CPE 487 Tic Tac Toe

CPE 487 – Tic Tac Toe
Alex Gaskins
Jackie Fang
Nikola Ciric
Rayhan Howlader
Riya Shrestha

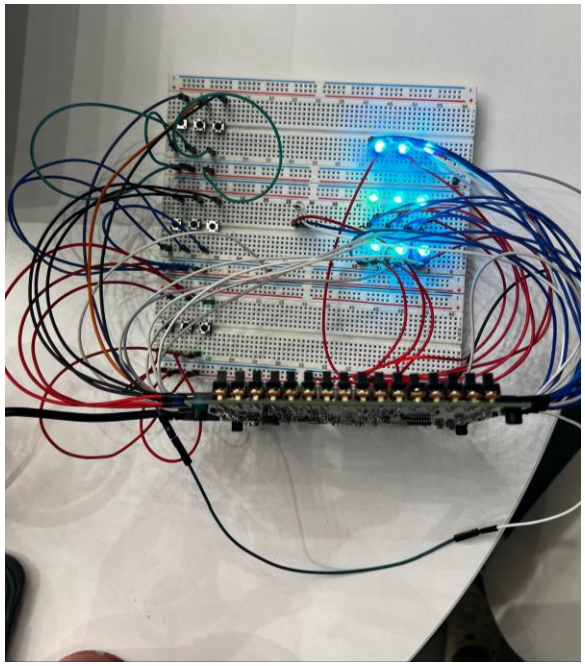
Project Description

- Tic Tac Toe

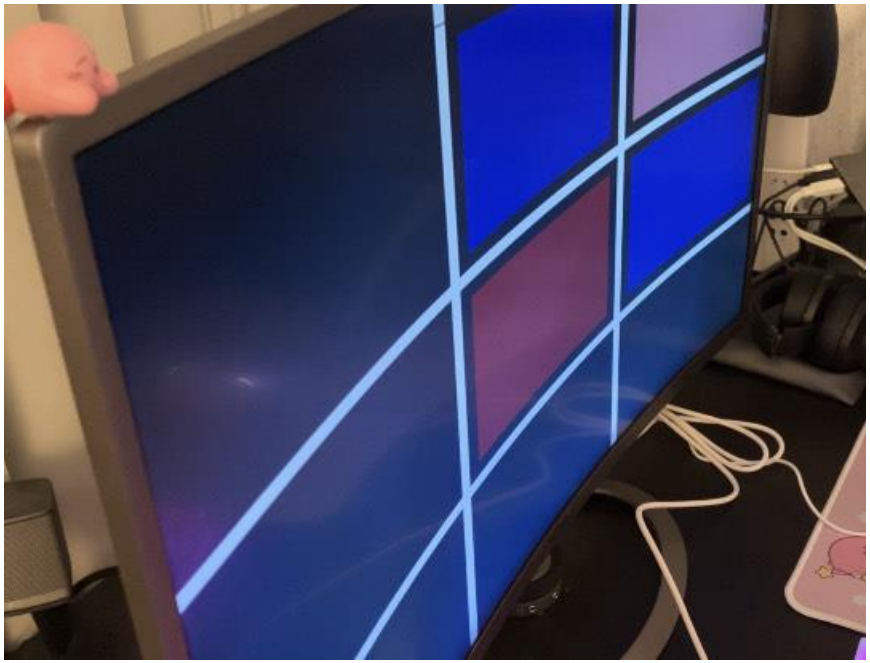
Project is split into two off-board designs (LEDs and VGA)

- Popular game two player game played on paper.
- Game is powered by the FPGA board. Board is displayed by LED lights, and the user interfaced is done through buttons on the breadboard.
- For now, the project is displayed on a monitor and works using a keyboard.

Pictures of Project

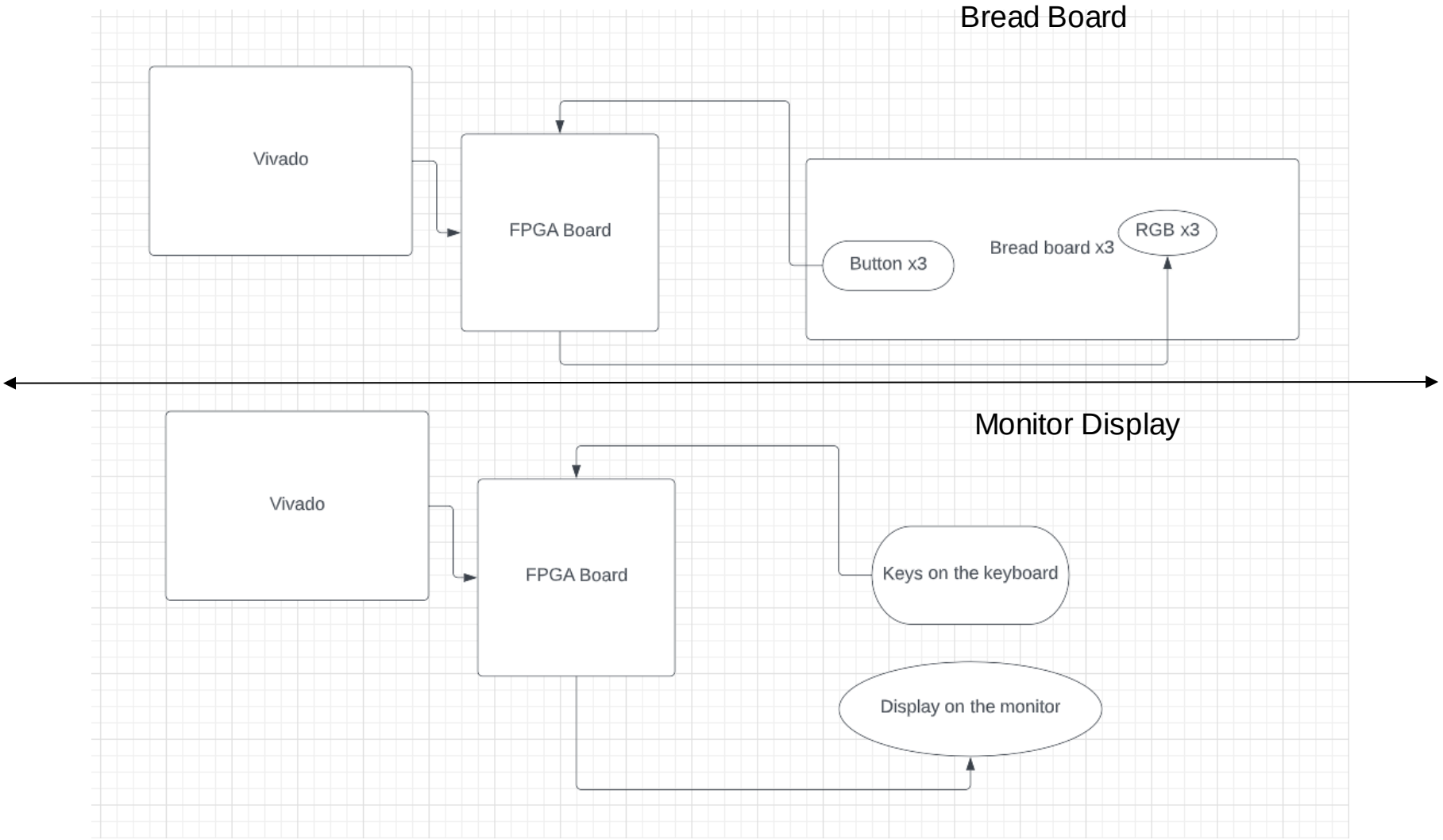


Breadboard Design



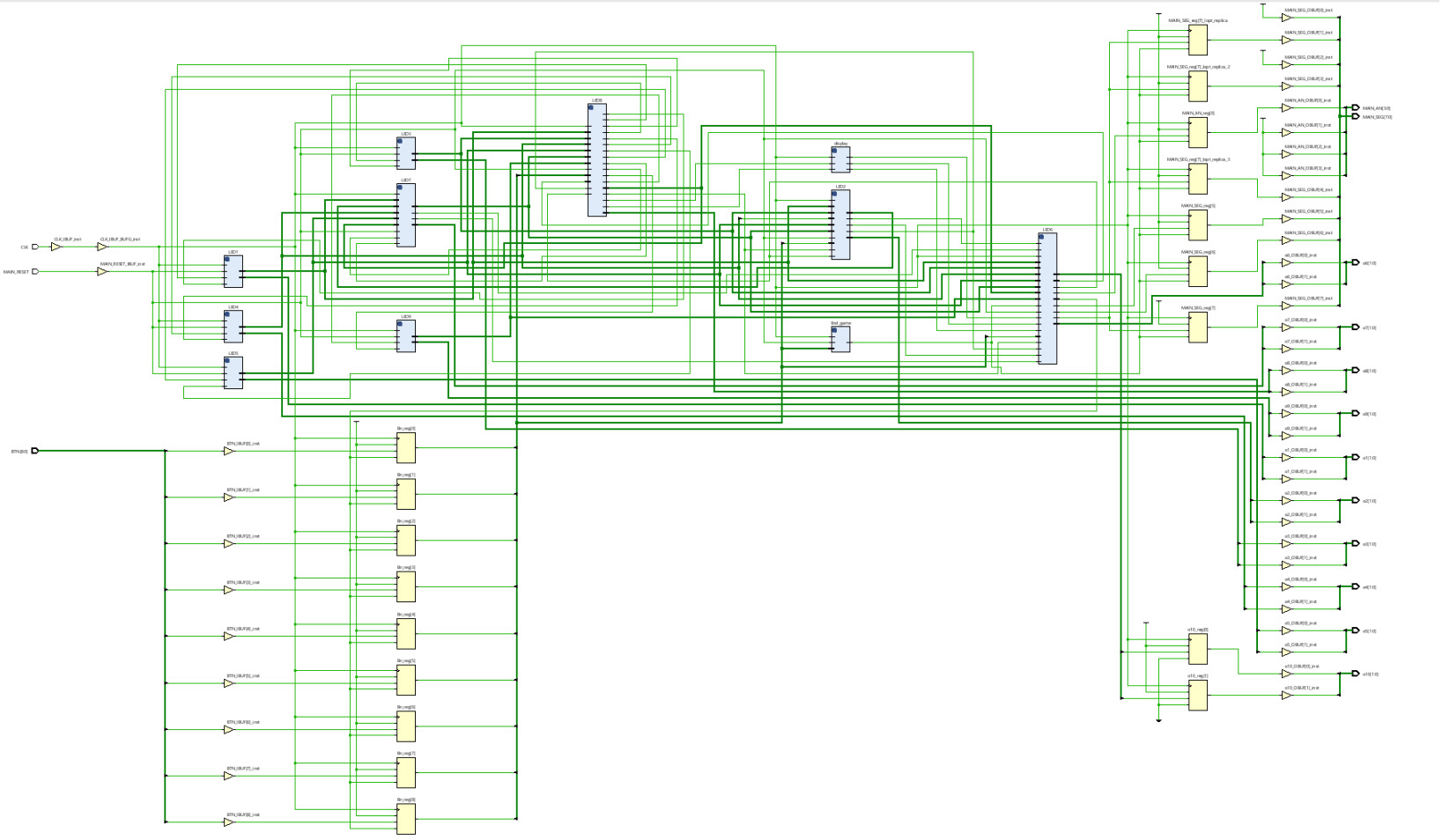
On-Screen Design

Block Diagrams



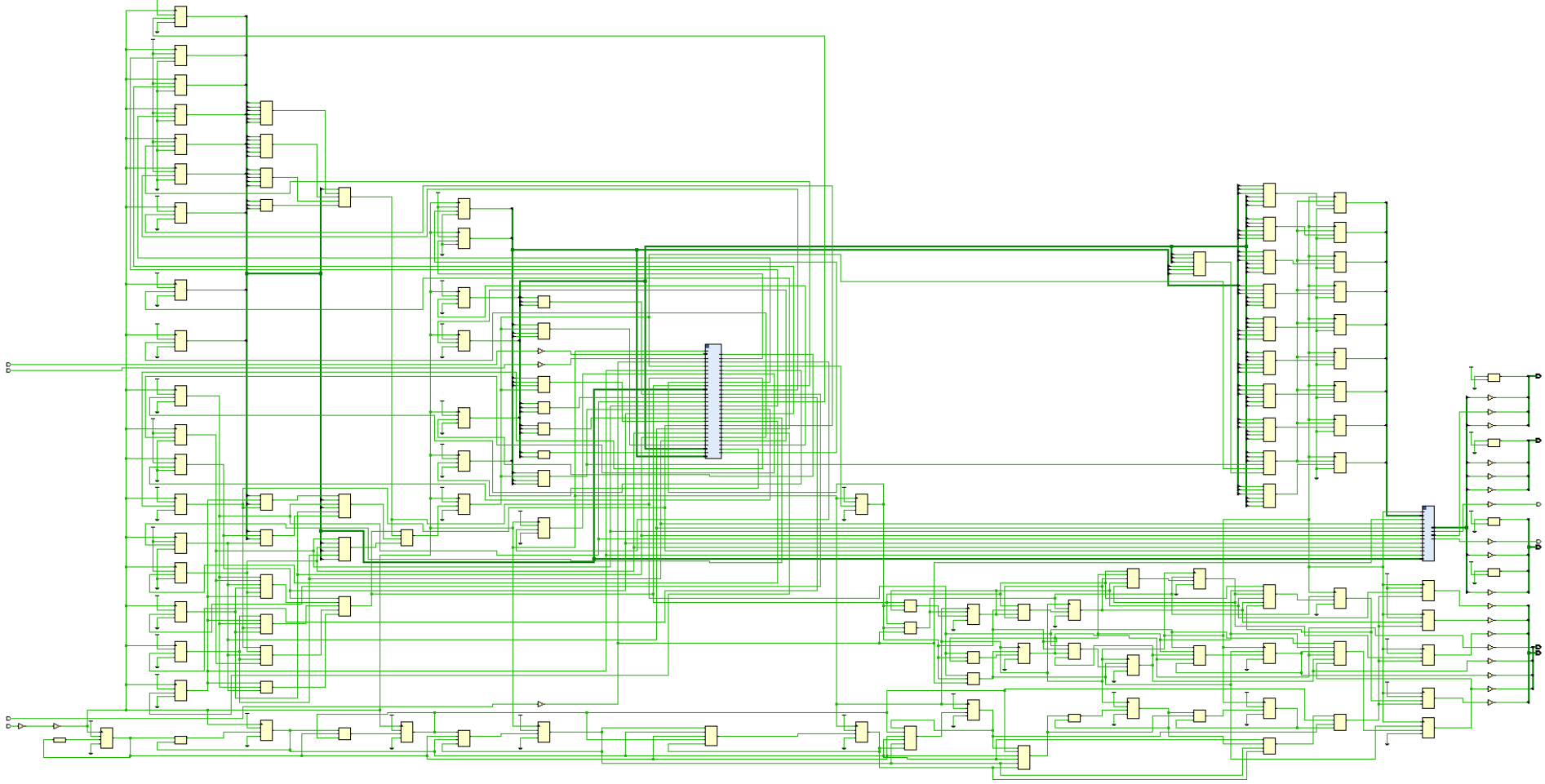
Schematic and Block Diagrams

- Schematic for Breadboard Design



Schematic and Block Diagrams

- Schematic for On-Screen Desian



VHDL Architecture

- Breadboard code
  - Configuring circuit layout and button functionality
  - LED response setup to light up upon pressing a specific button
  - Multiple LED support allows for different signals to be sent to RGB lights depending on which player is playing

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Main_Game_Board is
    port (CLK         : in std_logic;
          BTN         : in std_logic_vector(8 downto 0);
          MAIN_RESET : in std_logic;
          o1            : out std_logic_vector(1 downto 0);
          o2            : out std_logic_vector(1 downto 0);
          o3            : out std_logic_vector(1 downto 0);
          o4            : out std_logic_vector(1 downto 0);
          o5            : out std_logic_vector(1 downto 0);
          o6            : out std_logic_vector(1 downto 0);
          o7            : out std_logic_vector(1 downto 0);
          o8            : out std_logic_vector(1 downto 0);
          o9            : out std_logic_vector(1 downto 0);
          o10           : out std_logic_vector(1 downto 0);
          MAIN_AN       : out  STD_LOGIC_VECTOR(3 downto 0);
        MAIN_SEG    : out  STD_LOGIC_VECTOR(7 downto 0));
end Main_Game_Board;

architecture Modular of Main_Game_Board is
```

VHDL Architecture

- On-Screen Driver Code
  - Playing and checkWinner process is the same as on the breadboard
  - The most important code was that which generated graphics and keyboard input

```
[3:0] i_VGA_R,
[3:0] i_VGA_G,
[3:0] i_VGA_B,
[9:1] i_selected_square_pos,
[9:1] i_player_1_square_pos,
[9:1] i_player_2_square_pos,
[9:0] i_x, // current pixel x position: 10-bit value:
[8:0] i_y  // current pixel y position:  9-bit value:
```

// Variables being generated on the screen

```
display_sq_1_p1  = ((i_x >   0  + square_offset) & (i_x < 208 - square_offset) & (i_y >   0  + square_offset) & (i_y < 155 - squa
display_sq_2_p1  = ((i_x > 218 + square_offset) & (i_x < 411 - square_offset) & (i_y >   0  + square_offset) & (i_y < 155 - squa
display_sq_3_p1  = ((i_x > 421 + square_offset) & (i_x < 640 - square_offset) & (i_y >   0  + square_offset) & (i_y < 155 - squa
display_sq_4_p1  = ((i_x >   0  + square_offset) & (i_x < 208 - square_offset) & (i_y > 165 + square_offset) & (i_y < 315 - squa
display_sq_5_p1  = ((i_x > 218 + square_offset) & (i_x < 411 - square_offset) & (i_y > 165 + square_offset) & (i_y < 315 - squa
display_sq_6_p1  = ((i_x > 421 + square_offset) & (i_x < 640 - square_offset) & (i_y > 165 + square_offset) & (i_y < 315 - squa
display_sq_7_p1  = ((i_x >   0  + square_offset) & (i_x < 208 - square_offset) & (i_y > 325 + square_offset) & (i_y < 480 - squa
display_sq_8_p1  = ((i_x > 218 + square_offset) & (i_x < 411 - square_offset) & (i_y > 325 + square_offset) & (i_y < 480 - squa
display_sq_9_p1  = ((i_x > 421 + square_offset) & (i_x < 640 - square_offset) & (i_y > 325 + square_offset) & (i_y < 480 - squa
display_player_1_square = display_sq_1_p1 | display_sq_2_p1 | display_sq_3_p1 | display_sq_4_p1 | display_sq_5_p1 | display_sq_
```

// Parametric equations used to instantiate grid and square positions

VHDL Component Reuse

Single Led Controller

- D-Flip Flop: used to store data at a predetermined time and hold it until it is needed
- Each of the flip flops will share a clock, the input of the button, and a reset button
- The press of the button will latch in the D value of each flip flop and outputs it
- We will be using 9 of these flip flops for each button

VHDL Component Reuse

Multi-Led Controller

- Used to determine who goes next
- We have decided to use XOR logic gates to do the function of switching between each players
- Output ‘1’ if there is an odd number of 1's and output a ‘0’ if there is an even number of 1's

VHDL Digital Circuits

Victory Conditions

- Victory conditions consist of: vertical, horizontal, diagonal (total of 8 possible conditions
- Used "if" statements inside of processes to handle all 3 conditions
- Second process that compares the P1 and P2 win state to determine if there was a tie
- Finally, we must pass the values stored in the signals to our outputs

VHDL Digital Circuits

Main Game Board

- Connect everything spoken before to work together
- We have 9 separate LED outputs plus one to show which player won at the end
- Integrate all the modules using lots of components
- Finally, we needed to create signals and pass them between the inputs and outputs of the modules written before

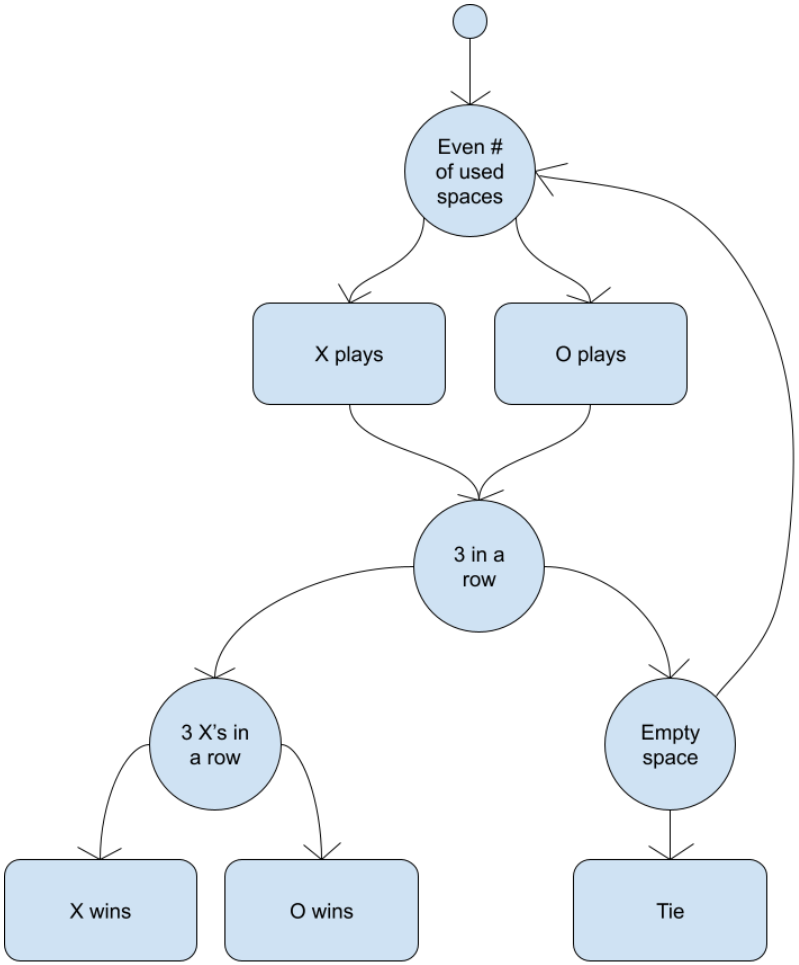
VHDL Digital Circuits

Ending the game in a tie

- To prevent the user to change their input we will use 9 D flip flops
- D set to '1' automatically to determine if the buttons have been pressed or not
- The module output will get the result of all the outputs of the D flip flops together

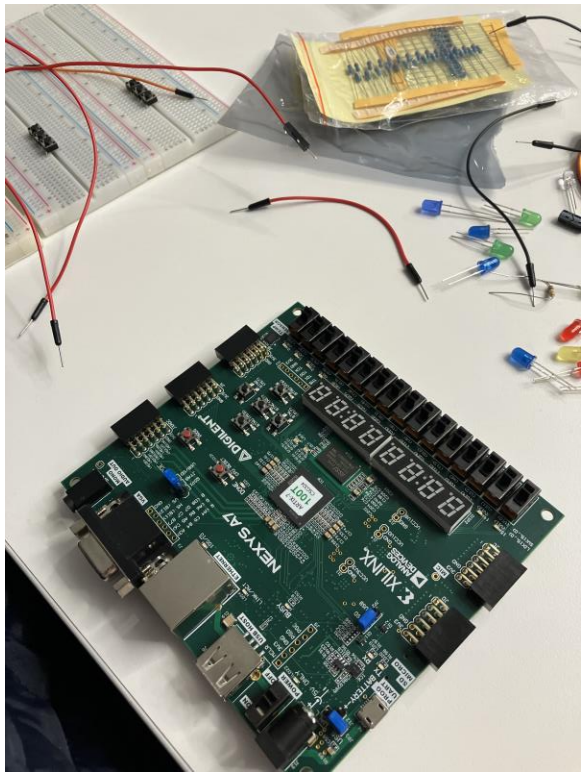
State Machines

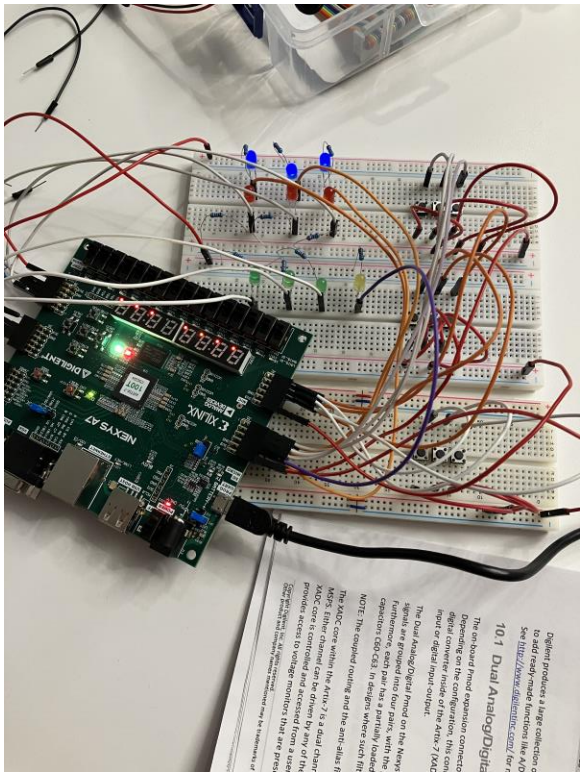
- Because both the inputs and the state impact the system output, this is a Mealy machine



Input 1	Input 2	Input 3
Input 4	Input 5	Input 6
Input 7	Input 8	Input 9

Testing


NEXYS A7
XILINX


10.1 Dual Analog/Digita


NEXYS A7
DIGILENT
XILINX

Off-Board Circuitry

- We had gone off the board using a breadboard

- There are multiple ways to display the tic tac toe game to the user. This is exemplified as we have explored two methods (On-Screen and On-Board)
- We had decided to use the breadboard that would use buttons that would light up LED lights as that would be an interactive way for users to play as well as an interesting way to incorporate other parts of this class into the project
- Which we plan to do before the 22nd of December!

Demonstrate

We will now demonstrate our project on a monitor