Tic Tac toe:

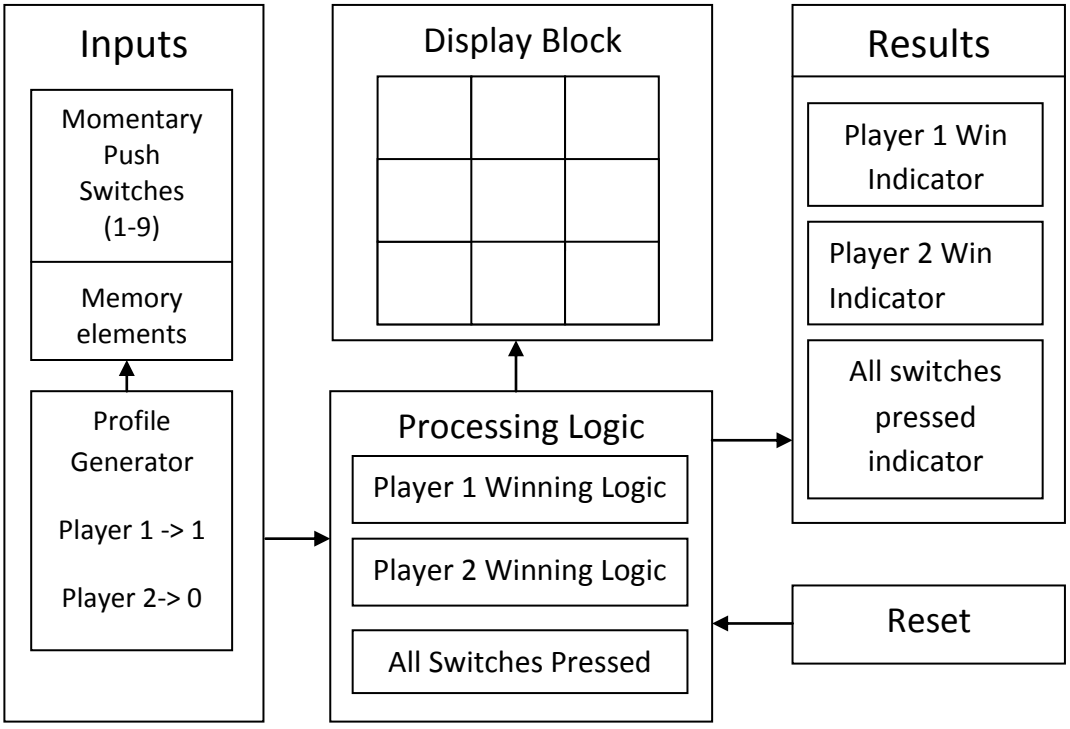
The aim was simple, to design a tic tac toe game, with logic Ic's only.

Requirement:

1. Dual color LED's should be in place, RED should denote player1 and Green should denote player2.
2. The inputs should be got only using momentary micro switches. This makes it easy to reset the game at any point. If we had pole switch or push buttons – it would become an overhead for the player to go and undo all the moves.
3. A player should not be able to override other player's move. For example if player 1 selects the first cell, player 2 should not be able to select the same cell again.
4. Player's should have the ability to reset the game at any point.
5. Once a player wins, the "result" state should be locked. i.e. If player 1 wins first and then player2 makes a moves and satisfies the wining criterion – the output should show only "Player 1" as the winner.
6. If all the switches were pressed or if there is a draw situation – a separate LED should glow.

Block design:

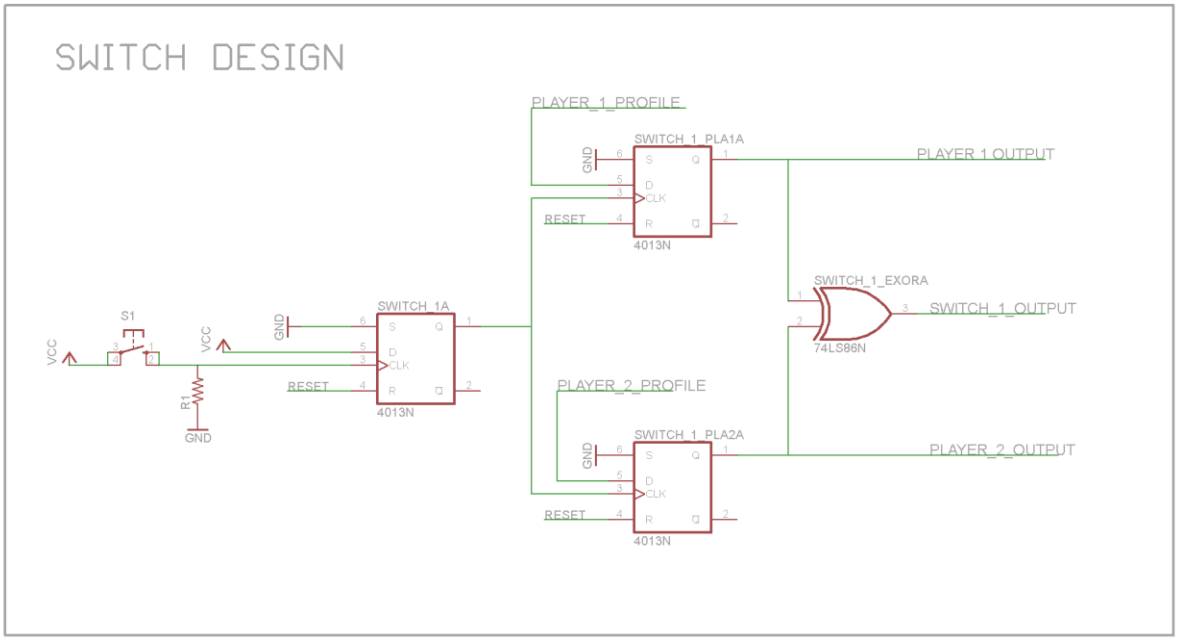
Let's start off with a block representation of the whole project,



Explanation:

Input Block:

The switches gets input from user, these are momentary micro switches, so to know which switch was pressed, we use flip flops (D flip flop) to keep track of input. Also we 'store' which player's move it was, below is the schema we used to achieve this purpose. This schema is duplicated for all nine switches.



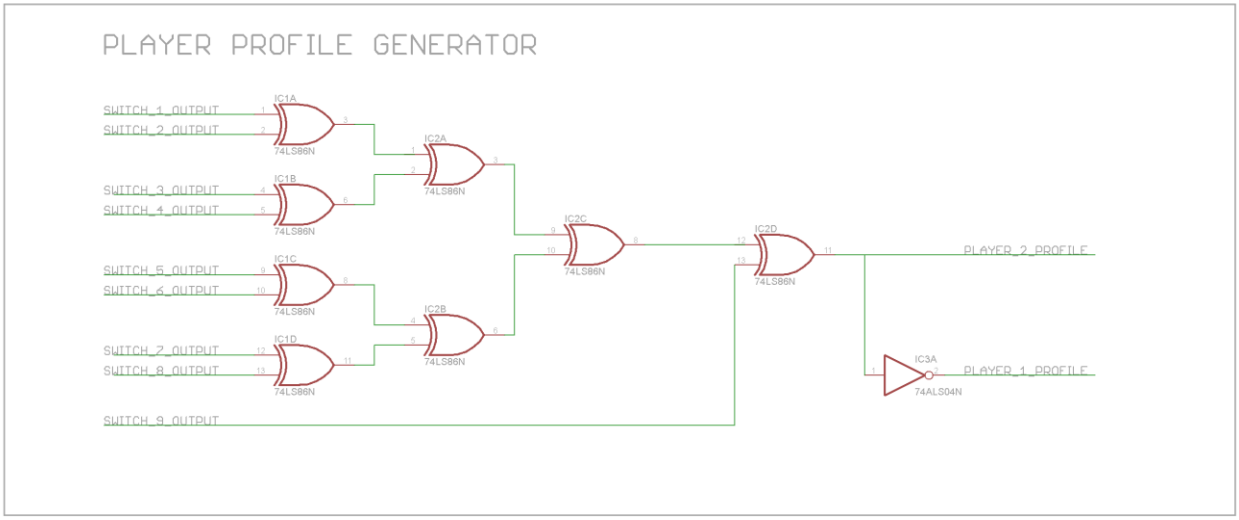
The D flip flops used here are positive edge triggered. SWITCH_1A is the master flip flop which triggers the flip flops SWITCH_1_PLA1A and SWITCH_2_PLA2A. When user presses switch S1, output 'Q' of SWITCH_1A goes from 'low' to 'High' triggering other two flip flops.

Data for other two flip flops comes from the 'Profile Generator Circuit'. A short description of the circuit is that it will have two lines, 'PLAYER_1_PROFILE' and 'PLAYER_2_PROFILE'. If it's player1's turn to play, 'PLAYER_1_PROFILE' line goes 'high' & PLAYER_2_PROFILE' line goes 'low' and vice versa.

Now when the switch S1 is pressed we have the below two cases:
If it is player1's turn, output Q from SWITCH_1_PLA1A would go high.
If it is player2's turn, output Q from SWITCH_1_PLA2A would go high.

Note1: It is not at all possible for output 'Q' from both SWITCH_1_PLA1A and SWITCH_1_PLA2A to go high at the same time.
Note2: The XOR could be avoided by directly taking the output Q from SWICTH_1A

Profile Generator:



This circuit helps us find which player's turn it is currently. It consists of XOR's so if no one has played all the inputs will be 0; hence output for PLAYER_2_PROFILE will be 0 and PLAYER_1_PROFILE will be 1 (as the output is inverted).

Once player1 plays, one of the inputs goes to 1, now the overall XOR output goes to 1, this makes player2_profile to 1 and PLAYER_1_PROFILE to 0.This keeps happening and we get alternate player1 and player2 results. Basically the output toggles for every input change.

Display Block:

This block has 3x3 dual color LED's arranged in a matrix fashion. The outputs of 'switch block' 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 player1 and if the led glows 'GREEN' that move was made by player2

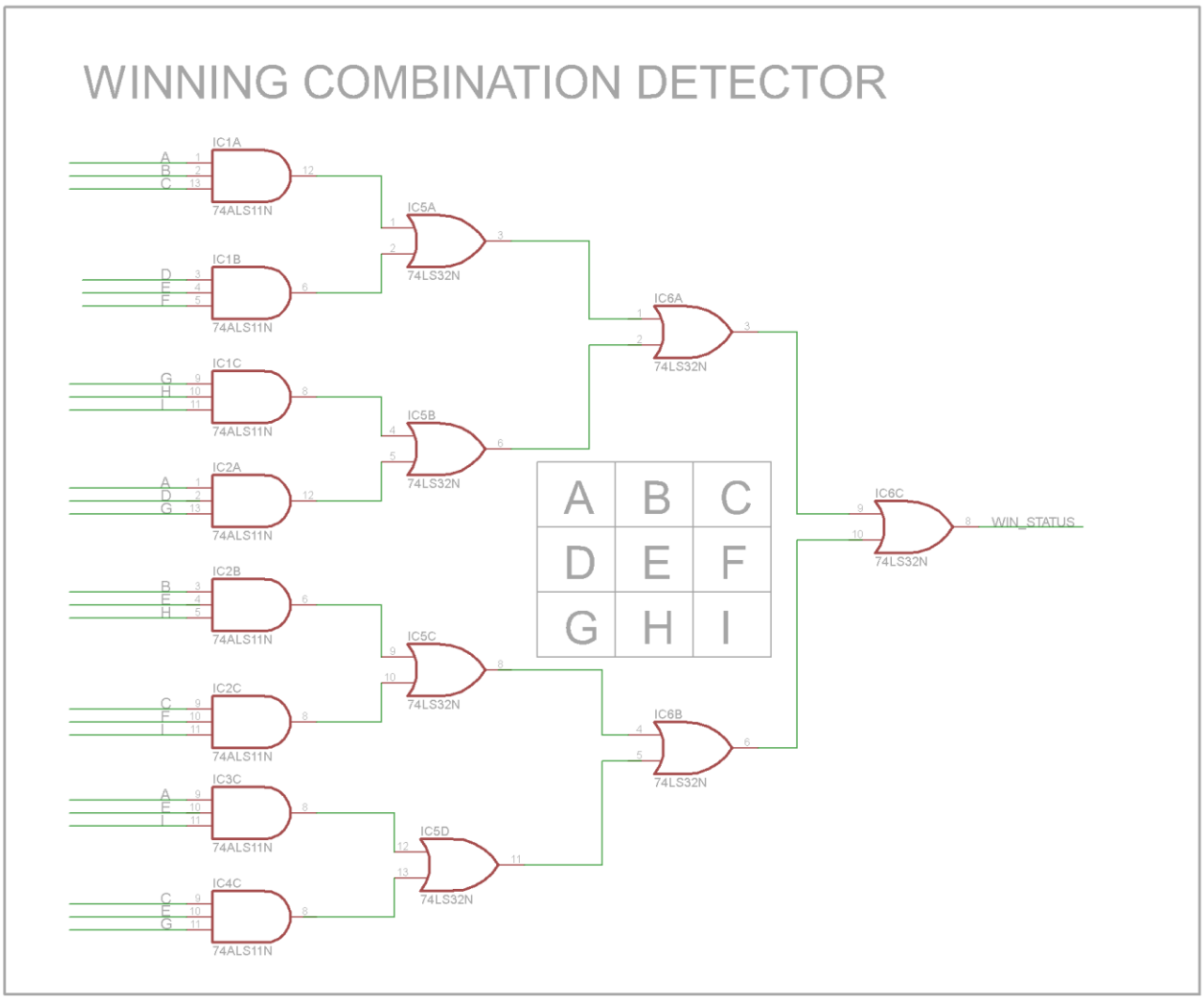
Processing Logic:

Now we have the inputs, we need to process and find out who won. The game says that if a player get 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 Or'ed together.

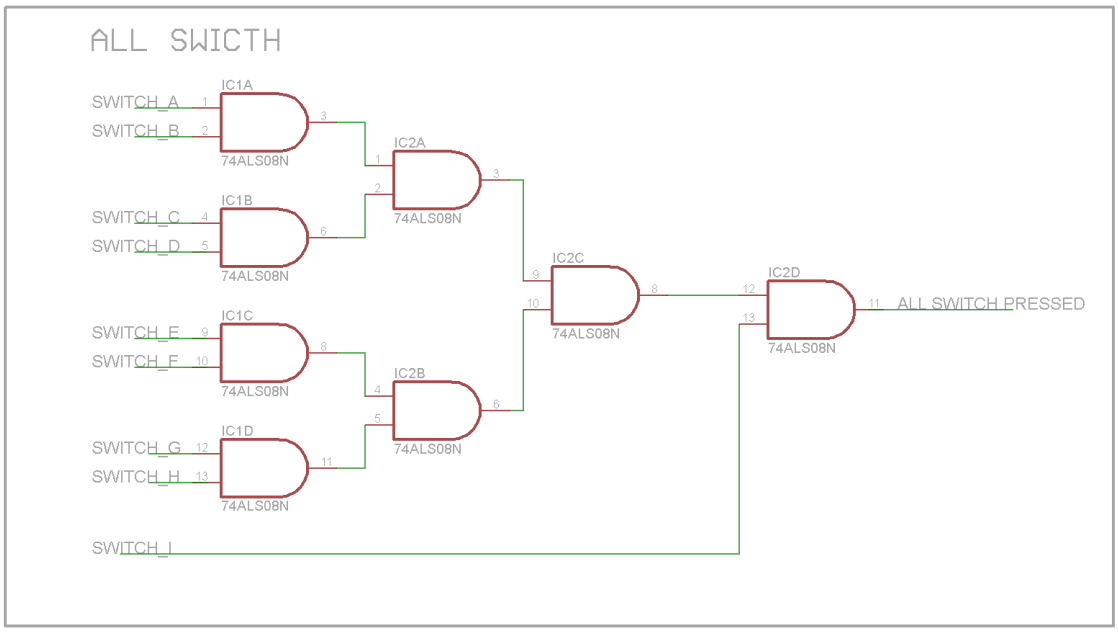
This setup is duplicated for both Player1 and Player2.

If the output of this block is high, means we have a winner!



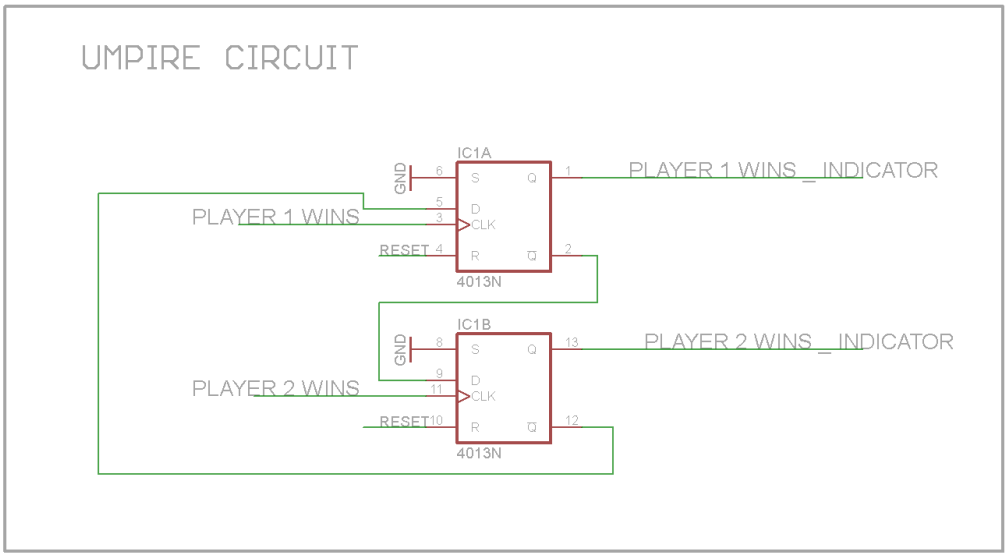
All switches pressed:

Here we just AND all the outputs from the switches. When all the switches are pressed, the output of the circuit goes high, Meaning it's a draw or that a new game should be started.



Result Block:

We can tell who has won the game, but we don't stop 'Player2' from winning after 'Player1' has won and vice versa. This is against the rules, so to put a check the below block is used. The Q^{o} of player1 is connected to data of player2 and vice versa. So only if player2 has not won only then player1 can win. This way the result state is locked.



~ Circuitchef.com ~