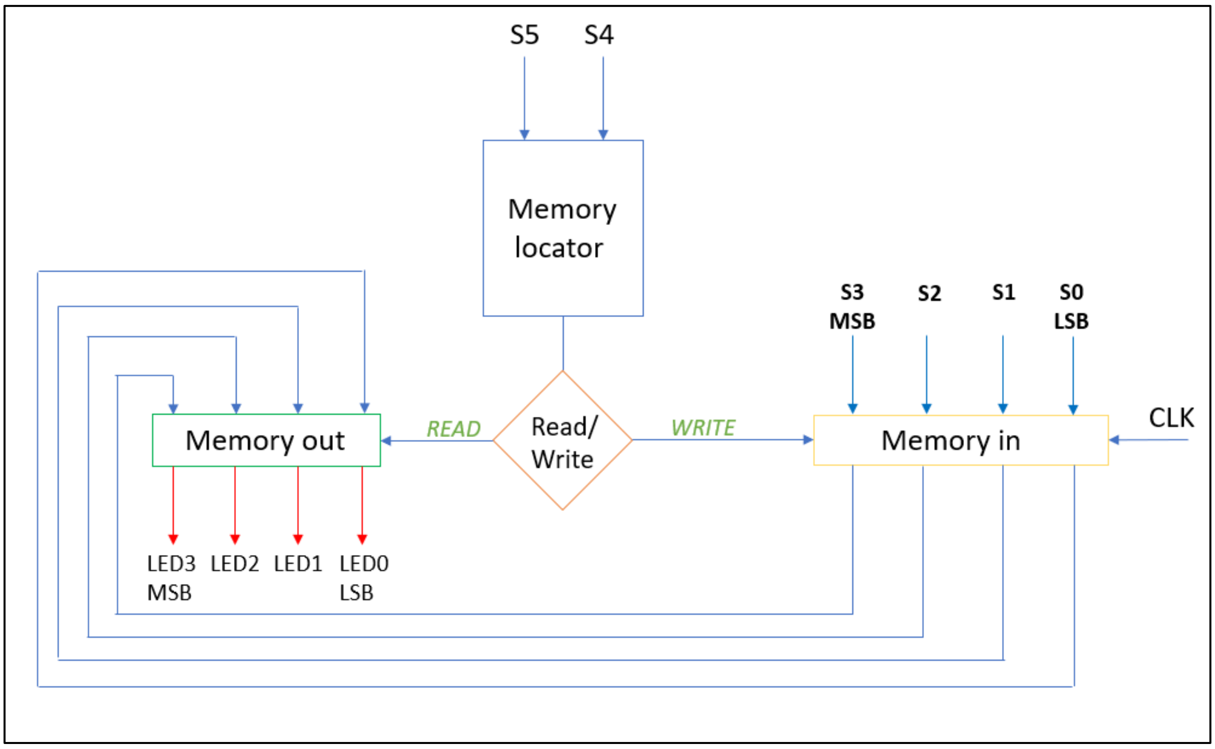


Figure 2: Block diagram

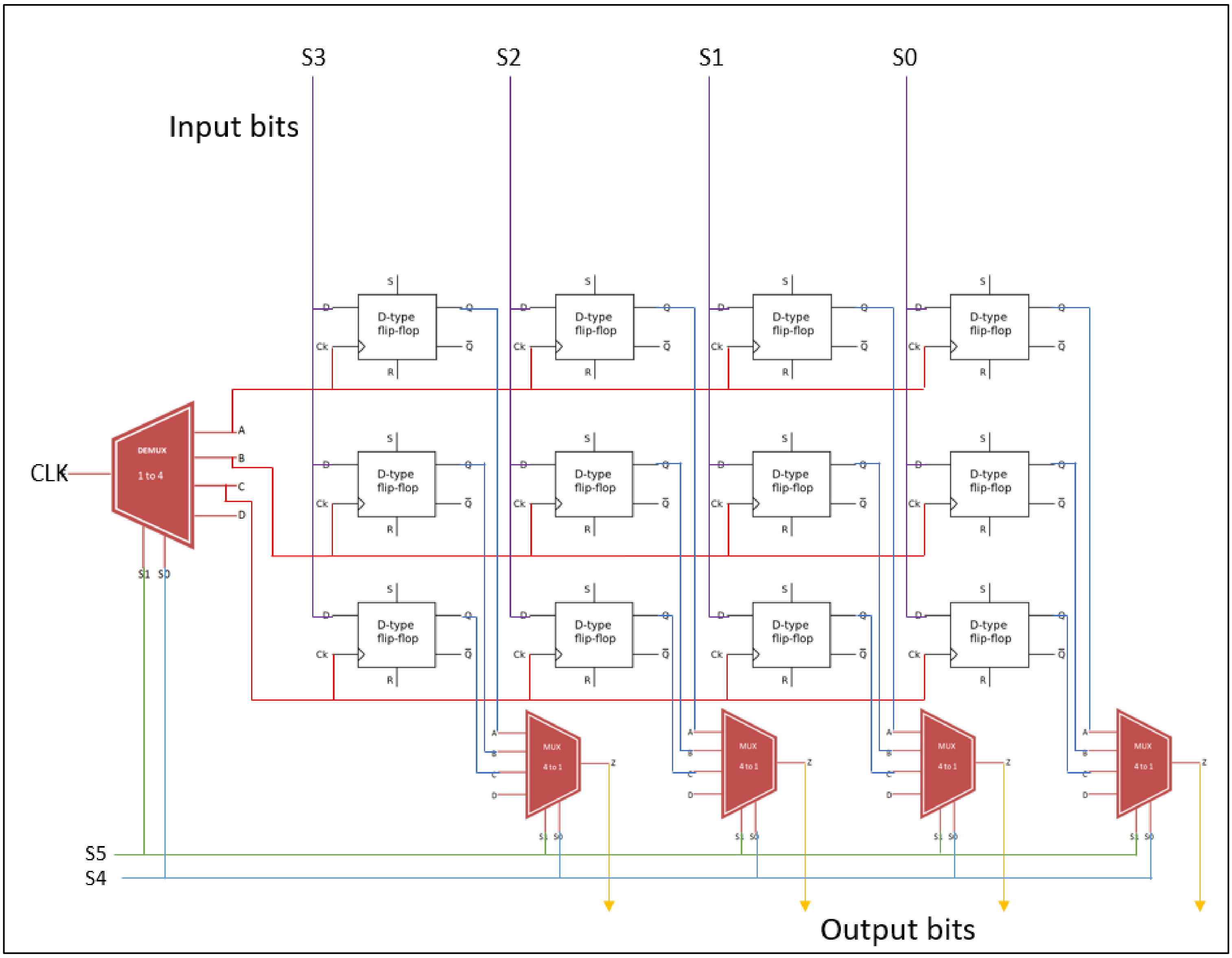


Figure 3: Circuit diagram

5. Design

5.1 Verilog implementation steps/flowchart

1) We classified the inputs and the outputs first.

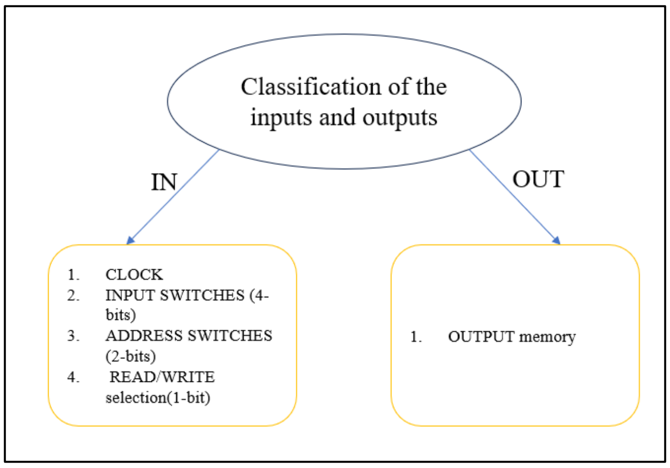

Classification of the inputs and outputs
IN
OUT
1. CLOCK
2. INPUT SWITCHES (4-bits)
3. ADDRESS SWITCHES (2-bits)
4. READ/WRITE selection(1-bit)
1. OUTPUT memory

Figure 4: classification of input and outputs for the module

2) Next step is related to the identification of an array to store the input information. Thus, we have declared a register array (4 by 4) in size where the rows are the different memory locations while the first column represents the MSB of all the memory locations and the last or the 4th column represent the LSB.

3) The next step deals with the always block as we are using the behavioural modelling. We are using the positive edge triggering. We used the if-else statements for declaring the operations under read and write category→

   a) When the rw is 1, we describe the logic for the write operation i.e. the array of memory stores the information at the positive edge of clock for different addresses.

   b) When the rw is 0, the read operation occurs wherein the output is declared to be the memories fed in till the last write positive highs.

5.2 Circuit design

We did not require any truth table. We decided to use the D flip flop because of the conveniency that the D flip-flop offers. This is depicted in the figure 5.

Q	D	Q(t+1)
0	0	0
0	1	1
1	0	0
1	1	1

Figure 5: characteristic table D flip-flop

6. Details of All Hardware Components Used and Its Justification

List of all the hardware used→

1. 74175 quad D flip flop ICs (total 3 used)
2. 74139 1*4 DeMux/Decoder IC (1 used)
3. 74153 4*1 Mux IC (total 4 used)
4. 7408 quad AND gate (1 used)
5. Push button pulse generator
6. Jumpers and hook-up wires.
7. Toggle Switches (total 6 used)
8. LEDs (total 4 used)
9. Breadboards (3 used)

On the hardware, we have used 16 D flipflops (3 ICs each having 4 flipflops; each IC act as a particular memory location) for remembering the information given during the write mode. The DeMux helps in choosing the specific IC i.e. a specific memory location. The switches S3, S2, S1, S0 are used to give the inputs. The switch S5, S4 are given to the selection lines of the DeMux. The read/write operation are toggled by toggling the switch S6. The LEDs show the output.

Cost

Each IC costed around rupees 20-35.

Total cost of ICs is rupees 310. Total cost of breadboards rupees 140.

Total cost of project is rupees 520 (450+70). Here,70 rupees is the cost of the spare ICs and other components which were purchased for demonstration at home.

7. Simulation Results

7.1 Modelsim Simulation

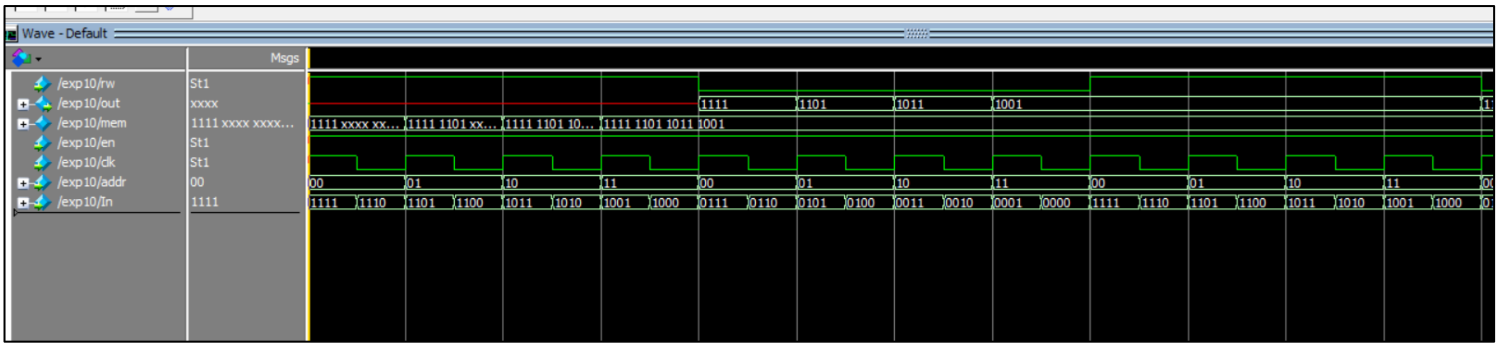


Figure 4: Required waveforms on Multisim

Only when the en (enable) is on, the read and write operations are initiated. When the rw is 1, the write operation is happening and since there is no initial block, the output is also indeterminate. During this time, the information 1111, 1100,1101,1100 are respectively written over addresses 00,01,10,11. During the read operation, the same inputs are reflected over respective addresses.

7.2 Circuit Simulation

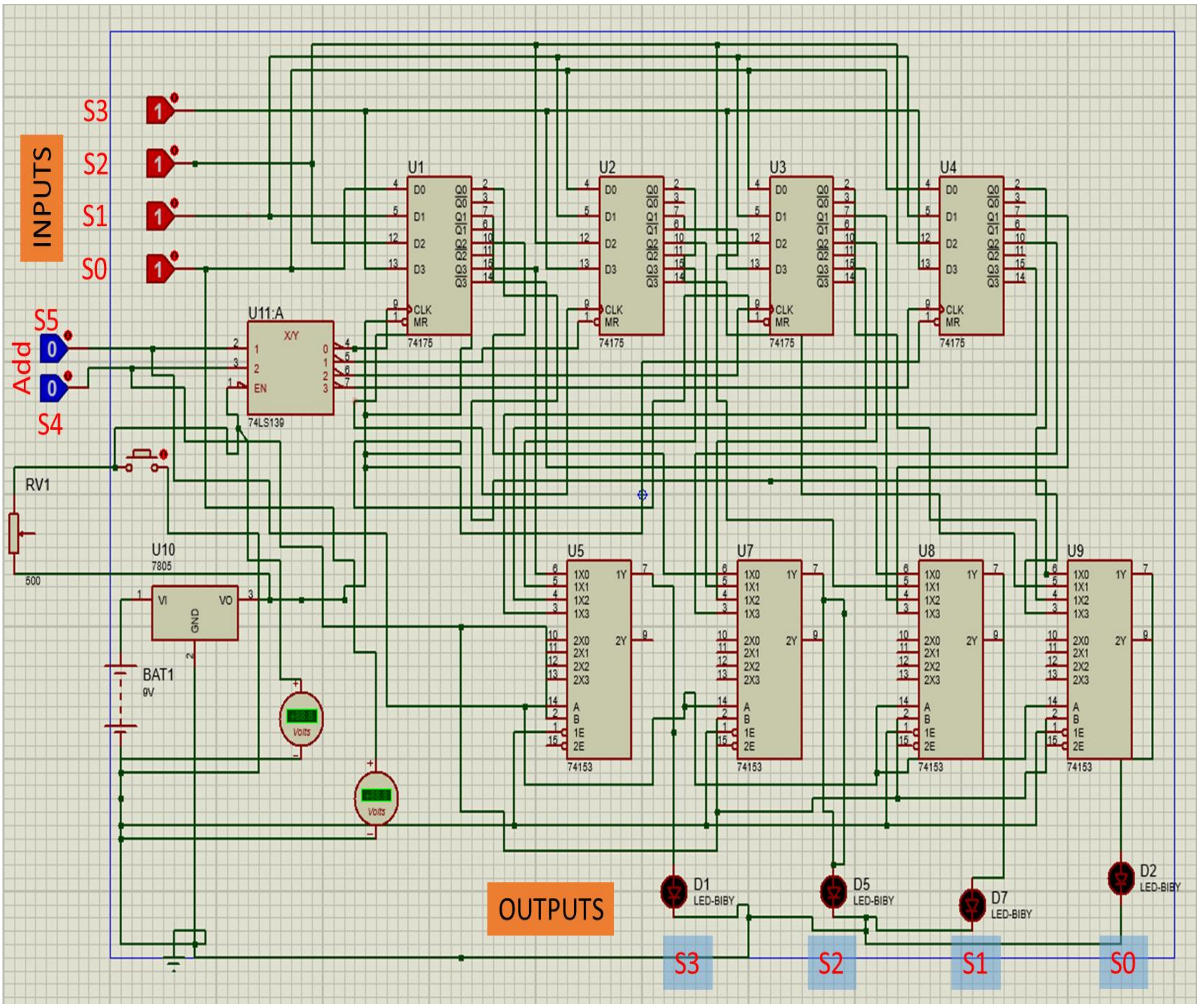


Figure 5: CIRCUIT DIAGRAM ON PROTEUS

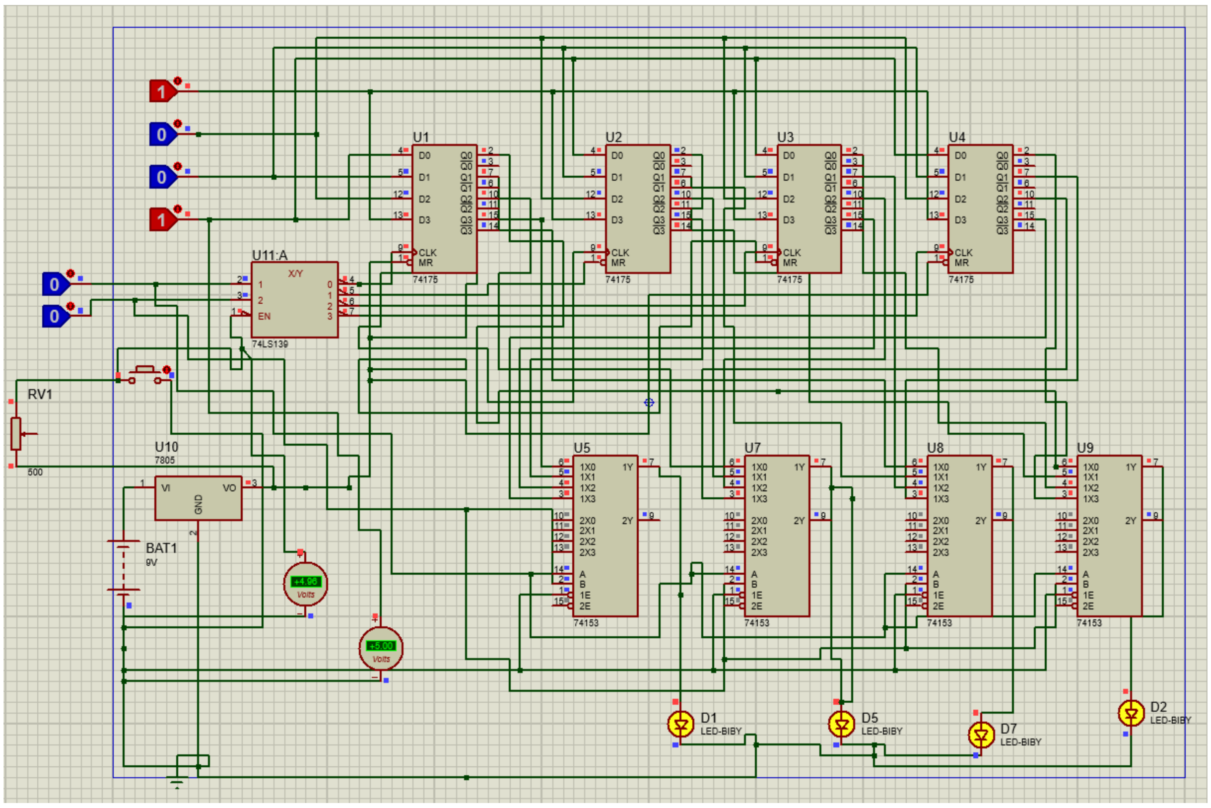


Figure 5A: For memory location 00

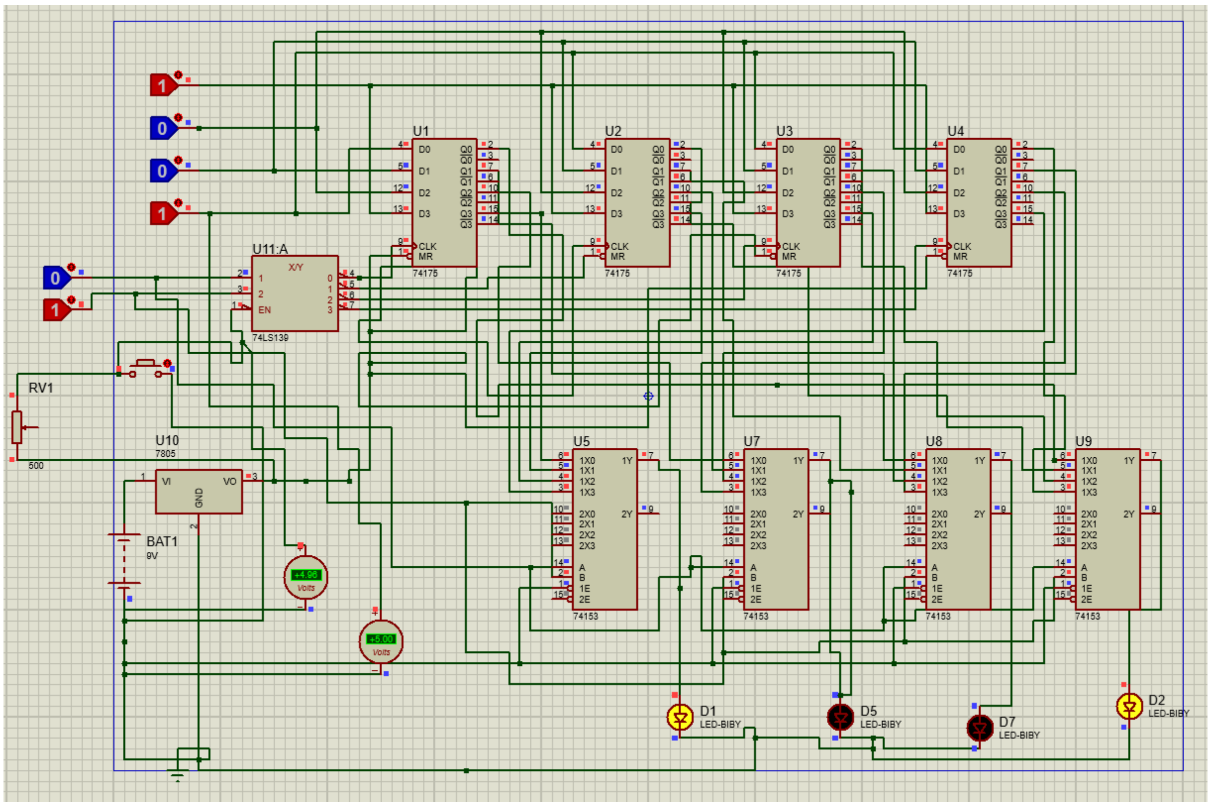


Figure 5B: For memory location 01

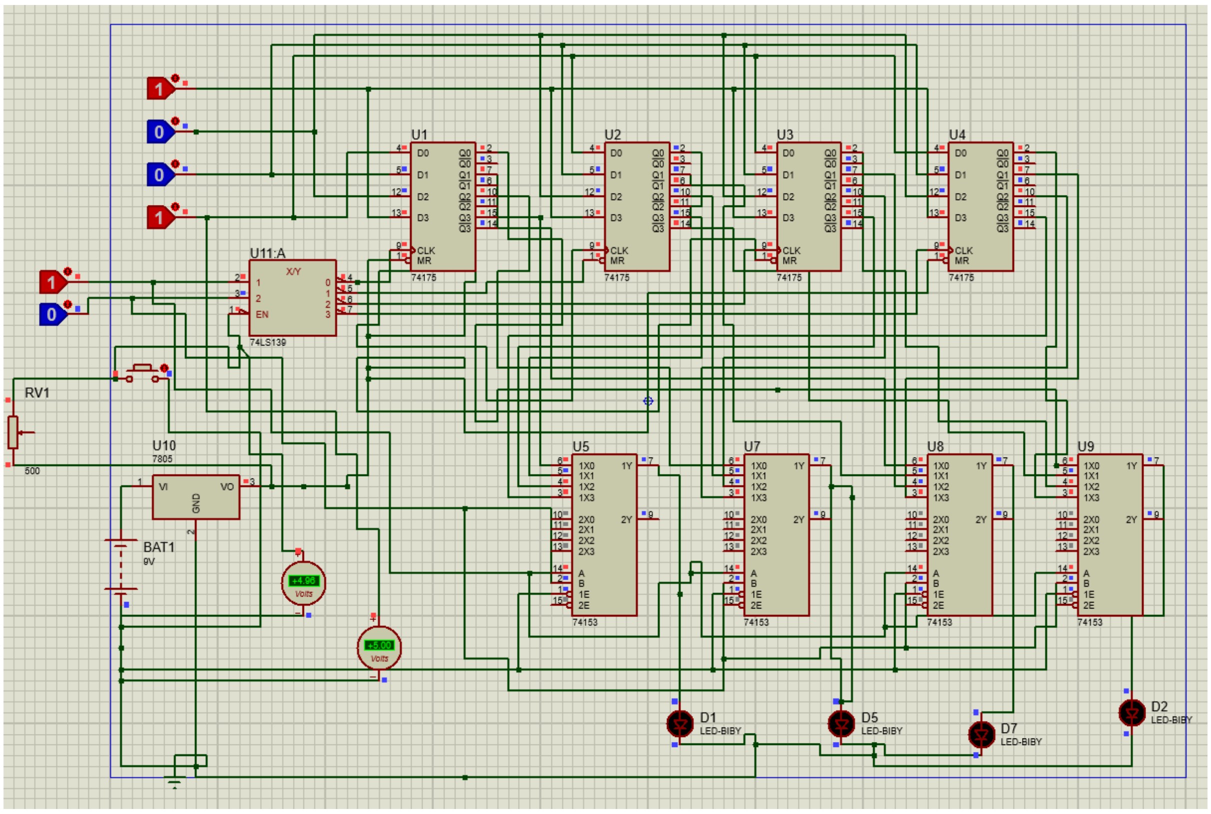


Figure 5C: For memory location 10

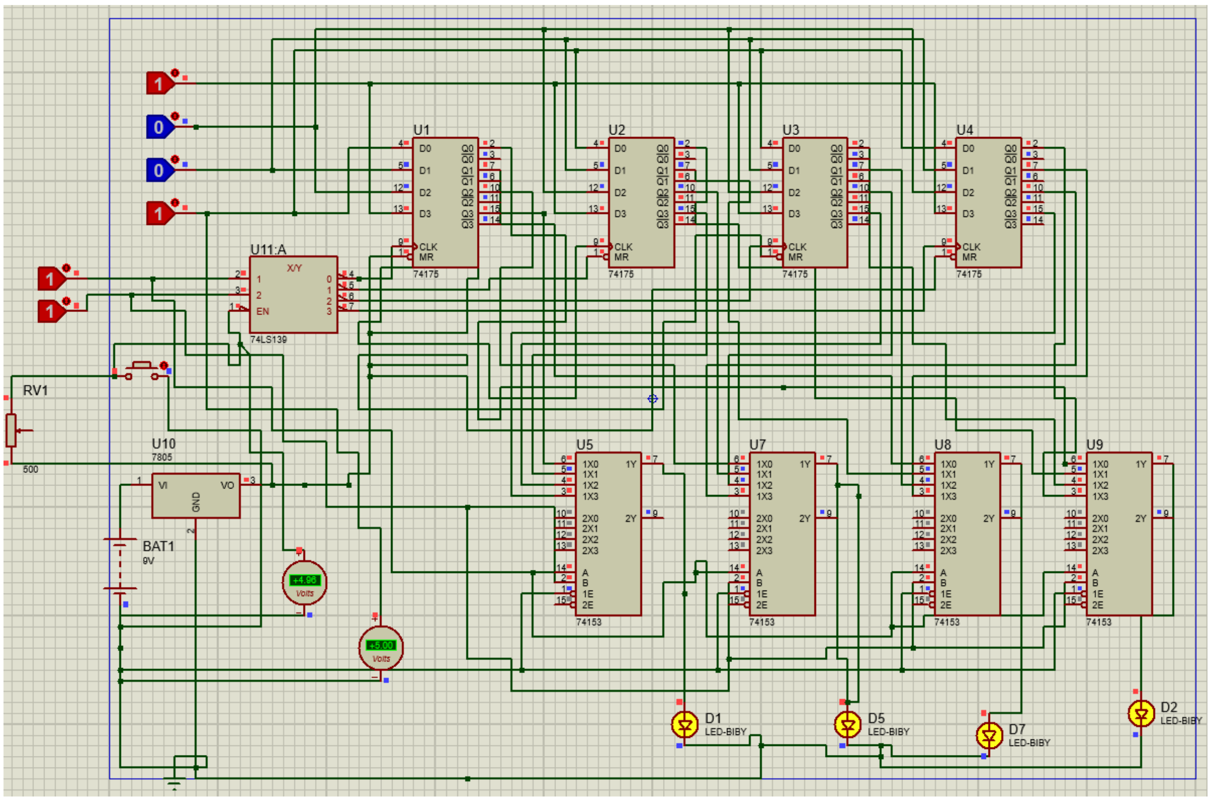


Figure 5D: For memory location 11

8. Results and Measurements

- **RTL View :**

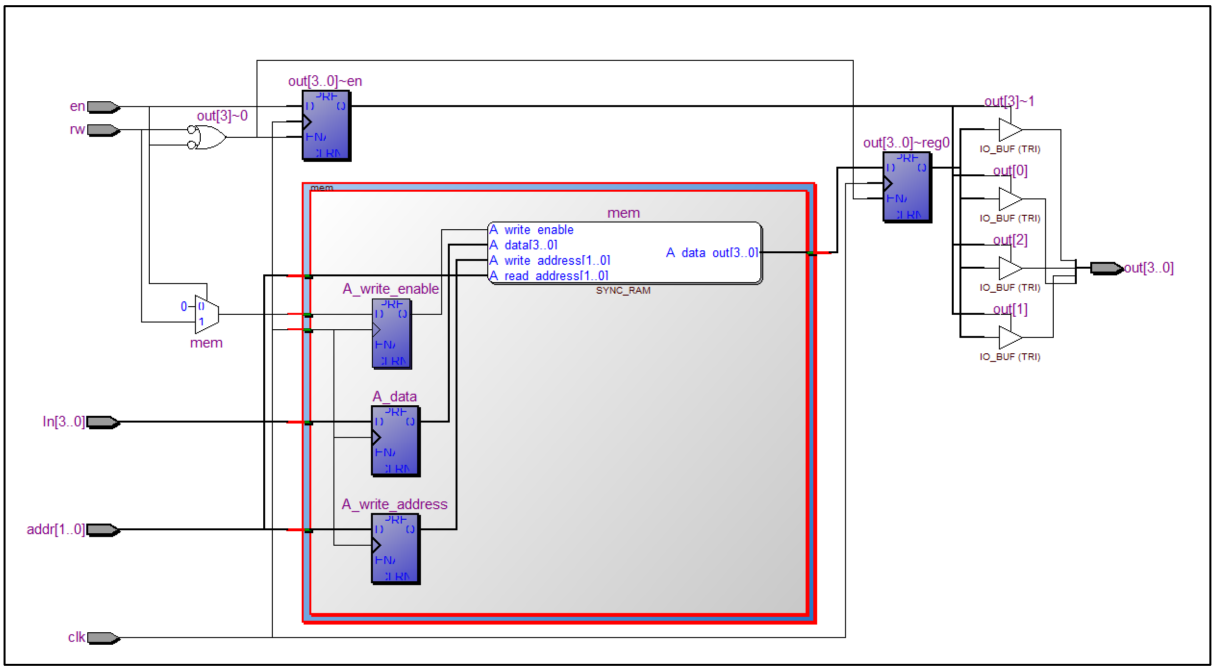


Figure 6: RTL View

All the things are already explained, few things newin this RTL View are as follows→

- If en is 1, only then the mem submodule array works to store the inputs and the read operation follows the same.
- A_write_enable, A_data, and A_write_address: These are signals or latches that buffer the write enable, input data, and write address before feeding them into the main memory. This buffering ensures that only the intended data and control signals reach the memory during write operations.
- These buffers are enabled during read operations and put in high-impedance state in the write state, allowing the output bus to be controlled depending on the rw and en signals.

- **Technology Map View (Post Mapping):**

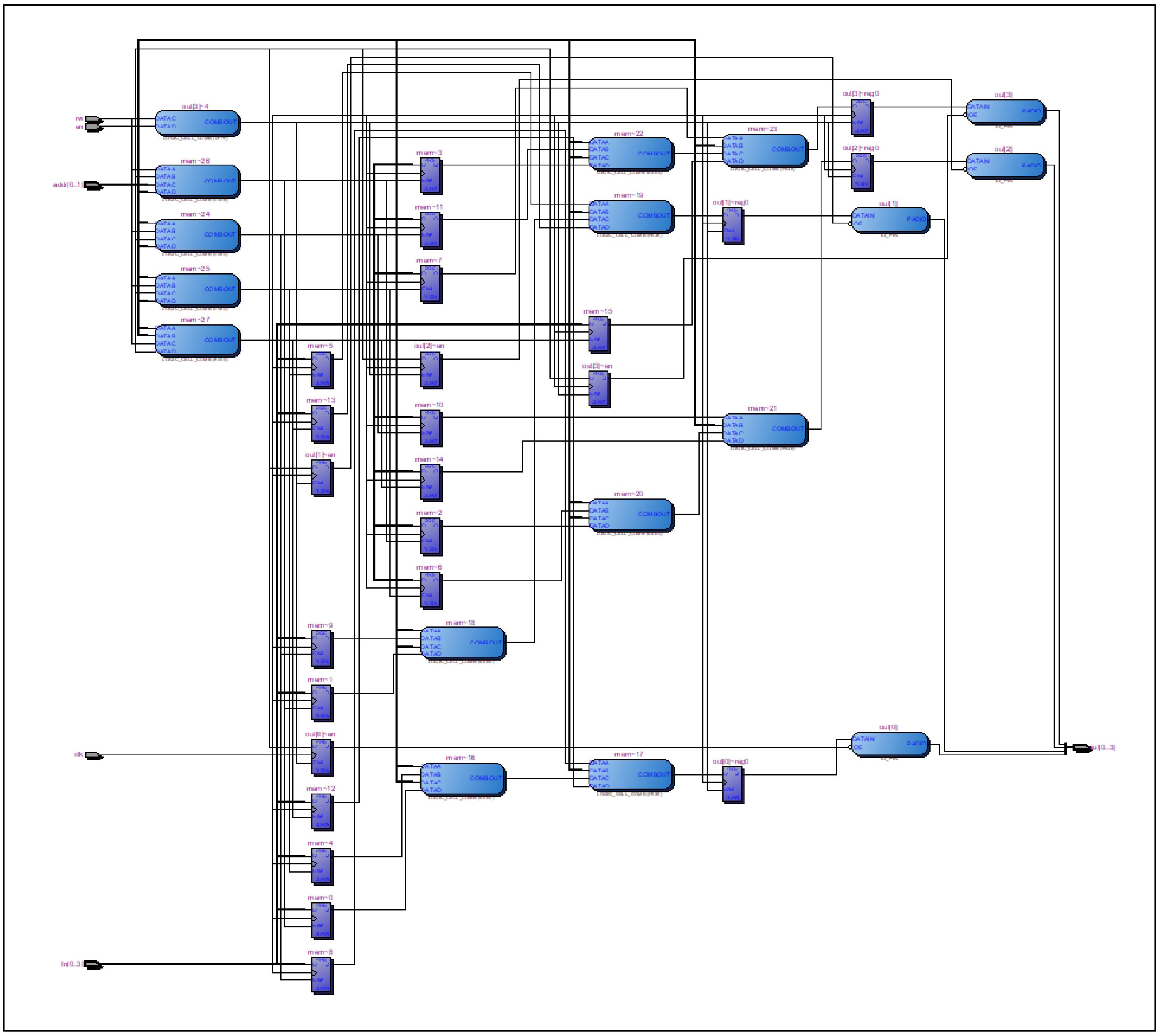


Figure 6: Technology Map View (Post Mapping)

9. Conclusion, Learning Outcome, and Future Scope

- **Conclusion**:

  We tried to implement the read and write operation which is soul of the computing world. It is very important to simulate all the problems on softwares like Proteus, TinkerCAD, etc. before jumping onto the physical hardware.

- **Learning Outcome**:

    i) We learnt that the flip-flops (especially the D flip-flop) are very important digital components used for the memory allocation and updation.
    ii) The other things learnt are related to use of multiplexers and demultiplexers for the purpose of selecting a specific memory.
    iii) Importantly, we understood the importance of clock pulse in putting a particular information into the memory.
    iv) The Verilog implementation portion is significant. We don't need to make the modules for the functionality of each of the IC and instantiate it. Instead, we directly devised a logic and developed the code accordingly. RTL View and other Netlist view shows the use of the hardware components that we used on the breadboard.

- **Future Scope**:

    i) Cascading more memory locations and input switches can help in storing more memories.
    ii) The next challenge could be how to integrate the ALU i.e. providing the input regarding the operation that we need to perform. To illustrate, let's say that we intend to perform the summing operation on the information present at the 2nd and 4th memory locations. It can involve providing other operations like binary subtraction, logical operations like XOR, NAND, etc.

10. References

[1] Chapter -6 Synchronous Sequential Logic from *'Digital Logic And Computer Design By M. Morris Mano (2nd Edition)'* Pearson Publications

[2] A. Kumar, *Fundamentals of Logic Design*. New Delhi, India: Prentice-Hall, 2005.

[3] A brief information about the Turing machine by geeksforgeeks
https://www.geeksforgeeks.org/turing-machine-in-toc/

[4] 'How Turing Machines Work' by BitMerge
https://youtu.be/cDc6Gfo3egk?si=qRrdnp_Aap7LbIit

[5] 'Von Neumann architecture' by BBC
https://www.bbc.co.uk/bitesize/guides/zhppfcw/revision/3

Digital Logic And Computer Design By M. Morris Mano (2nd Edition)

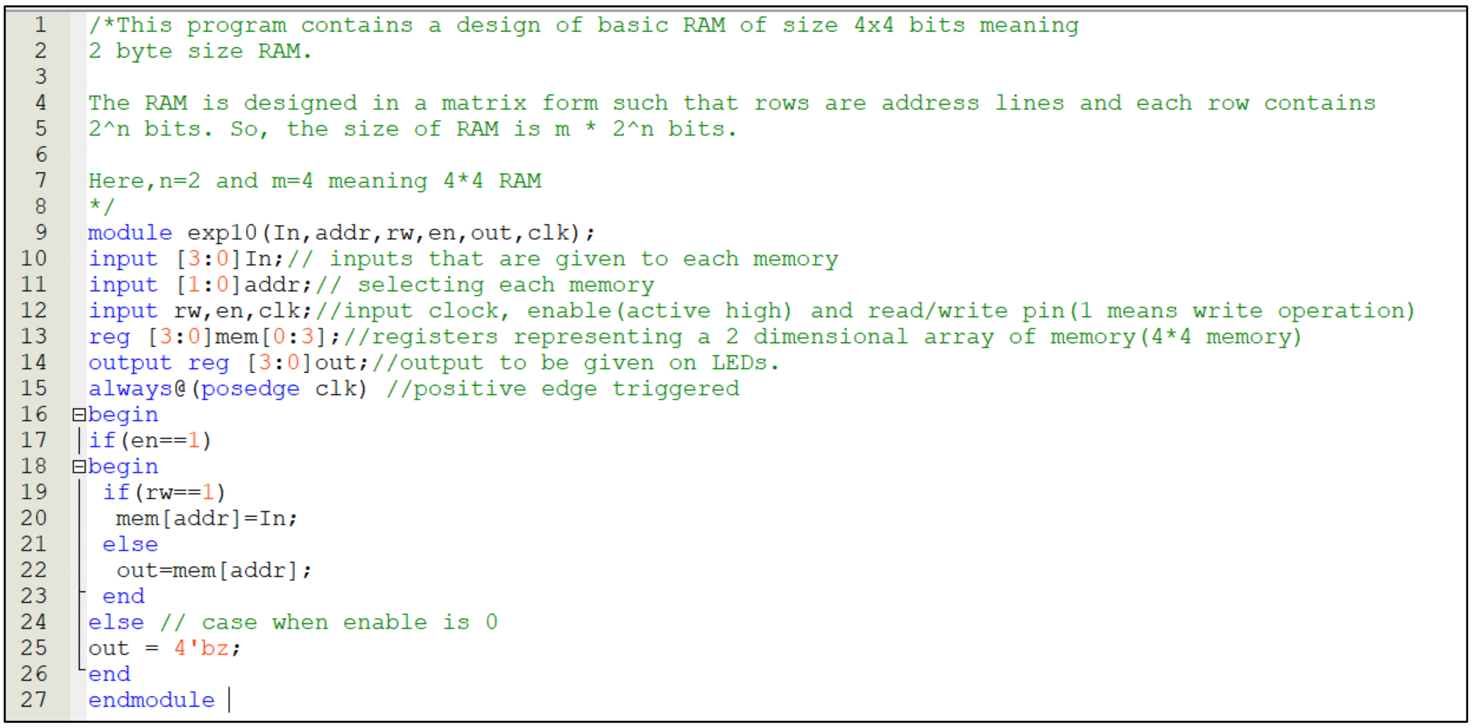
11. Appendix

Code:

```
/*This program contains a design of basic RAM of size 4x4 bits meaning
2 byte size RAM.

The RAM is designed in a matrix form such that rows are address lines and each
row contains
2^n bits. So, the size of RAM is m * 2^n bits.

Here,n=2 and m=4 meaning 4*4 RAM
*/
module exp10(In,addr,rw,en,out,clk);
input [3:0]In;// inputs that are given to each memory
input [1:0]addr;// selecting each memory
input rw,en,clk;//input clock, enable(active high) and read/write pin(1 means
write operation)
reg [3:0]mem[0:3];//register representing a 2 dimensional array of memory(4*4
memory)
output reg [3:0]out;//output to be given on LEDs.
always@(posedge clk) //positive edge triggered
begin
if(en==1)
begin
 if(rw==1)
  mem[addr]=In;
 else
  out=mem[addr];
 end
else // case when enable is 0
out = 4'bz;
end
endmodule
```



```
/*This program contains a design of basic RAM of size 4x4 bits meaning
2 byte size RAM.

The RAM is designed in a matrix form such that rows are address lines and each row contains
2^n bits. So, the size of RAM is m * 2^n bits.

Here,n=2 and m=4 meaning 4*4 RAM
*/
module exp10(In,addr,rw,en,out,clk);
input [3:0]In;// inputs that are given to each memory
input [1:0]addr;// selecting each memory
input rw,en,clk;//input clock, enable(active high) and read/write pin(1 means write operation)
reg [3:0]mem[0:3];//registers representing a 2 dimensional array of memory(4*4 memory)
output reg [3:0]out;//output to be given on LEDs.
always@(posedge clk) //positive edge triggered
begin
if(en==1)
begin
 if(rw==1)
  mem[addr]=In;
 else
  out=mem[addr];
 end
else // case when enable is 0
out = 4'bz;
end
endmodule
```

Figure 7: Verilog implementation code