Olexiy Burov
1398470
oburov@ucsc.edu
Lab #2

Section 02(Monday,Friday)

The lab problem was to check whether the one of 3-input ANDS was true for 8 such AND gates(which represent line combination in tic tac toe). If one of the 8 was true, then we had a winner(X or O winner). If one of the first 8 ANDs and one of the the other 8 ANDs, then we say it's a tie, and there is no winner.

I represented the game field in a coordinate manner. I created logic for two players, putting AND gates to check each line combination for each player (16 ANDs total). Each 8 AND gates are connected to OR, so that if at least one is true, then we have a winner, because one true AND gate basically means, that we have a line filled with identical values(all O or all X).

I thought about using less logic then I did, it would probably include checking COMMON points for some lines. Because some points are present in three or two lines at the same time, therefore we only need to check the ones that are left to complete the line.

To reduce the number of transistors, we can use NAND gates, we can break up the sum of products we have now into NANDS instead of ANDS and NAND instead of OR using De'Morgan laws. That would allow us to save on transistors and space and power(if we talk about the space on the chip).

First of all I used a very messy way to solve it. I intended to test my logic to see, whether it detects lines in the right way. Once I connected it all and made it work, I started thinking on optimal layout. I decided to use senders and receivers instead of nodes for wires because they result in a neater layout, which has much less wires on the screen. I put all my logic on a different level from where the board is, so I divided logic and user interface, and made them communicate to each other, in a way.

The first thing, which became apparent is that solving problems through gates and logic is so much different from programming to solve that problem. I would've gone through a more efficient way to check combinations if it was in Java or C, but here I had not much freedom to have loops or containers to store states of my lines. Therefore, I chose a more blunt approach, which is check every possible combination for O and X.