EE224 Project: Mess Food Management System

Advaith Suresh (200260004), Anurag Abhijit Pendse (200260008), Lokesh Mishra (200260026), Mehul Vijay Chanda (200260029), Niare Doyom (200260032), Samyak Jain (200260046)

Spring 2022

Abstract

The current system of food distribution in hostel messes is inefficient and has several points of direct human involvement causing it to be susceptible to inaccuracies. Using the digital circuits that have been taught in the course EE224 Digital Systems, we propose a *Mess Food Management System* (MFMS) using which we aim at automating a large part of the food distribution process and reduce the above mentioned inefficiencies and inaccuracies.

---

1 Current System of Food Distribution

The current system of food distribution involves the use of mess cards. The following process if followed:

a. A student enters the mess and picks their mess card out of a rack.

b. They go to a counter where they submit their mess card and collect a plate and a spoon after verification by the mess staff present. This ensures that the same student does not take food twice and also that they belong to the mess' hostel.

c. They get food and eat it.

d. The student submits their plate and spoon.

e. Once the meal timings are over, the mess staff put the mess cards back into the racks.

f. This process is repeated for each of the 4 meals (breakfast, lunch, evening snacks and dinner).

The current system has several drawbacks. We list them down below and also mention how the MFMS we propose will resolve them:

a. Verification of the mess cards by the mess staff is slow and results in crowding at the plate counter. Our system will utilize a bar code scanner which will speed up the process greatly.

b. Misplacement of cards, putting cards into the wrong racks are common issues which result in extra work for both the mess staff and students. In MFMS, there will be no concept of mess cards and all the data will be stored digitally.

c. Before the start of every semester, students pay a mess advance fee for the food they eat in the mess. Many times students skip meals in the mess. Ideally, they are supposed to get a rebate for this. But due to poor management of this data, they do not and end up paying extra money in the mess advance. In MFMS, we will store the information regarding the meals availed. This will allow the students to know what rebate can they get.

Thus, to summarize the current system, it uses physical mess cards to check whether the student is from the hostel and has not availed the meal before on that day. All processes involving the mess cards are manual.

2 Overview of the Logic

The following flowcharts explain the basic logic of the system designed.

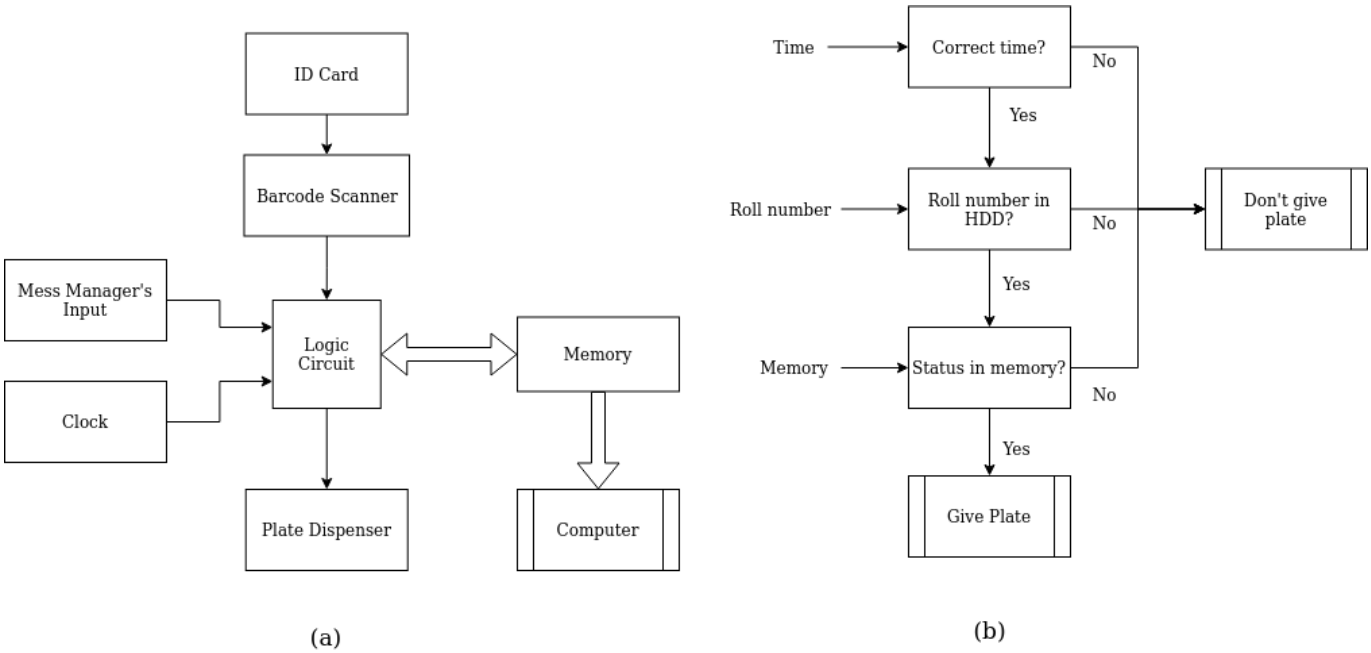


Figure 1: Figure (a) shows a flowchart representing the overall working of the system with its various components. Figure (b) shows the basic logic being followed by the logic circuit present in the system.

3 Description of Components

Below we describe each of the boxes in the flowcharts above.

a. **The ID card:**
Each student possesses an institute ID card which has a bar-code. The bar-code can be scanned to procure the student's roll number.

Figure 2: Representation of a bar-code on an ID card

b. **Bar-code Scanner:**
A bar-code scanner scans the ID card and procures the roll number. The roll number might be obtained in decimal digits or binary digits depending on scanner's design. We will assume that the scanner returns a decimal valued number of 9 digits (eg. 200260029_{10}). We then convert this decimal number to a binary number. We will make use of 29 bits. Using 29 bits, we can represent decimal numbers from 000000000_{10} to 536870911_{10}.

c. **Clock:**
Clock is a simple clock which stores time in $H_1H_2 : M_1M_2$ format. H_1, H_2, M_1 and M_2 can be stored in a digital form. H_1H_2 goes from 00 to 24 and therefore, we make use of 5 bits for storing it. M_1M_2 goes from 00 to 59 and therefore, we make use of 6 bits for storing it.

d. **Mess Manager's Input:**
The mess manager can manually input the start time (t_s) and end time (t_e) of the mess - between t_s and t_e students can take food from the mess. The time inputs are given in the same format as what is stored in the clock i.e. $H_1H_2 : M_1M_2$. The mess manager will also send a message indicating whether the meal being served is breakfast, lunch, evening snacks or dinner.

Meal	Code
Breakfast	00
Lunch	01
Evening snacks	10
Dinner	11

Therefore, the manager's input contains a total of 24 bits.

e. **Logic circuit:**
The logic circuit determines whether the student is eligible to avail a meal at the hostel. Following are the inputs given to the circuit:

i. 1 or 0 depending on whether the student's roll number is present in the hostel's database.

ii. The start and end times of the mess in the format specified in previous sections.

iii. The current time t obtained from the clock.

The input obtained in part (a) will act as a sort of $ENABLE$ bit for the circuit. The times will be compared and if $t_s < t < t_e$, where t is the current time, then the final output will simply be 1 if enable is 1 and 0 if it is 0. If current time is not between the start and end times, then the output will be 0.

f. **Memory:**
All of the data that is supposed to be stored for a long time, like roll numbers and meals, will be stored on a hard disk (HDD). The primary memory of MFMS will be a static random access memory (SRAM). The data in the SRAM will be accessed using binary addresses.

g. **Plate Dispenser:**
Plate dispenser is a simple machine which provides plates and spoons if it gets the input 1 and does not when it gets the input 0.

h. **Computer:**
The computer is used to access the data stored in the HDD. The mess manager or any of the students can access the computer's directories to retrieve and even rewrite the information in the HDD, if at all needed.

4 Working of MFMS

a. The system components can be majorly divided into two parts - *dormant* and *active* circuits. The active circuits, as the name suggests, are always active i.e. powered on. The bar-code scanner circuit along with the mess manager's input terminal is always active. The computer, the SRAM, and most of the other logical circuitry is dormant. They are powered on only when their functionality is needed.

b. Before every meal and after the end of the previous meal, the mess manager inputs the start time and the end time of the next meal along with the meal code.

c. If anyone tries scanning their ID card with the bar-code scanner, the bar-code circuitry will first send a signal to the logical circuitry to check if the meal duration is on. If no, the scanner rejects the ID card and flashes a red light, indicating that meal is being denied. If the meal duration is on, the bar-code scanner procures the roll number and passes it on the further circuitry.

d. We explain the working of the logical circuitry in detail below:

- We firstly wish to find the truth value of the statement $(t_s \leq t)\ \ AND\ \ (t_e \geq t)$. This logic is implemented through the circuit shown in 5.
- Here, the 'Actual Hour' and 'Actual Minute' are the H_1H_2 and M_1M_2 components of the current time. The 'Start Hour' and 'Start Minute' are the H_1H_2 and M_1M_2 components of the start time. The 'End Hour' and 'End Minut' are defined similarly.
- The output D1 is high if the first input (the upper 5 or 6 lines) is greater than the second input, D2 is high if they are equal and D3 is high if the second input is larger. In each pair, the M_1M_2 components are compared only if the H_1H_2 components are equal.
- The first and third blocks have 10 input lines (5 internal 1-bit comparators) while the second and fourth block have 12 input lines (6 internal comparators). The H_1H_2 comparison blocks have a external enable line which will allow us to stop the comparison externally as well.
- The blocks used are N bit $(N = 5, 6)$ comparators which have the following internal circuit. The circuit shown in 3 is stacked N times to get an N bit comparator. The enable input for the MSB's comes externally.
- It can be seen that in the comparator, the subsequent bit is compared only if it cannot be found with certainty the relation between the two inputs after looking at the previous bits (This essentially reduces the net number of operations the circuit needs to perform to obtain the output).
- This logic is also used in comparing the start, current and end times. If we can determine simply by looking at the H_1H_2 components of the times we wish to compare, then we do not need to compare the M_1M_2 components at all.
- The output of these N bit comparators can be obtained from 3 separate output lines which become active if the first input is greater than, equal to or less than the second input.
- In 4, the outputs from the 3 lines from each comparator are sent to 3 N-bit OR gates. The output of those operations will be the final output of the N-bit comparator.
- In the end, the enable bit (talked about at the beginning of this section) and the output of this comparison are sent through an AND gate and the output is sent to the plate dispenser.

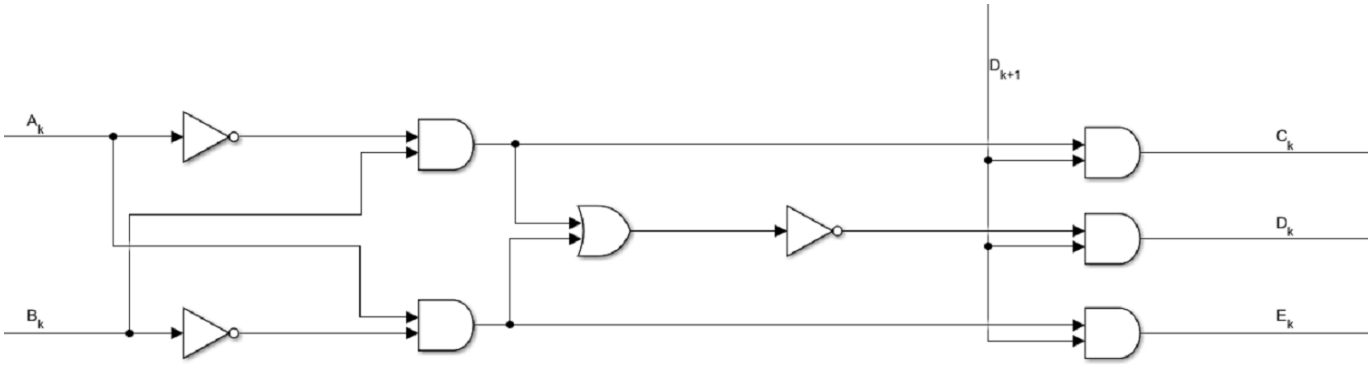


Figure 3: One bit comparator with external $ENABLE$ bit

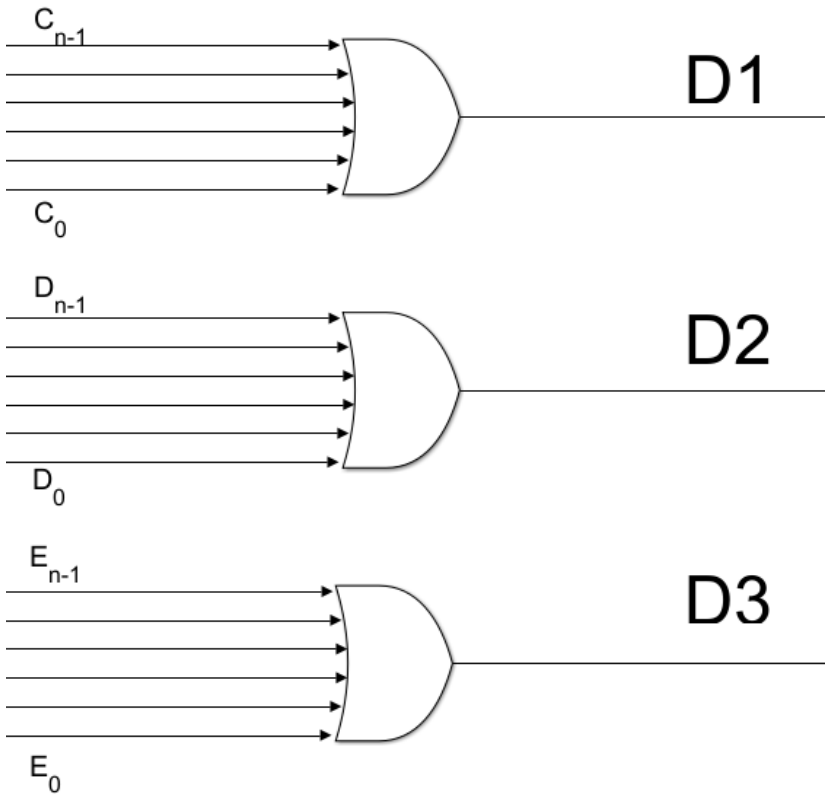


Figure 4: The final stage of the comparator circuit

5 Future Improvements

There are several aspects of the MFMS which can be improved:

a. The times at which each meal is availed by the student can also be stored in the memory to help the mess manager know when the most crowding occurs.

b. The restriction of the hostel in which the meal is availed by the students can be removed and the data about the hostel in which a particular meal is availed can be stored and the mess advance paid by the student can be appropriately distributed in between the hostels in which the meals were availed.

c. The data on the extra food items availed by the students can also be stored and the final cost of those can be removed from the semester mess advance rather than the students having to crowd near the counter to pay each time they wish to avail the extras.

d. Since the student's data is stored after obtaining the information by scanning the bar-code on the ID card, the same process can be extended to form a central system amongst all the eateries (including the messes) present on campus. Every time a student avails a particular meal from some eatery, they will scan their ID cards at the eatery and the cost of the meal availed will be added to the net bill which will be paid at the end of the semester. This will reduce the hassle at the eateries and it will save time for both the students and the owners of the eateries.

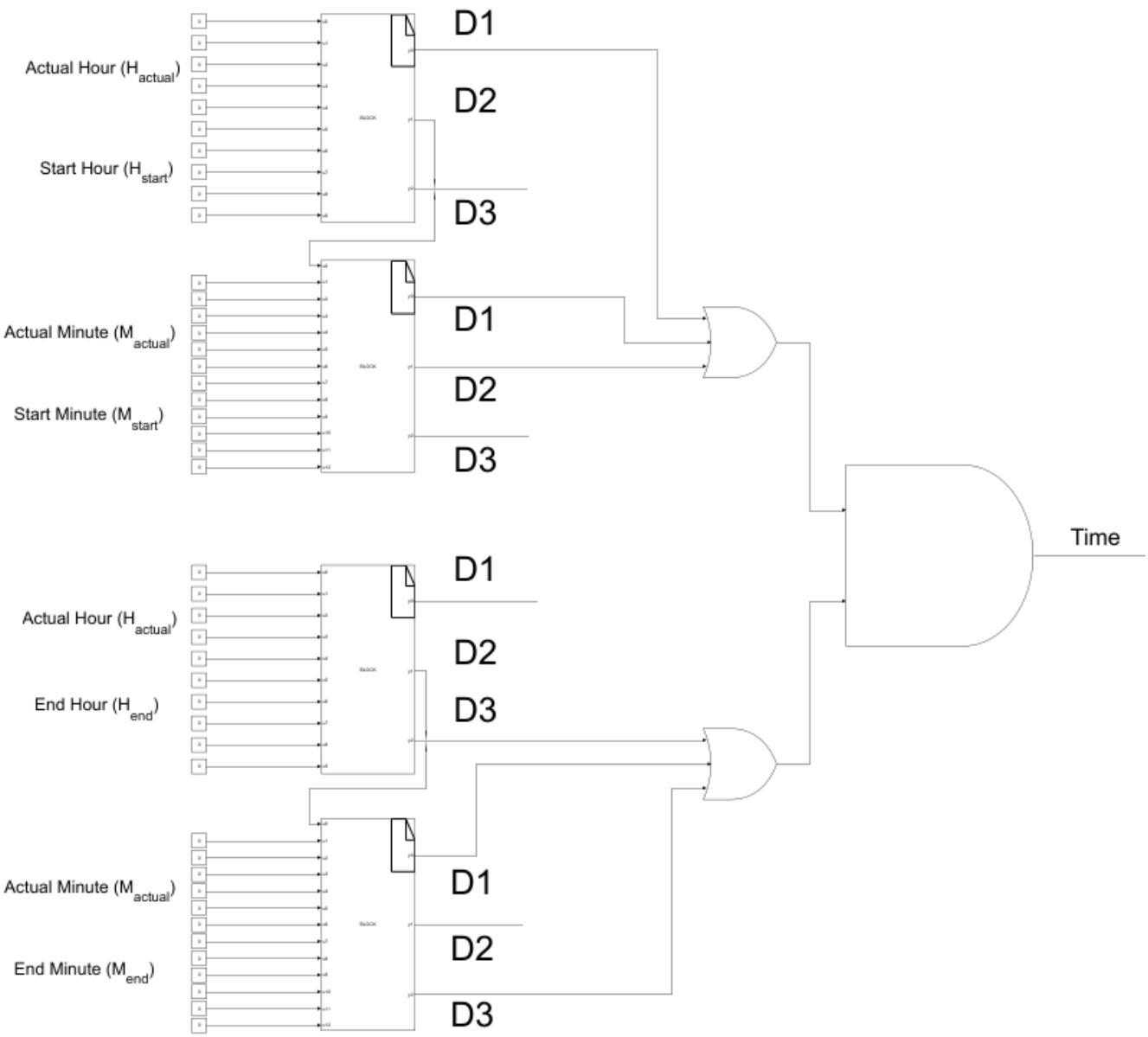


Figure 5: Circuit determining whether current time lies in the mess duration

- **Comparators:**
  Comparators are circuits that compare two numbers to check which one is larger. The simplest comparator circuit compares two 1-bit numbers. The output is given on three lines. We can see the circuit of a comparator in 6.

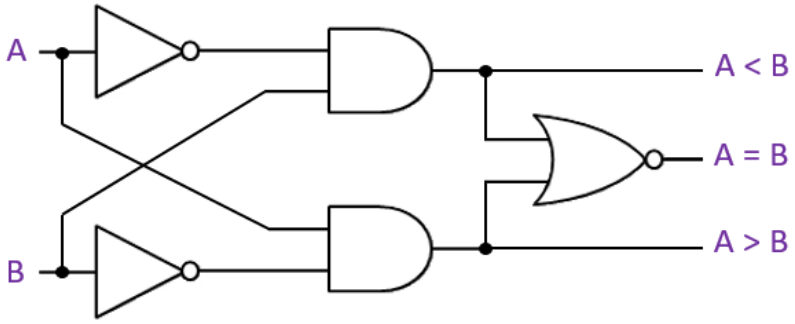


Figure 6: 1-bit Comparator

We can see that there are 3 output lines, one for each of the 3 cases.These one bit comparators can be stacked to make N-bit comparators.

- **RAM:**
  Primary digital memory can be largely divided into two types:
  - Read Only Memory (ROM)
  - Random Access Memory (RAM) or Read-Write Memory

As has been seen above, we are using only RAM in MFMS and therefore will briefly discuss it.
RAM is a *temporary memory* that helps the system with it's processes while it is running. It requires power to function and loses all its contents whenever power is turned off. Hence it is called *volatile*. RAM is composed of multitudes of data storing units called *cells*, which individually store a bit. We can access the state of a cell directly if we know it's *address*. RAM has both read and write options. 7b is a block diagram depicting a RAM.

There are address lines which are used to access a cell, and data lines which gives the value contained in the cell. If we have K address lines and N data lines, then the size of the RAM is $2^K \times N$ bits. $READ/\overline{WRITE}$ line is used to specify if we want to read or write. The CE (Chip Enable) line activates or de-activates the RAM.
The two main types of RAM are SRAM (Static RAM) and DRAM (Dynamic RAM) -

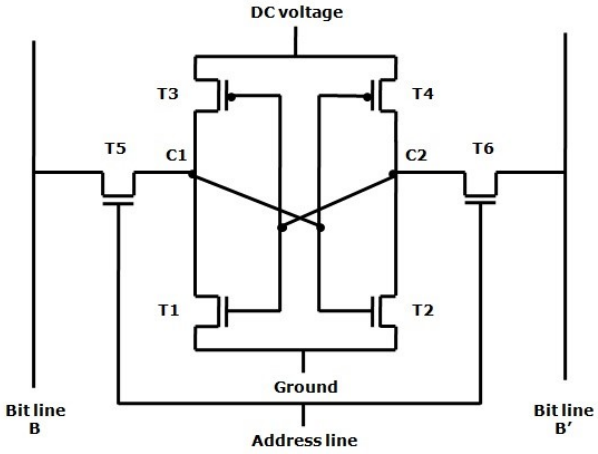
1. SRAM uses latches to store data. As long as power is supplied to the latch, the value is stored.
2. DRAM uses capacitors and transistors. The charge in the capacitor determines the state of the cell. The transistor is used for switching the state of the capacitor. This also requires power, however the capacitors leak charge over time, and hence require a periodic charge top-up, giving it's "dynamic" name. DRAM is slower than SRAM but is cheaper, and hence is more widely used.

Now we look at the working of a SRAM cell.
As seen in 7a, it consists of 6 transistors:

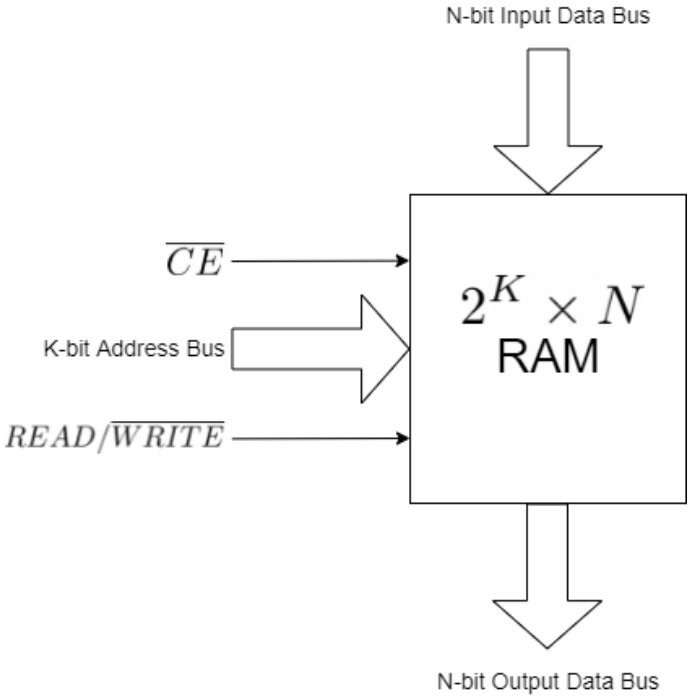
- The transistors T1, T2, T3, and T4 perform the task of a storage cell.
- The transistors T5, T6 work as access transistors that control the access to the storage cell.

Transistors T1 and T2 form an inverter and so do T3 and T4. Both the inverters are coupled in such a way that they store a single bit (C1) and its complement (C2).

[1] https://www.101computing.net/binary-comparators-using-logic-gates/



(a) Circuit of a cell in SRAM [2]



(b) Block diagram of RAM

Figure 7

There are 3 operations that the SRAM cell can perform:

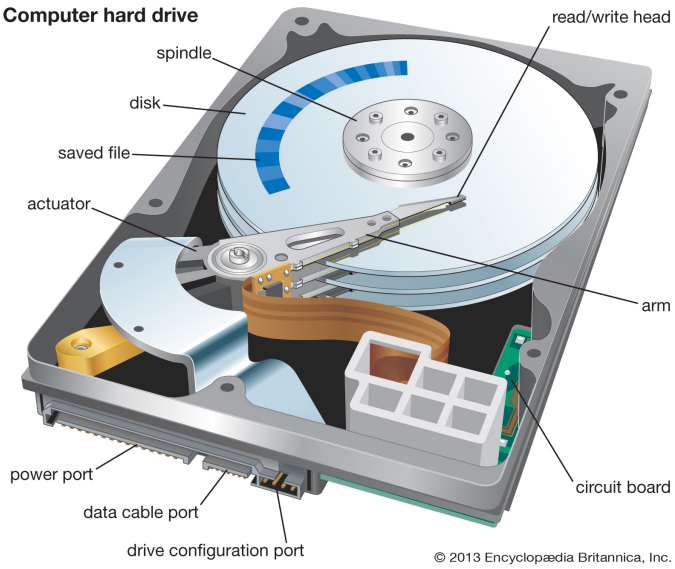
a. **Hold**
With the address line turned OFF, the NMOS transistors T5 and T6 essentially act like floating wires, which implies the data is confined to the storage cell.

b. **Read**
With the address line turned ON, the NMOS transistors T5 and T6 act as a short circuit, and the data at C1 and C2 are collected at B and B' respectively. However, bit lines are relatively long and have large parasitic capacitance. So, to speed up the process we use a sense amplifier, which is essentially an Op-Amp comparator with B connected to + input and B' connected to − input. We start out by pre-charging both B and B' to HIGH voltage. Then when the address line enables both T5 and T6, the voltage at one of the bit lines will drop slightly, which causes a slight voltage difference at the input of the Op-Amp comparator causing the output to instantly correspond to the state at C1.

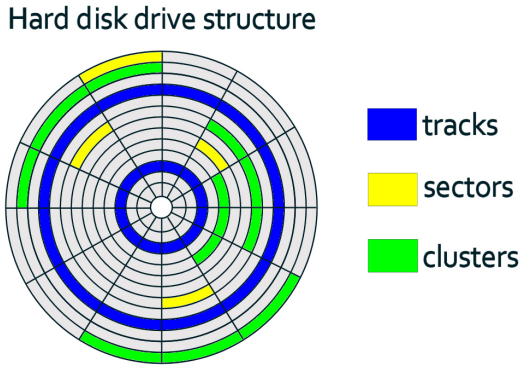
c. **Write**
We start out by setting the value that we want to store, on B and the complement of that value on B', the address line is then turned ON to store the data inside the storage cell.

[2] https://computerhindinotes.com/difference-between-sram-and-dram/static-ram-cell/

- **HDD:**
  Hard Disk Drive (HDD) is a secondary memory device which stores data *permanently*. We can both read and write data on a HDD. It uses a magnetic disk and readers to store data. A magnetised cell is read as binary 1, while a non-magnetic cell is read as binary 0. This makes it easy to write and retrieve. A hard disk contain platters that are aluminium or glass disks that contain a magnetic layer on them. Above the disk is a read/write head held on an arm. The arm has one dimensional radial motion, while the disk spins using motors. Sectors and tracks are used to distinguish different cells on the disk, as seen in 8b



(a) Labeled diagram of a modern HDD [3]



(b) Sectors and Tracks in a HDD [4]

Figure 8

Hard disk drives are advantageous as they have a longer life span as compared to other data storage devices and are readily available in the market at affordable prices.

References

[1] Albert Paul Malvino, Jerald A. Brown, *Digital Computer Electronics*, McGraw-Hill, 1977

[2] https://studiousguy.com/working-principle-hard-disk/

[3] https://computer.howstuffworks.com/hard-disk.htm

[4] https://www.explainthatstuff.com/harddrive.html

[5] https://digitalthinkerhelp.com/sram-circuit-design-and-operation-read-write-working-of-sram/

[6] https://www.britannica.com/technology/static-random-access-memory

[7] https://computer.howstuffworks.com/ram.html

[8] https://www.geeksforgeeks.org/random-access-memory-ram-and-read-only-memory-rom/

[3] https://suronto.blogspot.com/2020/06/hard-disk-diagram.html
[4] https://geekgirls.com/2010/03/why-defrag/

Contribution Report

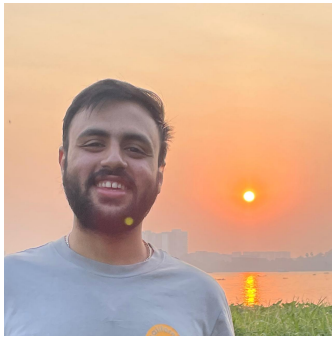
Following were the contributions of each team member individually:

- Advaith Suresh: Studied the parts of memory involving RAM and HDD.
- Anurag Abhijit Pendse: Designed the logic circuits, wrote the write up on the current system and was involved in writing the report in LaTeX.
- Lokesh Mishra: Worked on writing the report in LaTeX.
- Mehul Vijay Chanda: Studied the memory system to be used, wrote the description of the different elements of the MFMS used and was involved in writing the report in LaTeX.
- Niare Doyom: Studied the part of memory involving SRAM.
- Samyak Jain: Drew the circuits in Simulink and studied the memory system to be used.

(a) Advaith Suresh

(b) Anurag Abhijit Pendse

(c) Lokesh Mishra



(d) Mehul Vijay Chanda

(e) Niare Doyom

(f) Samyak Jain

Figure 9: Members of the team