**Course Code: CS 220, Course Title: Digital Logic Design**

**Evaluation Rubric**

**Group Members:**

| Student No. | Name | Roll No. | Batch |
|---|---|---|---|
| S1 | **Safia Faiz** | **CS-58** | **2020** |
| S2 | **Syed Muhammad Mubashir Rizvi** | **CS-71** | **2020** |

| **CRITERIA AND SCALES** | | | |
|---|---|---|---|
| **Criterion 1: To what level has the student understood the problem? [CPA 1]** | | | |
| 0 - 2 | 3 - 4 | 5 | |
| Problem understanding is minimal. | Problem is understood partially | Problem is completely understood | |
| **Criterion 2: To what extent the student implemented the solution? [CPA 1]** | | | |
| 0 | 1-3 | 4-5 | |
| The solution has not been implemented | The solution is incomplete. The outputs are erroneous | The solution is complete. Desired outputs have been achieved | |
| **Criterion 3: What level of creativity is evident in the proposed design? [CPA 2]** | | | |
| 0 - 1 | 2-3 | 4-5 | |
| No / Poor design | The design is run of the mill | The design is innovative | |
| **Criterion 4: What is the student's level of confidence with the Simulation Tool Interface? [CPA 3]** | | | |
| 0 | 1-2 | 3-4 | 5 |
| The student is unfamiliar with the tool | The student is familiar with the visible features of the tool | The student is familiar with the unexposed features of the tool | The student is proficient with the tool |
| **Criterion 5: How well has the student interconnected the circuit components / hardware resources?** | | | |
| 0 | 1-3 | 4-5 | |
| Student has no idea how to connect the circuit components / hardware resources | Circuit components / hardware resources are not connected properly | Circuit components / hardware resources are properly connected | |
| **Criterion 6: Answer to questions related to the design (Hardware part)** | | | |
| 0 | 1-2 | 3-4 | 5 |
| The student did not answer any question | Few questions were answered | The student answered most of the questions | The student answered all the questions |
| **Criterion 7: Answer to questions related to the design (Simulation Part)** | | | |
| 0 | 1-4 | 5-8 | 9-10 |
| The student did not answer any question | The student answered a few questions | The student answered most of the questions | The student answered all the questions |

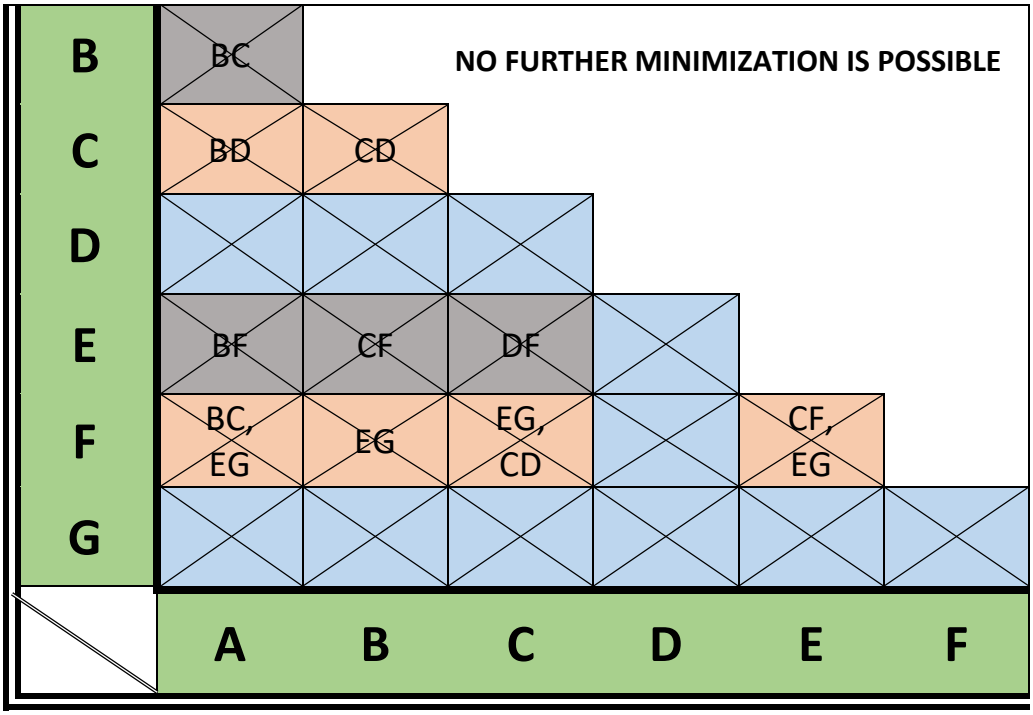
## 6. TRUTH TABLE

| INPUT | PRESENT STATE | | | NEXT STATE | | | OUTPUT | | FF INPUT | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| X | $Q_{2(t)}$ | $Q_{1(t)}$ | $Q_{0(t)}$ | $Q_{2(t+1)}$ | $Q_{1(t+1)}$ | $Q_{0(t+1)}$ | $Z_1$ | $Z_0$ | $D_2$ | $D_1$ | $D_0$ | |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | $Z_o$ |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | |
| 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | |
| 0 | 1 | 1 | 1 | X | X | X | X | X | X | X | X | |
| 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | |
| 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | |
| 1 | 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | |
| 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | |
| 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | $Z_1$ |
| 1 | 1 | 1 | 1 | X | X | X | X | X | X | X | X | |

## 4. STATE TABLE

| X | 0 | 1 |
|---|---|---|
| A | E , 00 | B , 00 |
| B | E , 00 | C , 00 |
| C | E , 00 | D , 00 |
| D | E , 01 | D , 00 |
| E | E , 00 | F , 00 |
| F | G , 00 | C , 00 |
| G | E , 00 | F , 10 |

## 5. STATE MINIMIZATION




| | |
|---|---|
| | This colour represents the first phase/stage of the state minimization. |
| | This colour represents the second phase/stage of the state minimization. |
| | This colour represents the third phase/stage of the state minimization. |

# 7. K – MAPS & STATE EQUATIONS

| $Z_1$ | $Q_1Q_0$ | | | | |
|---|---|---|---|---|---|
| $XQ_2$ | | 00 | 01 | 11 | 10 |
| | 00 | | | | |
| | 01 | | | X | |
| | 11 | | | X | 1 |
| | 10 | | | | |

$Z_1 = x'Q_1Q_0$

| $Z_0$ | $Q_1Q_0$ | | | | |
|---|---|---|---|---|---|
| $XQ_2$ | | 00 | 01 | 11 | 10 |
| | 00 | | | 1 | |
| | 01 | | | X | |
| | 11 | | | X | |
| | 10 | | | | |

$Z_0 = xQ_2Q_1$

| $D_2$ | $Q_1Q_0$ | | | | |
|---|---|---|---|---|---|
| $XQ_2$ | | 00 | 01 | 11 | 10 |
| | 00 | 1 | 1 | 1 | 1 |
| | 01 | 1 | 1 | X | 1 |
| | 11 | 1 | | X | 1 |
| | 10 | | | | |

$D_2 = x' + Q_2Q_0'$

$Q_{2(t+1)} = D_2 = x' + Q_2Q_0$

| $D_1$ | $Q_1Q_0$ | | | | |
|---|---|---|---|---|---|
| $XQ_2$ | | 00 | 01 | 11 | 10 |
| | 00 | | | | |
| | 01 | | 1 | X | |
| | 11 | | 1 | X | |
| | 10 | | 1 | 1 | 1 |

$D_1 = Q_2Q_0 + xQ_0 + xQ_2'Q_1$

$Q_{1(t+1)} = D_1 = Q_2Q_0 + xQ_0 + xQ_2'Q_1$

| $D_0$ | $Q_1Q_0$ | | | | |
|---|---|---|---|---|---|
| $XQ_2$ | | 00 | 01 | 11 | 10 |
| | 00 | | | | |
| | 01 | | | X | |
| | 11 | 1 | | X | 1 |
| | 10 | 1 | | 1 | 1 |

$D_0 = xQ_1 + xQ_0'$

$Q_{0(t+1)} = D_0 = xQ_1 + xQ_0'$

# COMPLEX ENGINEERING PROBLEM (CEP)

**DIGITAL LOGIC DESIGN – CS-220**


## CEP TASK:

SEQUENCE DETECTOR

SUBMITTED BY:

SAFIA FAIZ – CS-20058
SYED MUHAMMAD MUBASHIR RIZVI – CS-20071

SUBMITTED TO:

MS. ANITA ALI

## PROBLEM DESCRIPTION

A sequential combination door lock of a building opens it only when correct password (combination) is entered by the authorized user. Each door has two authorized users who can unlock the door with their secret combination without revealing the same to other.
You are required to design a state machine for the combination door lock.

## COMBINATION PAIR

Since our roll numbers are 58 and 71, adding them together resulted with the least significant digit as "9", so we were assigned the last ($4^{th}$) combination which is **"1110" and "0101".**

# FINITE STATE MACHINE DESIGN PROCEDURE

## 1. MACHINE, FLIP FLOP AND IC SELECTION

➢ MACHINE SELECTION:

We chose to work on **"MEALY MACHINE MODEL"** due to the following facts:

- It is faster as compared to Moore Machine, as we receive our output as soon as we give our input to the machine, but only in that specific CLK cycle and correct code.
- Less number of states are formed; hence it results in a smaller number of gates, i.e less hardware to be used (cost-effective).

➢ FLIP FLOP SELECTION:

We chose to work on **"DELAY FLIP FLOP (D-FF)"** because:

- The state equation of it is very simple which is "**$D = Q_{t+1}$**" (i.e same as the next state).
- It has a single input i.e design process becomes easier which also makes it simpler to use.

➢ IC SELECTION:

In our hardware we have used:

1. One 7411 – because we had 3-three input variables in our equations of $Z_0$, $Z_1$ and $D_1$.
2. Two 7408 – because there were 5-two input variables in our equations of $D_2$, $D_1$ and $D_0$.
3. One 7432 – because we had to OR all the AND-ed outputs together, and we only needed 4 OR gates (i.e 1 IC).
4. One 7404 – because we needed a NOT gate to invert the input (i.e x).
5. Two 7474 – the D-FFs ICs, in which we took 1D of both ICs and 2D of the first D-FF only, as we needed only 3 D-FFs in our circuit, the OR-ed outputs from IC no. 7432 is given input here at 1D of both the circuits and 2D of the first IC, and also used the 1Q of both the ICs and 2Q of the first IC and their complement as the inputs for the circuit.

# 2. STATE DIAGRAM

➢ WHAT PRINCIPLE DOES THIS MACHINE WORK ON?

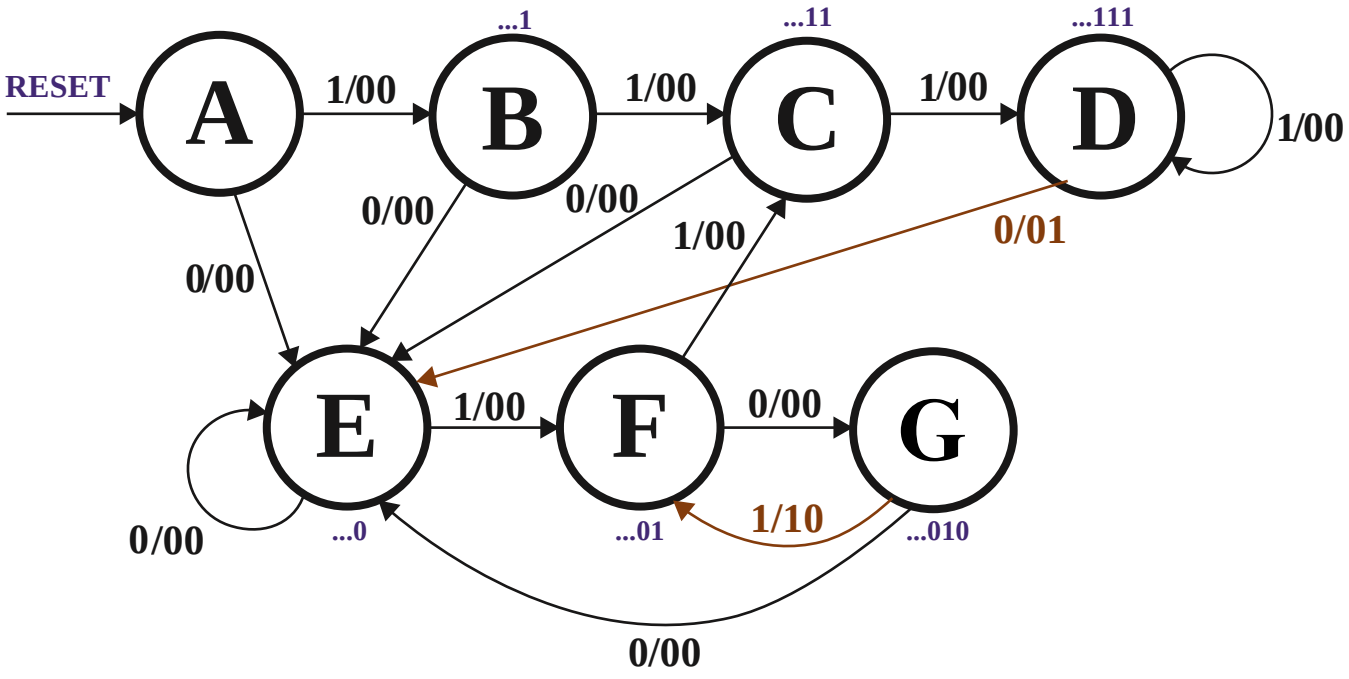
This machine works on the principle of an **"OVERLAPPING MACHINE MODEL"**, where even if the correct code has been entered, it does not relapse back to the initial state, i.e the previous bits are still significant and will be considered by this machine.

➢ POSSIBLE OUTCOMES:

There are **THREE** possible outcomes that can be yield from this machine, which are mentioned below:

1. When an invalid or an incorrect code is entered : **00**
2. When 1110 is found : **01** (our second bulb will glow)
3. When 0101 is found : **10** (our first bulb will glow)

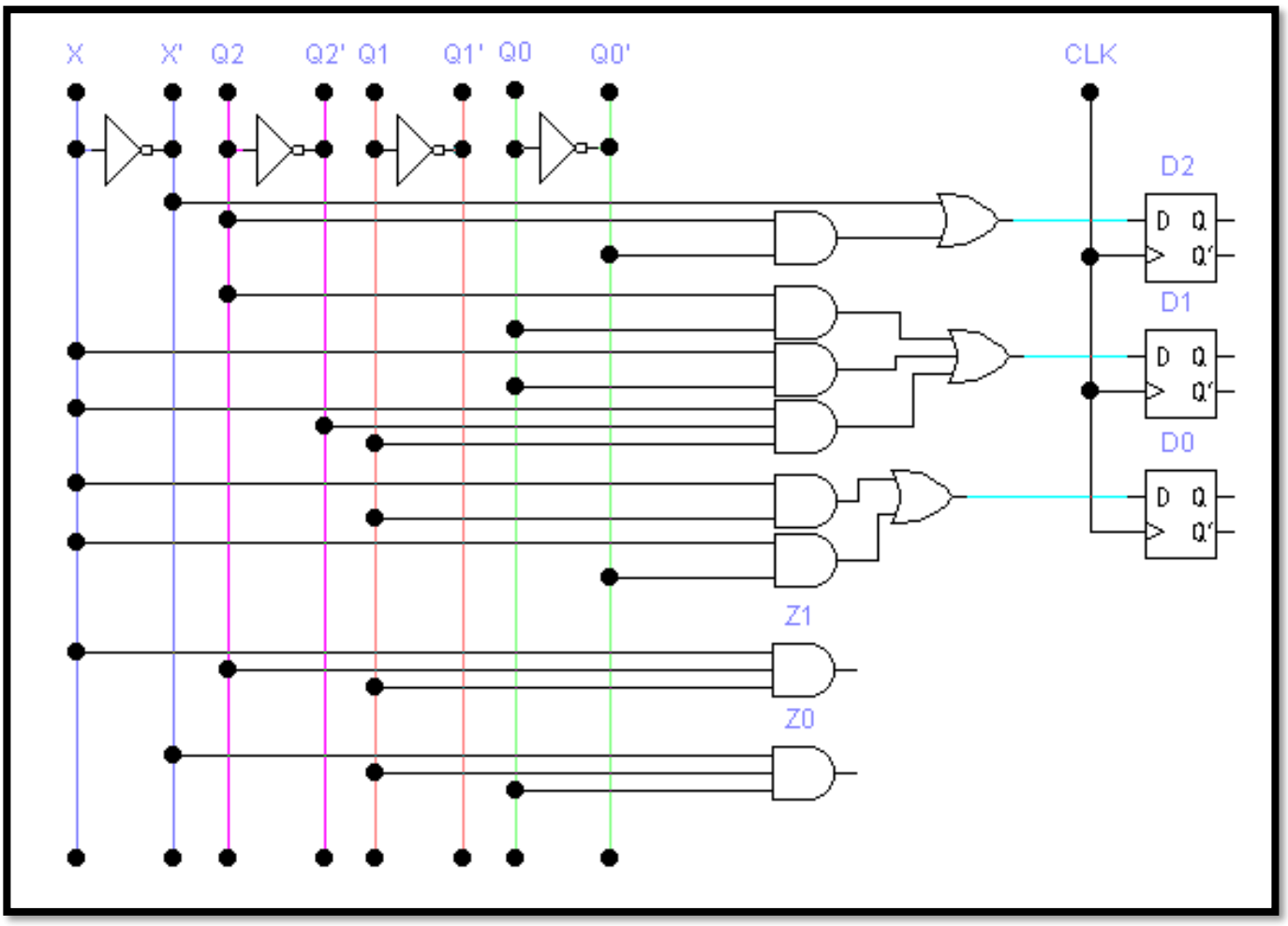
➢ DIAGRAM:




# 3. STATE ASSIGNMENT

| | | | |
|---|---|---|---|
| A = 000 | B = 001 | C = 010 | D = 011 |
| E = 100 | F = 101 | G = 110 | |

## 8. LOGIC DIAGRAM

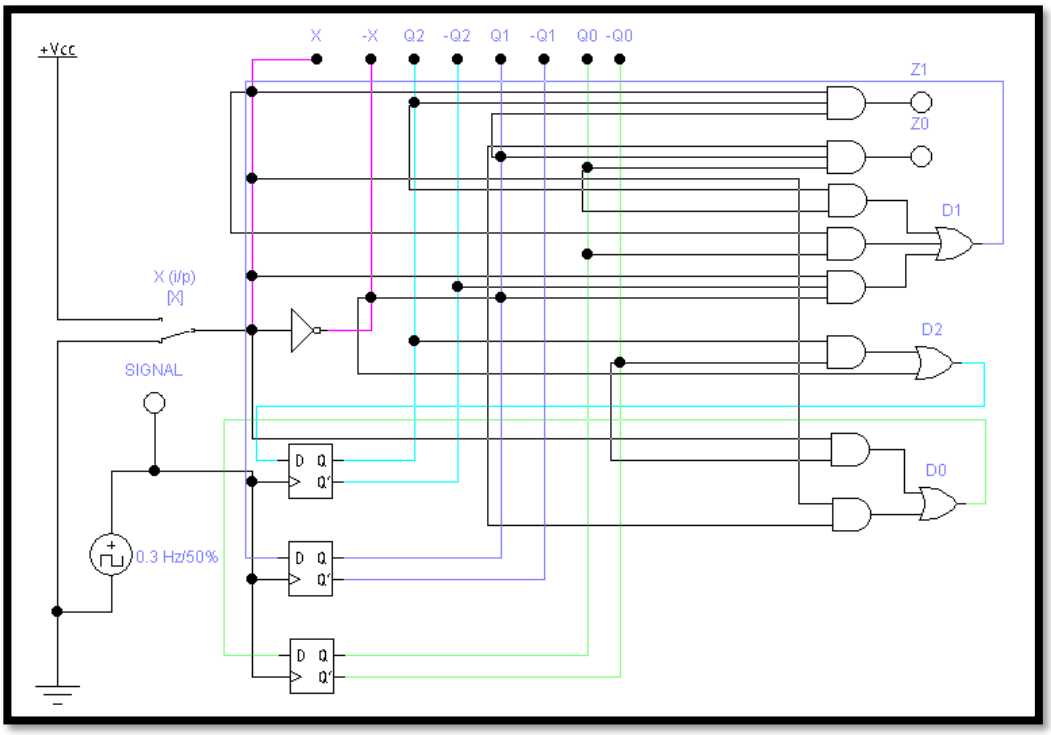



# LOGICAL/REALISTIC ASSUMPTION OF THE FSM:

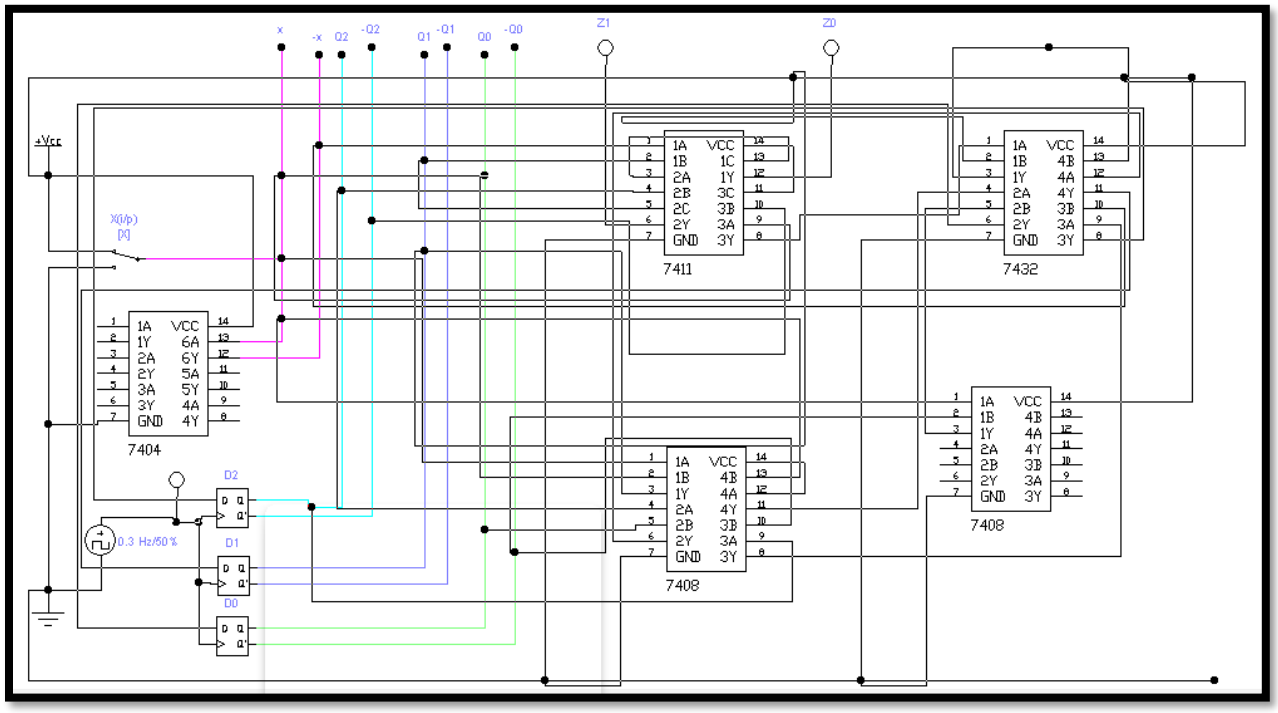
This FSM is designed for a safe lock door which is **accessed quite rarely** in the building (we assumed it to be a money safe place). Both the authorized users know that this FSM (or door lock) also **considers the bits which were previously entered by them**. As they rarely visit the room, they can also perhaps forget their passcode too, to which they can randomly add zeros and ones (assuming they know what they are giving as input, for e.g; IS:000100**1110**) and then identify their passcode. Also, if not used for 24 hrs, the door lock shuts off and when it turns back on, it resets itself to its initial (empty) state. Though, there's a vague possibility if both the users tried entering their codes after the other, they might overlap, for example if "1110" is entered, then there's no need to enter "0" from "0101" and the same applies for the other code as well, but this shortcoming will be overcome soon.

# SIMULATION OF THE FINITE STATE MACHINE:

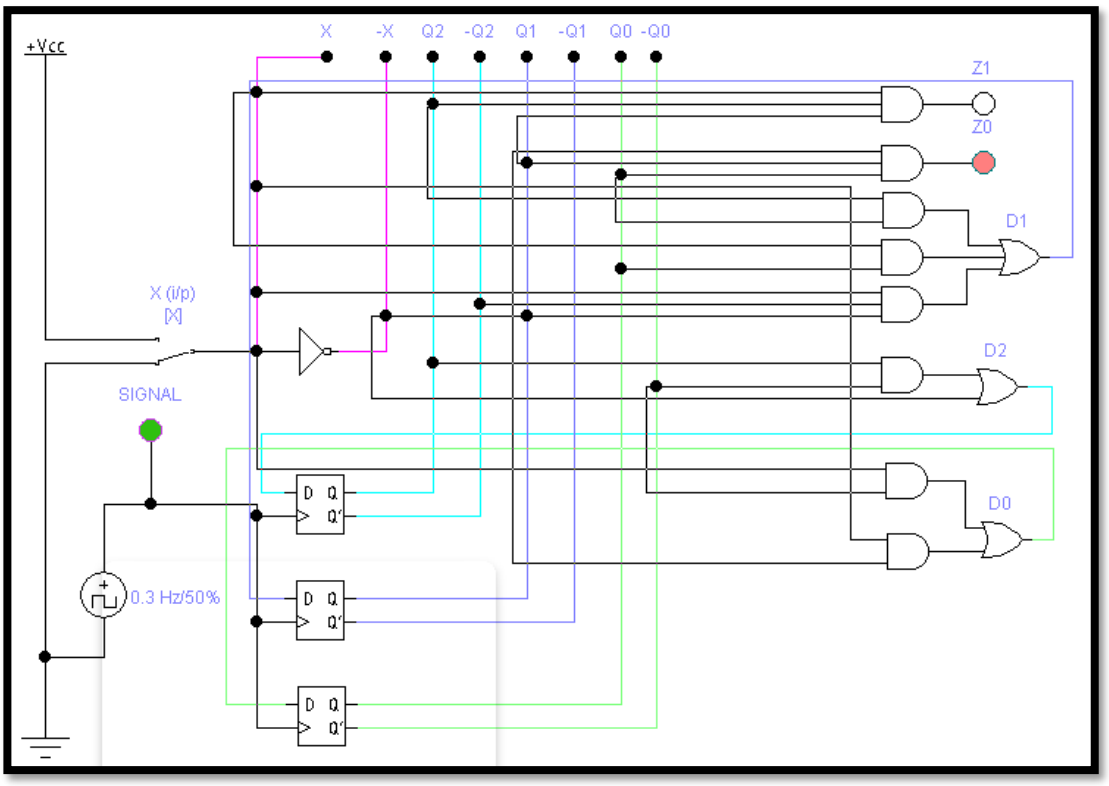
## THE COMPLETE CIRCUIT (USING GATES):




## THE COMPLETE CIRCUIT (USING ICS):



## WHEN 1110 IS ENTERED:




## WHEN 0101 IS ENTERED: