Tic-Tac-Toe Game

Sabrina Hsia, Jackie Huang

Introduction

- Tic-tac-toe is a game in which players alternate turns to complete a row, a column, or a diagonal with three markings in a 3x3 grid.
- This project will combine our knowledge gained from current and previous courses to implement this game using Verilog and display it using breadboards and RGB LEDs. The game will be played using the switches on the Nexys A7-100T board.
- Sequential circuits, finite state machines, and breadboarding are the concepts that this project will demonstrate.

Implementation

- There are 7 total modules needed to implement all the rules of the game and determine a winner.
- Position Tracker, Position Decoder, Illegal Move Detection, No Space Detect, Winner Detect, Winner Detector, and Turn Tracker.

Position Tracker

- Keeps track of what values are held in all 9 positions of the board
- Assigns empty spaces with a player’s marker based on inputs
- 3 values possible on the board
    - 2’b00 - Empty Space
    - 2’b01 - Player 1 (Blue)
    - 2’b10 - Player 2 (Red)

Position Decoder

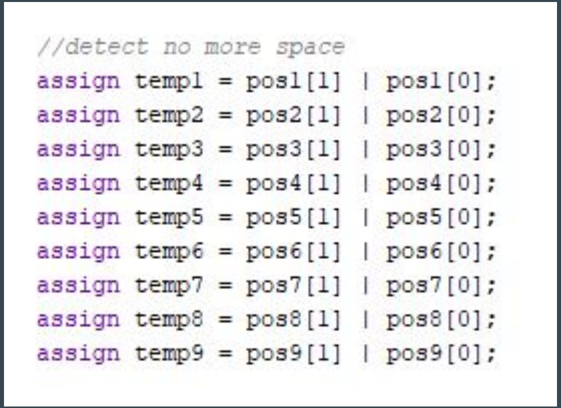
- Converts a 4-bit input from the player into a 9-bit output
- Each of the nth bits in the 9-bit output represents a position on the board
  - I.e 4’b1000 = 9’b100000000 = 9th position
  - 4’b0010 = 9’b000000100 = 2nd position

Illegal Move Detection

- Prevents a player from making any moves that violate the rules of the game
  - I.e. overwriting one of their own positions or the opponent’s

No Space Detection

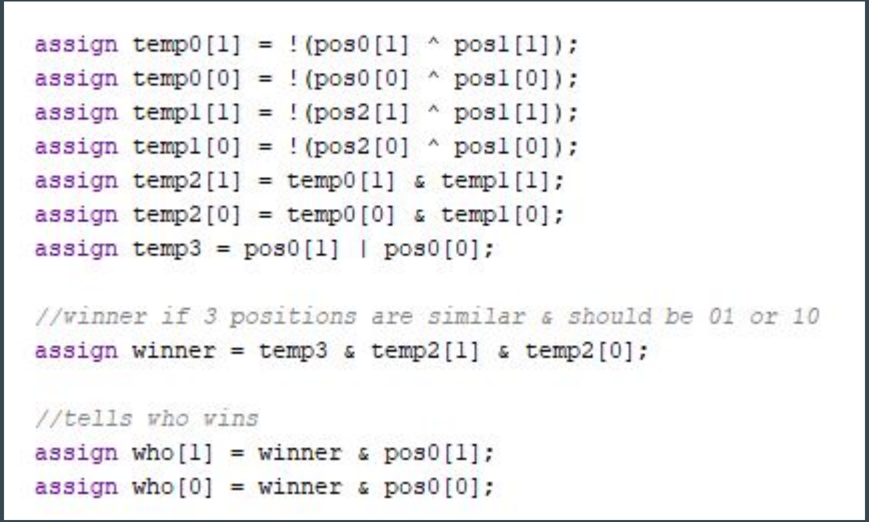
- Compares all 9 positions’ 2-bit inputs to see if there are any empty spaces left to play
- If there is no winner and no spaces left, the game ends in a tie.



```
//detect no more space
assign temp1 = pos1[1] | pos1[0];
assign temp2 = pos2[1] | pos2[0];
assign temp3 = pos3[1] | pos3[0];
assign temp4 = pos4[1] | pos4[0];
assign temp5 = pos5[1] | pos5[0];
assign temp6 = pos6[1] | pos6[0];
assign temp7 = pos7[1] | pos7[0];
assign temp8 = pos8[1] | pos8[0];
assign temp9 = pos9[1] | pos9[0];
```

Winner Detect

- Compares 3 positions on the board to see if they are similar
- Does not check for win conditions



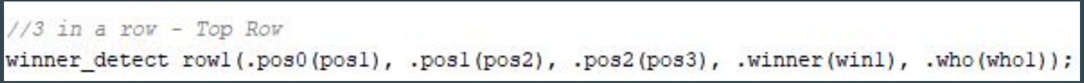
```
assign temp0[1] = !(pos0[1] ^ pos1[1]);
assign temp0[0] = !(pos0[0] ^ pos1[0]);
assign temp1[1] = !(pos2[1] ^ pos1[1]);
assign temp1[0] = !(pos2[0] ^ pos1[0]);
assign temp2[1] = temp0[1] & temp1[1];
assign temp2[0] = temp0[0] & temp1[0];
assign temp3 = pos0[1] | pos0[0];

//winner if 3 positions are similar & should be 01 or 10
assign winner = temp3 & temp2[1] & temp2[0];

//tells who wins
assign who[1] = winner & pos0[1];
assign who[0] = winner & pos0[0];
```

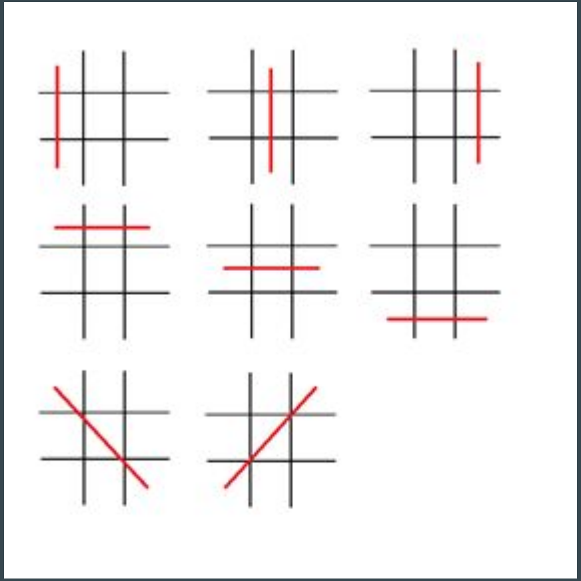
Winner Detector

- Uses the Winner Detect module to check all 8 ways a player can win the game



```
//3 in a row - Top Row
winner_detect row1(.pos0(pos1), .pos1(pos2), .pos2(pos3), .winner(win1), .who(who1));
```

Sample of Code



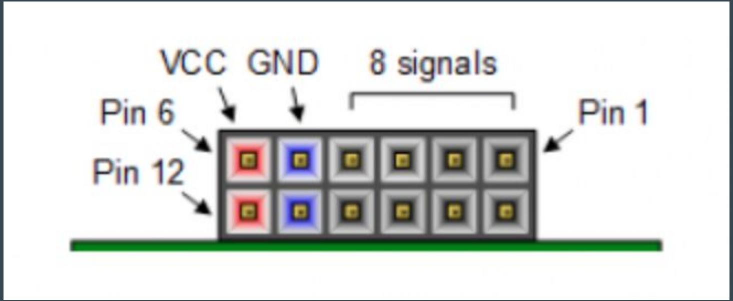
Turn Tracker

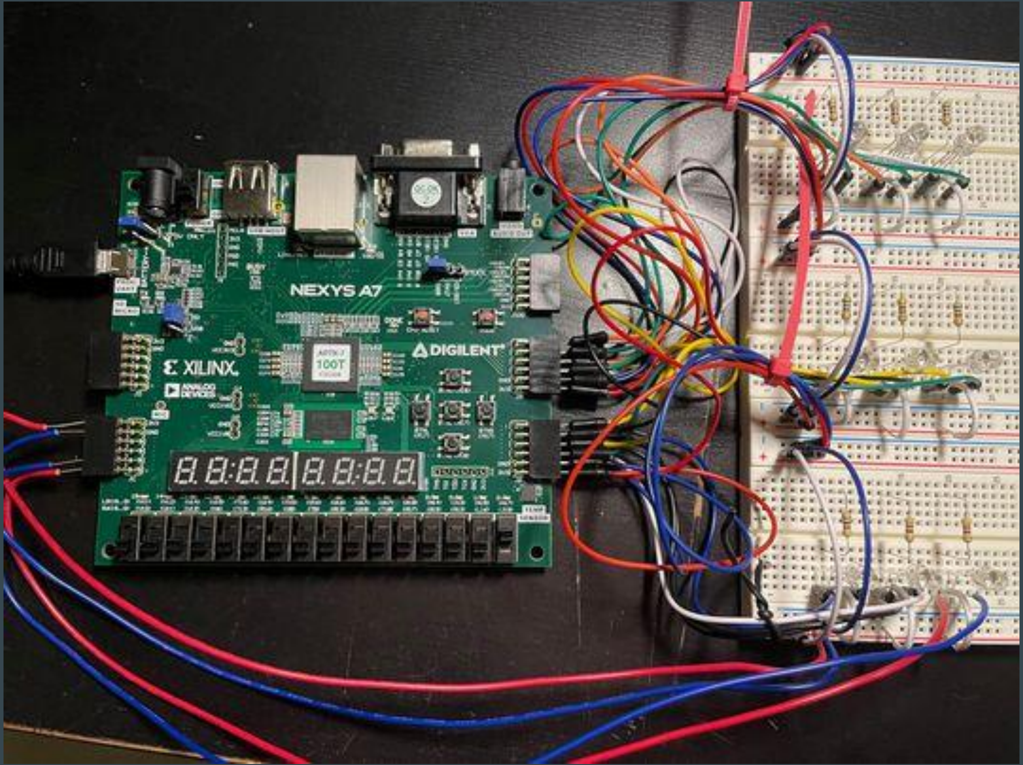
- Uses a finite state machine to determine which state the game is in
- There are 4 conditions the game can be in:
  - IDLE, Player 1 turn, Player 2 turn, and Game Over
- Uses Illegal Move, No Space Detector, Winner Detector to determine the state the game should be in

FPGA to Breadboard

- Pmod ports will be used to make the connections between the FPGA board and the breadboards.
- Each RGB LED will be in series with a resistor and the red and blue pins of the LEDs will each be connected to two pins of the Pmod ports.
- Resistors used are 470 Ω and 560 Ω.

PMOD to LED Connections





Game Controls