Islamic University of Technology

Digital Electronics 4307 Project: Snake & Ladder Game

Phase: A,B,C,D,E

Section : A
Department : EEE
Course No : EEE 4307

Student IDs

220021108 SHAZZAD AHMED

220021116 FAIAD FAISAL SARTHOK

220021123 FAHIM SHIFAT

220021127 AHNAF SAKIF

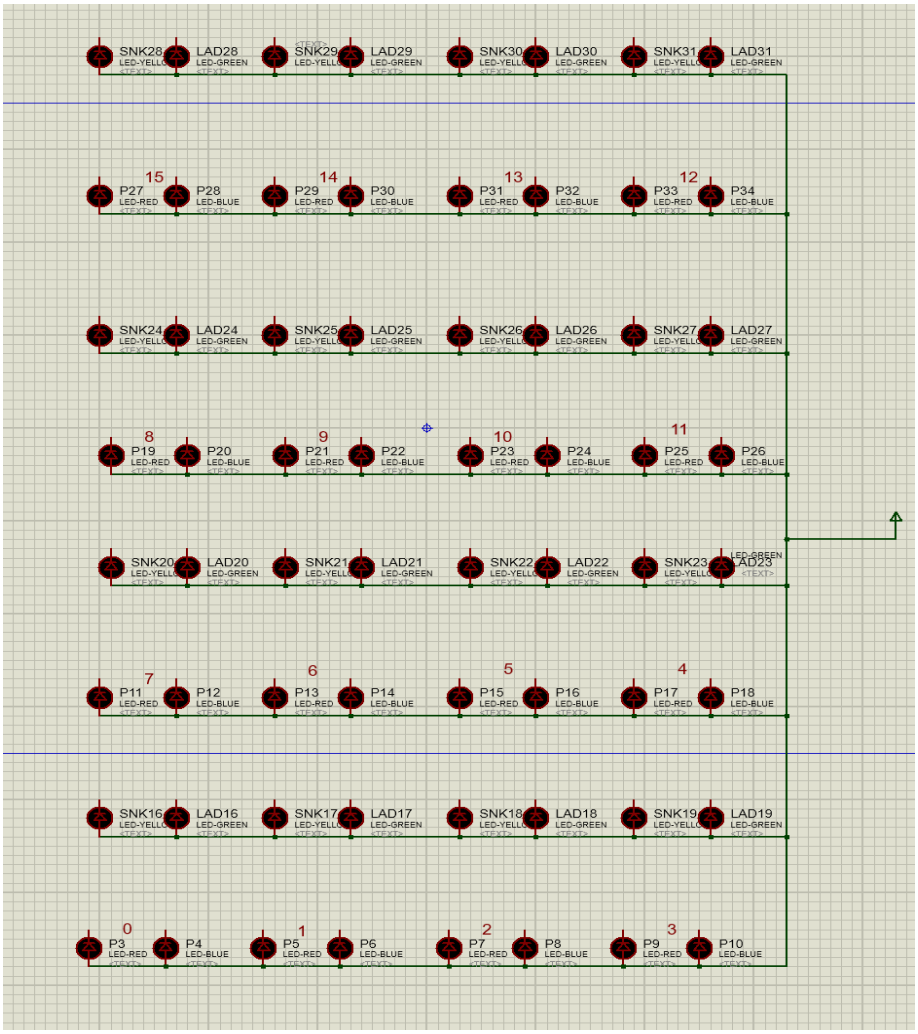
220021130 MAHIB ABTAHI

220021151 ABDULLAH JUBAYER

220021157 HASIB AHMED

Board module

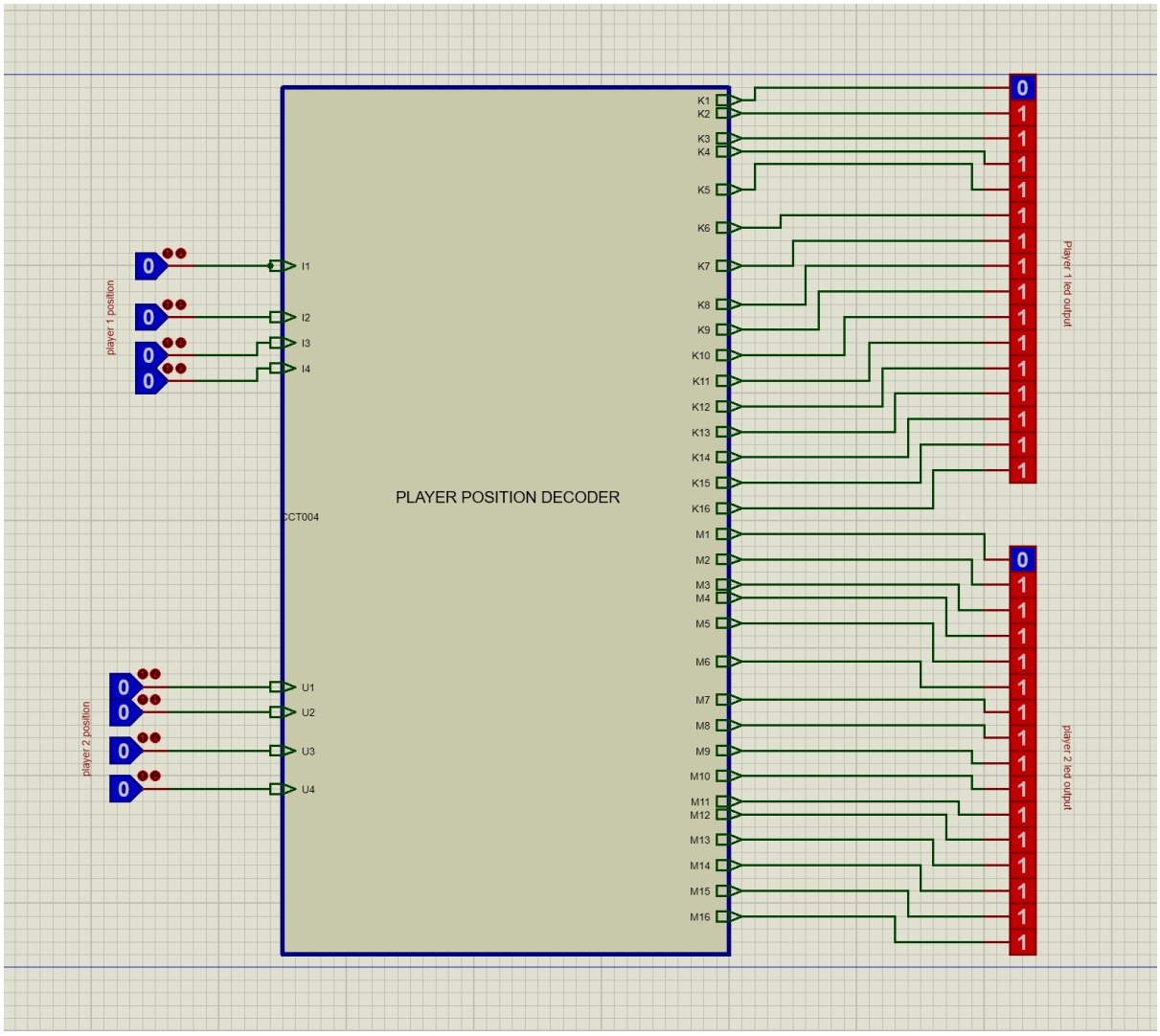
For Board module, single LEDs were used. Where the red led is for player one's position, blue is for player two's position, green is for ladder bottom and top, and yellow is for snake head and tail. There are 16 blocks total in this board where each block contains four LEDs for outputting the full information of the game. Every anode terminal of the LED is common powered since the LEDs are designed as active low components. Whenever a 4-bit binary number is taken as input for the LED board the corresponding position to that binary equivalent will be lit up.

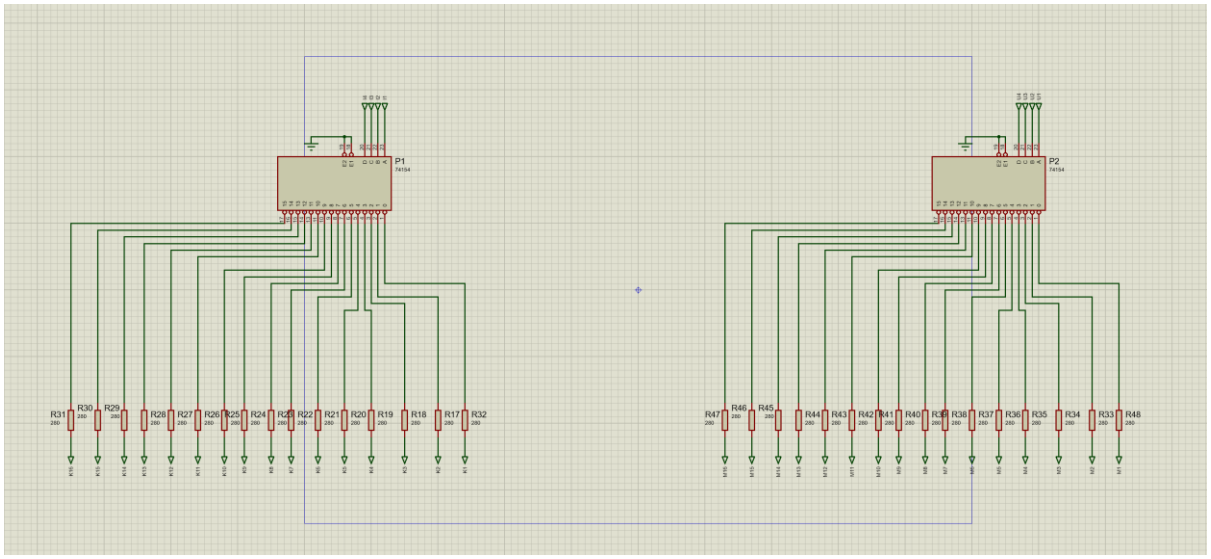


Player Position Decoder

Here Two decoders are connected through resistor. Since the red and blue color is selected for the player one and two respectively. Both

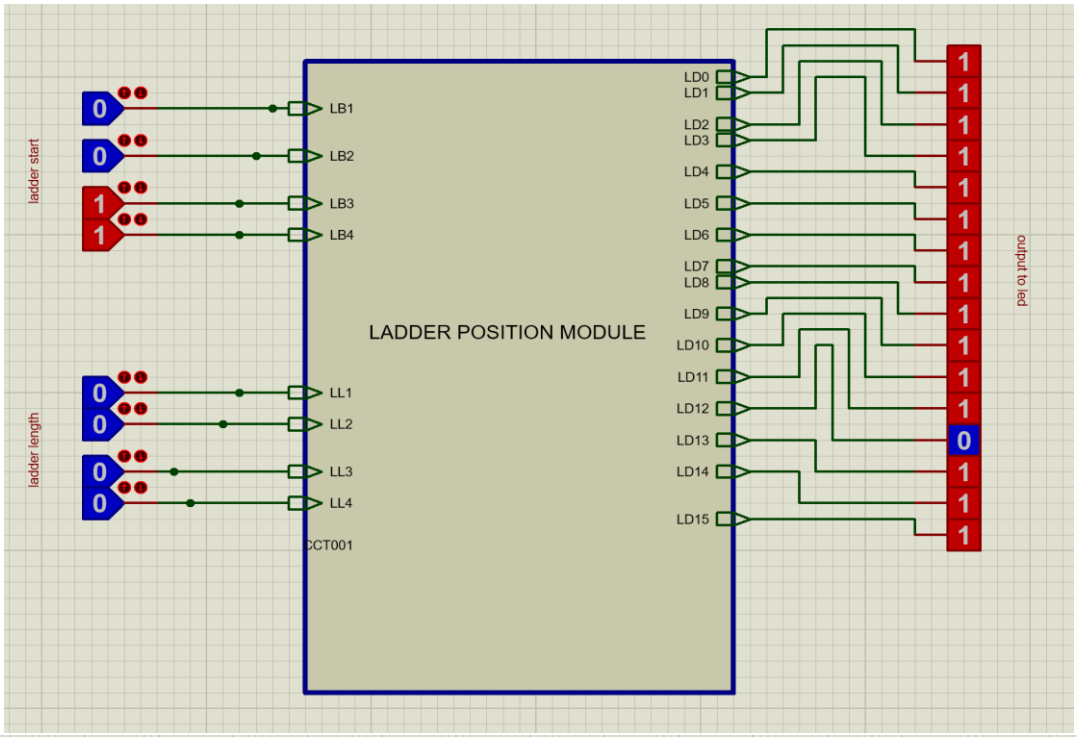
player position will start from zero and when we get a new position of any player, this will go through the decoder and it will light the led where it needs to be. Here the resistors are used to protect the led and get sufficient voltage to lit them brightly and here the led board is connected with power pin in common anode connection so the LEDs are only lit when the cathode voltage will lower than the cathode ones. Here we had earthing on the enabler pins of the input side as here we don't need any of them.

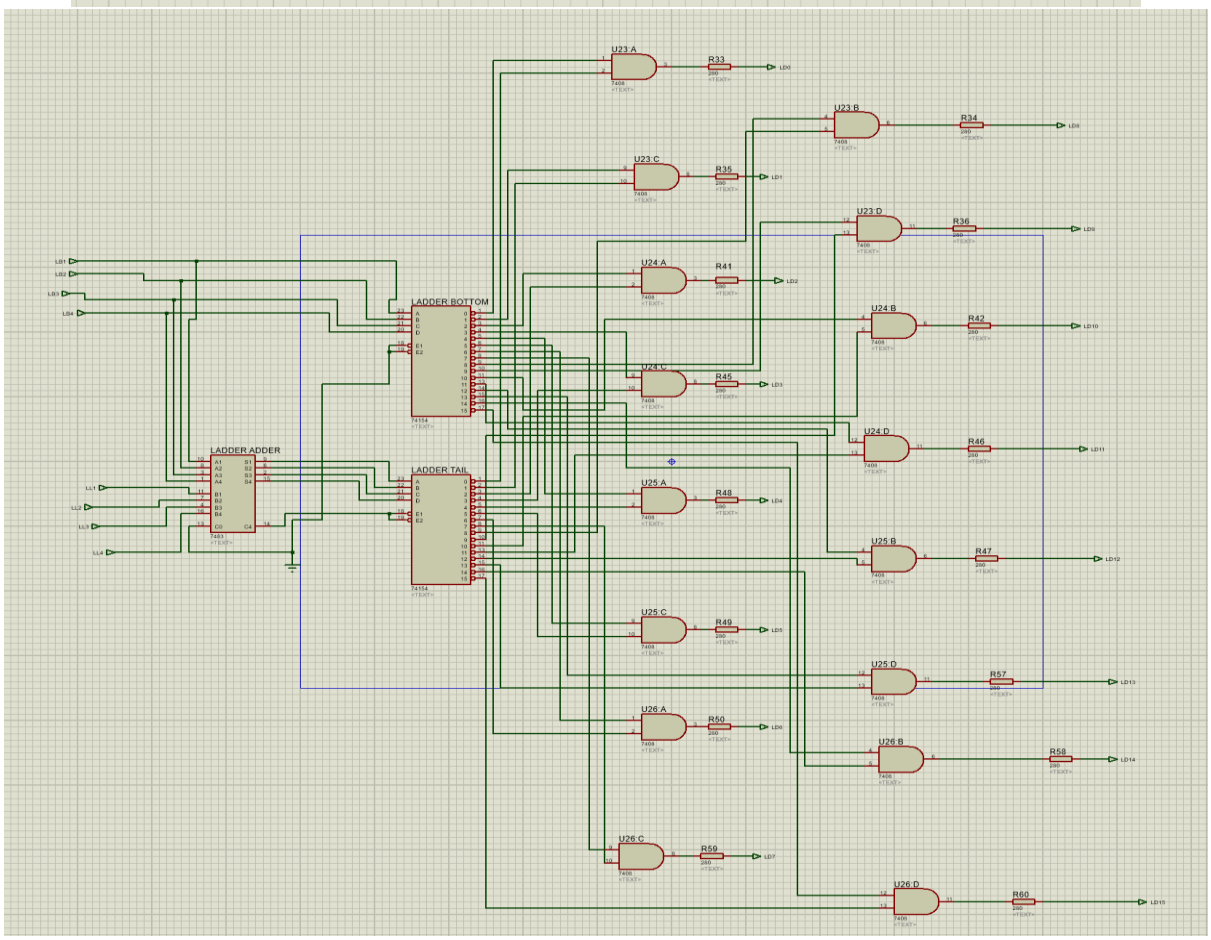




Ladder position decoder module

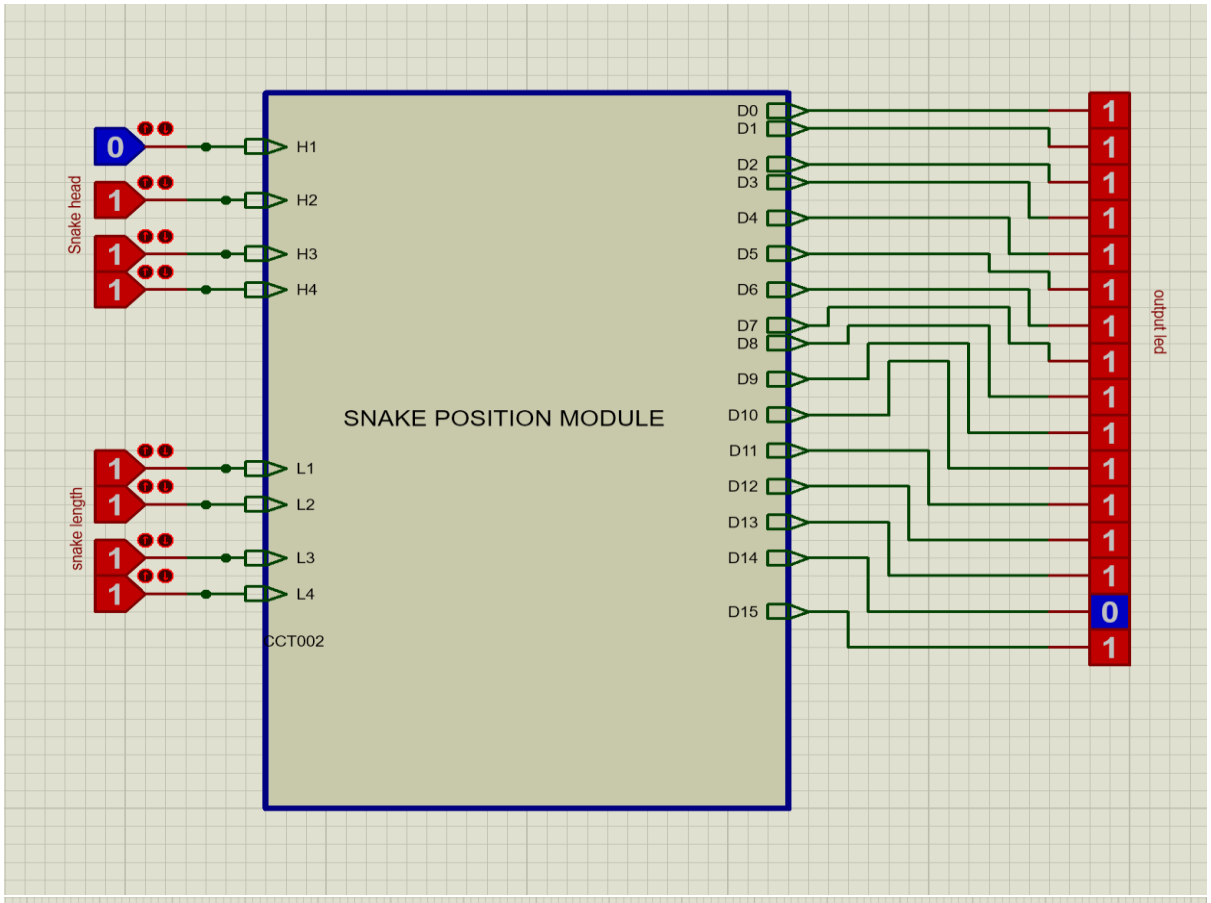
We first build a sub-circuit to design the ladder position decoder consisting of two 4-bit inputs and 16 outputs. LB1, LB2, LB3, and LB4 are the four inputs representing the user-defined ladder bottom position and LL1, LL2, LL3, and LL4 represent the user-defined ladder length. As we have 16 blocks, each with one LED to represent the ladder position, we need two 4×16 decoders. To implement this logic, we first take the four inputs of the ladder bottom position and transfer the inputs into a 4×16 decoder (ladder bottom decoder- IC74154). The output of this decoder will give us a 16-bit ladder bottom position. Then we let the ladder bottom position inputs and ladder length inputs pass through a 4-bit adder (IC7483). The output is then transferred to another 4×16 decoder (IC74154-named as ladder next position decoder). This decoder will represent the 16-bit ladder's new position. In Proteus, the decoder works on active low, and for the diode to be on, we have to keep its positive terminal at high voltage and negative terminal at low voltage. Then we placed an and get in combining two connections from the same pin number of the both decoders. So it will give output only when one of the inputs is active (Zero). Finally, we get 16 outputs which are then connected to 16 LEDs properly to represent the proper ladder position module. When ladder top is greater than 15 then only ladder bottom will light up.

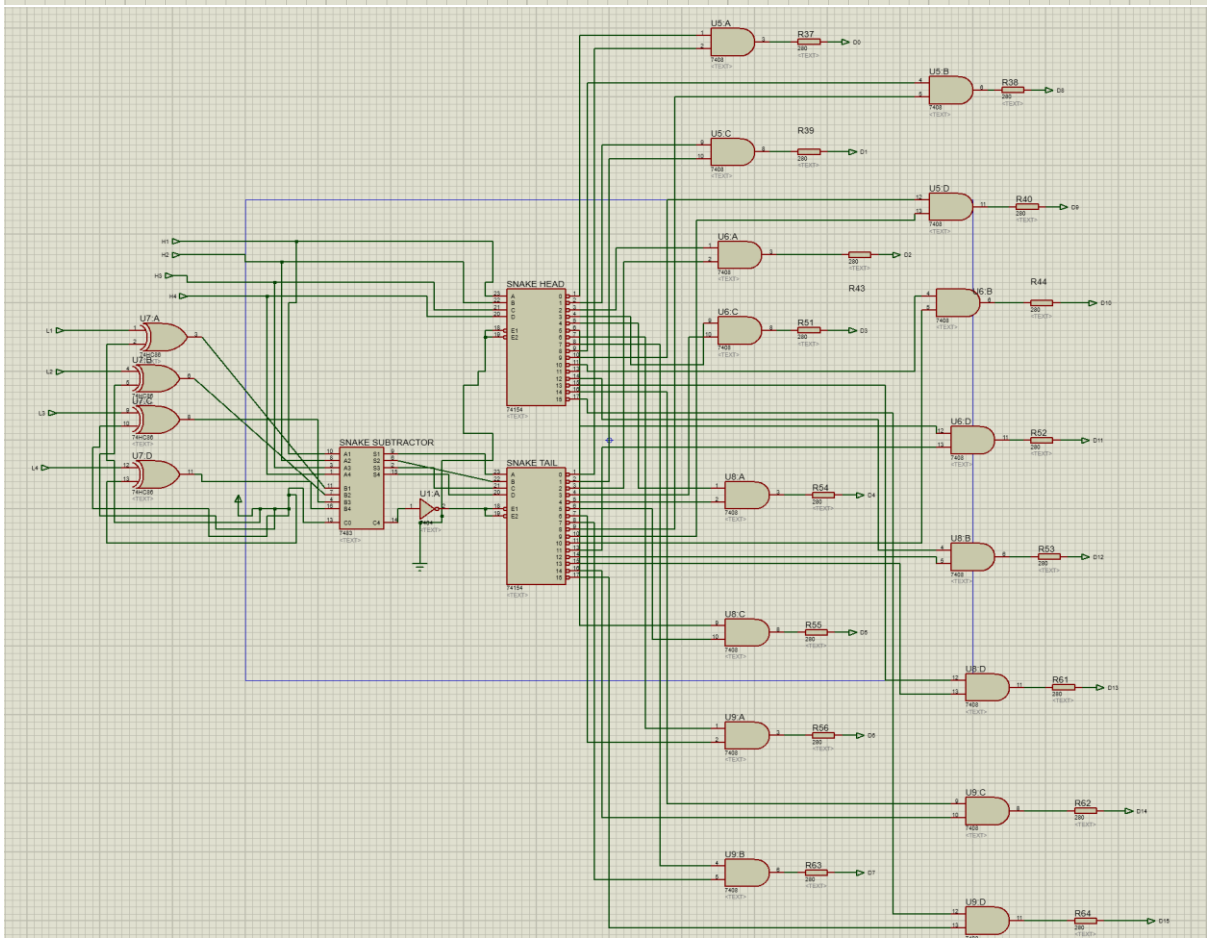




Snake position decoder module

The logic circuit of the snake position decoder module is almost similar to that of the ladder position decoder. We first build a sub-circuit to design the snake position decoder consisting of two 4-bit inputs and 16 outputs. H1, H2, H3, and H4 are the four inputs representing the user-defined snake head position and L1, L2, L3, and L4 represent the user-defined snake length. As we have 16 blocks, each with one LED to represent the snake position, we need two 4×16 decoders. To implement this logic, we first take the four inputs of the snake head position and transfer the inputs into a 4×16 decoder (snake-head decoder- IC74154). The output of this decoder will give us a 16-bit snake head position. Then we let the snake head position inputs and snake length inputs pass through a 4-bit subtractor (typically adder-IC7483, we implement adder along with X-OR gate so that the adder will act as a subtractor). The output is then transferred to another 4×16 decoder (IC74154-named as snake-head decoder). This decoder will represent the 16-bit snake's tail. Then we placed an and get in combining two connections from the same pin number of the both decoders. So, it will give output only when one of the inputs is active (Zero). Finally, we get 16 outputs which are then connected to 16 LEDs properly to represent the proper snake position module. When the snake tail position is below zero only the snake head will light up.

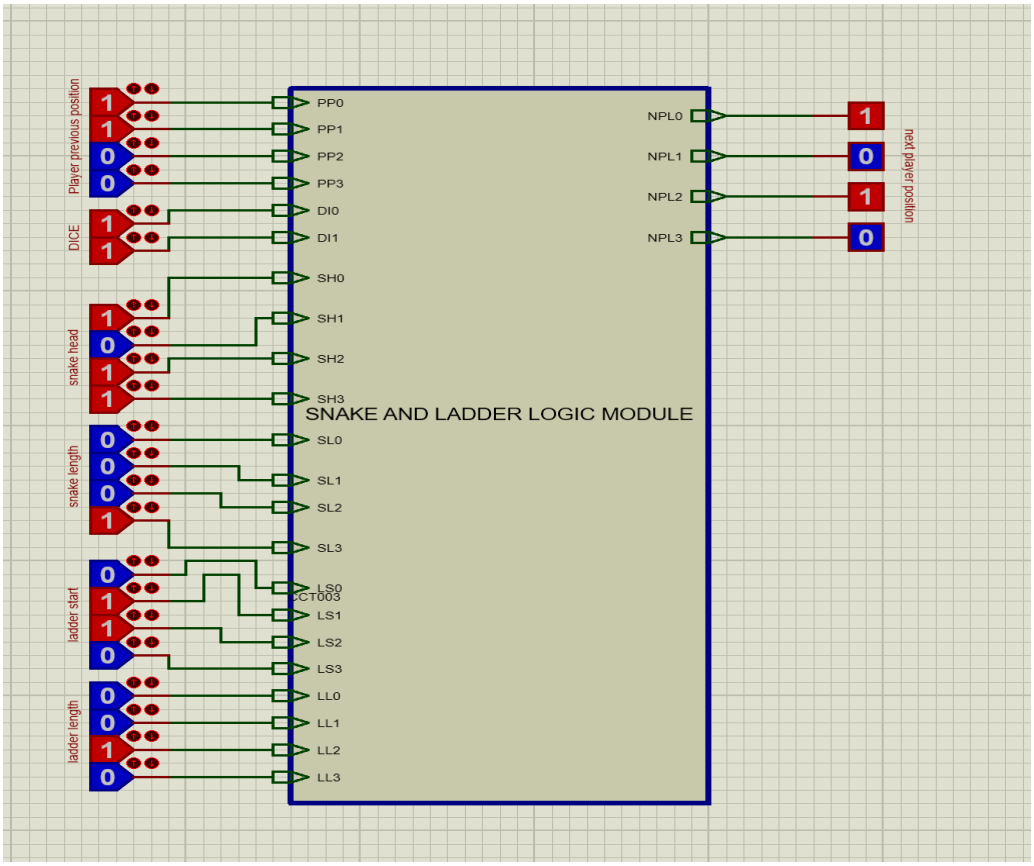

Snake head
H1
H2
H3
H4
snake length
L1
L2
L3
L4
CCT002
SNAKE POSITION MODULE
D0
D1
D2
D3
D4
D5
D6
D7
D8
D9
D10
D11
D12
D13
D14
D15
output led

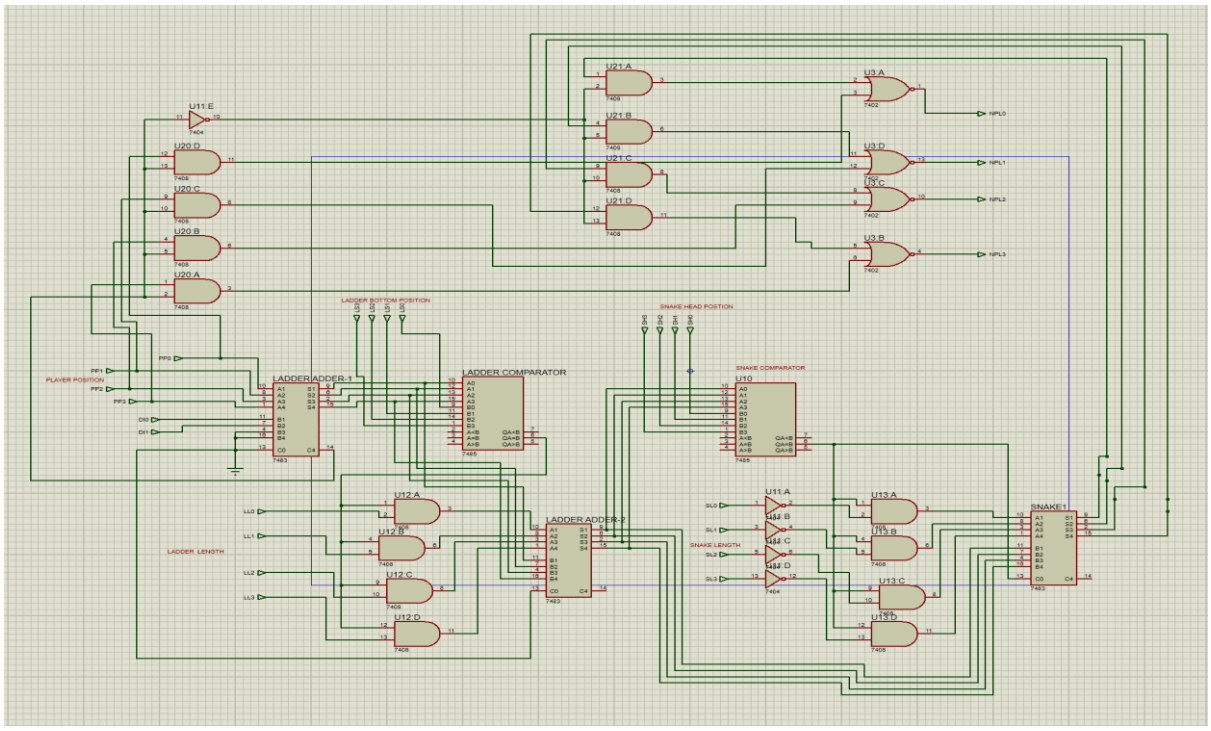

SNAKE HEAD
SNAKE TAIL
SNAKE SUBTRACTOR

Snake-Ladder logic module

To make the snake-ladder logic module we first build a sub-circuit consisting of 22 inputs and 4 outputs along with a carry output. Firstly, we add the player's current position input to the dice input using an adder (IC-7483). Then we will get an interim player position which has to be compared with the ladder bottom position and snake head position respectively. According to the lab manual if both a snake and a ladder condition might be met, prioritizing ladder movements over snakes to maintain. So, we first compare it with the ladder's bottom position. After comparing, if the interim player position matches with the ladder bottom position, then compare will output 1 which is connected to one of the terminals of an AND gate, and the other terminals are connected with the 4 inputs of the ladder length. Then the outputs from the AND gate have to be added with the player interim position, that's why we use an adder (IC-7483, named LADDER ADDER-2). If we don't use AND gate and if the interim player position doesn't match with the comparator, then the adder will keep continuing to add the interim player position with the ladder length which will be a major error. So, to mitigate this problem, using the AND gate is crucial because in this case, the comparator will give output 0, and from the truth table of the AND gate if one of the inputs is 0, the output will be 0. It will not add ladder length. So, AND gate satisfies both conditions. So, using the AND gate is a must in this case. Now, we find the new player position which has to be compared with the snake head position, the logic is almost similar to before. But in this case, if the comparator (named snake comparator) output is 1 then the current position has to be subtracted from the snake length (we use adder IC-7483 as subtraction operation, named snake subtractor), and if the comparator (named snake comparator) output is 0, then it will simply give the current player position as output. Here

if players next position is above 15 than updated position will be equal to previous position.





Contributions:

· **Shazzad Ahmed**: Reviewed the entire project and worked on enhancing its efficiency.

· **Faid Faisal Sarthok:** Conducted a thorough review of the project to improve its overall efficiency.

· **Fahim rezwan Shifat:** Developed the Snake position decoder, Ladder position decoder, and the logic module for both Snakes and Ladders.

· **Ahnaf Sakif:** Implemented the player position decoder.

· **Mahib Abtahi:** Reviewed the entire project and focused on making it more efficient.

· **Abdullah al Jubayer:** Designed the board module and assisted in debugging phases A and B. Implemented and designed logic circuits it the ladder is greater than 15 and snake tail is less than zero. Also, if players next position is above 15 than updated position will be equal to previous position.

· **Hasib Ahmed :** Contributed to problem analysis and performed a comprehensive review of the project.

Phase_C :

Dice module :

In the Dice module We have used two D flipflop and use a 555 timer ic to generate random signal.In The 555 timer ic we can change the frequency by fixing the values of capacitor and the two resistor.The formula used here is

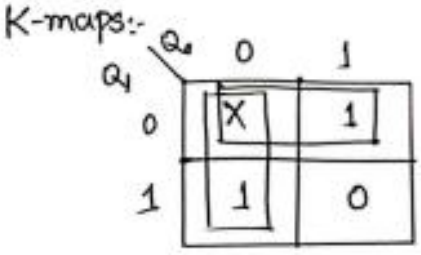
$$\text{Frequency} = 1.44/\{C*(R1+2R2)\}$$

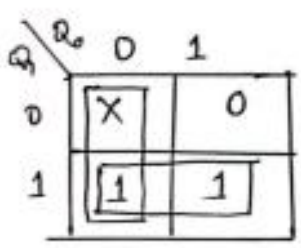
In this module We have to express 1,2,3 and for expressing these digit we need 2 bit number system which is the reason of using two SR flipflop. First a square signal generator having a suitable frequency for showing the output properly is connected to the CLK of two D flipfops .

Present State		Next State		Flipflop Input	
Q_1	Q_0	Q_1	Q_0	D_1	D_0
0	1	1	0	1	0
1	0	1	1	1	1
1	1	0	1	0	1

After doing the State table and K-map we have got the boolean functions like :

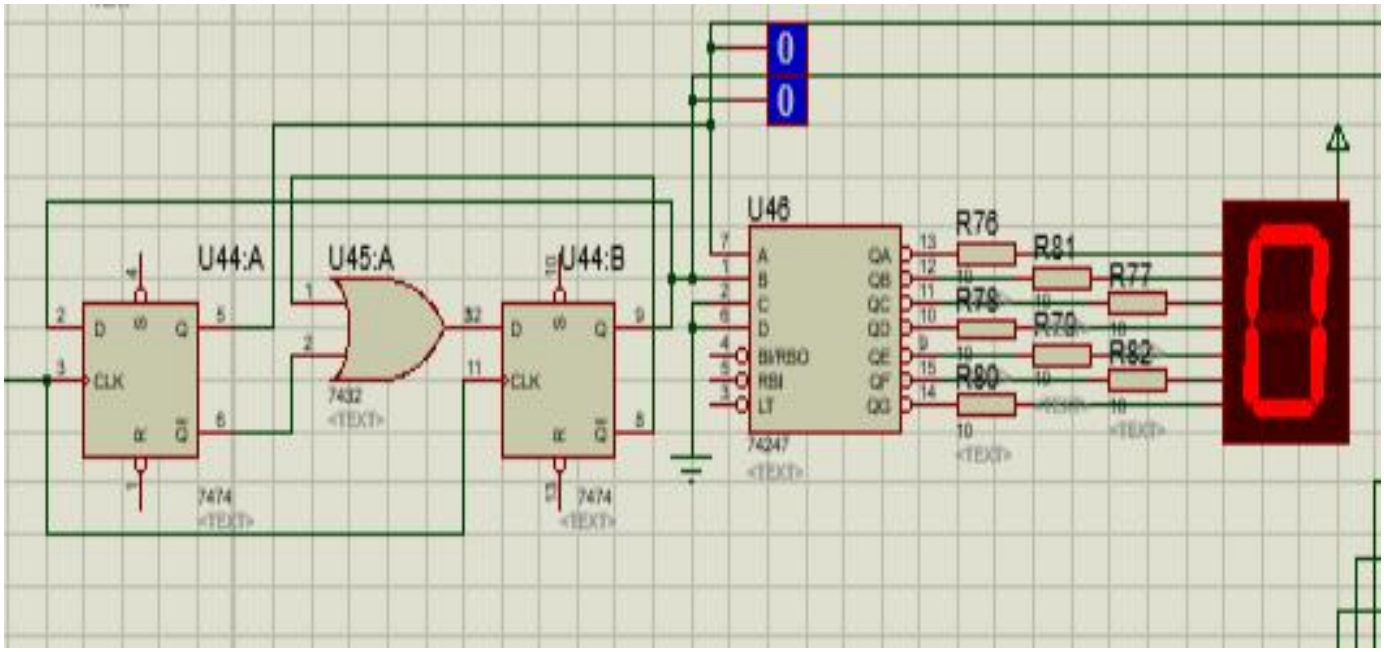
K-maps:-

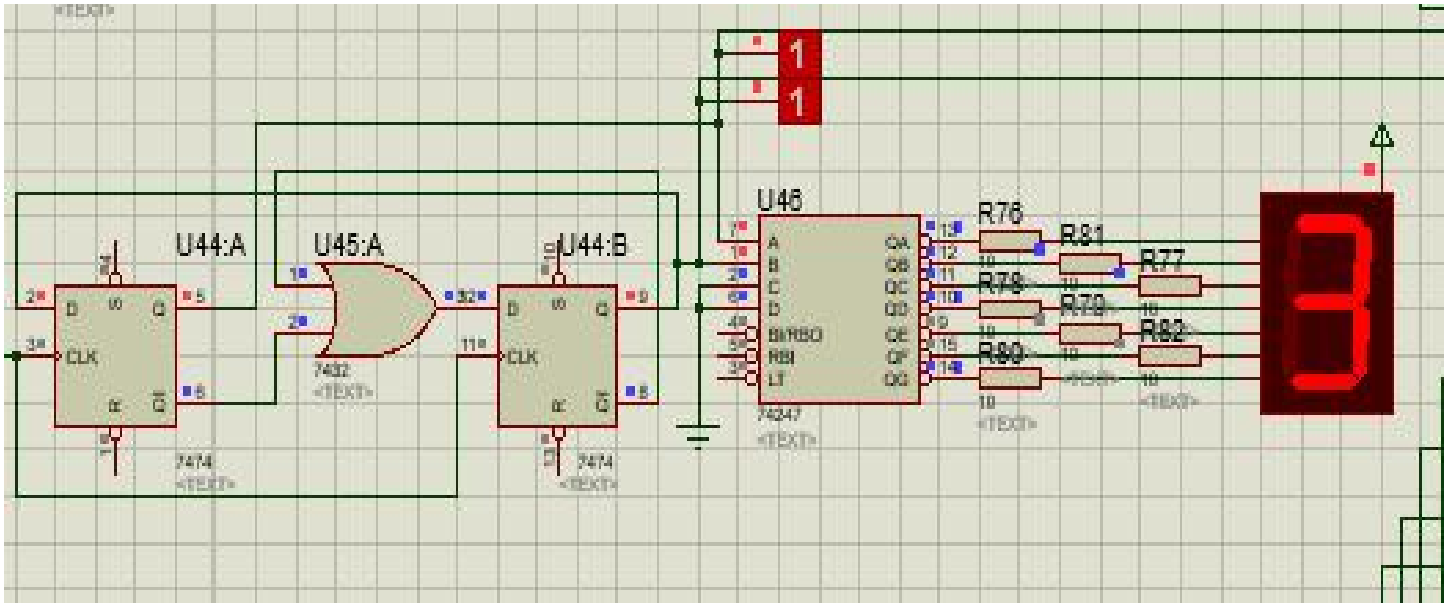




Implementing boolean function:-

To get this desired output we have joined all the connection between the D flipflops according to the boolean functions.When the switch is pressed, “1” signal is passed through the signal generator and both of the flipflops will be clocked that will show the value 1,2,3 simultaneously . When the switch is closed, it will stop at certain value and this will pass as Dice value through the “Snake and Ladder Logic Module”.Here a seven segment display is connected to the second D flipflops to show the Output and certain value of resistor is used to pass the signal accurately.

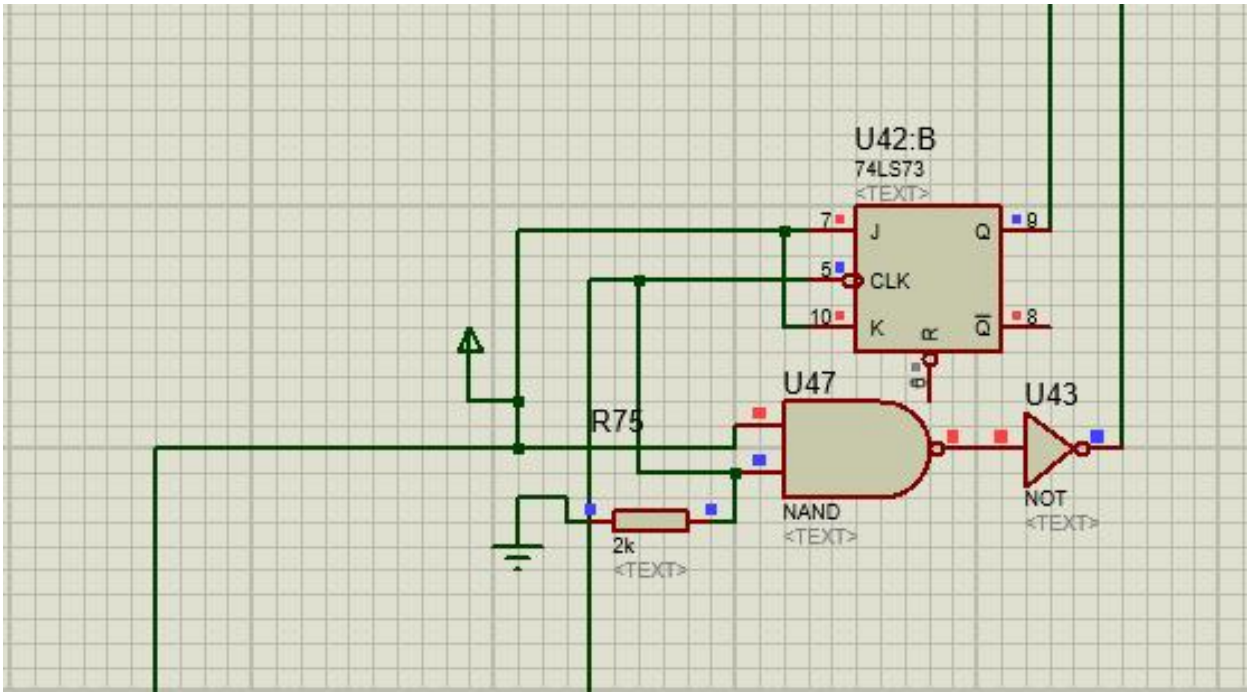




Phase_D+E :

Player Selector module :

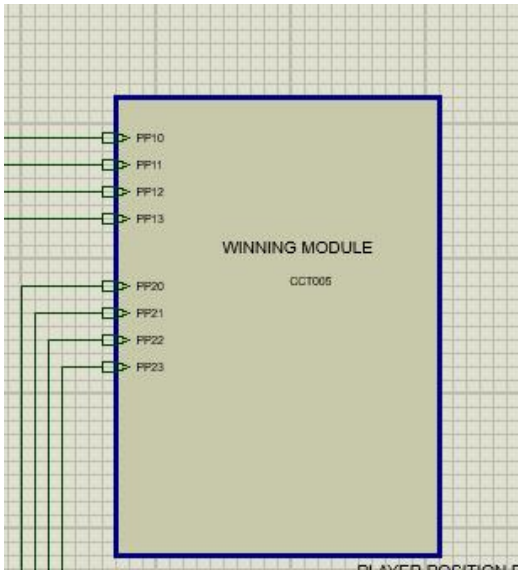
In this module we have used J-k flipflop to toggle the output as here we have set 0 for player and 1 for the second player. For the toggle operation of this Jk-flipflop here we have to keep 1 in both of the inputs J and K always and the Clk is connected to the power pin via a switch. When this switch is on , The clk of the flipflop will get 1 and it will toogle the output of flipflop and switches between player 1 and 2 . We have used only 1 output pin like Q as we need only one bit result to determine player 1 and 2 .

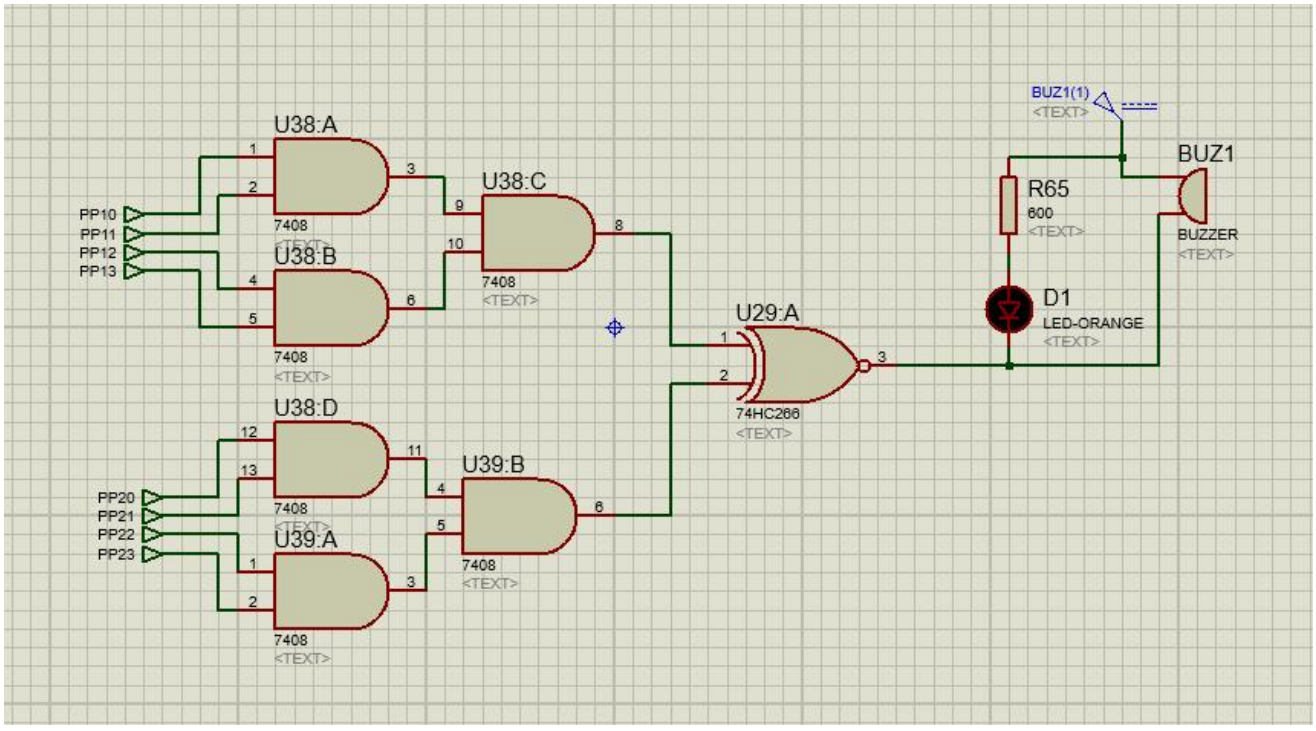


Winning Module :

In this module , we have used to comparator for two players and when any of the player position get 15 or in the last box of the game , the comparator will match the input A with the fixed value B(15) and it will give 1 through the QA=B(2nd Output pin) . Both of the players gets position 15 at a time isnt possible ,So if any of them get that

position it give output 1 the other one will be 0 .Both of the pins are connected with the X-NOR gate as it will give 0 only any one of the input pin will give 1 . At last a Buzzer and a led is connected with a DC source. When any of the player reaches winning position , the orange led will lit as it's cadhode portion have signal 0 and the buzzer will also buzz for the same reason.

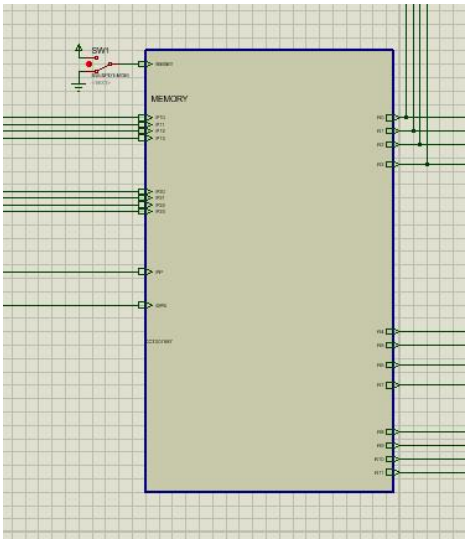




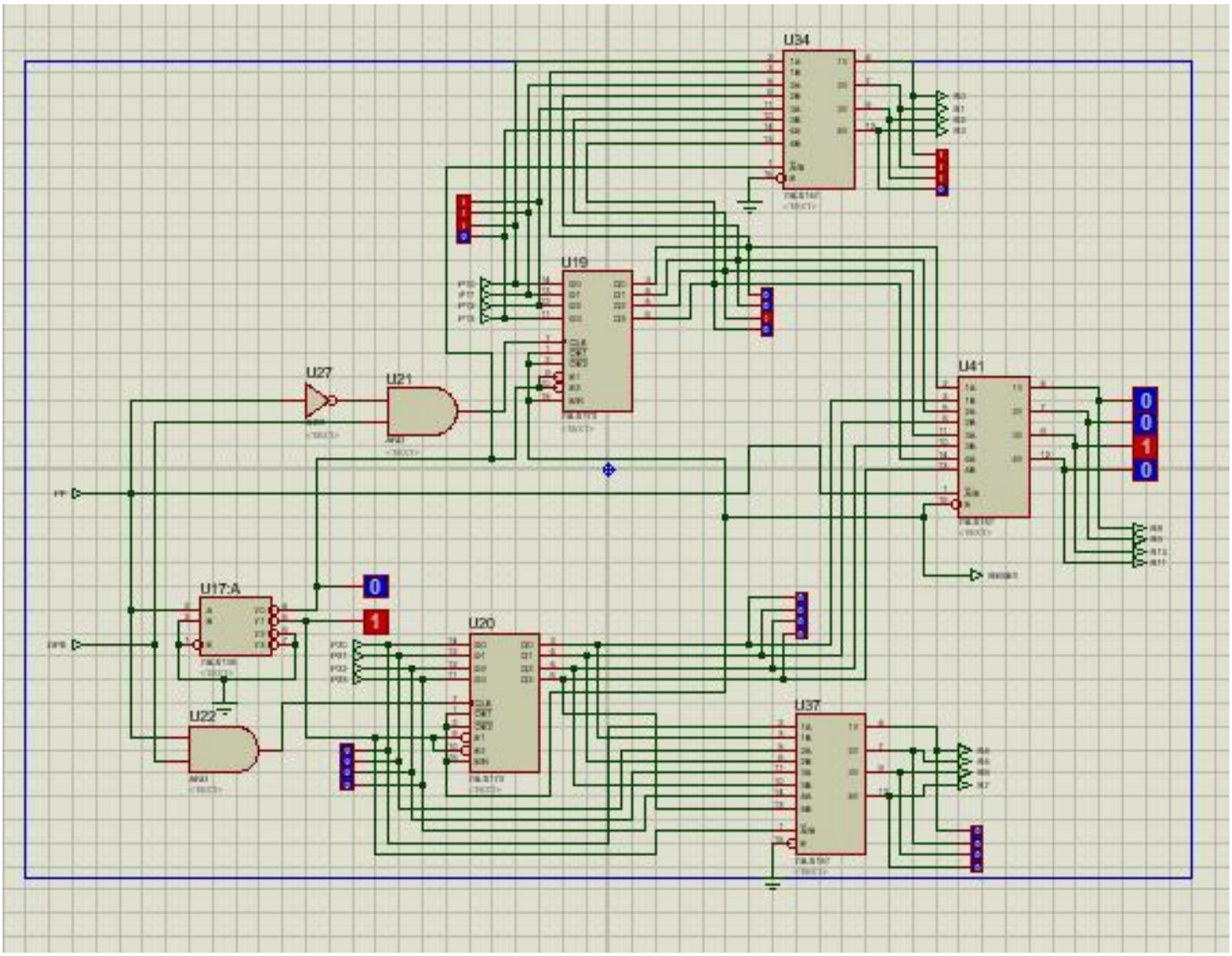
Memory Module :

Here we have used two shift register to store the previous data of each player . In the memory module , the player position pin and the dice pin connected with a 2x4 demultiplexer.here the one input pin and the enable pin (active low) of the multiplexer is connected with ground.Here only “A” pin connected with player selector module as it changes from 0 to 1, the output of the mux will also swap between 1 and 0 . By using this logic we can enable any one of shift register at a time . Here we have also used a NOT gate to player position and passed through with dice input in a AND gate which can control the clock of one shift register. Ont the other hand , The other shift register have a same connection except tha NOT gate as any one shift register get clock at a time .

The first two multiplexers are used to display the real-time output of players 1 and 2. The last one is used to alternate between players 1 and 2. First and foremost, the multiplexer presents the real-time output of player 1 as it occurs. We utilize the demultiplexer's output. When player 1 is playing, the demux output is 0, as 0 is sent to the mux selector. For zero, player 1's current location is displayed on the output. When it changes to player 2, the demux output for player 1 remains 1. As a result, player 1's location is saved in memory, and mux displays the memory output. Because of this mux output selection, it remains the same for player 1 and displays the player's real-time output. It functions similarly for player 2 mux as it does for player 1.



The last multiplexer is used to take any one shift register data at each play ,it will go to Snake and Ladder Logic module.



Reset Module :

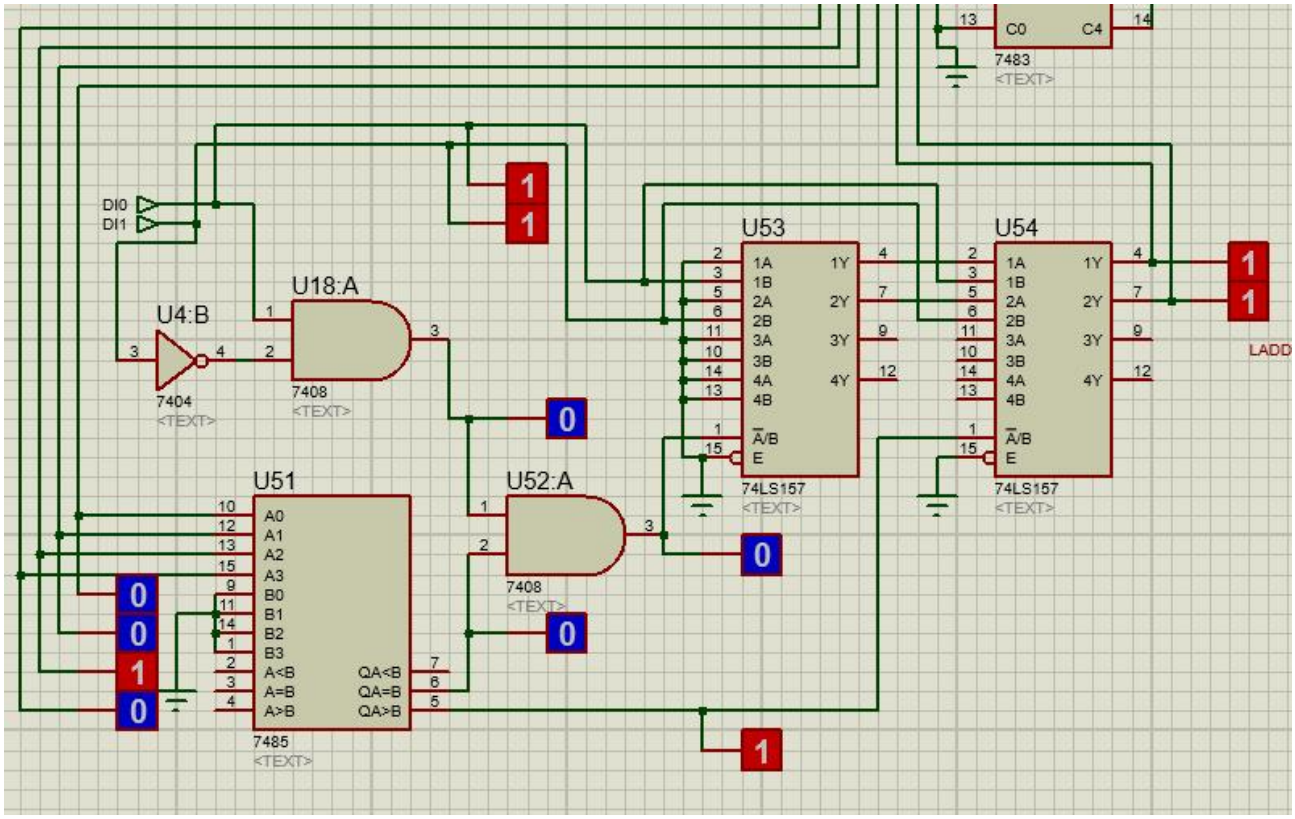
In the memory module , the two pins of Shift register name OEL1 ,OEL2 (active high) and the enable pin of last multiplexer is connected with SW-SPDT-MOM switch . The one of the terminal is a Power pin and the other one is ground . When switch is connected with power terminal, last multiplexer used in the memory module disabled and the two shift register is cleared. And when the switch is

connected to ground, mux will active and the shift register keeps the cleared value. (See the previous two screenshots)

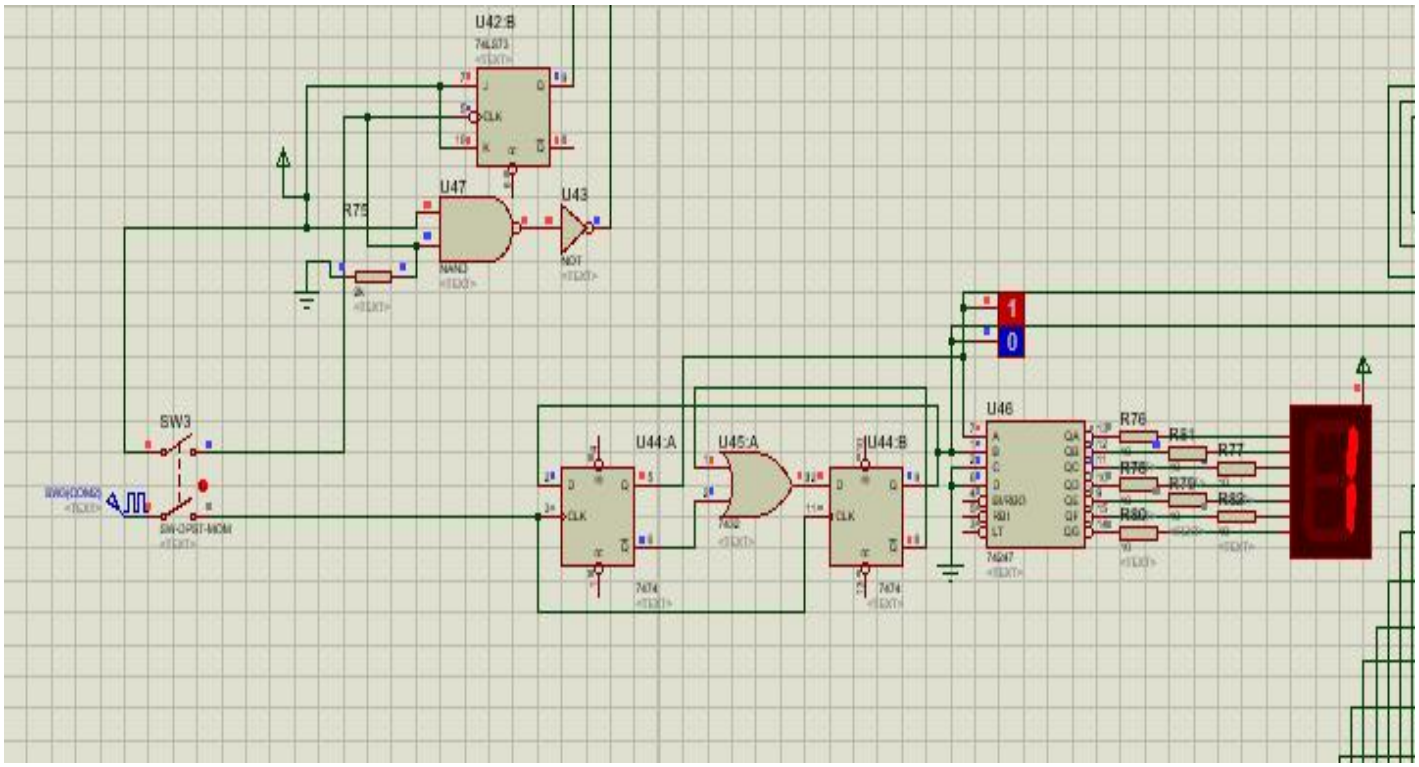
Special Feature :

1.**To start the game with a dice value of one**:

We utilize one comparator, two multiplexers, two AND gates, and one NOT gate. The comparator compares the player's current location to zero. If the player's position is equal to zero, output A=B returns one and A>B returns zero. Here, the NOT and AND gates return 1 if and only if the die value is 1. Another AND gate returns 1 if the dice value and player position are 1 and 0. The first multiplexer outputs the dice value if the AND gate returns 1, and 0 otherwise. If A>B=0, the second multiplexer returns the first multiplexer's output. For the rest, the dice value is output. So dice values function like this: If the dice value at the beginning of the game is one, the game will begin.It is implemented in the Snake And Ladder logic module.



2.Double switching in a single push : When we connect All the modules, in the dice module and player selector module we have used SW-DPST-MOM switch to connect both of them and they both can work at a same time as the input of that switch is connected with the square wave signal generator . By using this feature , the full game can be played using only this switch that enables both the dice module and player selector module and it will helps to swtich between the player when the dice is rolled each time.



3.Wait Until win :

we have design the snake ladder logic module in a way that if a player needs one or two to win the game and then if the dice come up with a larger value than the desired value, the player will wait at its place for the next roll of the dice to get the desired value,

4.Ladder Priority First :

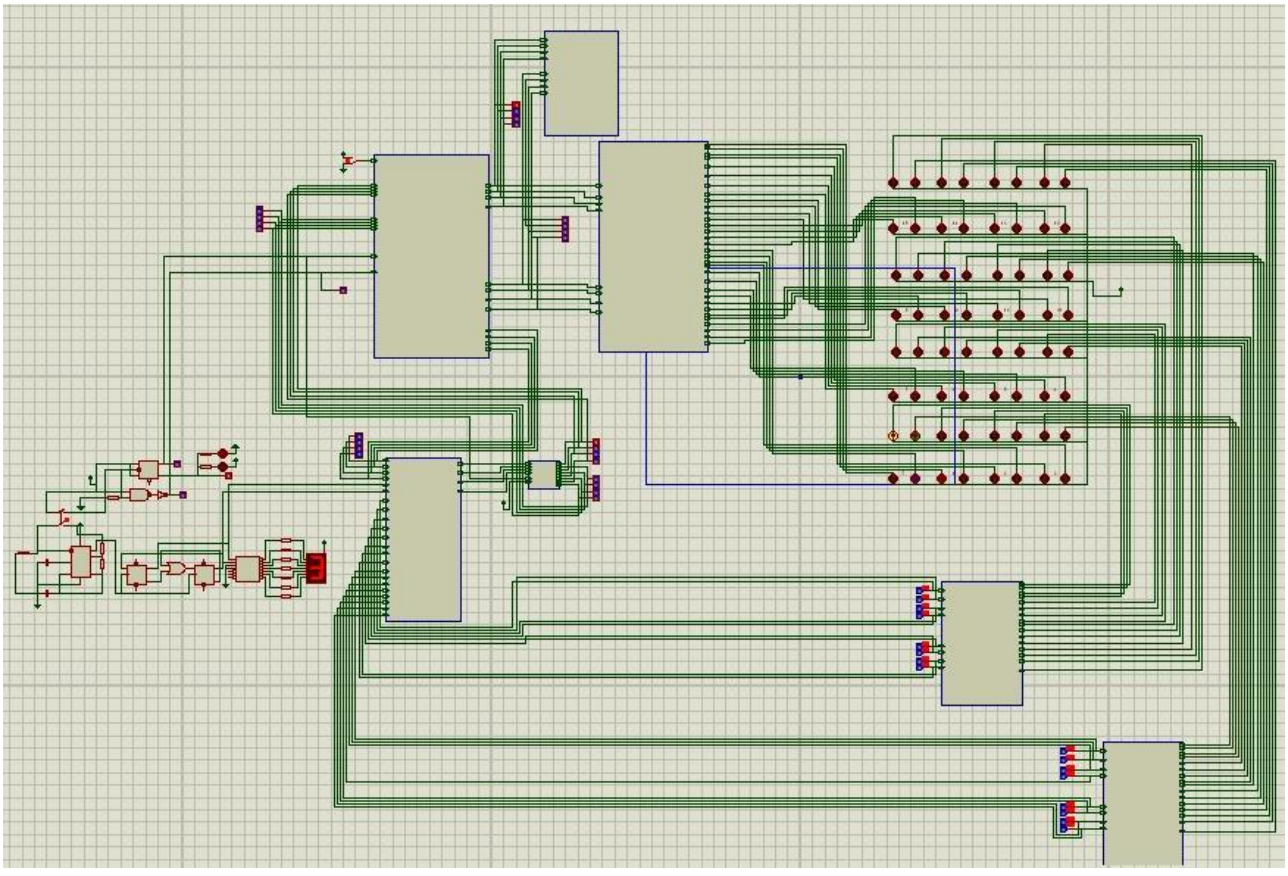
If the ladder head and snake mouth is placed In the same box , it will take the ladder function and ingnore the snake in that box . We have implemented that logic in the snake ladder logic module .

5.Impossible Snake and ladder positions:

If the value of ladder head is greater then the number of boxes in the board , it is not possible as well as if the snake mouth can not be placed in any negative box when the snake length get larger than the boxes left from the initial position . Both of the cases can not be possible if we want to play the game logically.So this two cases are also removed from the game as we have implemented that logic in the sanke ladder logic module.

_Integration of whole module :

We have previously described in every module that where the connection came from and where it will go . Here in this module we have simply joined the connections among the modules and completed the circuit to run the game properly.



Contributions:

· **Shazzad Ahmed:** Reviewed the entire project and worked on enhancing its efficiency.

· **Faid Faisal Sarthok:** Conducted a thorough review of the project to improve its overall efficiency.

· **Fahim Shifat:** Developed the Snake position decoder, Ladder position decoder, and the logic module for both Snakes and Ladders.

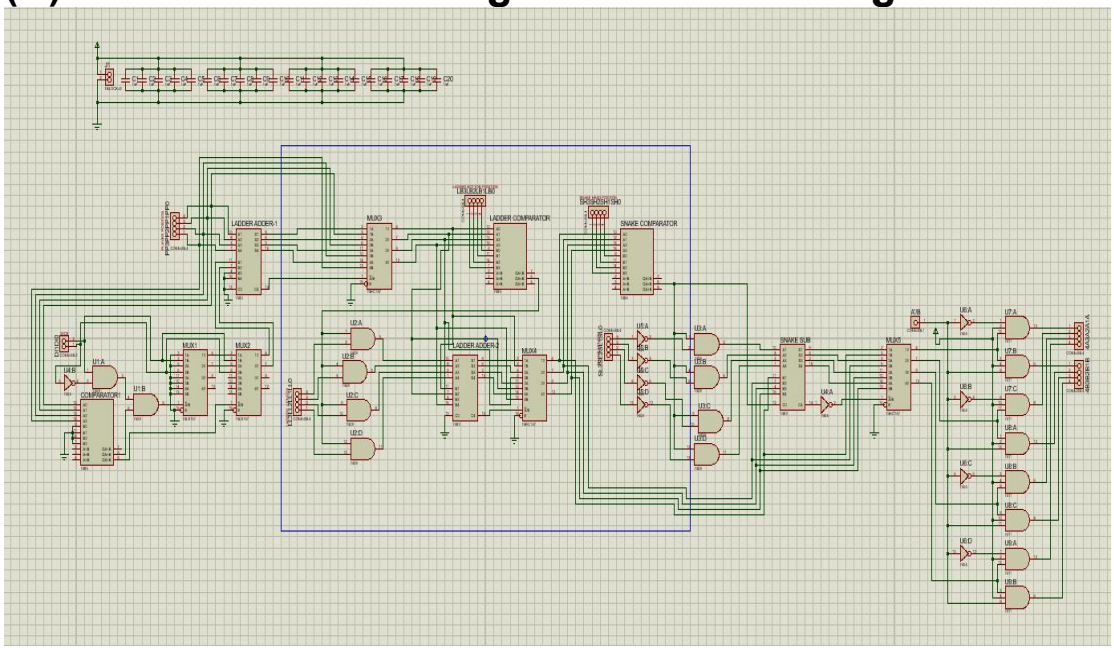
· **Ahnaf Sakif:** Implemented the player position decoder.

· **Mahib Abtahi:** Reviewed the entire project and focused on making it more efficient.

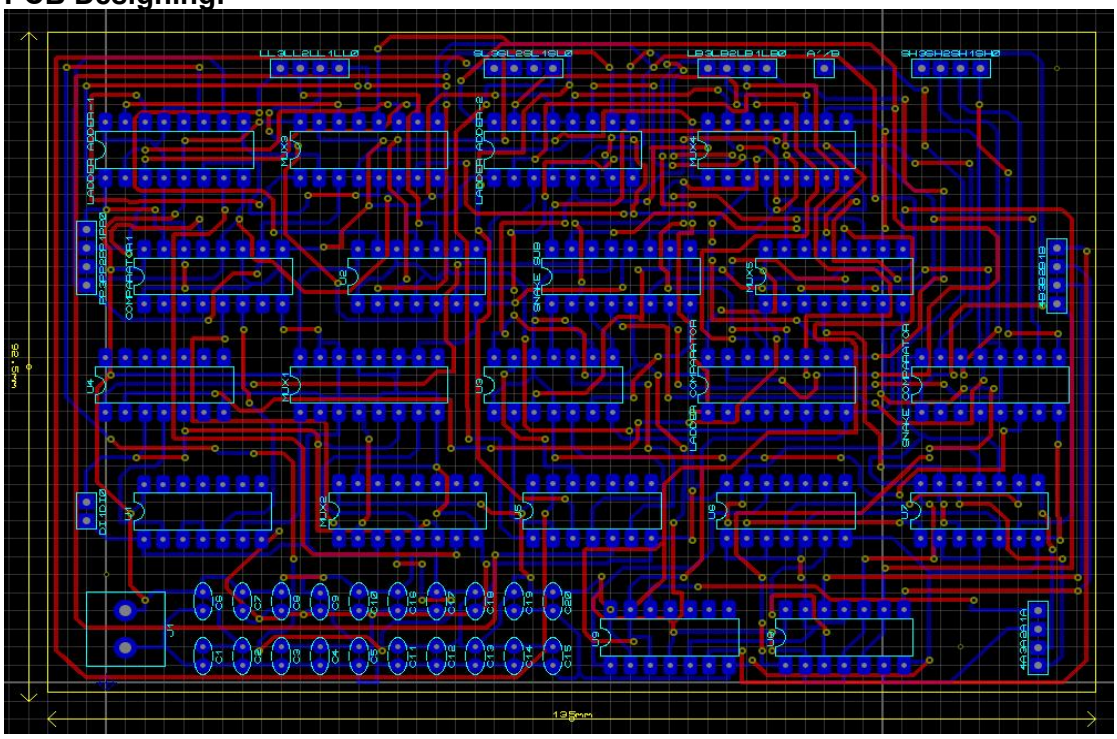
· **Abdullah Jubayer:** Designed the board module and assisted in debugging phases A and B.

· **Hasib Ahmed Anik:** Contributed to problem analysis and performed a comprehensive review of the project.

(A) Snake & Ladder Logic Schematic design:



PCB Designing:



Explanation:

In the PCB technology portion, Design Rule Manager part:

A)Design Rule:

- **Pad - Pad** Clearance: 20 th
- **Pad - Trace** Clearance: 30 th
- **Trace - Trace** Clearance: 30 th
- **Graphics** Clearance: 15 th
- **Edge/Slot** Clearance: 15 th

B)Net Classes:

Both **POWER** & **SIGNAL**. Here are the configurations **:** • **Routing Styles:**

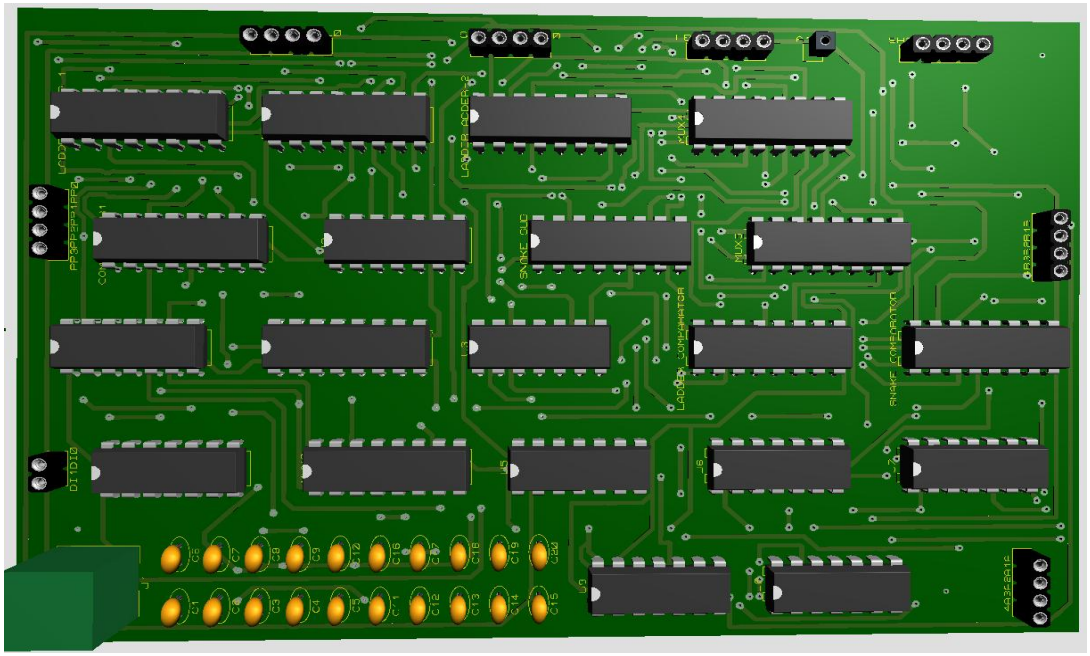
- **Trace** Style: T50
- **Via** Style: V80

• **Via Type: Thru-Hole**

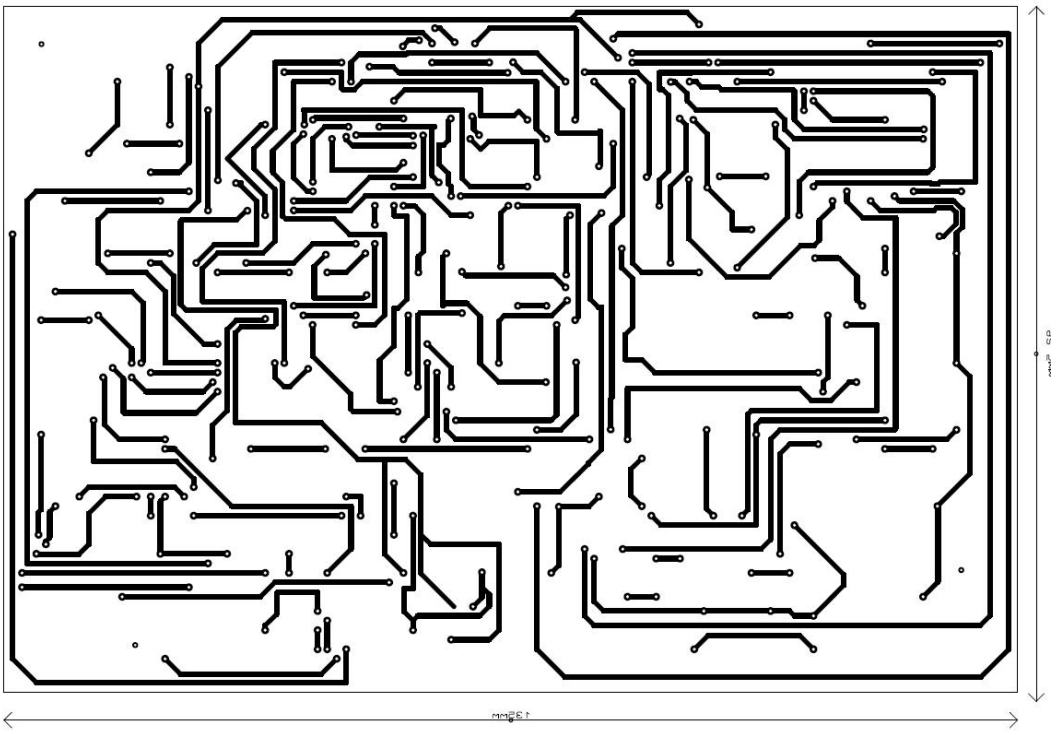
• **Ratsnest Display:**

- **Color:** Green

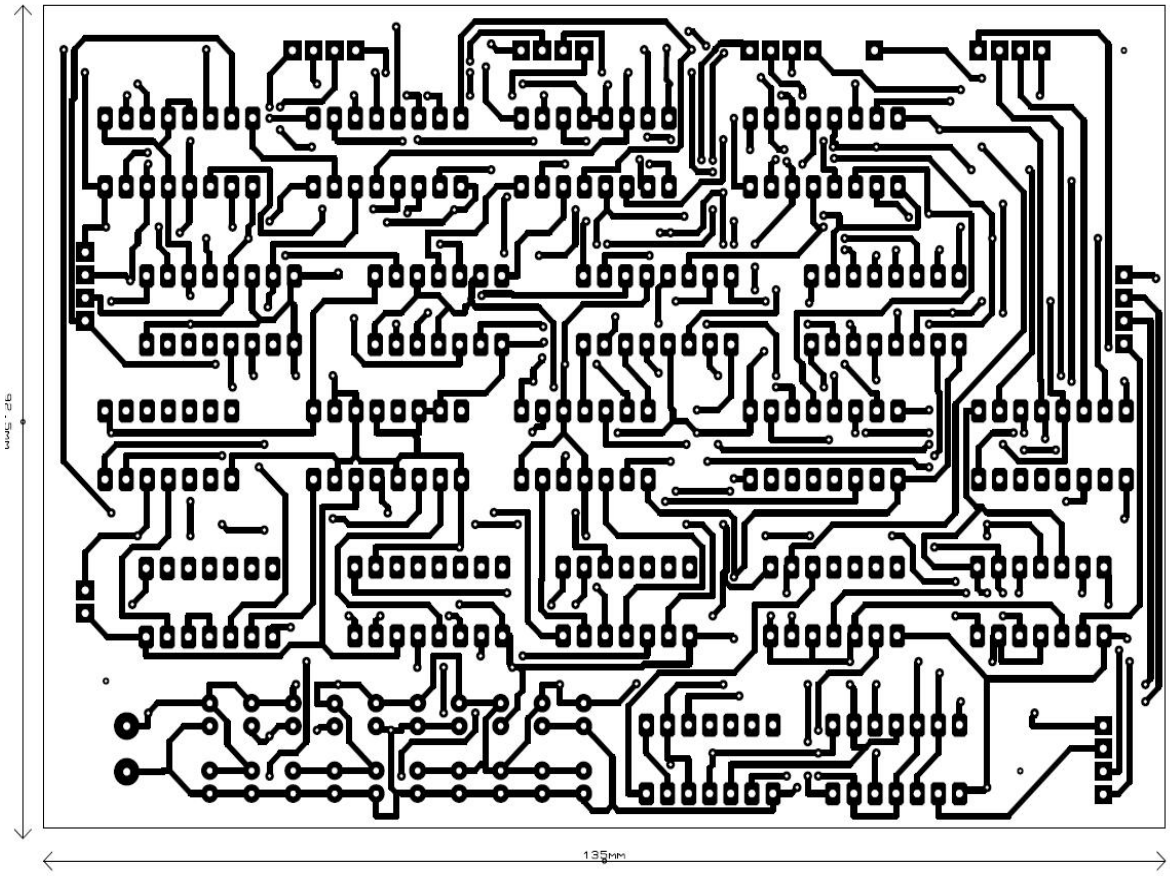
3D Visualizer:



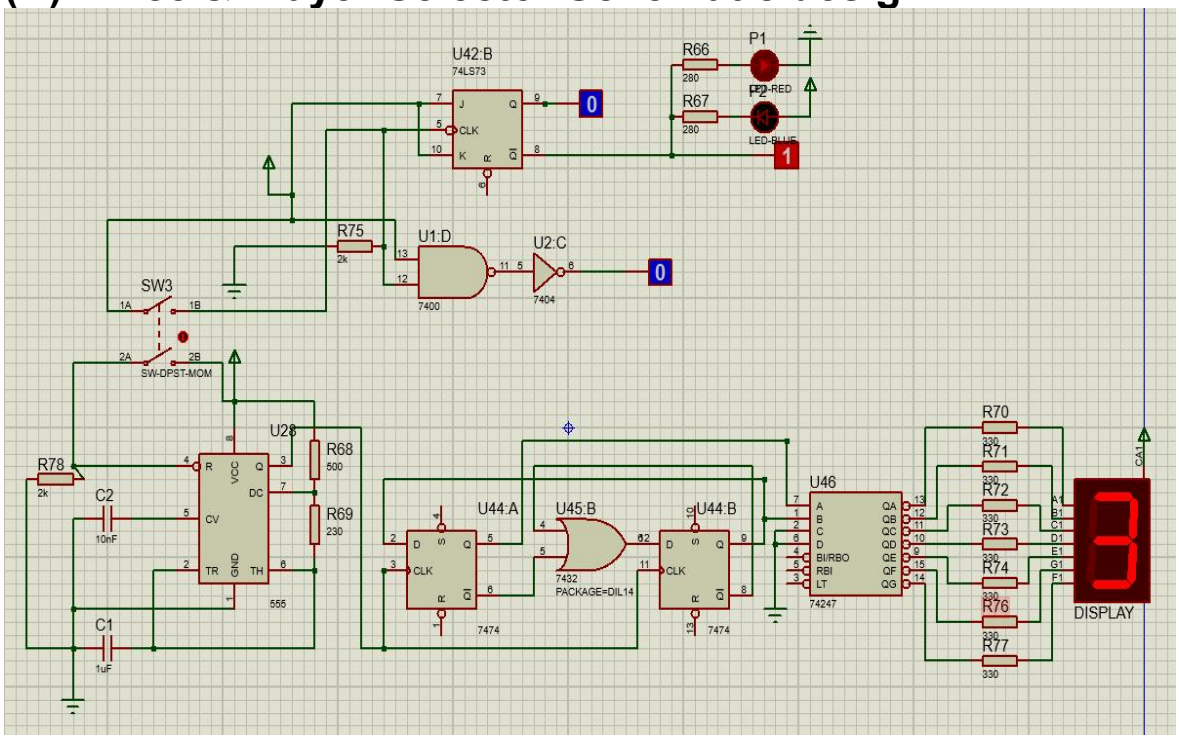
Top Layer Ouput :Mirror



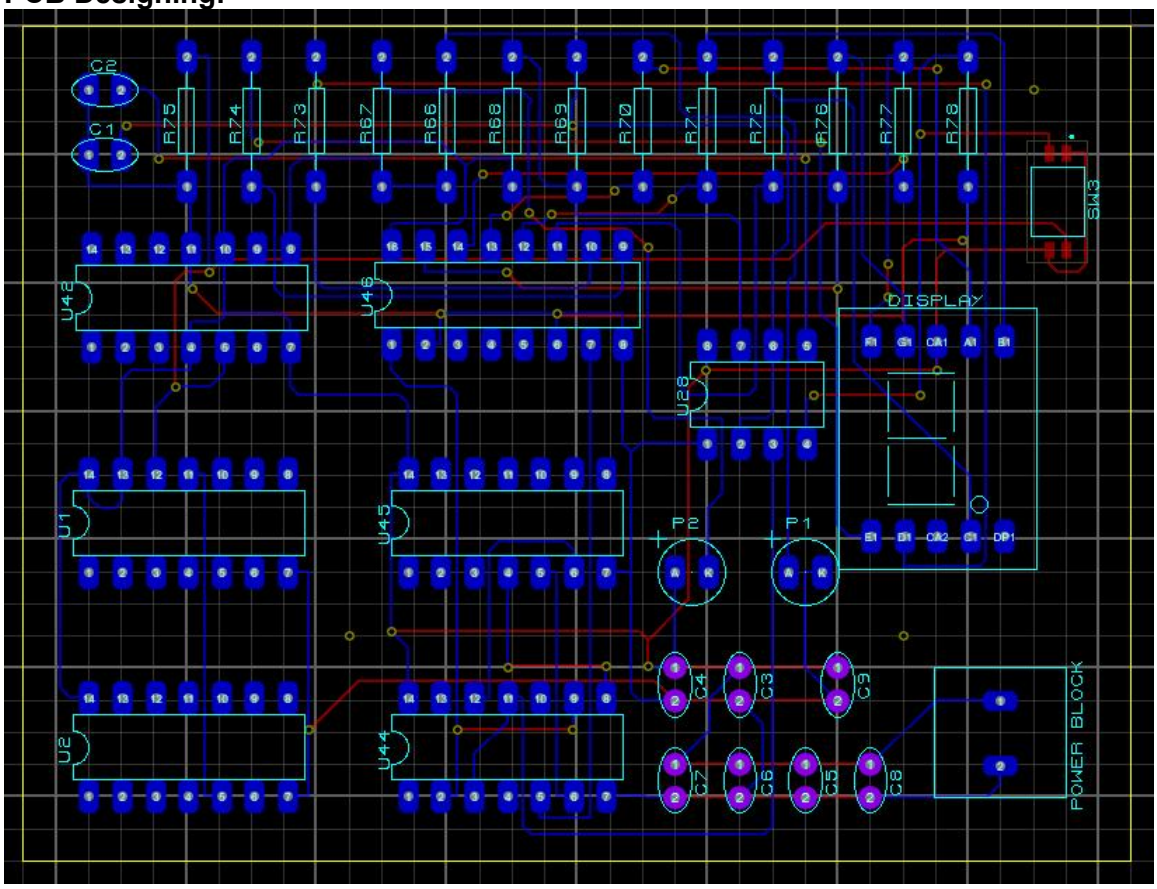
Bottom Layer Output :Normal



(B) Dice & Player Selector Schematic design:



PCB Designing:



Explanation:

In the PCB technology portion, Design Rule Manager part:

A)Design Rule:

- **Pad - Pad** Clearance: 10 th
- **Pad - Trace** Clearance: 10 th
- **Trace - Trace** Clearance: 10 th
- **Graphics** Clearance: 15 th
- **Edge/Slot** Clearance: 15 th

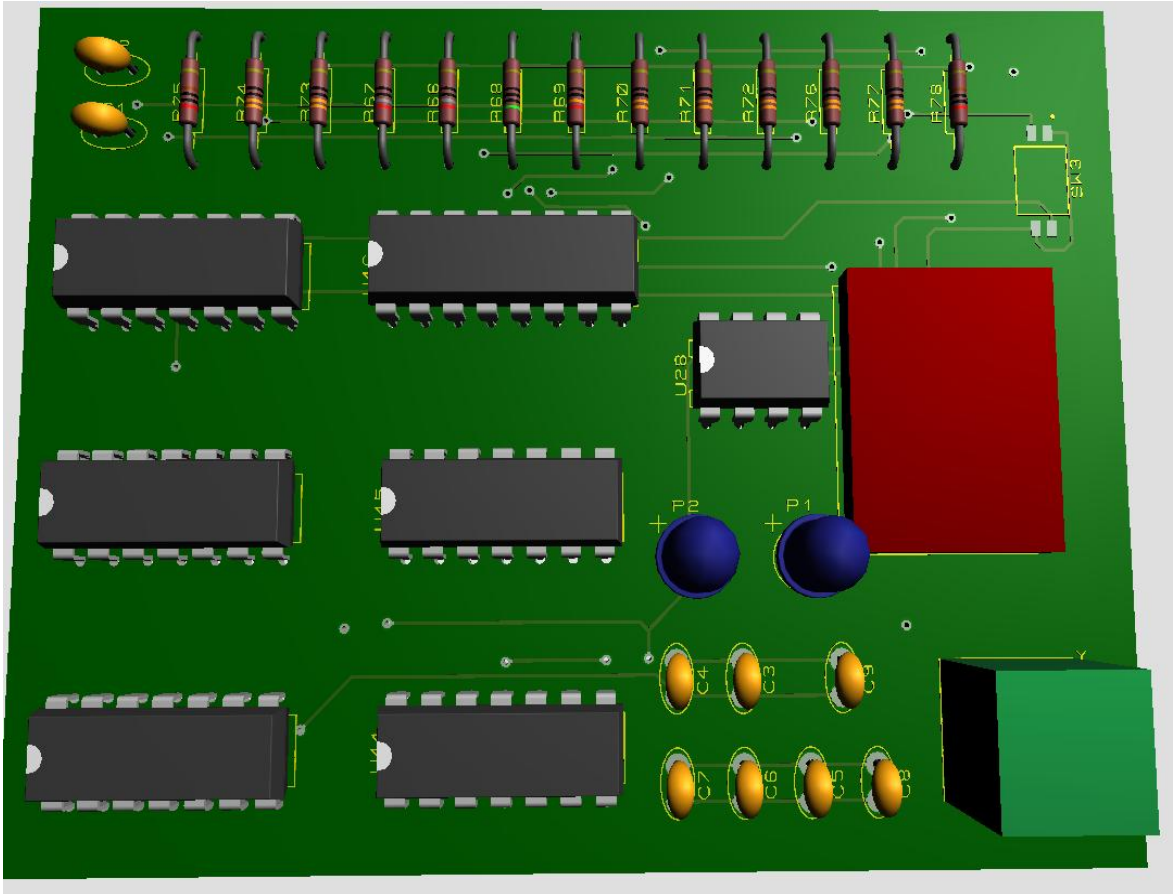
B)Net Classes:

Both **POWER** & **SIGNAL**. Here are the configurations **:** • **Routing Styles:**

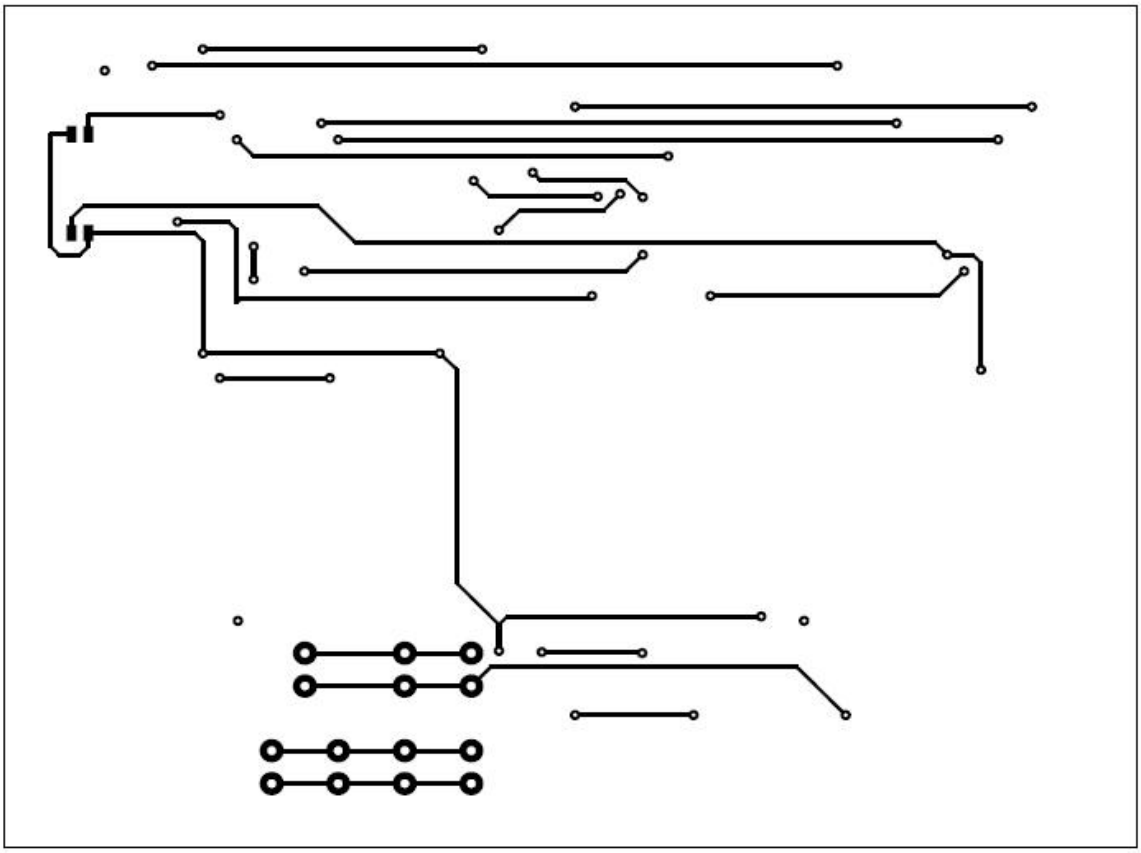
- **Trace** Style: DEFAULT
- **Via** Style: DEFAULT

- **Via Type: Thru-Hole**
- **Ratsnest Display:**
  - **Color:** Green

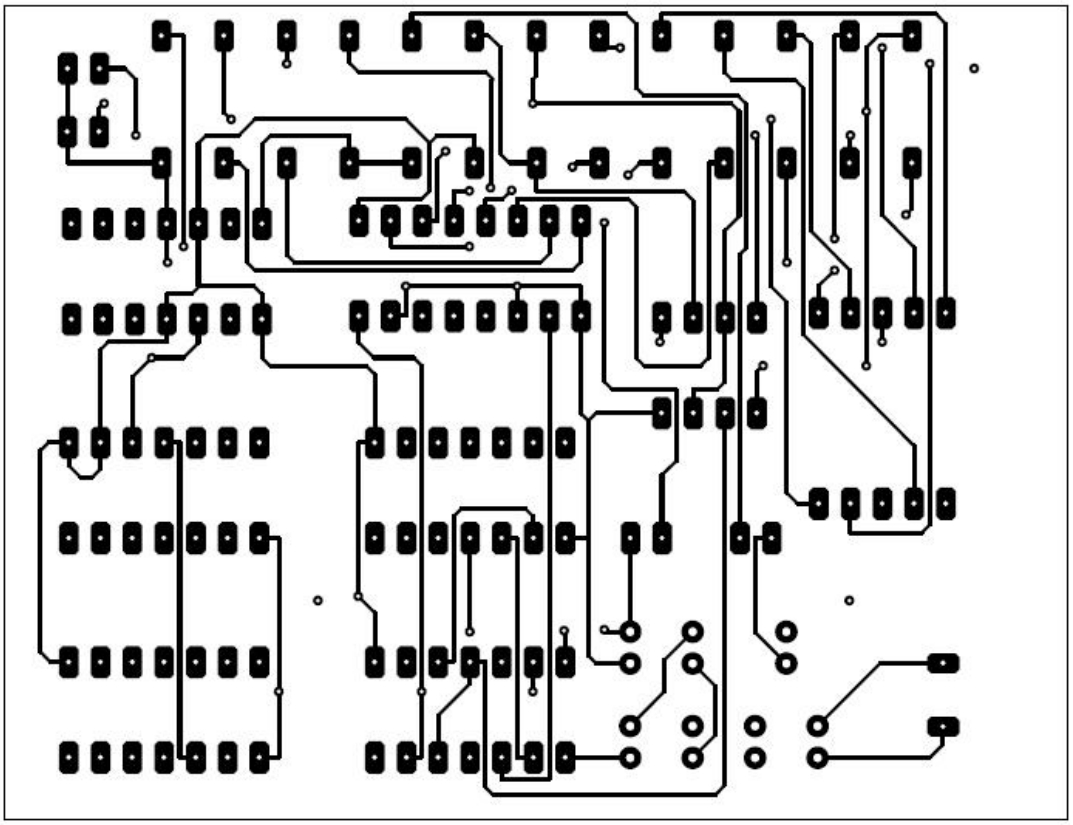
3D Visualizer:



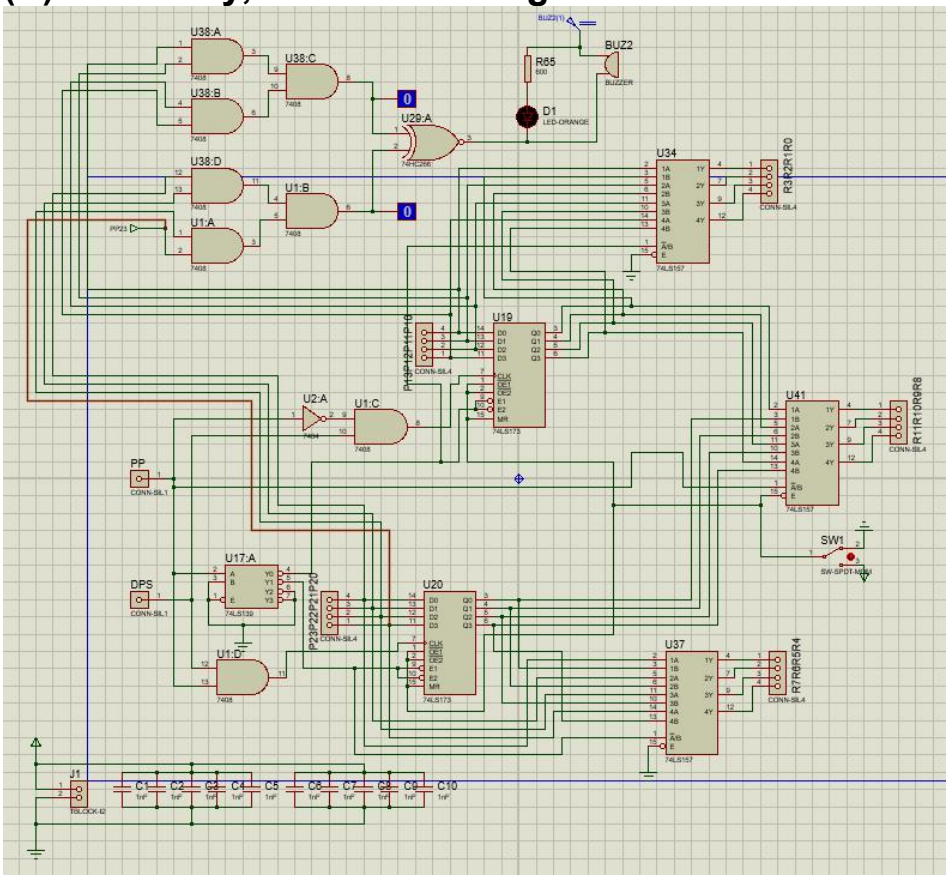
Top Layer Ouput :Mirror



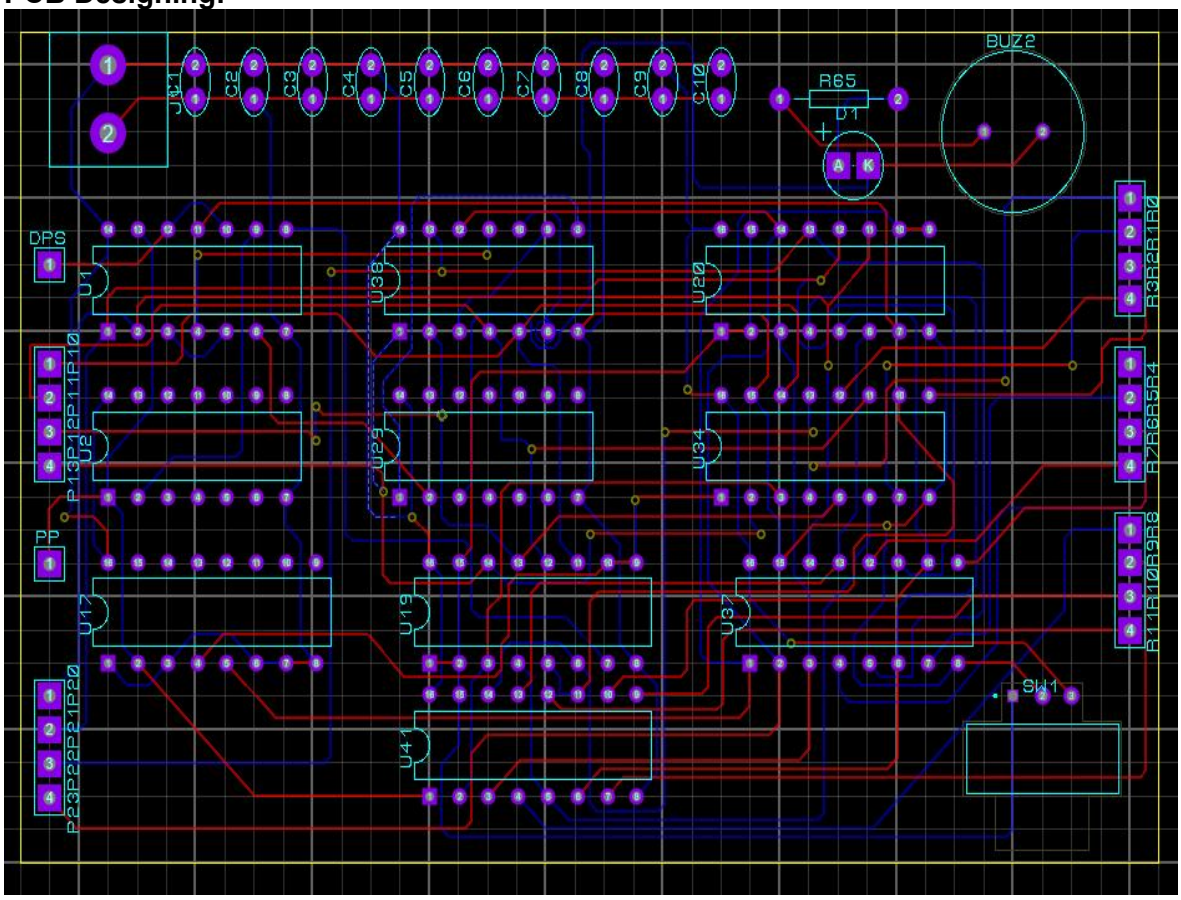
Bottom Layer Ouput :Normal



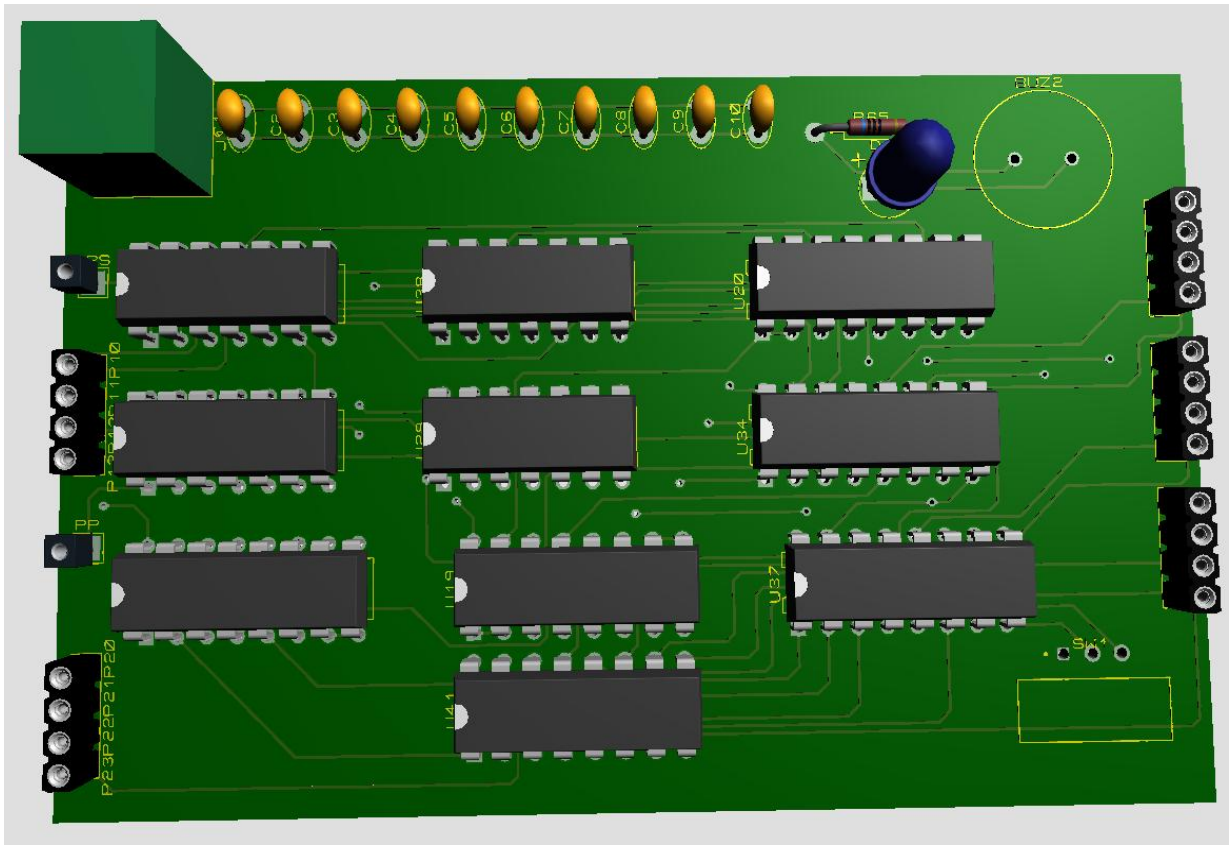
(C) Memory,Reset & Winning modul Schematic design:



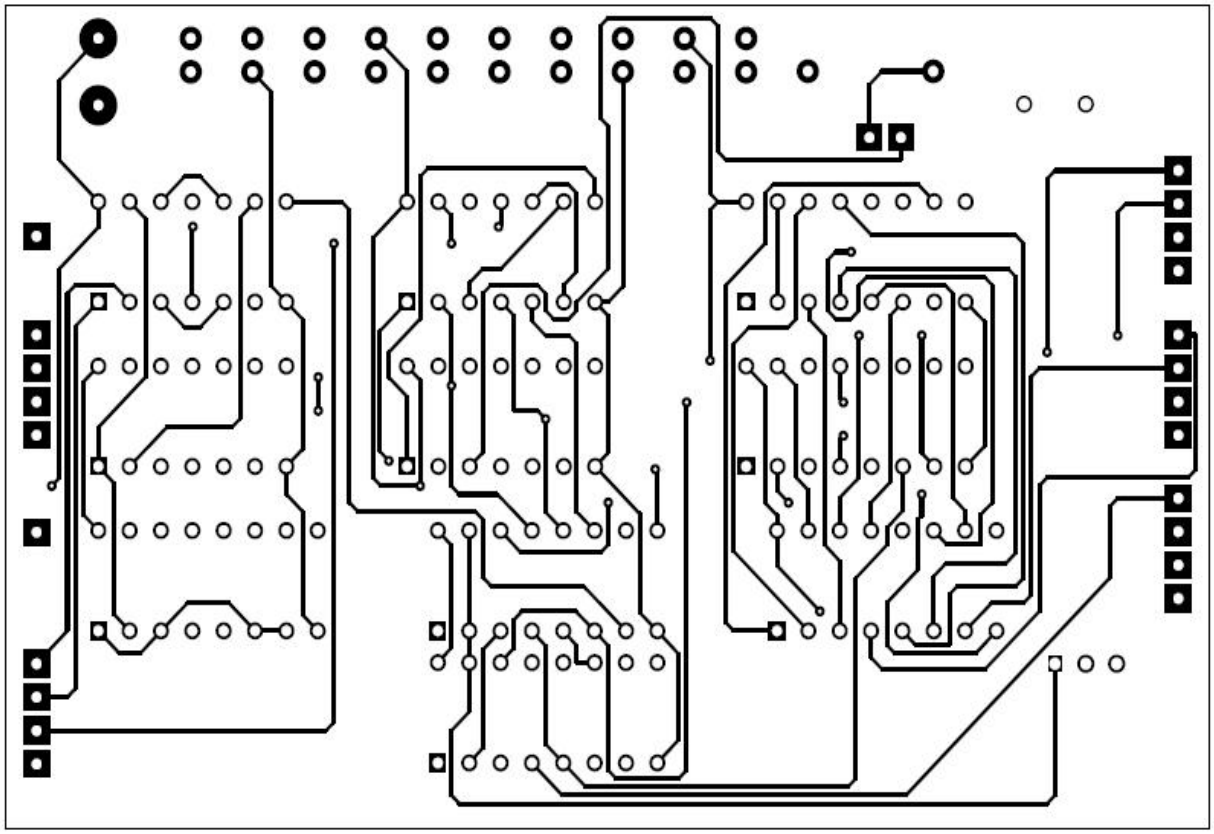
PCB Designing:



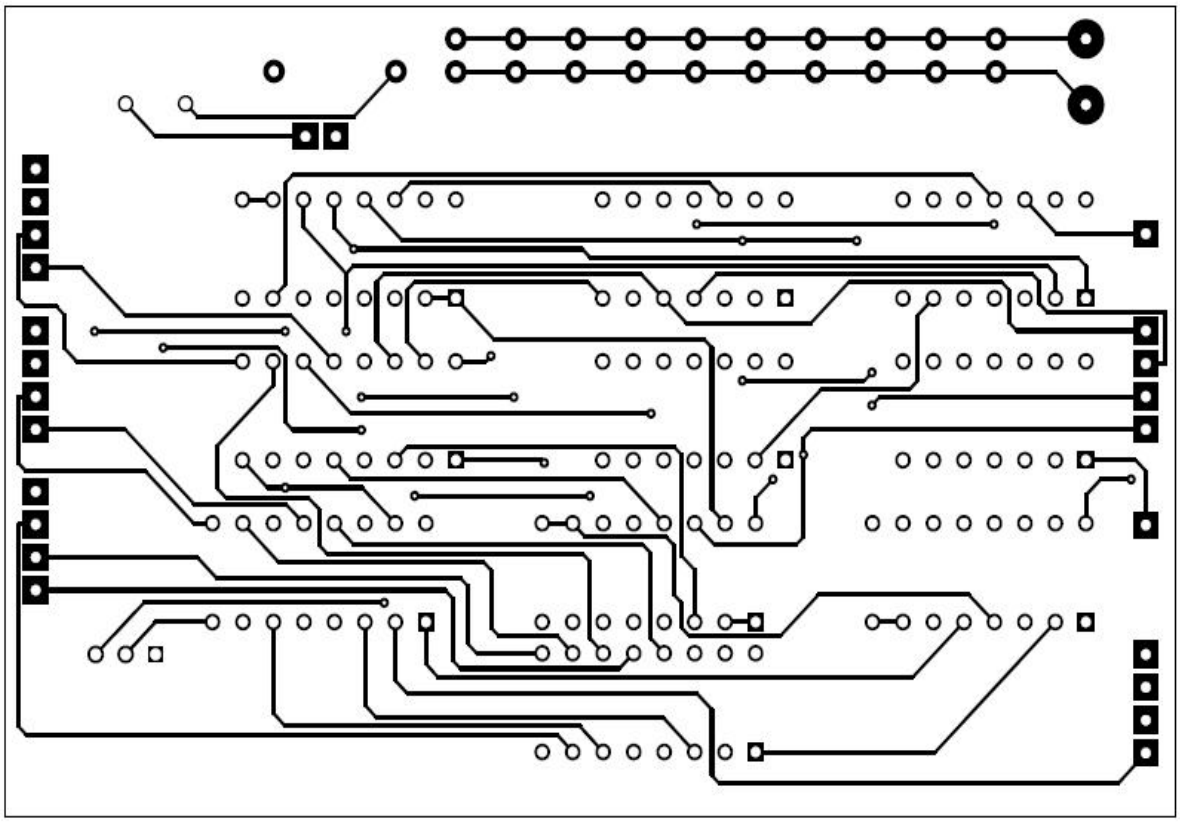
3D Visualizer:



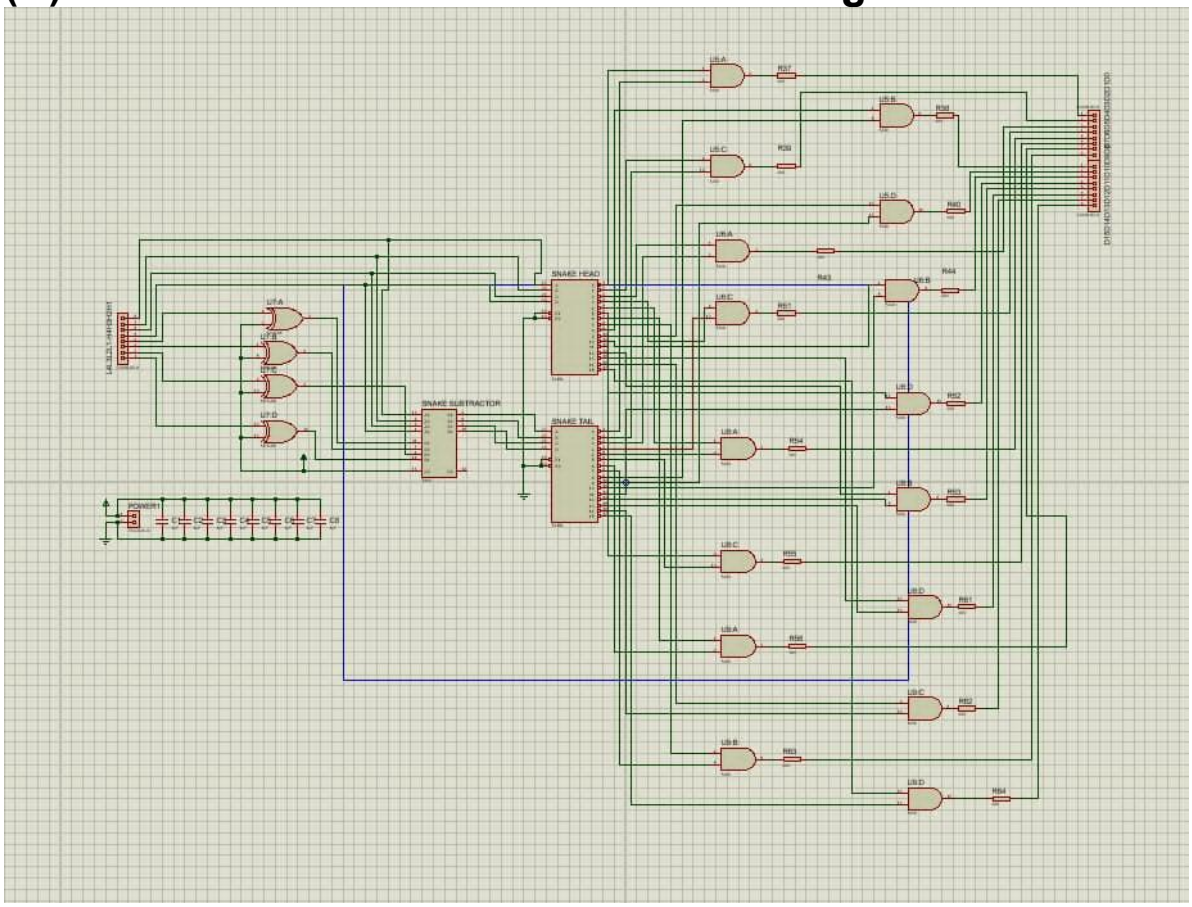
Bottom Layer Ouput :Normal



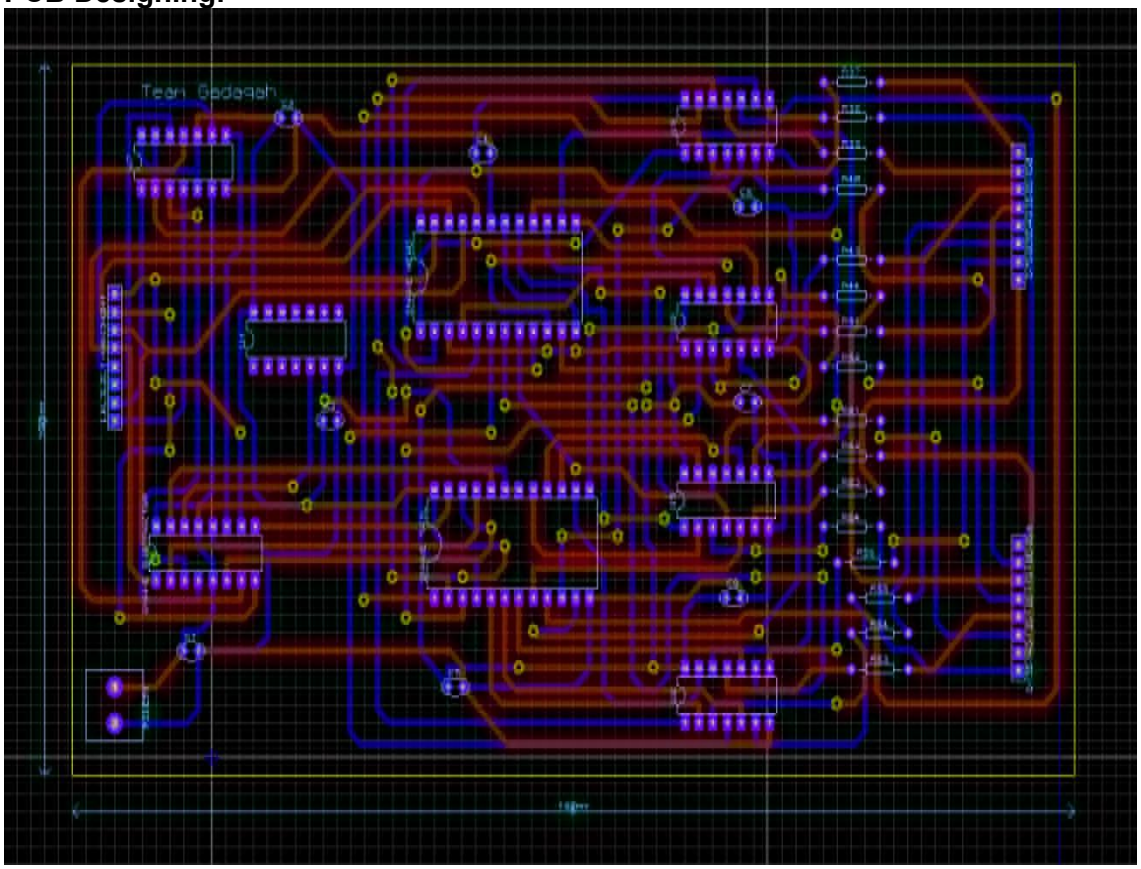
Top Layer Ouput :Mirror



(D)Snake Position ModuleSchematic design:



PCB Designing:



Explanation:

In the PCB technology portion, Design Rule Manager part:

A)Design Rule:

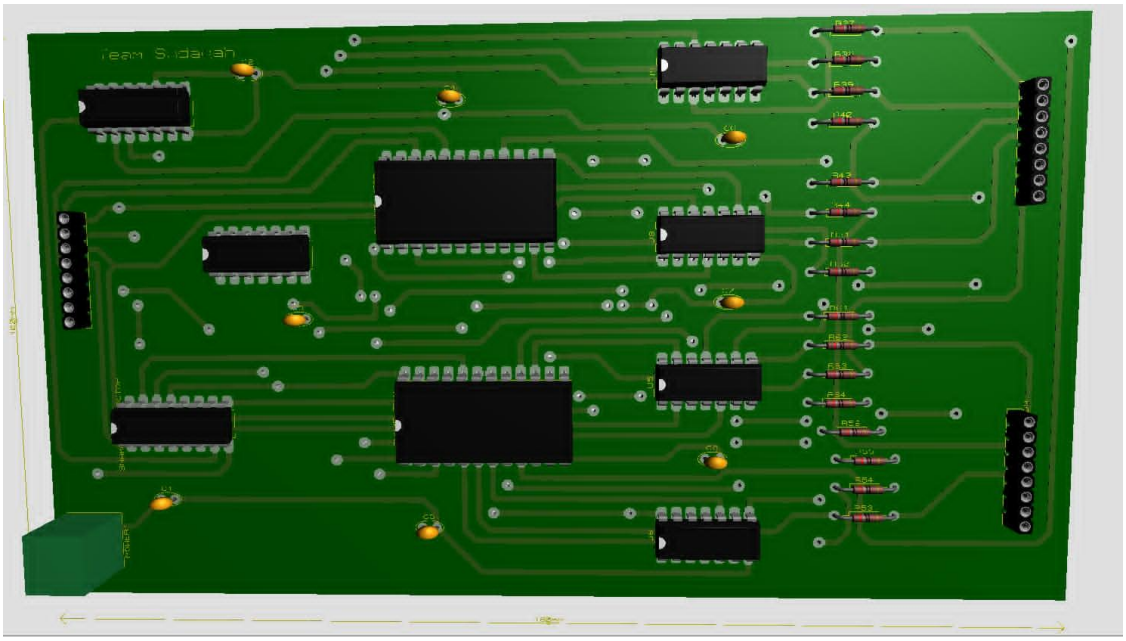
- **Pad - Pad** Clearance: 20 th
- **Pad - Trace** Clearance: 30 th
- **Trace - Trace** Clearance: 30 th
- **Graphics** Clearance: 15 th
- **Edge/Slot** Clearance: 15 th

B)Net Classes:

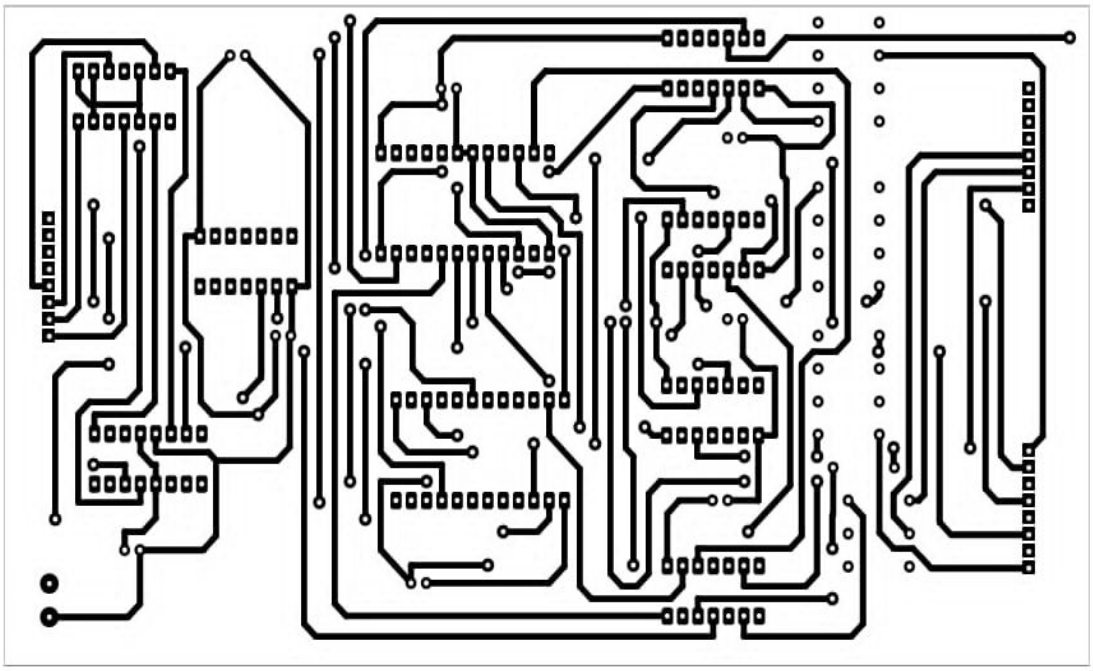
Both **POWER** & **SIGNAL**. Here are the configurations **:**

- **Routing Styles:**
  - **Trace** Style: T50
  - **Via** Style: V80
- **Via Type: Thru-Hole**
- **Ratsnest Display:**
  - **Color:** Green

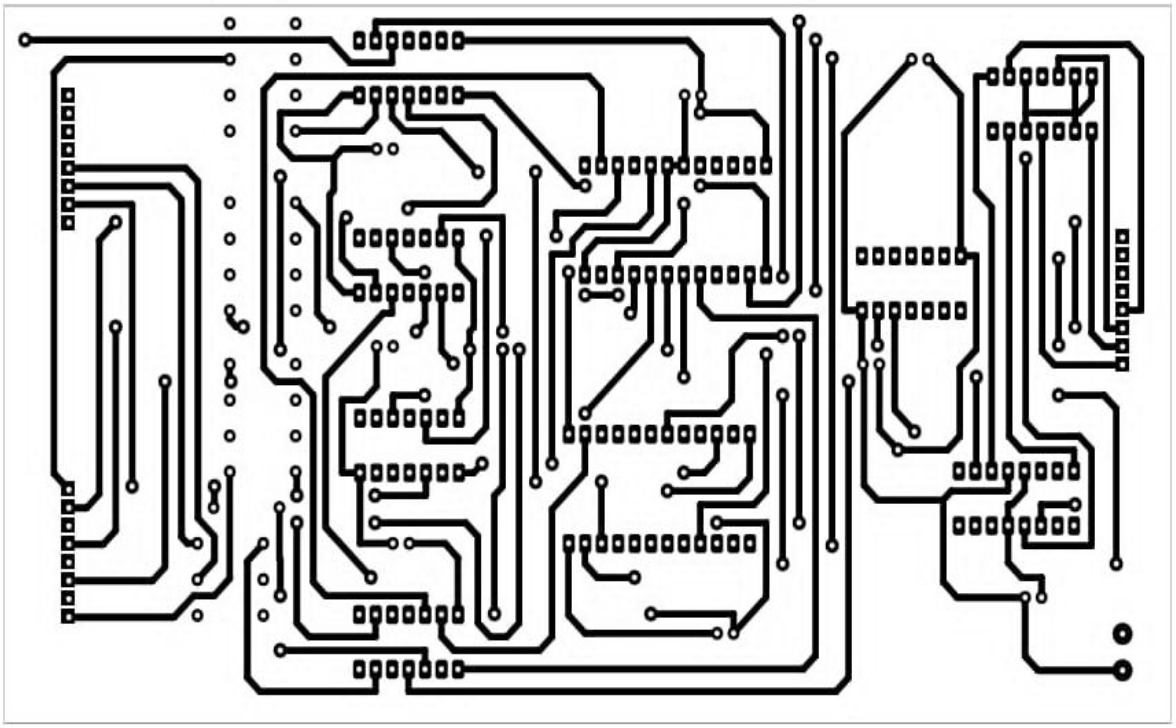
3D Visualizer:



Bottom Layer Ouput :Normal



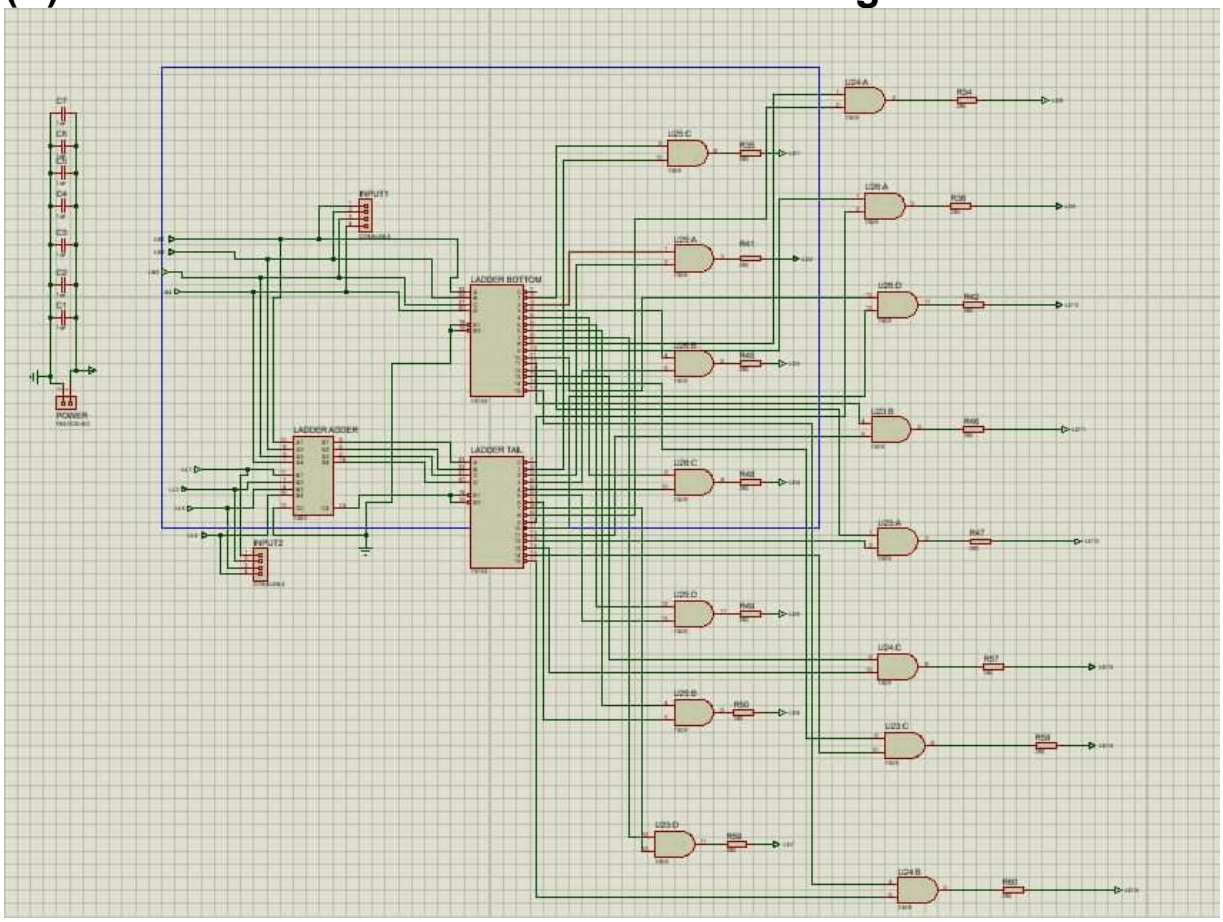
Top Layer Ouput :Mirror



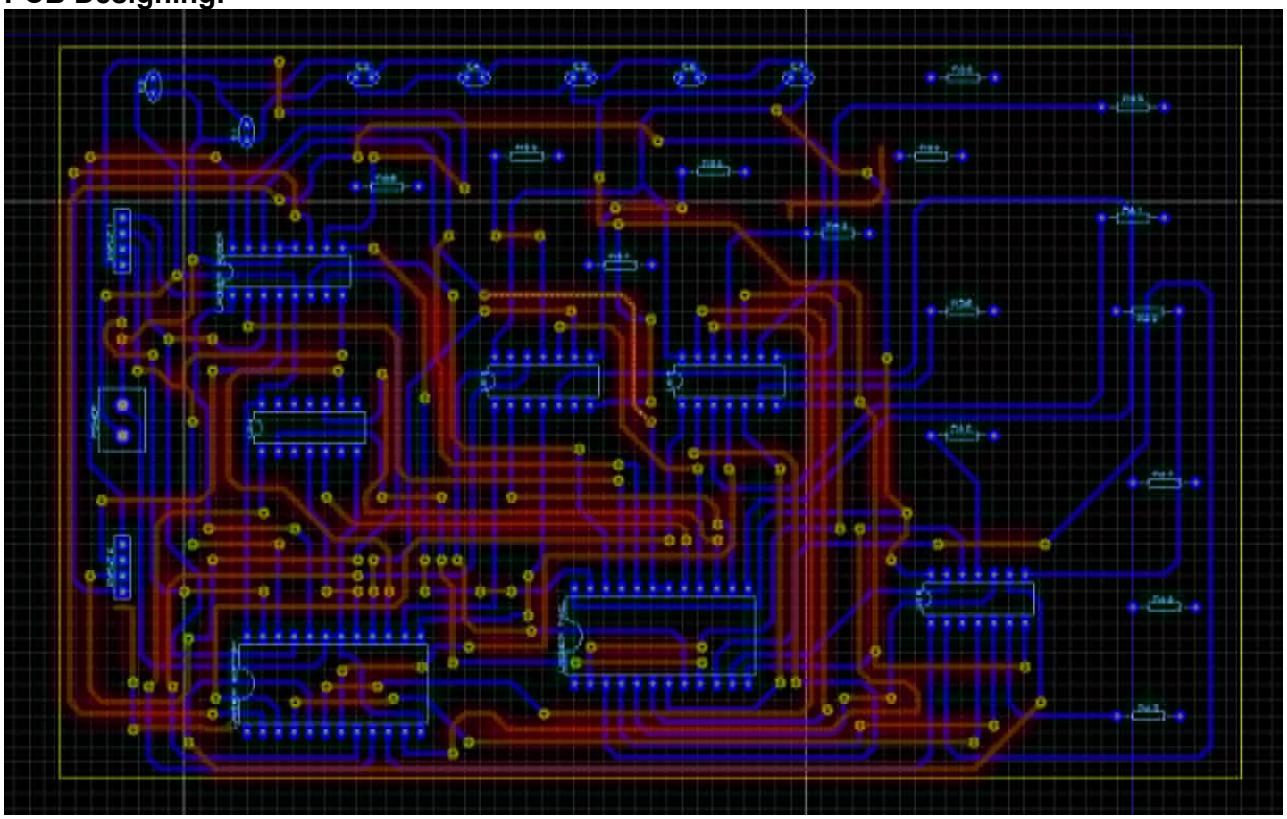
Discussion:

In this module, we have used 3 conn-sil4 and connected them with their corresponding IC's for our PCB design. A tblock has been taken for powering up the ICs. Eight(8) capacitors has been used as we have implemented the module using 8 ICs in the schematic. The capacitors are connected in parallel to the tblock to prevent the current overflowing. Once every component is in the PCB, we employ a gate-swap optimizer. We set the design rule manager then. DIL Pad, Square through hole pad, and round through hole pad have been used as they are larger. Some parameters were changed to avoid DRC error like we have used pad to pad 20th, pad to trace 30th, trace to trace 30th. Under the net classes Trace style T40, Via style V80 were implemented in power and signal . At first we found some DRC error while replacing the small pads by bottom copper in Pad. On the advice of one of our teachers we then implemented this using all copper and no DRC errors were found then. After this we took the help of auto router and connections were done automatically without any DRC and CRC errors. We have measured the board edge 180mm X 100mm.

(E)Ladder Position ModuleSchematic design:



PCB Designing:



Explanation:

In the PCB technology portion, Design Rule Manager part:

A)Design Rule:

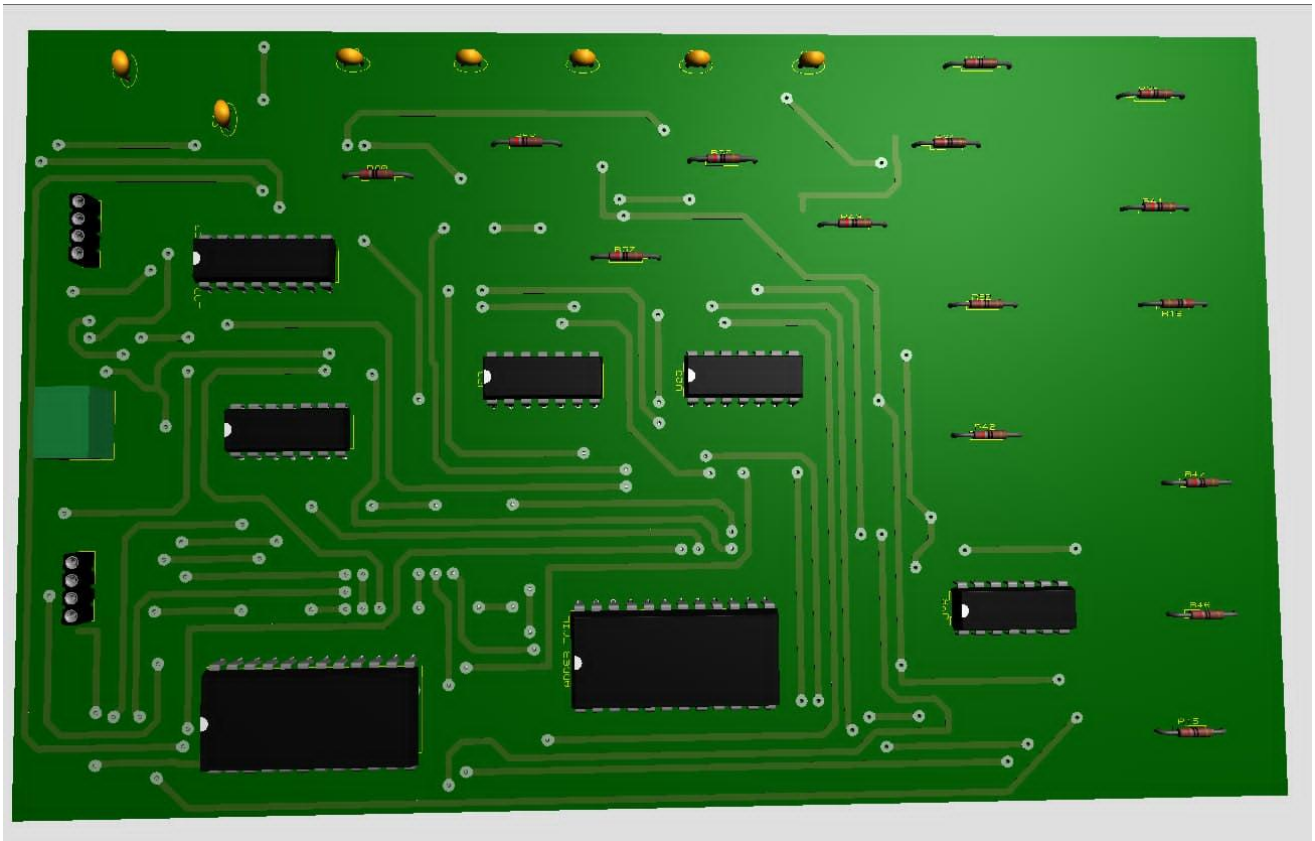
- **Pad - Pad** Clearance: 10 th
- **Pad - Trace** Clearance: 10 th
- **Trace - Trace** Clearance: 10 th
- **Graphics** Clearance: 15 th
- **Edge/Slot** Clearance: 15 th

B)Net Classes:

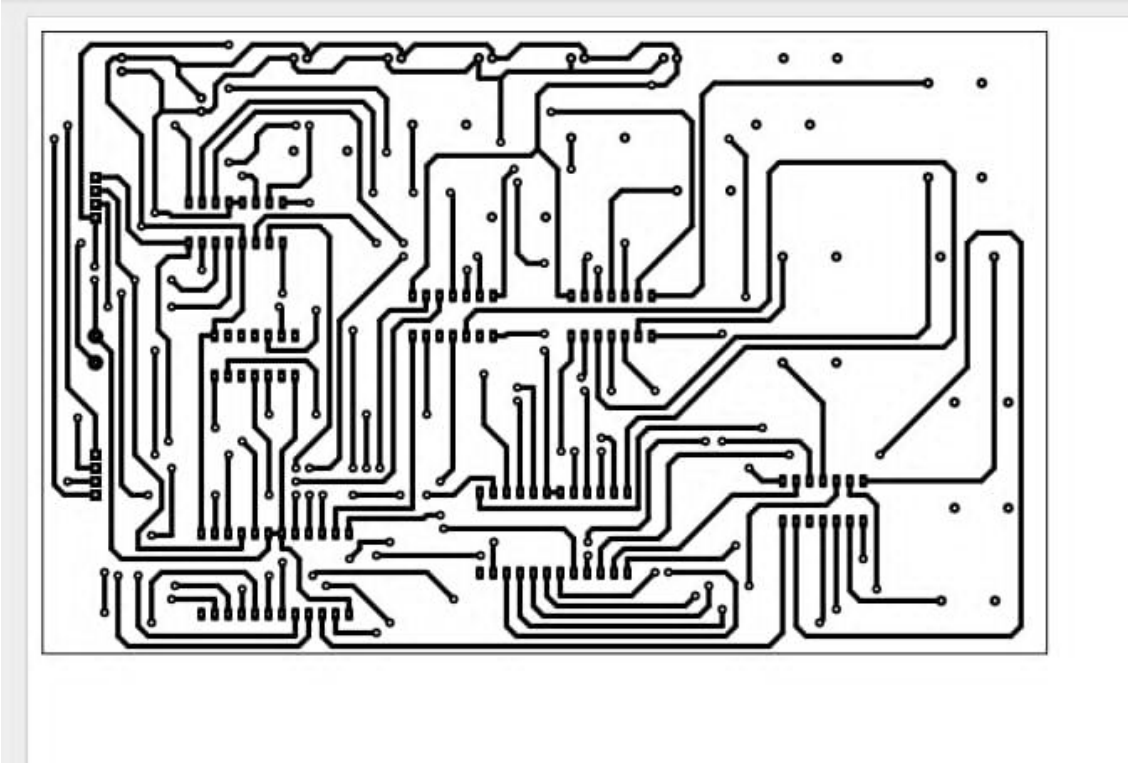
Both **POWER** & **SIGNAL**. Here are the configurations **:**

- **Routing Styles:**
  - **Trace** Style: DEFAULT
  - **Via** Style: DEFAULT
- **Via Type: Thru-Hole**
- **Ratsnest Display:**
  - **Color:** Green

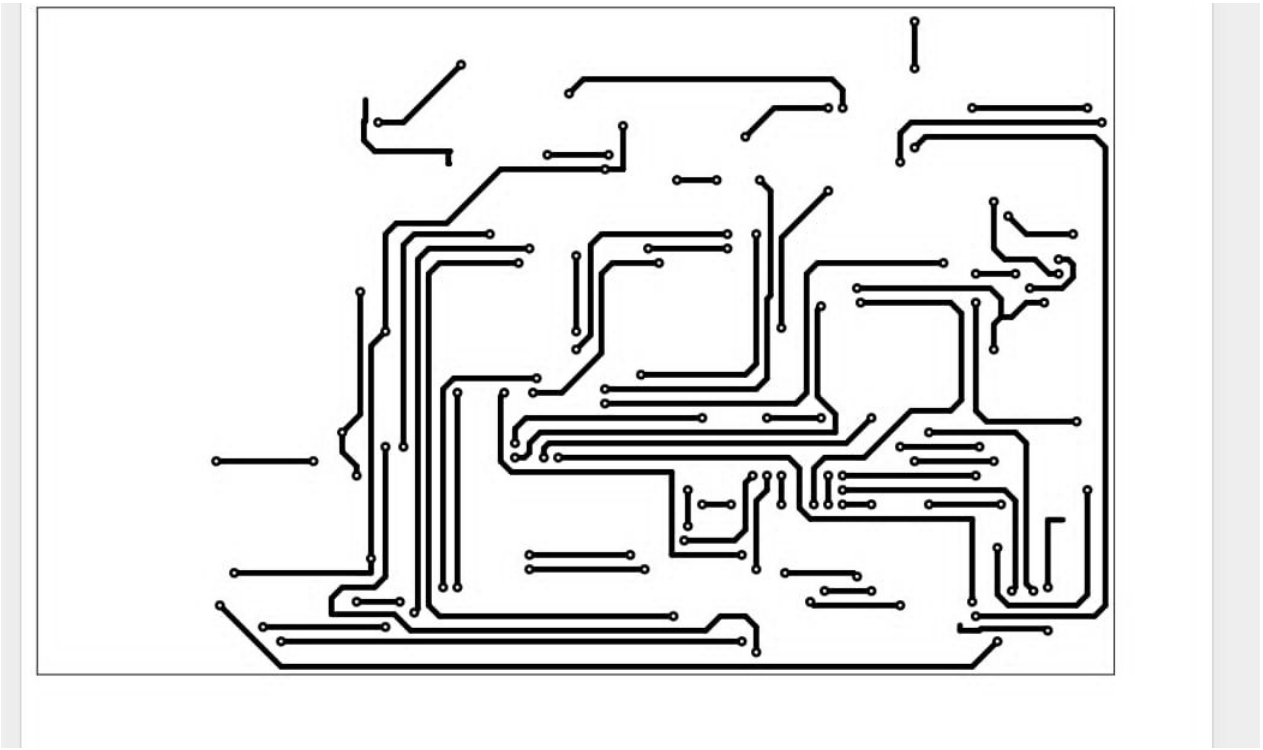
3D Visualizer:



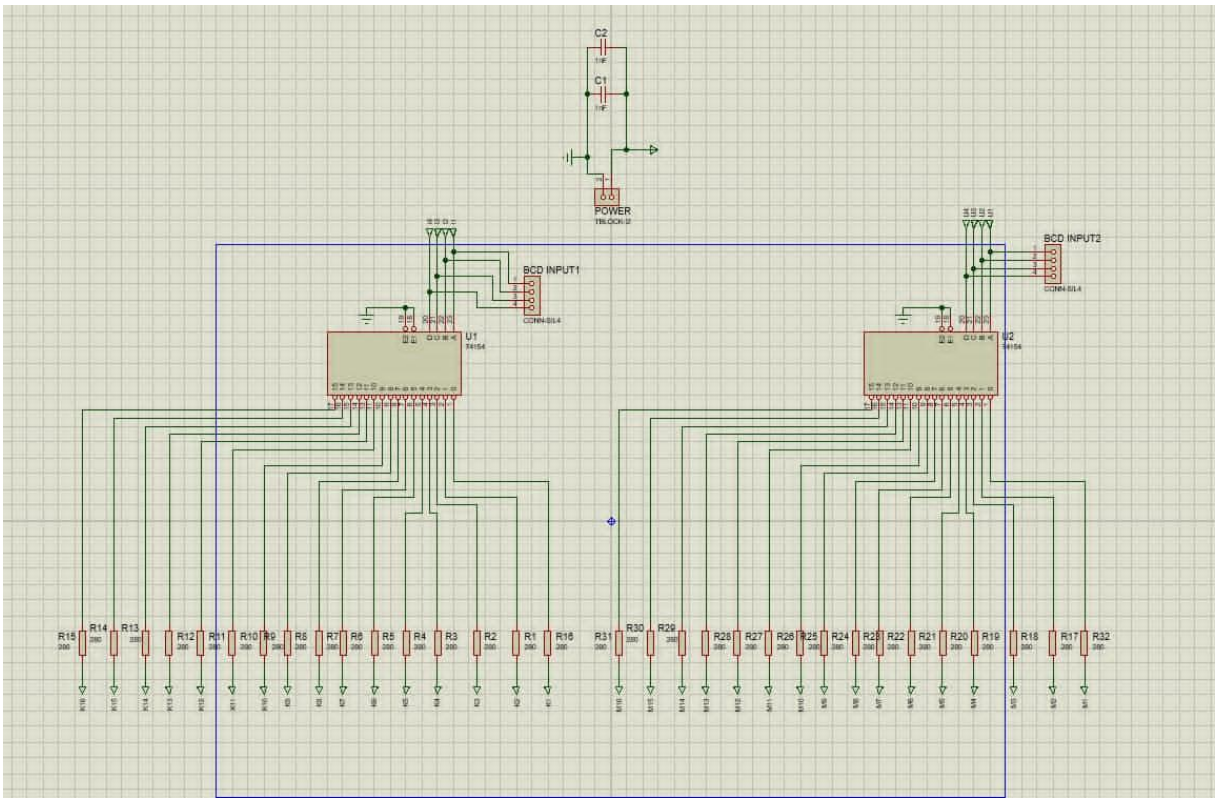
Bottom Layer Ouput :Normal



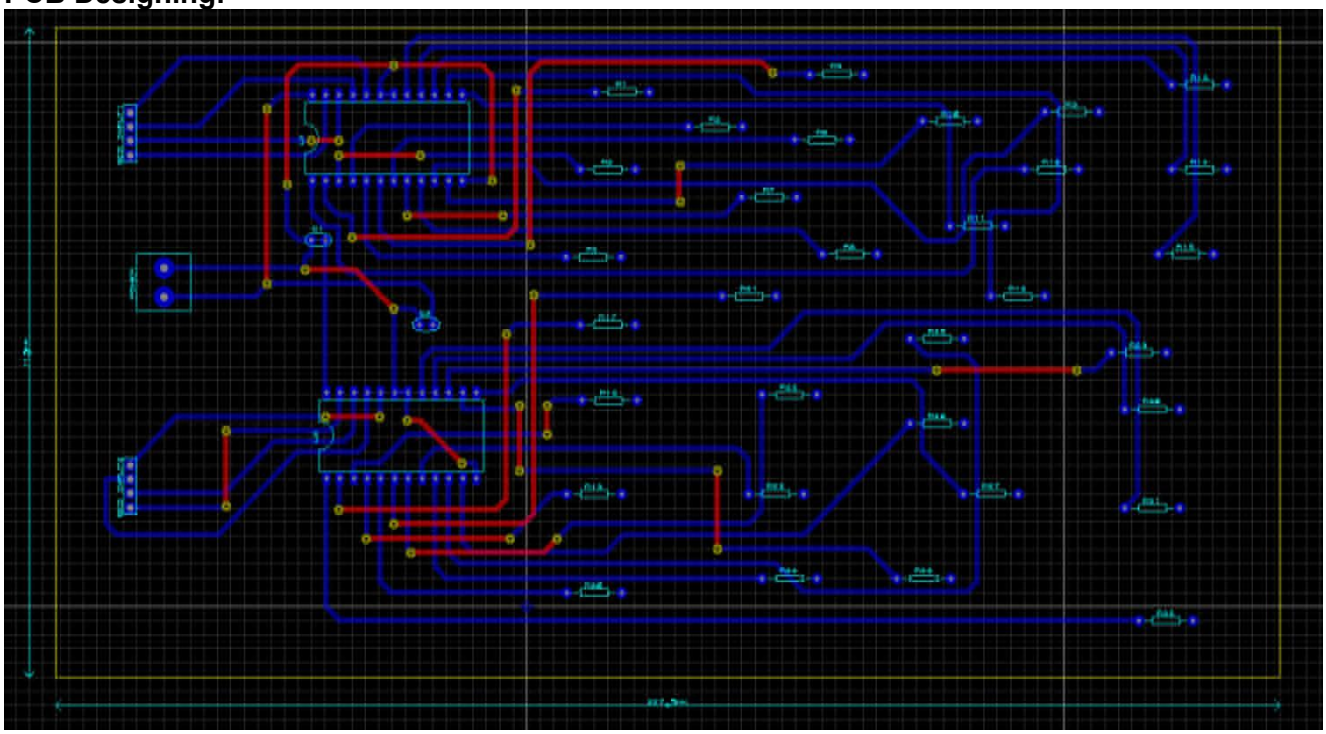
Top Layer Ouput :Mirror



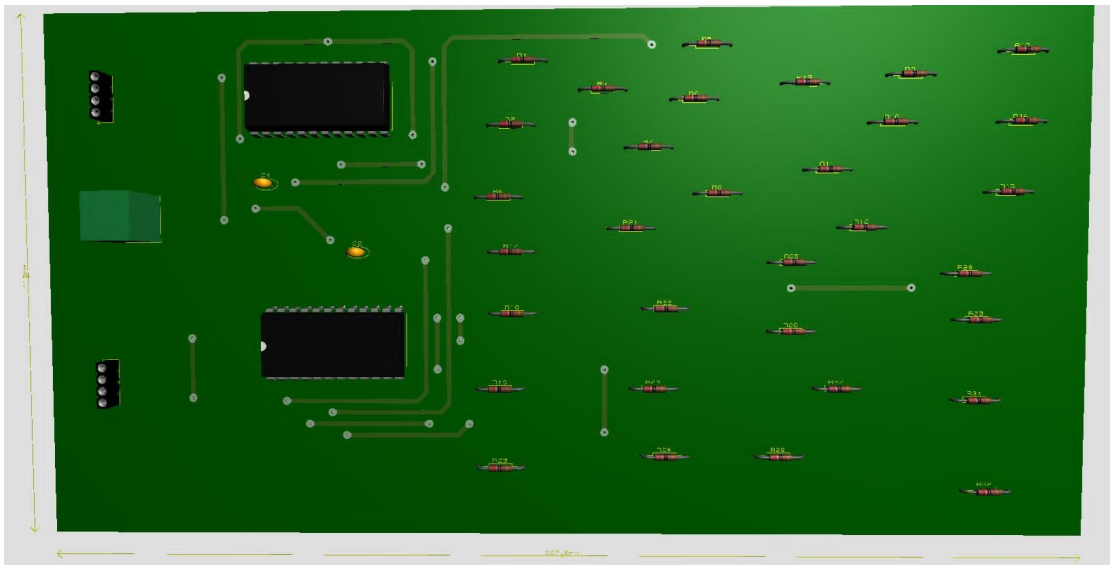
(F)Player Position Module Schematic design:



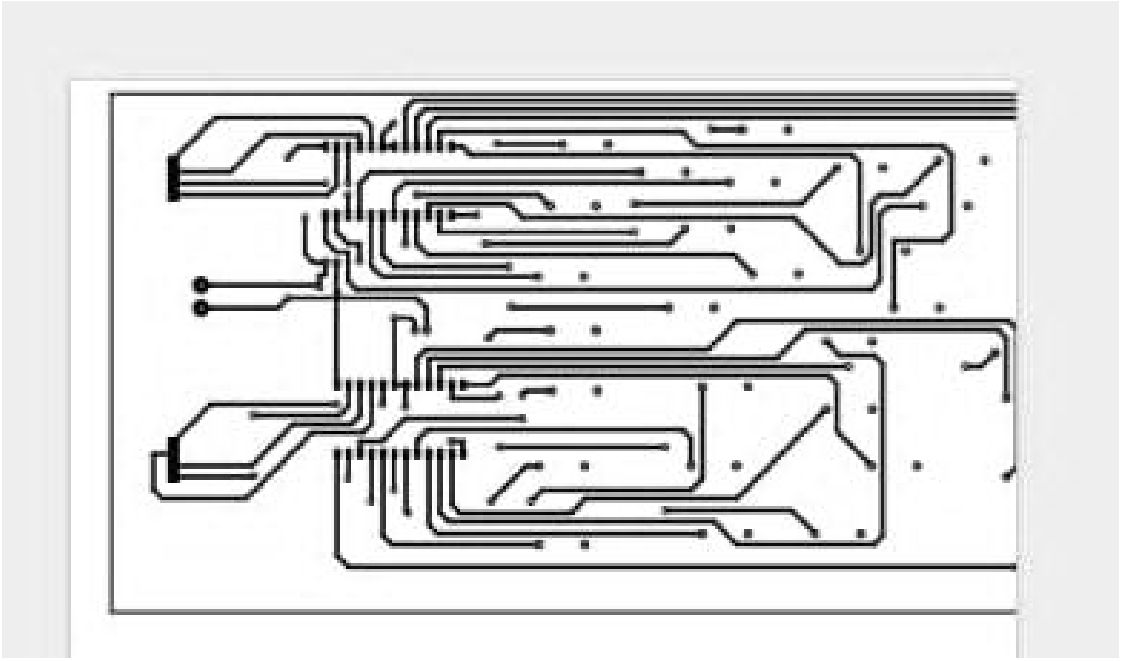
PCB Designing:



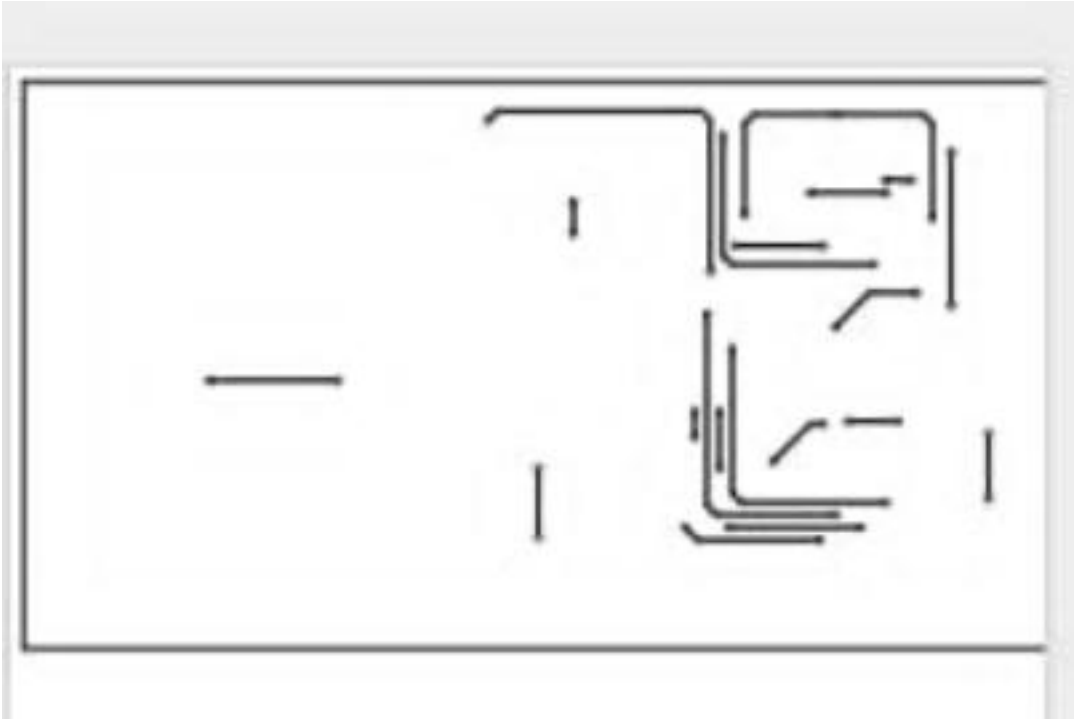
3D Visualizer:



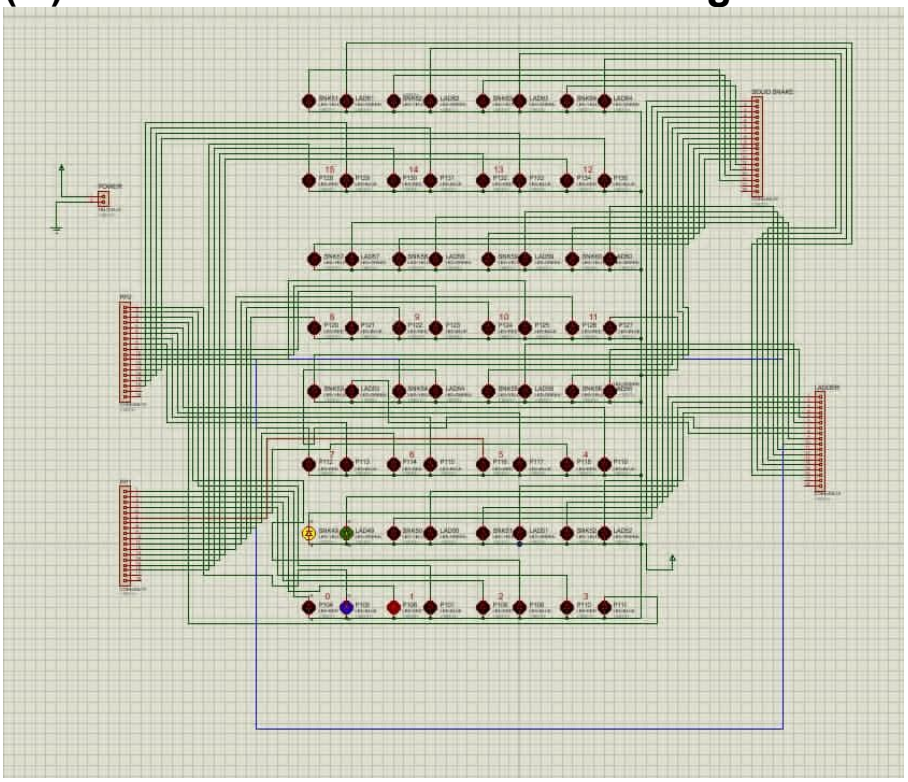
Bottom Layer Ouput :Normal



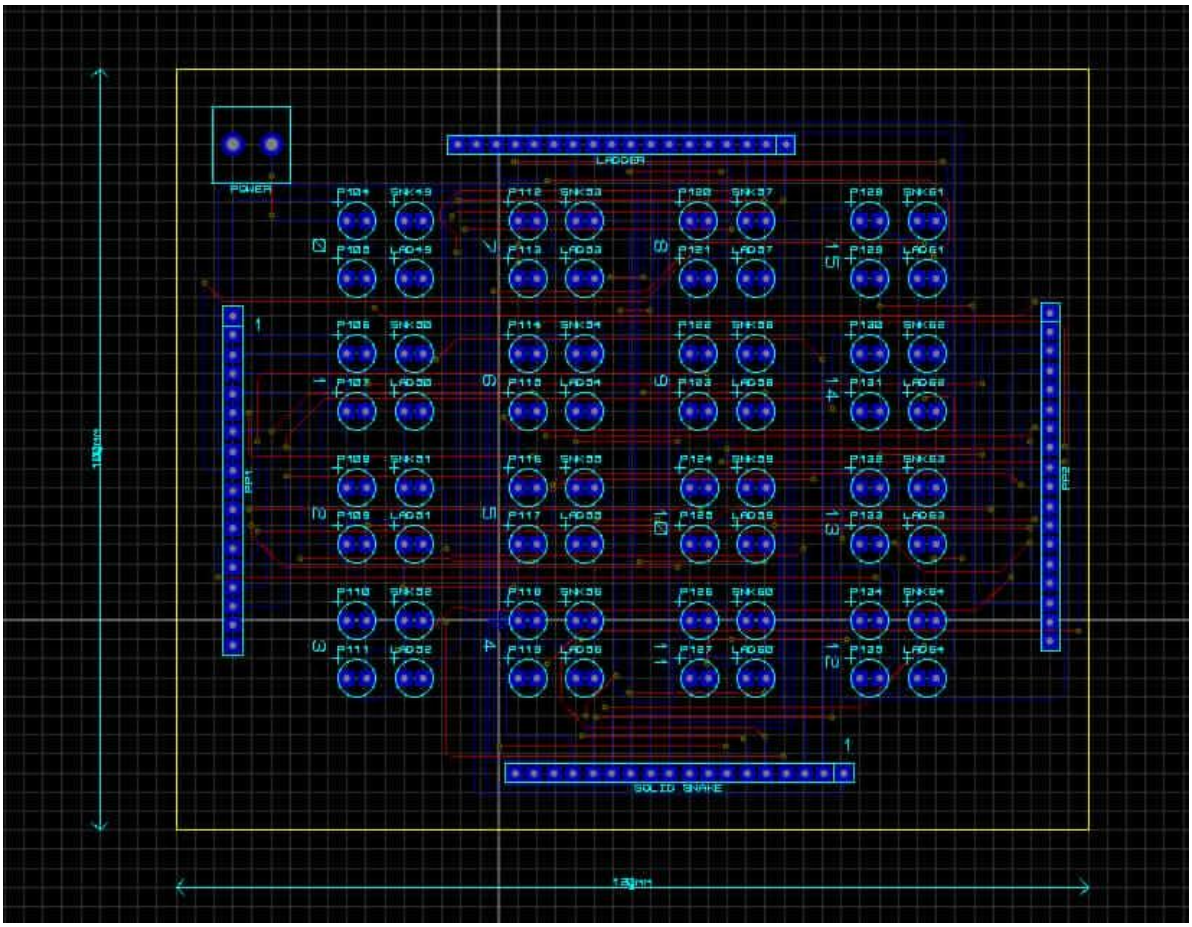
Top Layer Ouput :Mirror



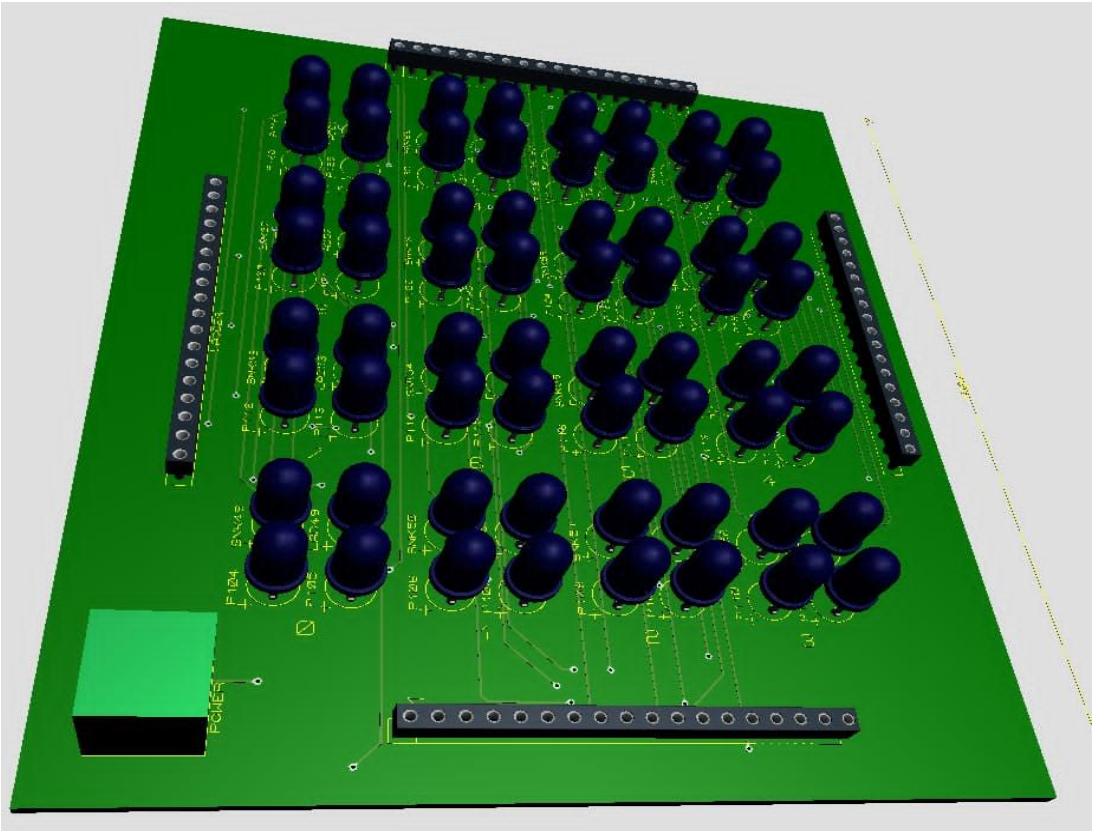
(G) Board Module Schematic design:



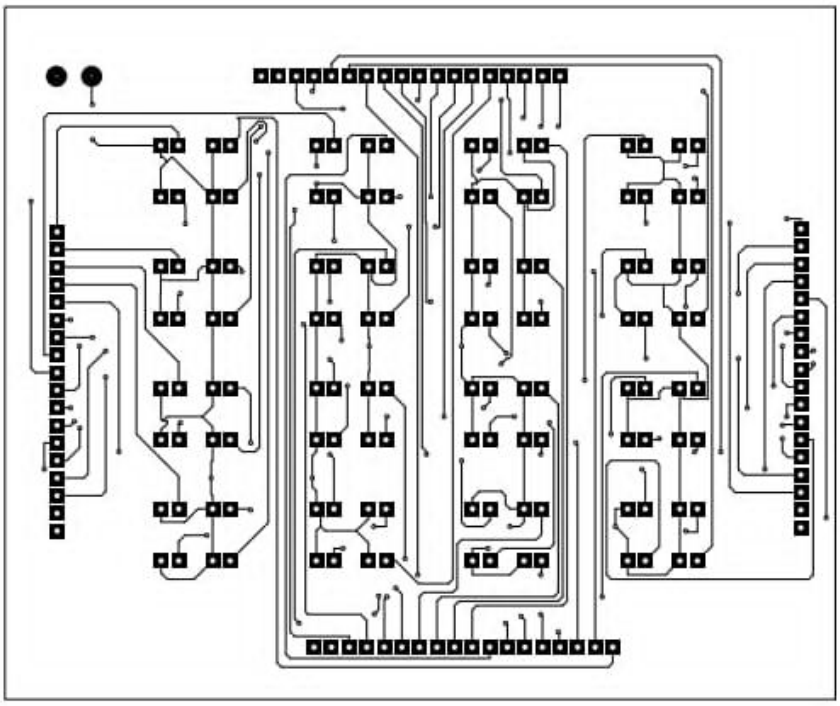
PCB Designing:



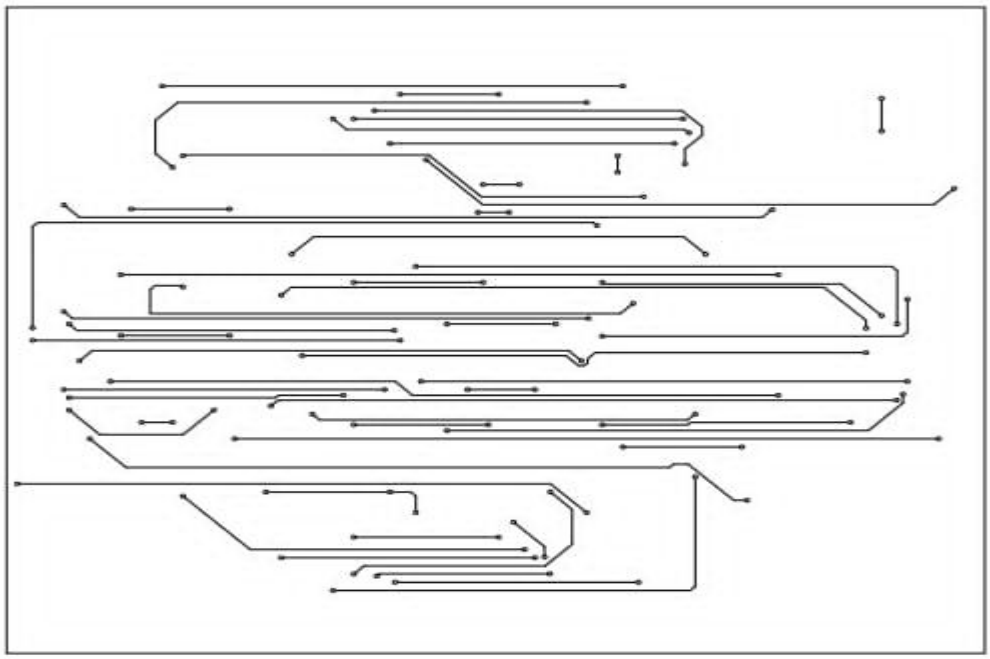
3D Visualizer:



Bottom Layer Ouput :Normal



Top Layer Ouput :Mirror



Discussion:

For designing the PCB of the board LED module, first, a board edge was selected with dimensions width x height = 120 mm x 100 mm. Then all of the components were placed on that board. The pads were replaced with only bottom copper pads to avoid any soldering issues. After aligning the LEDs and other components, design rule parameters were set. For power and signal net class, via type was selected as through hole, and all other settings were left at default. Then, the auto-router was applied. Finally, all the levels (level 0 to 15) were labeled in the PCB's top silk.

Group member contribution:

1. PDF written by Shazzad Ahmed Chowdhury, Mahib Abtahi, Ahnaf Sakif and Hasib Ahmed Anik
2. Snake & Ladder Logic PCB by Shazzad Ahmed Chowdhury
3. Dice Module PCB by Ahnaf Sakif
4. Snake Position PCB by Mahib Abtahi
5. Ladder Position PCB by Fahim Shifat
6. Board Module PCB by Abdullah Jubayer
7. Player position Decoder PCB by Faiad Faisal Sarthok and Fahim Shifat
8. Memory,Reset and Winning module PCB implemented by Shazzad Ahmed Chowdhury, Ahnaf Sakif