Tic Tac Toe

Digital Logic Design

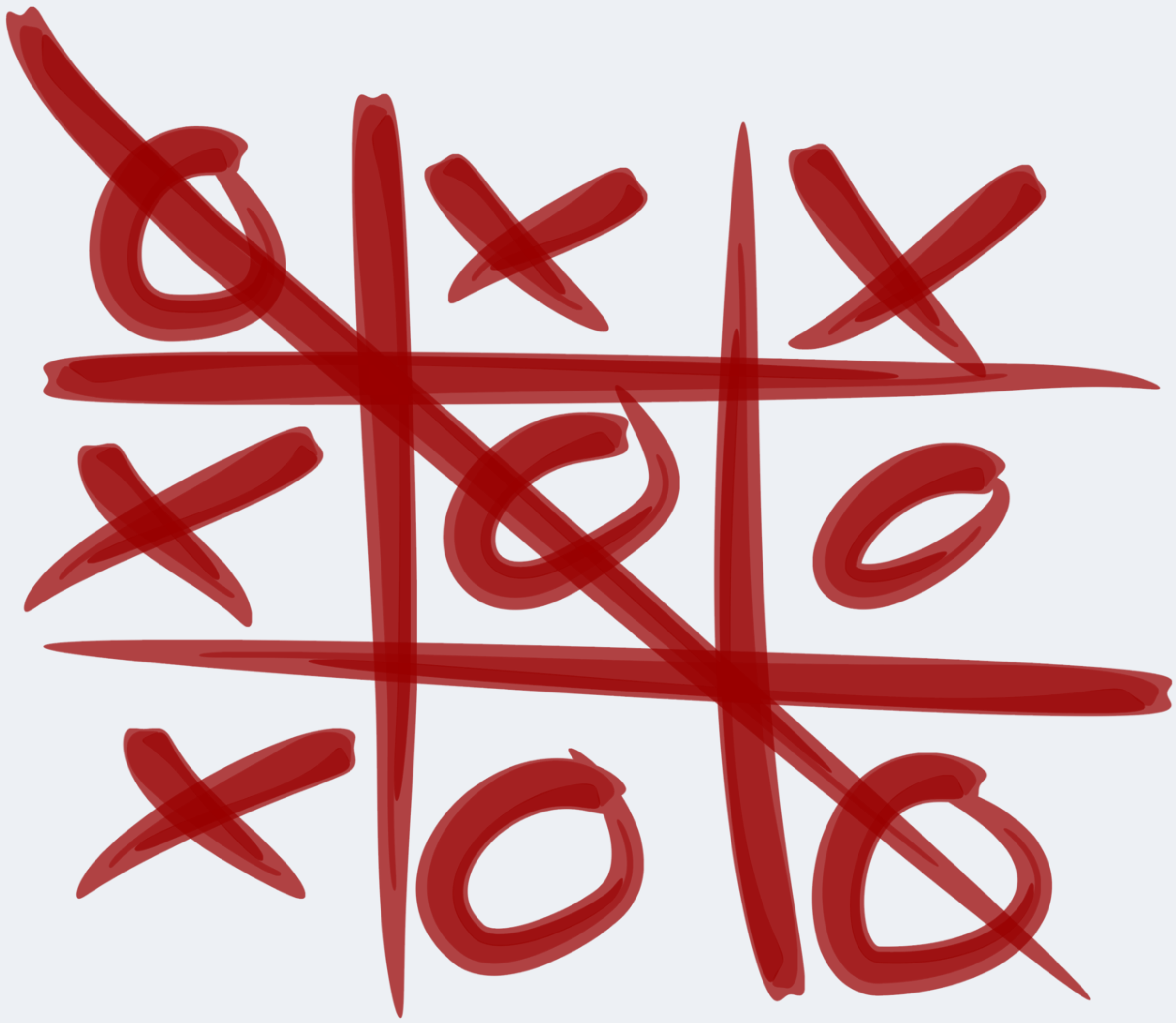
Submitted by:

Fahad Waheed (20i-0651)

Submitted to:
Shoaib Mehboob

Introduction

As a part of our Digital Logic Design Laboratory project we made a Digital Tic Tac Toe game. Tic Tac Toe is mainly an indoor game played with pen and papers. It is played between two persons. They choose two different symbols. There are nine square boxes and each player puts their symbol (mainly cross (+) or circle (o)) in the boxes turn by turn. Whenever three sequential symbols are put by any player, he is declared as the winner. If neither player wins, the match is drawn. We have made this whole game by using simple electronic components.



Our Approach

1. We made nine square boxes and selected symbols for the players using nine bi-color LED. We put them in a 3x3 matrix indicating the nine square boxes.

2. We chose red and green color as two symbols and used one bi-color (BIRG) and yellow LED for declaring the winner. When yellow LED lights up it means that the match has ended up in a draw. When red lights up, it means that the player who used red has won and vice versa.

3. We used nine logic toggles for playing. When one player pushes a button, the corresponding color lights up. Once a player wins, the “result” state should be locked. i.e. If player 1 wins first and then player 2 makes a move and satisfies the wining criterion – the output should show only “Player 1” as the winner.

4. There is a possibility of two players pushing the same button. So, we have taken necessary condition to prevent cheating.

Components and ICs

1. Bi color led (BIRG)

2. Yellow led

3. Logic toggles

4. AND gate

5. OR gate

6. NOT gate

7. XOR gate

8. Wires

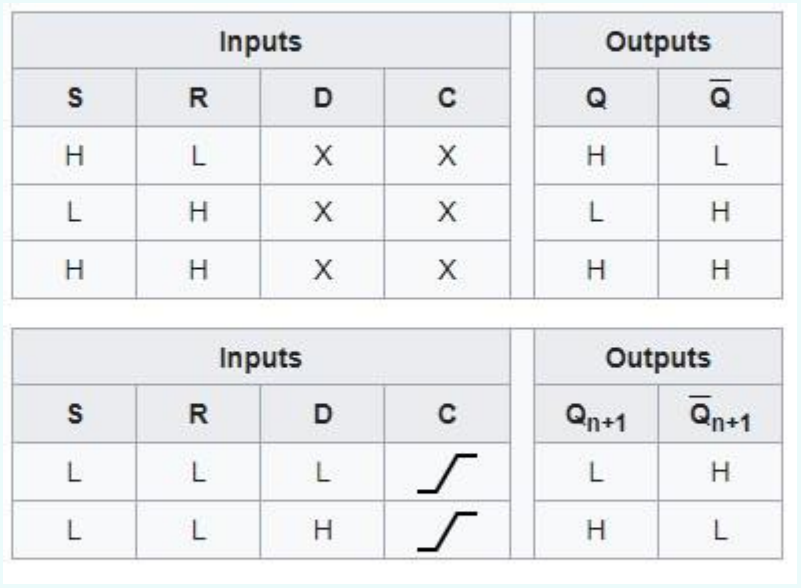
We have used following ICs.

1. 4013

2. 74174

Description of the ICs

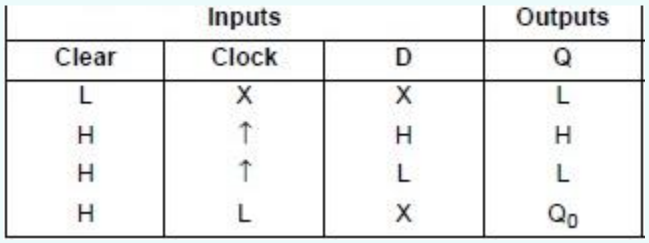
1. 4013 IC:
   The 4013 contains two independent D-type flip-flops with asynchronous set/reset inputs. Whenever the set or reset pins go high, the appropriate output is expressed immediately on the outputs. When set and reset are low, the output shows the data at the input at the time of the last low-to-high clock transition. This is then held until the next transition.



Inputs				Outputs	
S	R	D	C	Q	$\overline{Q}$
H	L	X	X	H	L
L	H	X	X	L	H
H	H	X	X	H	H

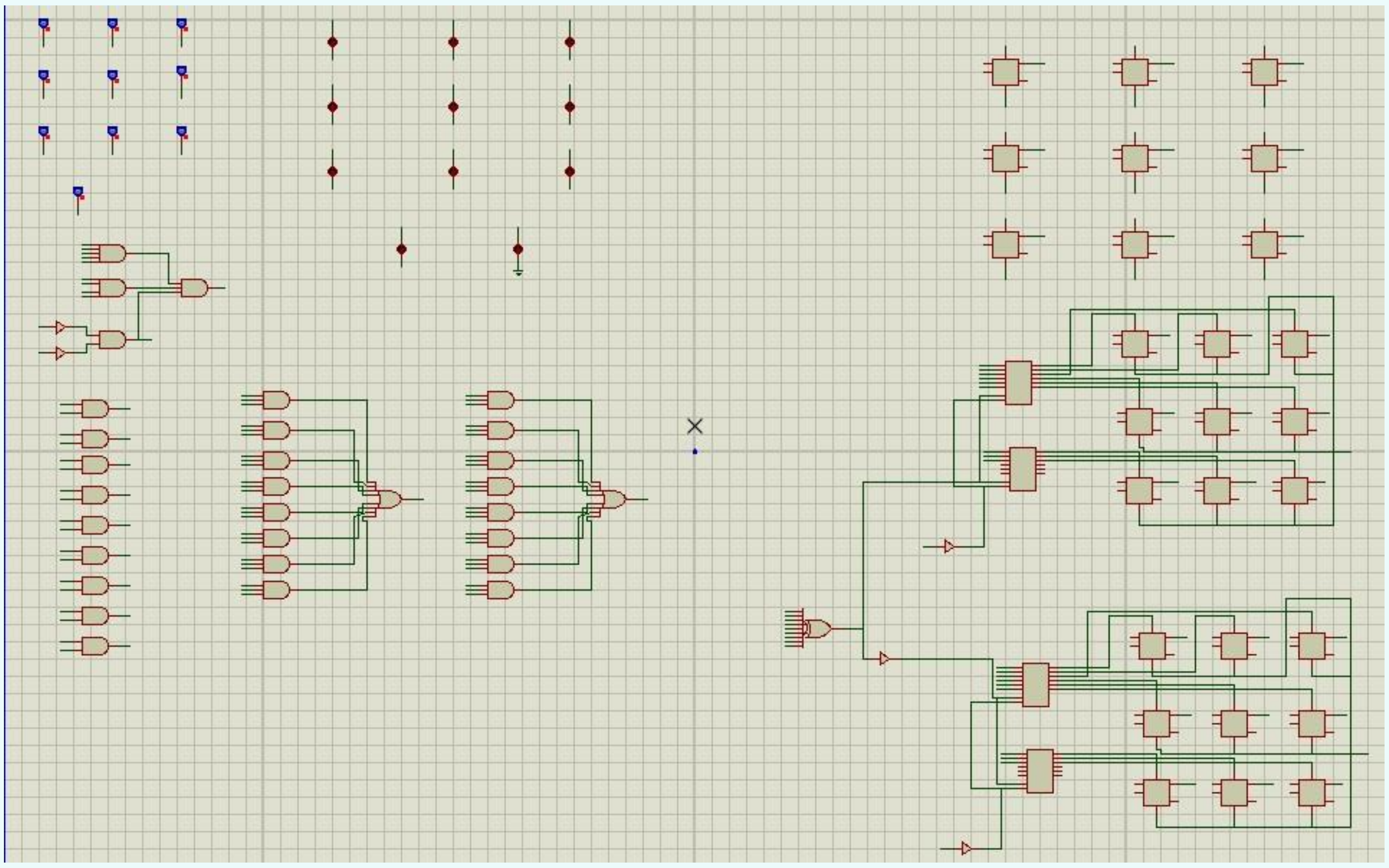
Inputs				Outputs	
S	R	D	C	Q_{n+1}	$\overline{Q}_{n+1}$
L	L	L	↑	L	H
L	L	H	↑	H	L

2. 74174 IC:
   These positive-edge triggered flip-flops utilize TTL circuitry to implement D-type flip-flop logic. All have a direct clear input. Information at the D inputs meeting the setup and hold time requirements is transferred to the Q outputs on the positive-going edge of the clock pulse. Clock triggering occurs at a particular voltage level and is not directly related to the transition time of the positive-going pulse. When the clock input is at either the HIGH or LOW level, the D input signal has no effect at the output.



Inputs			Outputs
Clear	Clock	D	Q
L	X	X	L
H	↑	H	H
H	↑	L	L
H	L	X	Q_0

Circuit Diagram



Input Block:

In our circuit we have used logic toggles for input.

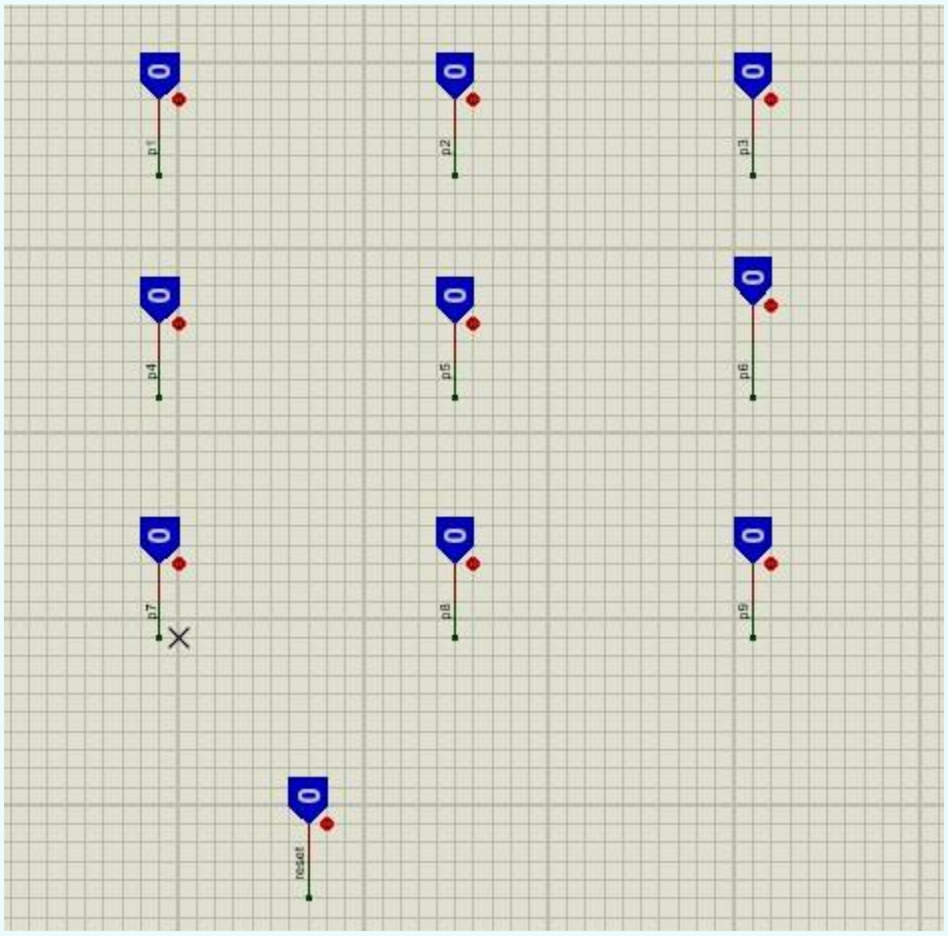


Fig: logic toggles for input

The logic toggles get input from user, these are momentary push buttons, so to know which button was pressed, we use D flip flops (IC 4013) to keep track of input. Output of each button goes the SET of flip flop which sets the output Q to HIGH whenever a button is pressed. As SET, will not become LOW until RESET is pressed the button memory is stored in the flip flops. So, pressing a button twice will not change the output of the flip flops. It is important to impede cheating.

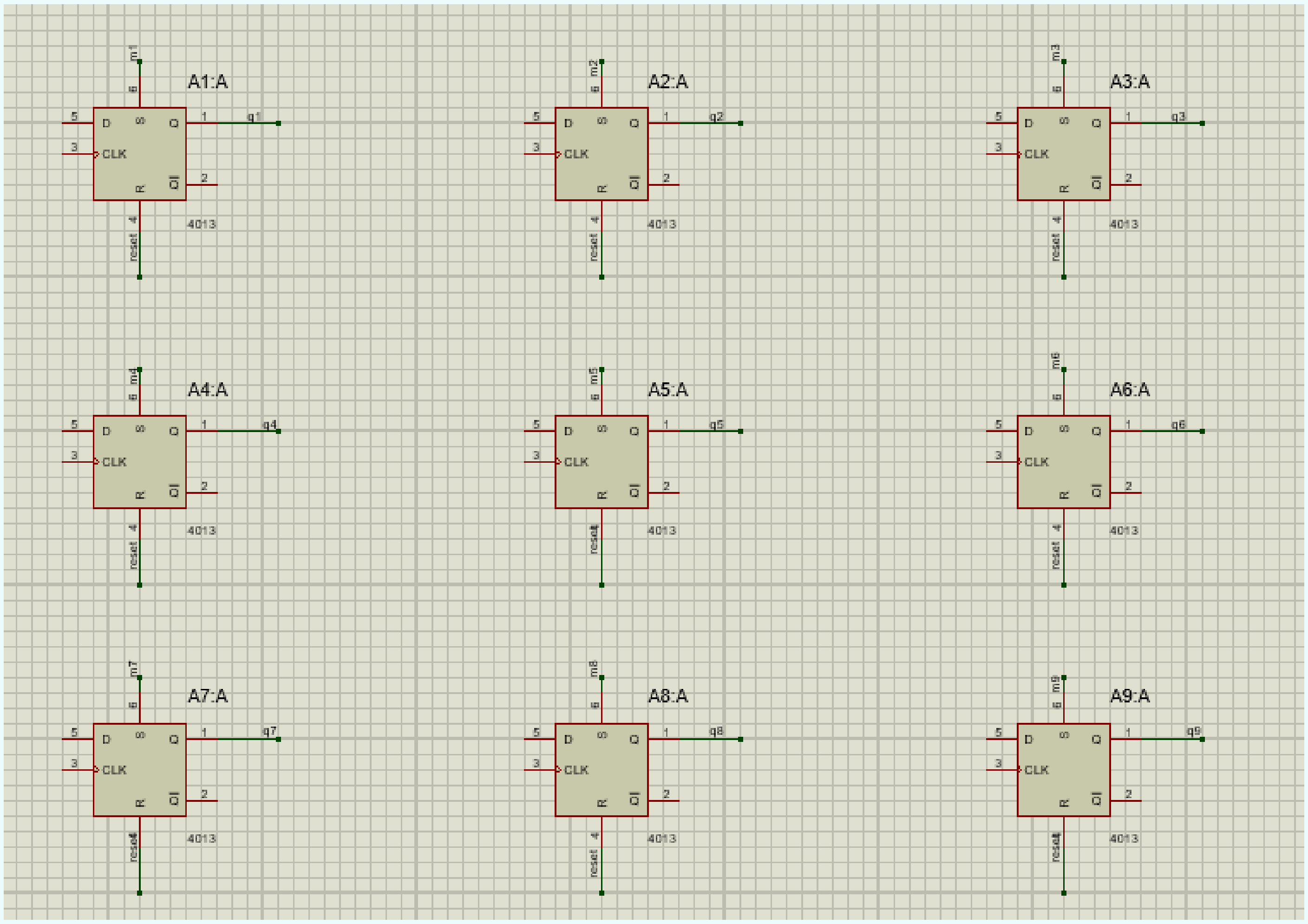


Fig: D Flip flops for storing button memory

Also, we ‘store’ which player’s move it was, below is the schematic we used to achieve this purpose.

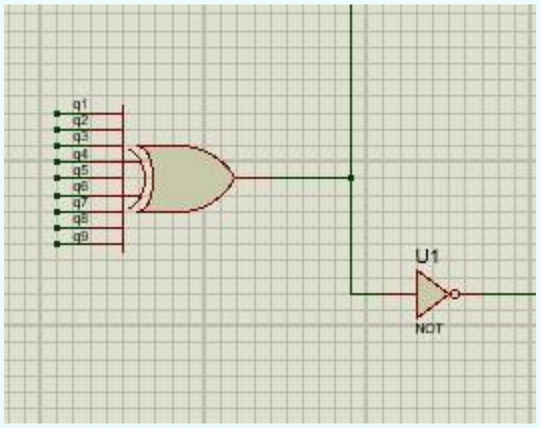


Fig: Player profile generator

Output of the flip flops are connected to the XOR gate of the player profile generator circuit. When first player presses any button, output “Q” of corresponding flip flop goes from “low” to “High” triggering the output “p1” of the player profile unit “High” and “p2” to “LOW”. The next input to the circuit results in opposite. A short description of the circuit is that it will have two lines, “p1” and ‘‘p2”. If it’s player 1’s turn to play, “p1” line goes “high” & “p2” line goes “low” and vice versa.
Note: It is not at all possible for output from both “p1’and “p2’ to go high at the same time.

Winning Logic:

Now we have the inputs, we need to process and find out who won. The game says that if a player gets any of the horizontal, vertical, or diagonal lines full (all 3 for himself) is declared the winner. So, the next task is to find out those winning combinations. For this we used 3 input AND gates. Whose outputs are attached with an OR gate. This setup is duplicated for both Player 1 and Player 2.

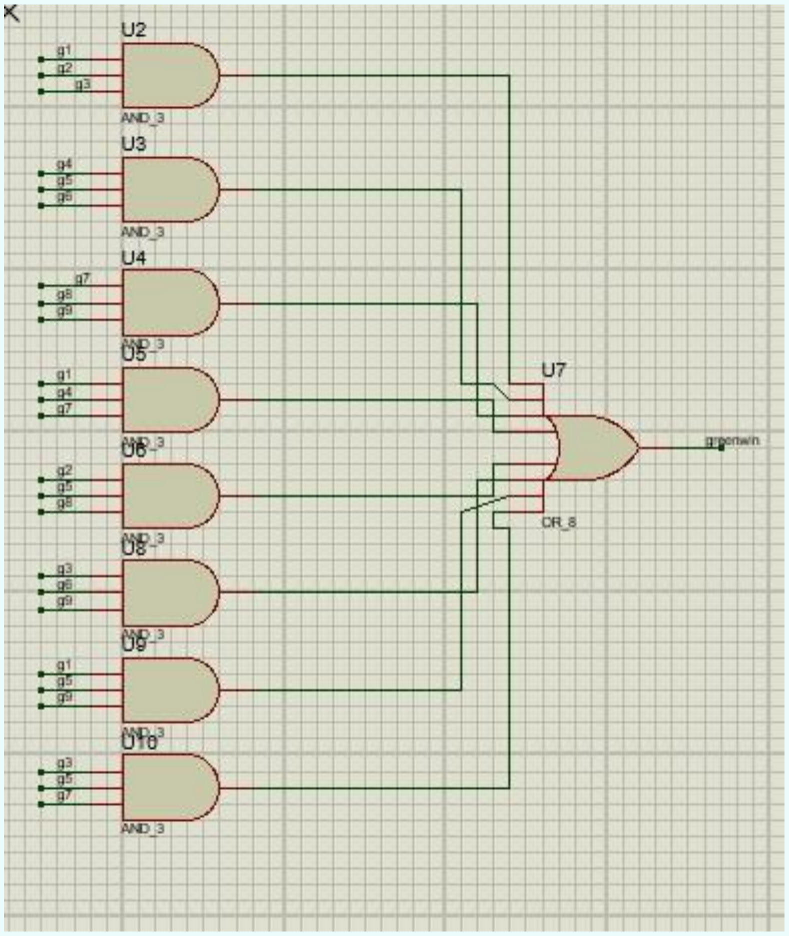


Fig: Green Player winning conditions

If the output of this block is high, means we have a winner! We have used a bi-color LED to indicate the winner. According to our algorithm, this LED will turn red if player red wins and will turn green if player green wins.

Display Block:

This block has 3x3 bi-color LED's arranged in a matrix fashion. The outputs of circuit are given to the corresponding LED. Player 1 output is given to one leg and player 2 output is given to another leg of the LED. So, if the LED glows "RED" it means that that move was made by player 1 and if the led glows "GREEN" that move was made by player 2. If one of the two led's below the matrix turns into "RED" it means red wins! And vice versa. If yellow led glows it means that the game is drawn.

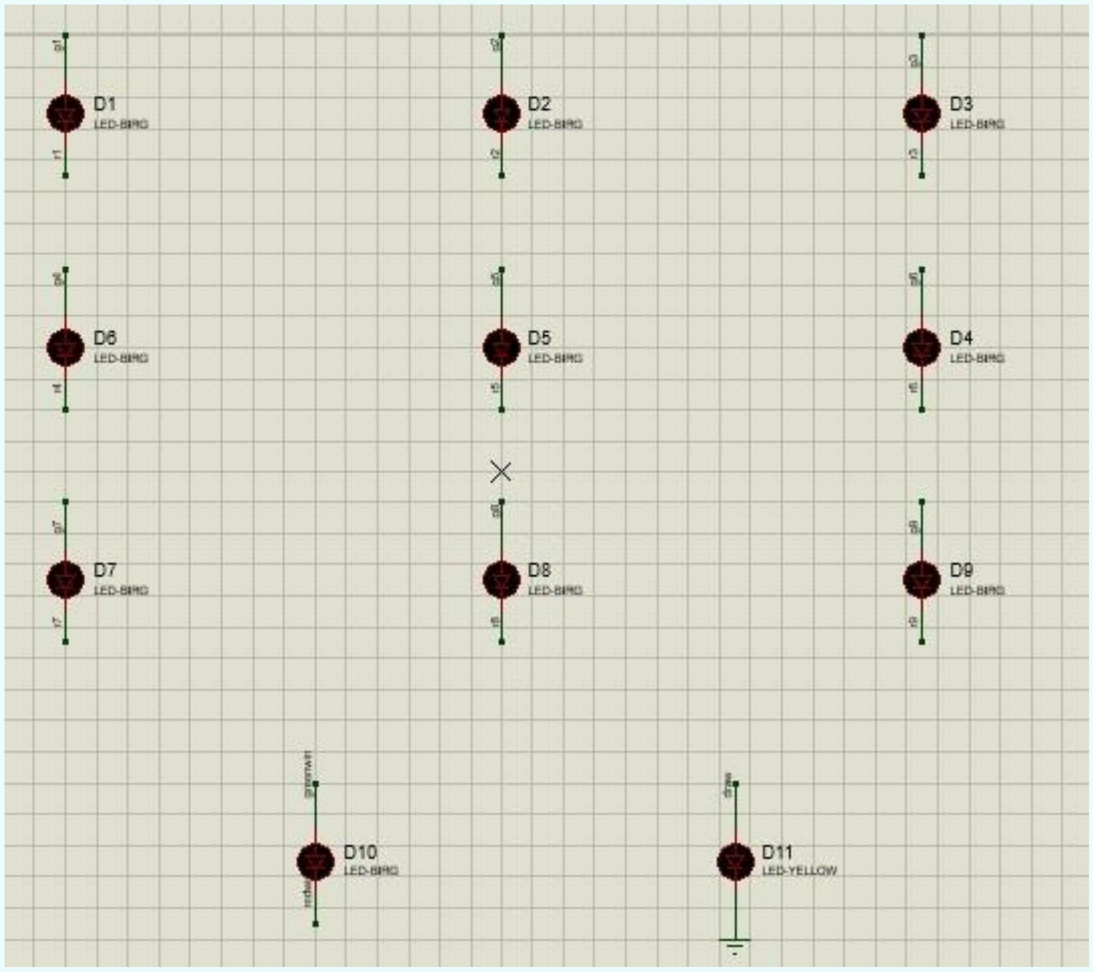


Fig: Display Block

Thank You