**Bimal parajuli (20BDS0405)**

# DIGITAL LOGIC DESIGN EXPERIMENT #6
# DESIGN WITH MULTIPLEXERS

## Objectives:

To design a combinational circuit and implement it with multiplexers. To use a demultiplexer to implement a multiple output combinational circuit from the same input variables.

## Apparatus:

- IC type 7404 HEX inverter
- IC type 7408 quad 2-input AND gate
- IC type 74151 8x1 multiplexer (1)
- IC type 74153 dual 4x1 multiplexer (2)
- IC type 7446 BCD-to-Seven-Segment decoder (1)
- Resistance network (1)
- Seven-Segment Display (1)

## Softwares Used:

- LogicWorks 5
- Proteus 8 pro

# IC Description:

74151 is an 8 line-to-1 line multiplexer. It has the schematic representation shown in Fig 1. Selection lines $S_2$, $S_1$ and $S_0$ select the particular input to be multiplexed and applied to the output.

Strobe S acts as an enable signal. If strobe =1, the chip 74151 is disabled and output y = 0. If strobe = 0 then the chip 74151 is enabled and functions as a multiplexer. Table 1 shows the multiplex function of 74151 in terms of select lines.

Table 1.

| Strobe | Select Lines | | | Output |
|---|---|---|---|---|
| S | $S_2$ | $S_1$ | $S_0$ | Y |
| 1 | X | X | X | 0 |
| 0 | 0 | 0 | 0 | D0 |
| 0 | 0 | 0 | 1 | D1 |
| 0 | 0 | 1 | 0 | D2 |
| 0 | 0 | 1 | 1 | D3 |
| 0 | 1 | 0 | 0 | D4 |
| 0 | 1 | 0 | 1 | D5 |
| 0 | 1 | 1 | 0 | D6 |
| 0 | 1 | 1 | 1 | D7 |

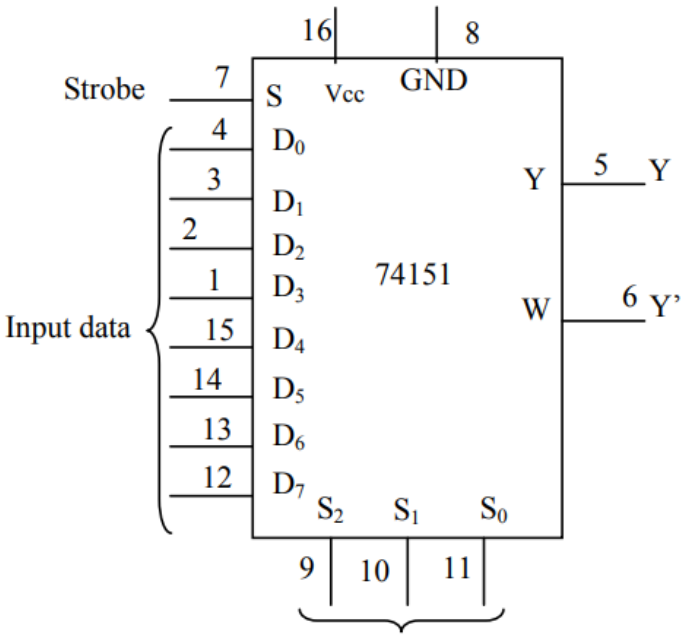



Fig.1 IC type 74151 Multiplexer 8×1

74153 is a dual 4 line-to-1 line multiplexer. It has the schematic representation shown in Fig 2. Selection lines S1 and S0 select the particular input to be multiplexed and applied to the output IY {1 = 1, 2}.
Each of the strobe signals *IG* {I = 1, 2} acts as an enable signal for the corresponding multiplexer.
Table 2. Shows the multiplex function of 74153 in terms of select lines.

Note that each of the on-chip multiplexers act independently from the other,

while sharing the same select lines $S_1$ and $S_0$.

Table 2

| | Multiplexer 1 | | |
|---|---|---|---|
| Strobe | Select lines | | Output |
| 1G | $S_1$ | $S_0$ | 1Y |
| 1 | X | X | 0 |
| 0 | 0 | 0 | $1D_0$ |
| 0 | 0 | 1 | $1D_1$ |
| 0 | 1 | 0 | $1D_2$ |
| 0 | 1 | 1 | $1D_3$ |

| | Multiplexer 2 | | |
|---|---|---|---|
| Strobe | Select lines | | Output |
| 2G | $S_1$ | $S_0$ | 2Y |
| 1 | X | X | 0 |
| 0 | 0 | 0 | $2D_0$ |
| 0 | 0 | 1 | $2D_1$ |
| 0 | 1 | 0 | $2D_2$ |
| 0 | 1 | 1 | $2D_3$ |

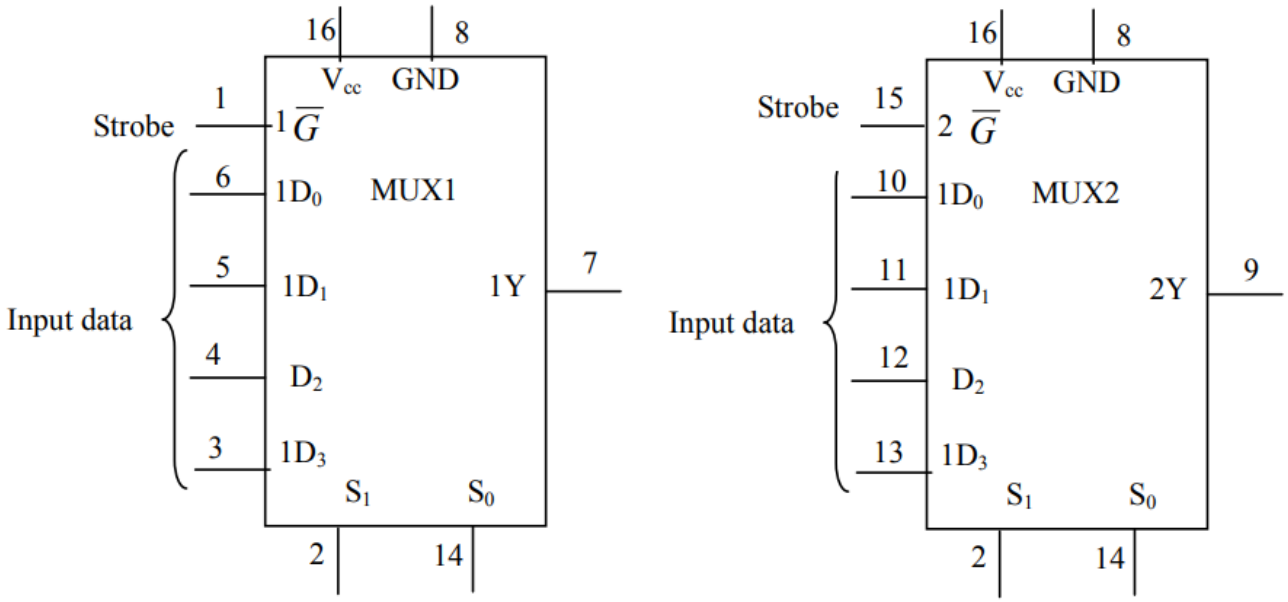



Fig.2 Chip 74153

IC 7446 is a BCD to seven segment decoder driver. It is used to convert the combinational circuit outputs in BCD forms into 7 segment digits for the 7 segment LED display units. Just like experiment #4.

# Procedure:

## *Part I: Parity Generator:*

a) Design a parity generator by using a 74151 multiplexer. Parity is an extra bit attached to a code to check that the code has been received correctly.

Odd parity bit means that the number of 1's in the code including the parity bit is an odd number. Fill the output column of the truth table in Table 2 for a 5-bit code in which four of the bits (A, B, C, D) represents the information to be sent and fifth bit (x), represents the parity bit. The required parity is an odd parity.

The inputs B, C and D correspond to the select inputs of 74151. Complete the truth table in Table 3 by filling in the last column with 0, 1, A or A'.

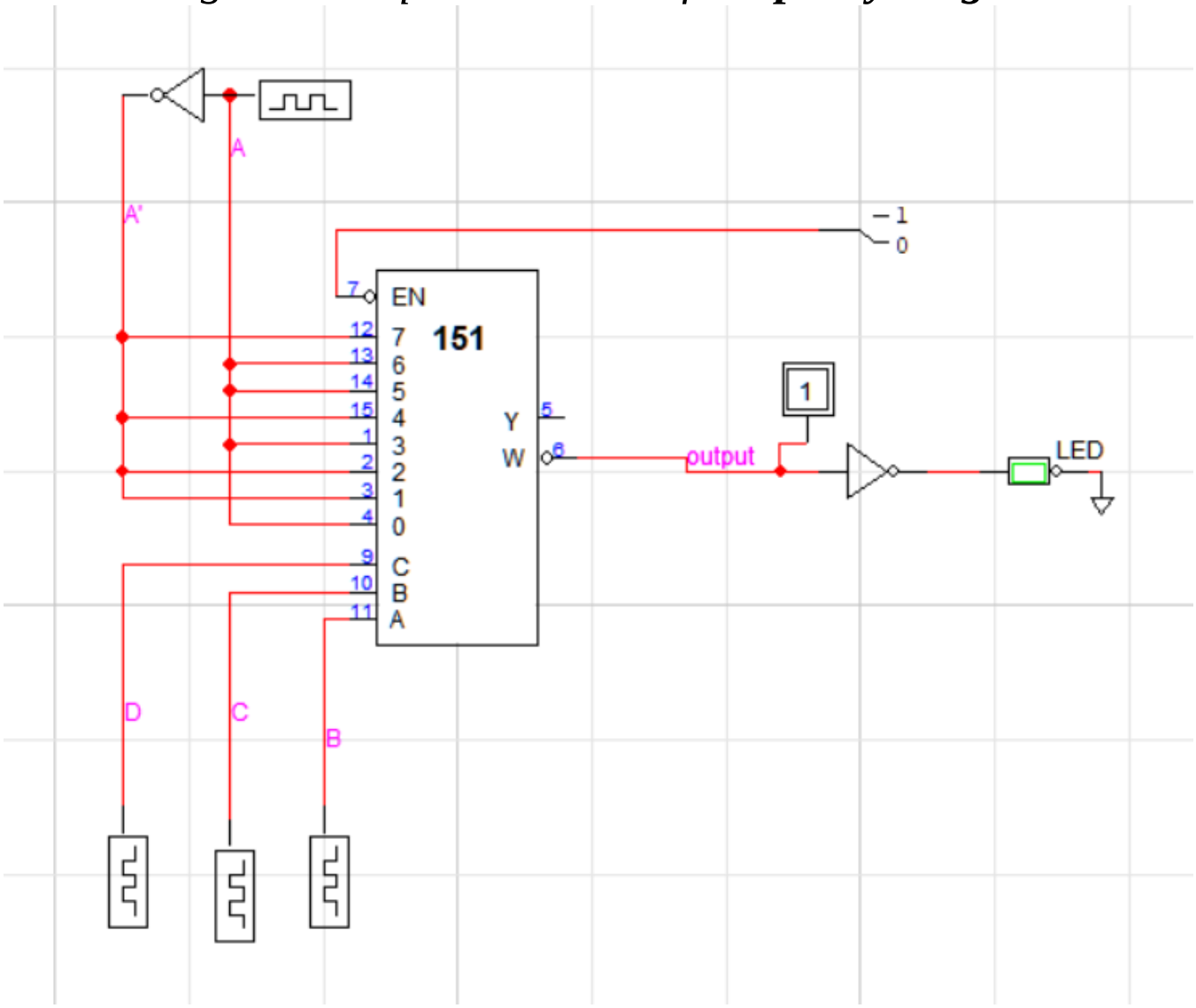
Ans:

Following is the table that demonstrates the corresponding parity bit along with the 4 original bits for **odd parity check**

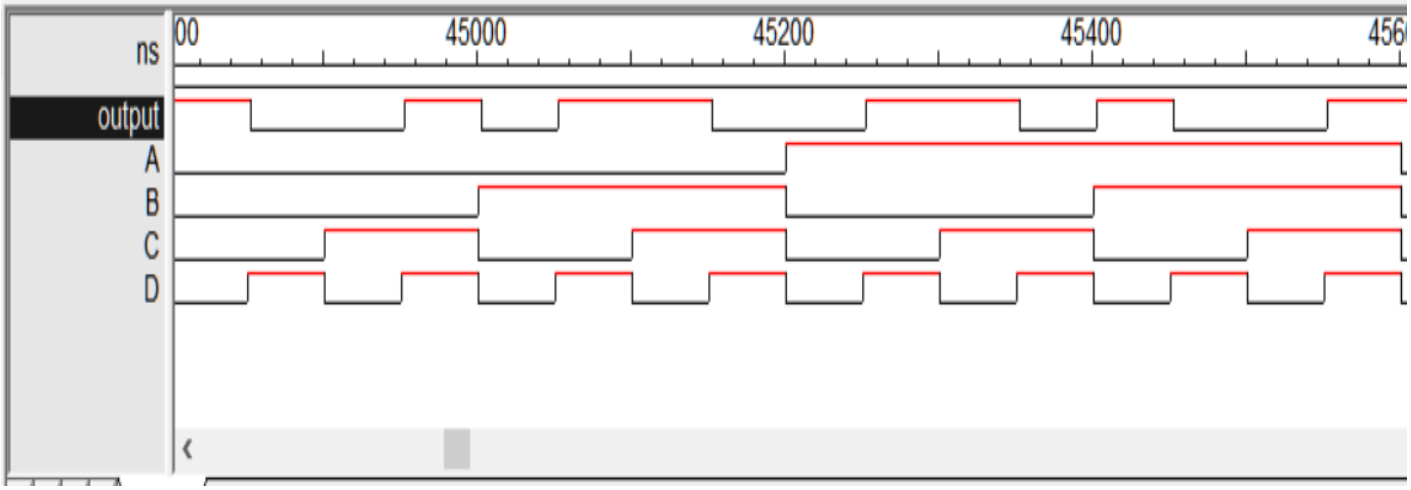
| Inputs | | | | Output | Connects Data to |
|---|---|---|---|---|---|
| A | B | C | D | x | |
| 0 | 0 | 0 | 0 | 1 | A’ |
| 0 | 0 | 0 | 1 | 0 | A |
| 0 | 0 | 1 | 0 | 0 | A |
| 0 | 0 | 1 | 1 | 1 | A’ |
| 0 | 1 | 0 | 0 | 0 | A |
| 0 | 1 | 0 | 1 | 1 | A’ |
| 0 | 1 | 1 | 0 | 1 | A’ |
| 0 | 1 | 1 | 1 | 0 | A |
| 1 | 0 | 0 | 0 | 0 | A’ |
| 1 | 0 | 0 | 1 | 1 | A |
| 1 | 0 | 1 | 0 | 1 | A |
| 1 | 0 | 1 | 1 | 0 | A’ |
| 1 | 1 | 0 | 0 | 1 | A |
| 1 | 1 | 0 | 1 | 0 | A’ |
| 1 | 1 | 1 | 0 | 0 | A’ |
| 1 | 1 | 1 | 1 | 1 | A |

b) Simulate the circuit using LogicWorks, use 74-151 multiplexer and Binary switches for inputs and Binary Probes for outputs. The 74151 has one output for Y and another inverted output W. Use A and A' for providing values for inputs 0-7. The internal values "A, B, C" are used for selection inputs B, C, and D. Simulate the circuit and test each input combination filling in the table shown below. In the Lab connect the circuit and verify the operations. Connect an LED to the multiplexer output so that it represents the parity bit which lights any time when the four bits input have even parity.

*Following is the required circuit of the* ***parity bit generator:***




***Circuit assembled in LogicWorks 5***




***Simulated in LogicWorks 5***

# ***Part 2: Vote Counter:***

A committee is composed of a chairman (C), a senior member (S), and a member (M). The rules of the committee state that:

- The vote of the member (M) will be counted as 2 votes
- The vote of the senior member (S) will be counted as 3 votes.
- The vote of the chairman (C) will be counted as 5 votes.

Each of these persons has a switch to close ("l") when voting yes and to open ("0") when voting no.

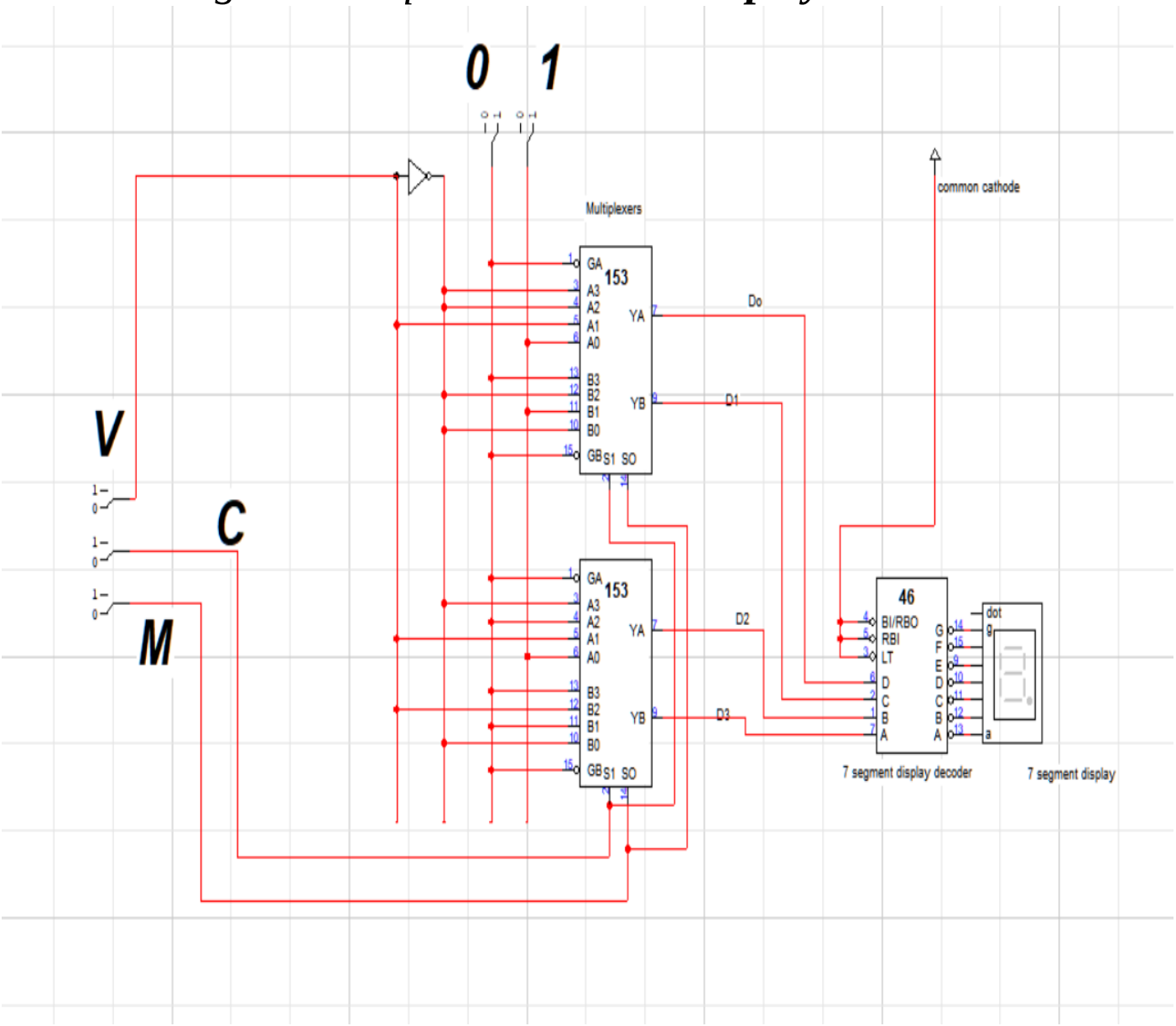
It is necessary to design a circuit that displays the total number of votes for each issue. Use a seven segment display and a decoder to display the required number.

If all members vote no for an issue the display should be blank. (Recall from Experiment #5, that a binary input 15 into the 7446 blanks all seven segments).

If all members vote yes for an issue, the display should be 0. Otherwise the display shows a decimal number equal to the number of 'yes' votes. Use two 74153 units, which include four multiplexers to design the combinational circuit that converts the inputs from the members' switch to the BCD digit for the 7446.

In LogicWorks use +5V for Logic 1 and ground for Logic 0 and use switches for C, S, and M. Use two chips 74153 and one decoder 7446 verify your design and get a copy of your circuit with the pin numbers to Lab so that you could connect the hardware in exactly the same way.

*Following is the required circuit to* ***display the vote count:***:




Here,

Let vote of Chairperson --> C (weightage, C = 5)

Let vote of senior member --> S (weightage, S = 3)

Let vote of general member --> M (weightage, M=2)

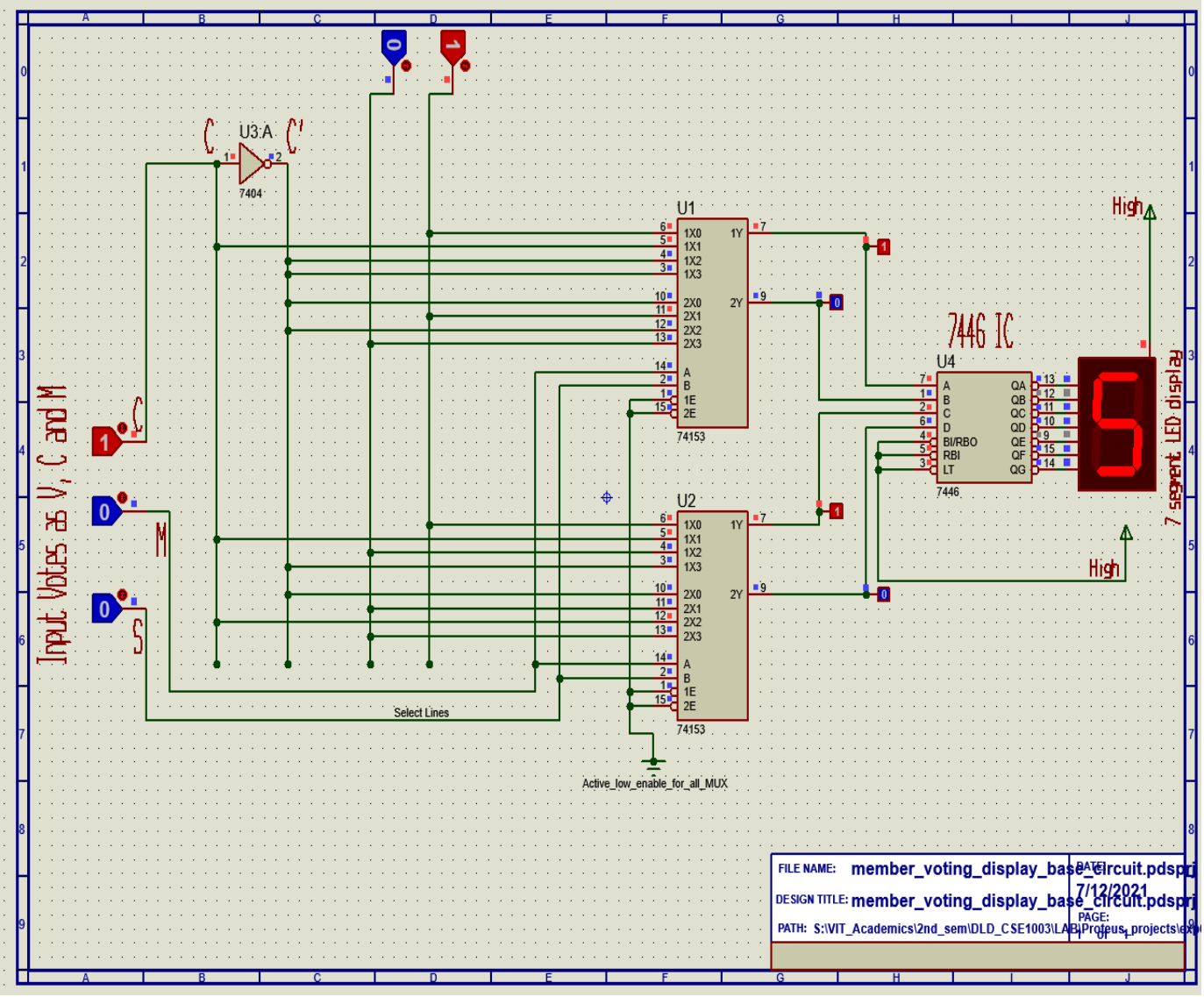
**When only C votes in favor,**

**C=1,**

**S=0,**

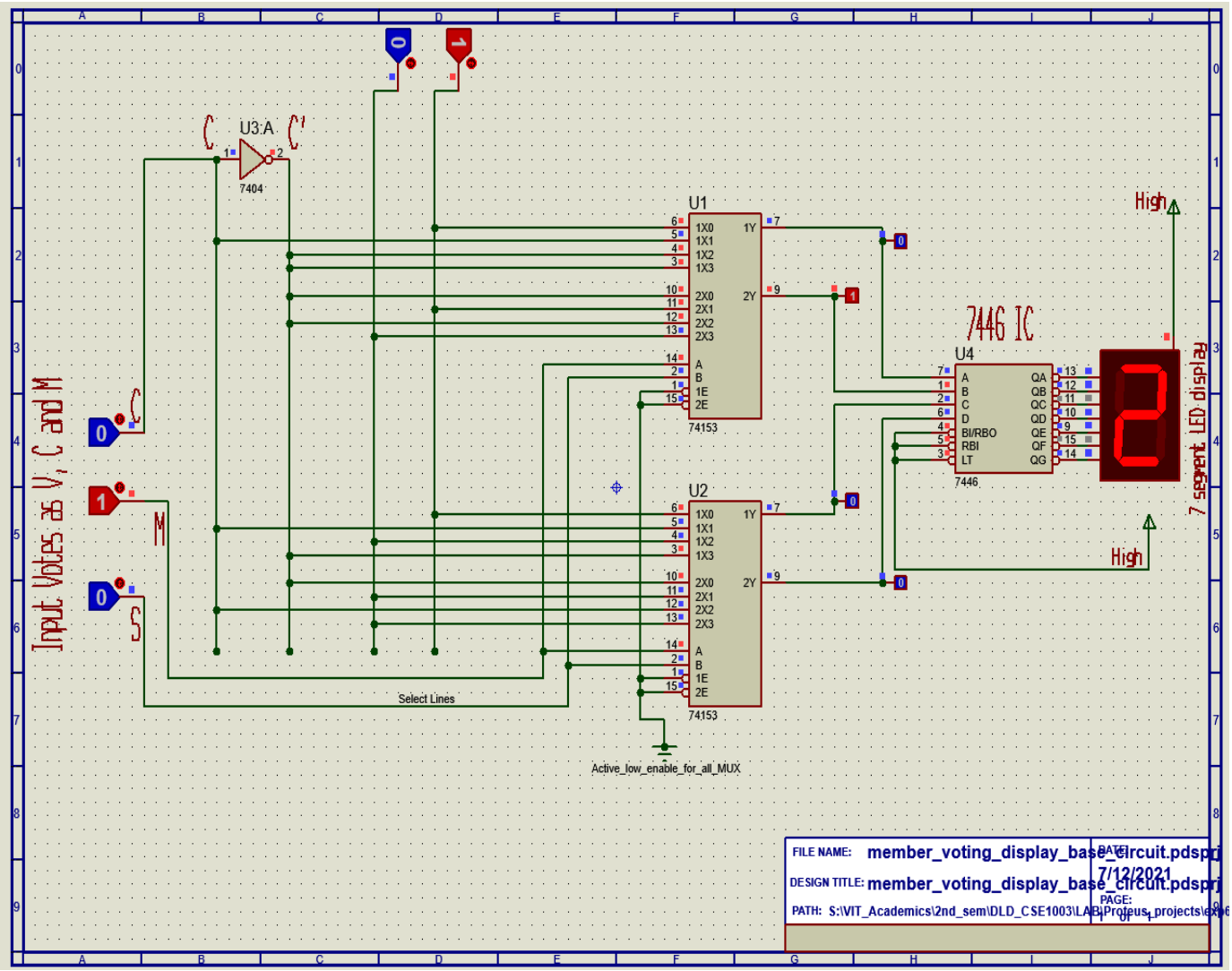
**M=0**

**Display out =5**




***Simulated in Proteus 8 pro***

**When only M votes in favor,**
**C=0,**
**M=1,**
**S=0**
**Display out =2**




***Simulated in Proteus 8 pro***

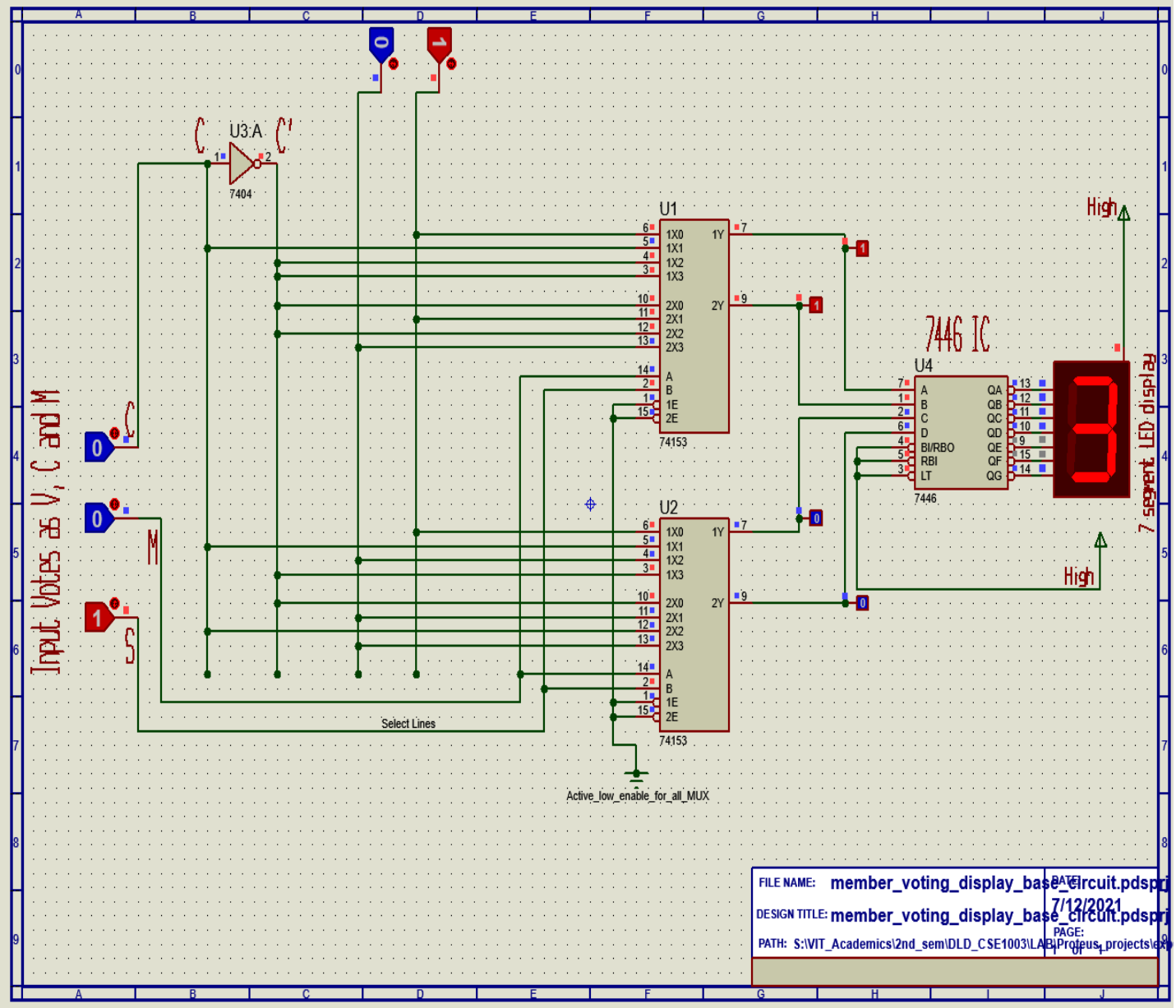
**When only S votes in favor,**

**C=0,**

**M=0,**

**S=1**

**Display out = 3**




***Simulated in Proteus 8 pro***

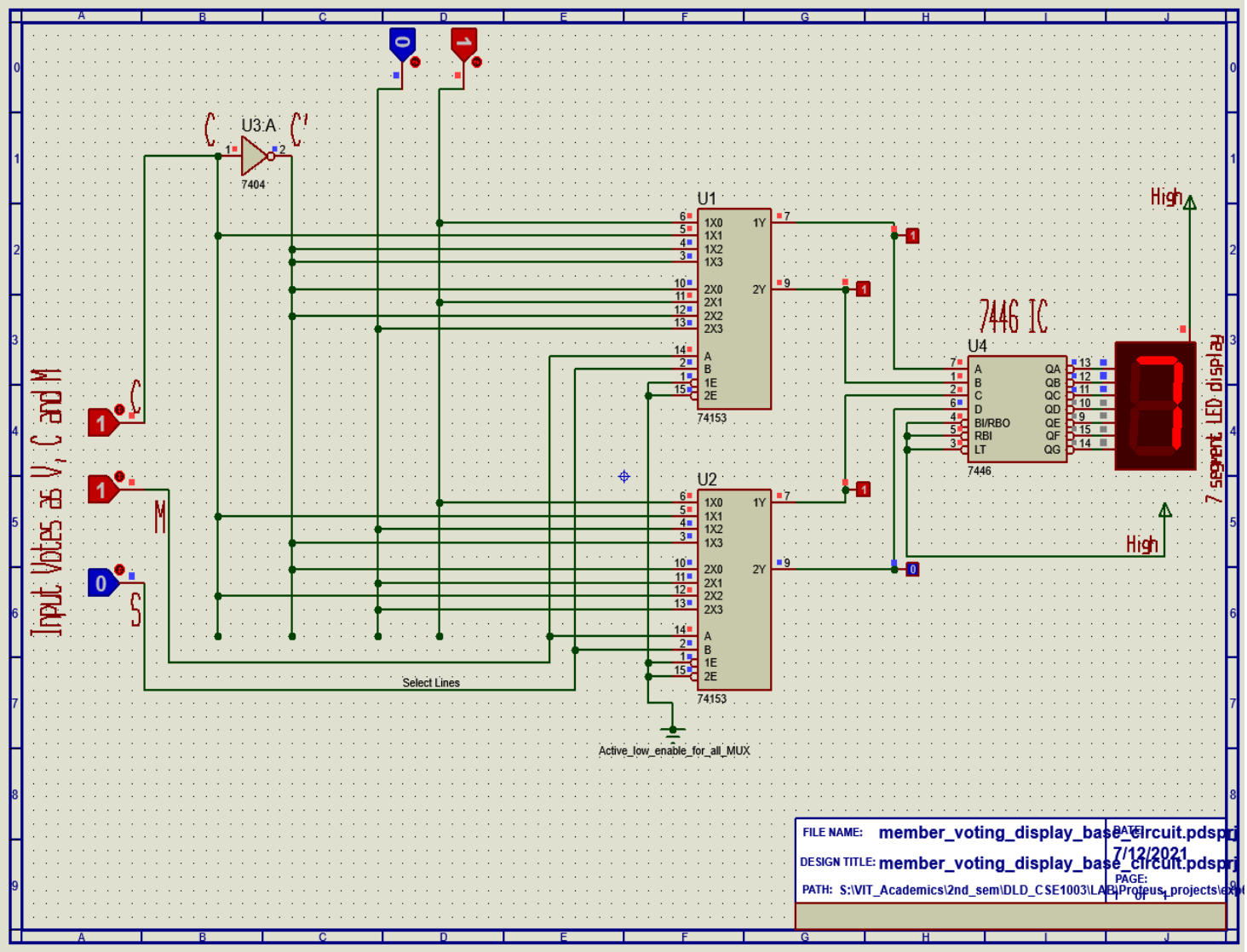
**When C and M votes in favor,**

**C=1,**

**M=1,**

**S=0**

**Display out = 7**




***Simulated in Proteus 8 pro***

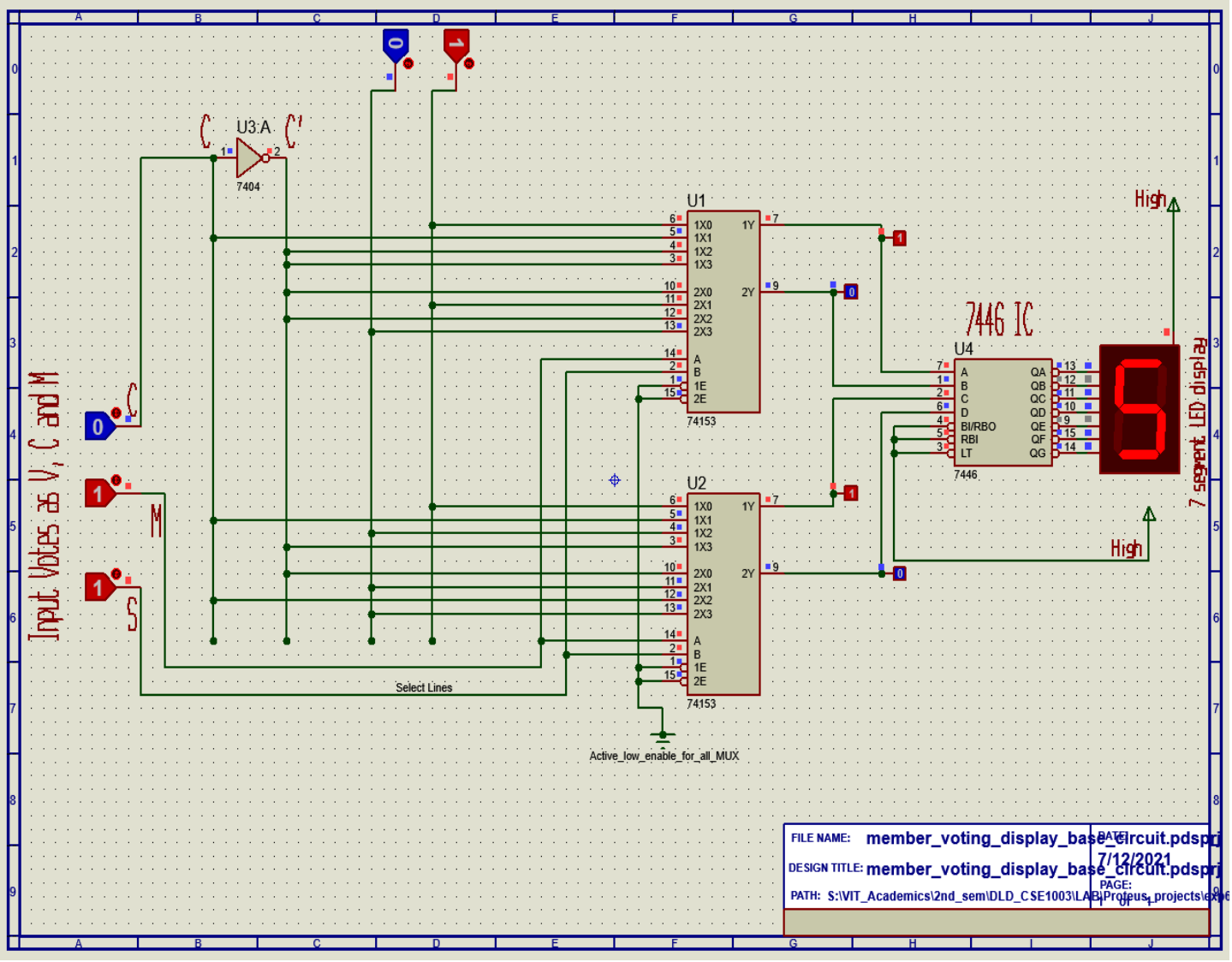
**When M and S votes in favor,**

**C=0,**

**M=1,**

**S=1**

**Display out = 5**




***Simulated in Proteus 8 pro***

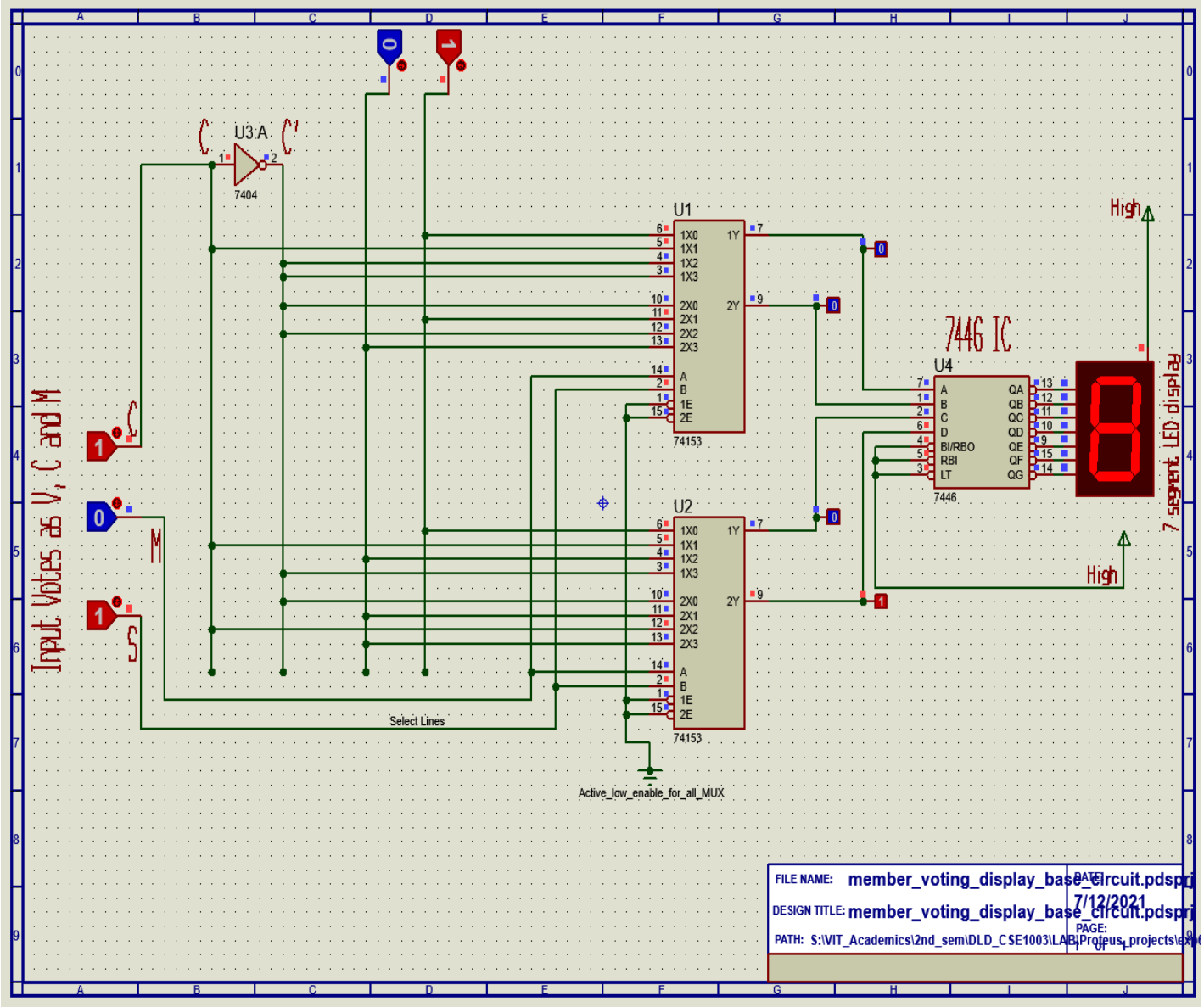
**When C and S votes in favor,**

**C=1,**

**M=0,**

**S=1**

**Display out = 8**




***Simulated in Proteus 8 pro***

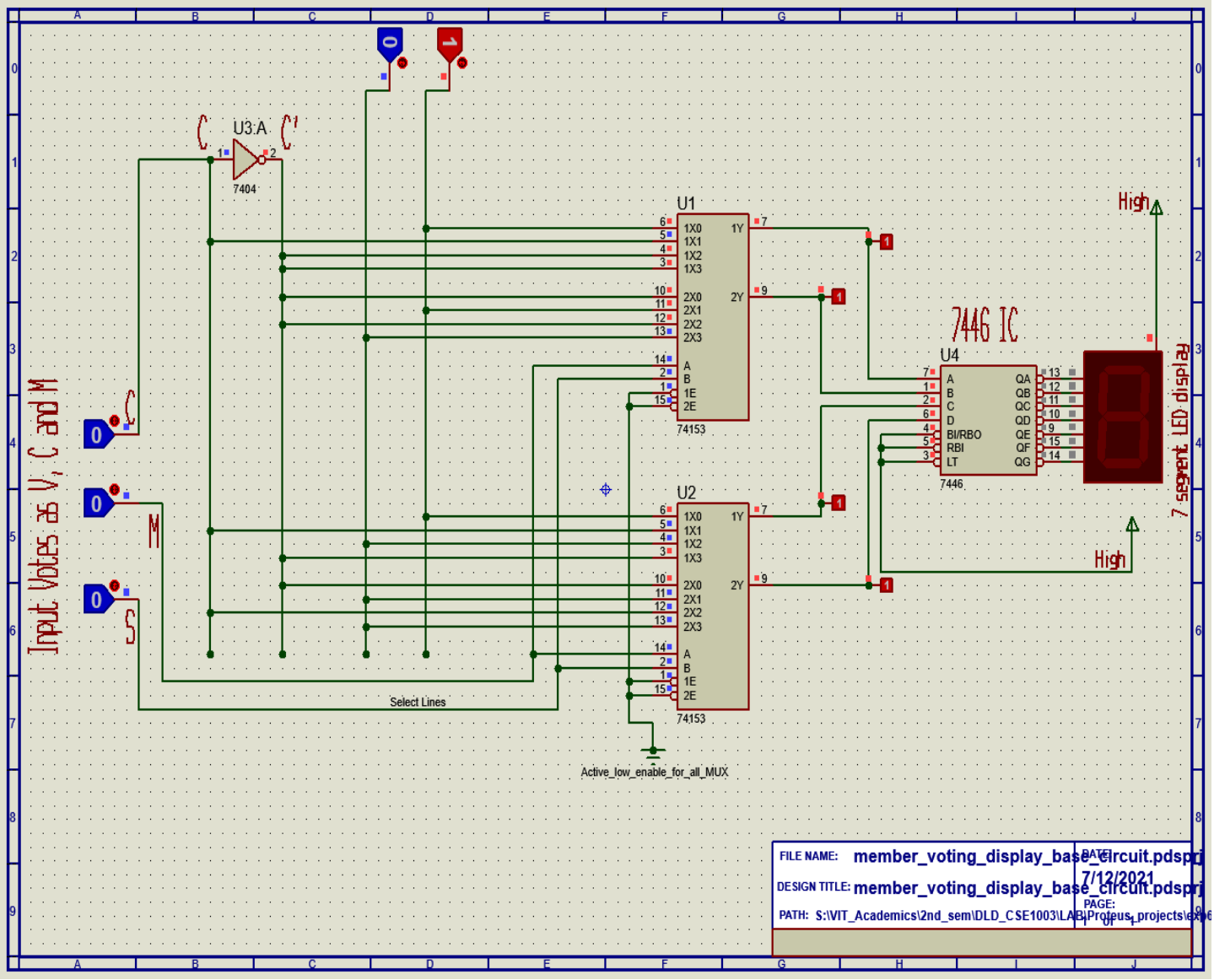
**When no one votes in favor,**

**C=0,**

**M=0,**

**S=0**

**Display out = null.**




***Simulated in Proteus 8 pro***

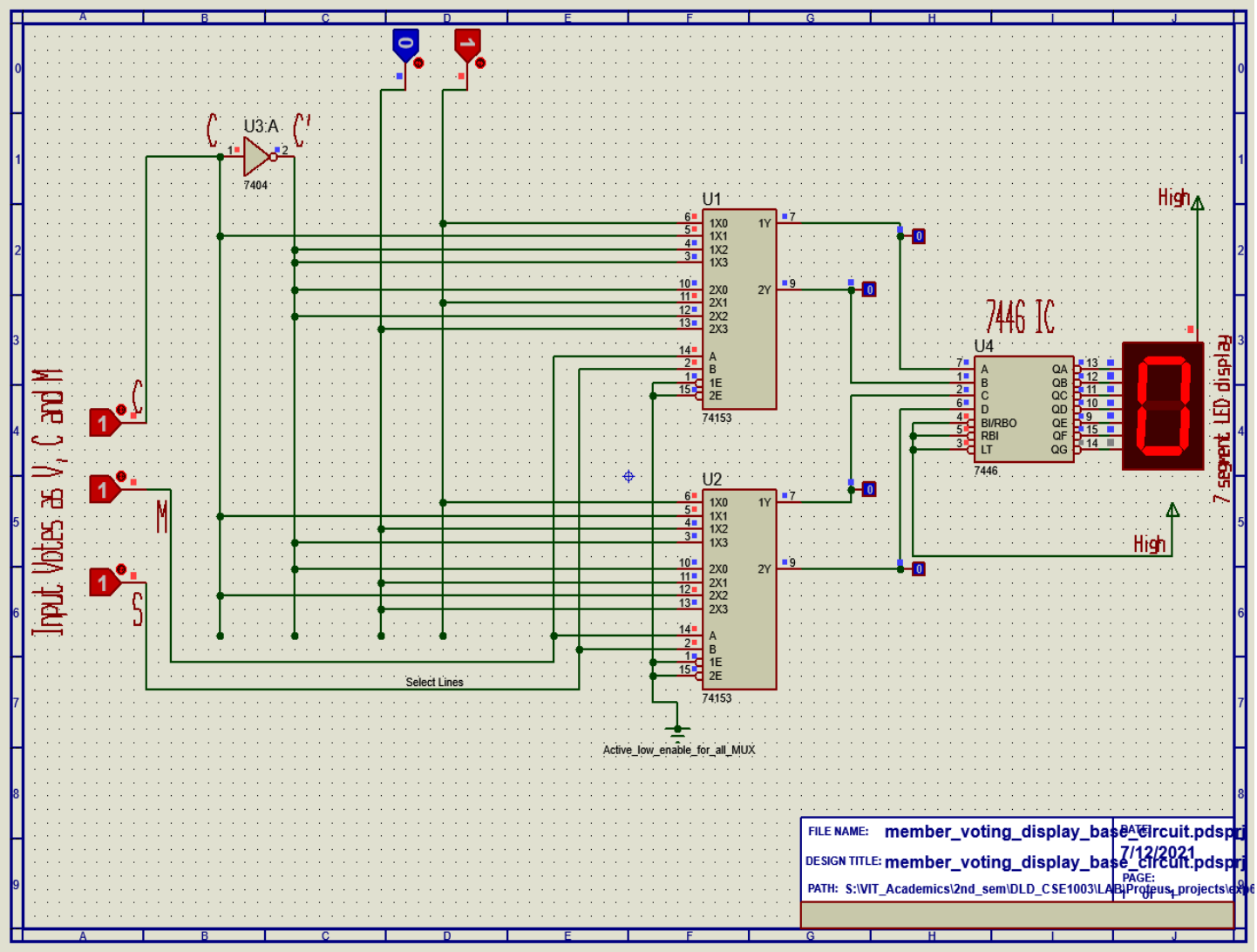
**When all votes in favor,**

**C=1,**

**M=1,**

**S=1**

**Display out = 0 (Although it's 10, 1 as carry can't be displayed).**




***Simulated in Proteus 8 pro***

--*THE END*--