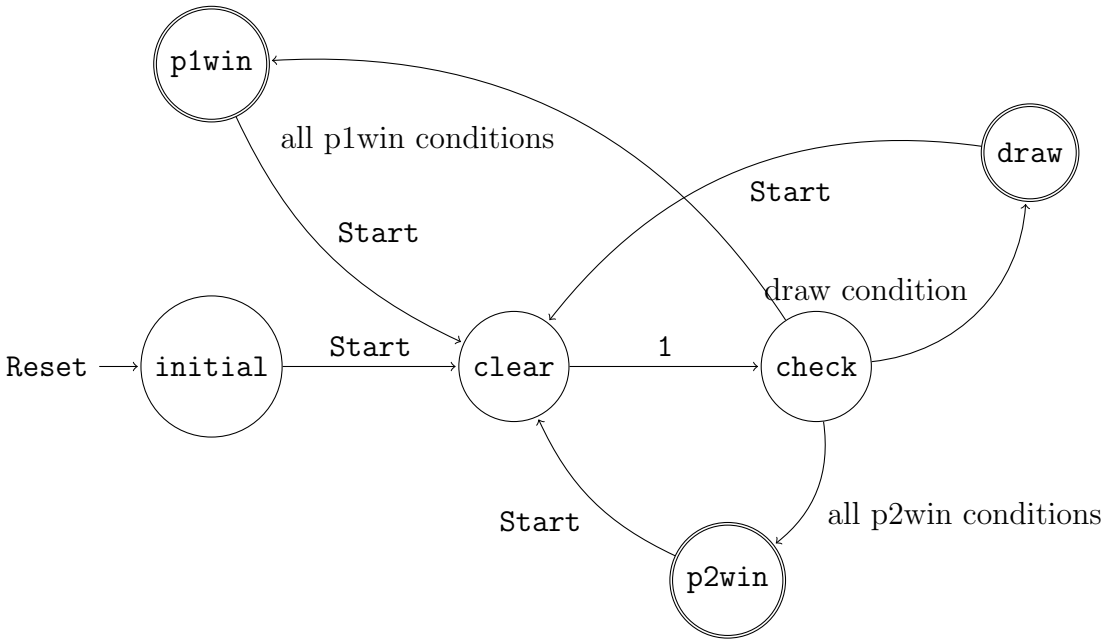
Minh La, Luke Yang
EE 201
Instructor Gandhi Puvvada
Due Dec 3rd
Final Project Report

1. Summery

This final project report is on a TicTacToe game implemented on Nexys3 FPGA Board with a VGA display. It consists of a core machine that handles the gaming, and a front-end that coordinates the I/O signal, calculates all graphical content, and calls the `hvsync_generator` module to produce the display.

2. Backend



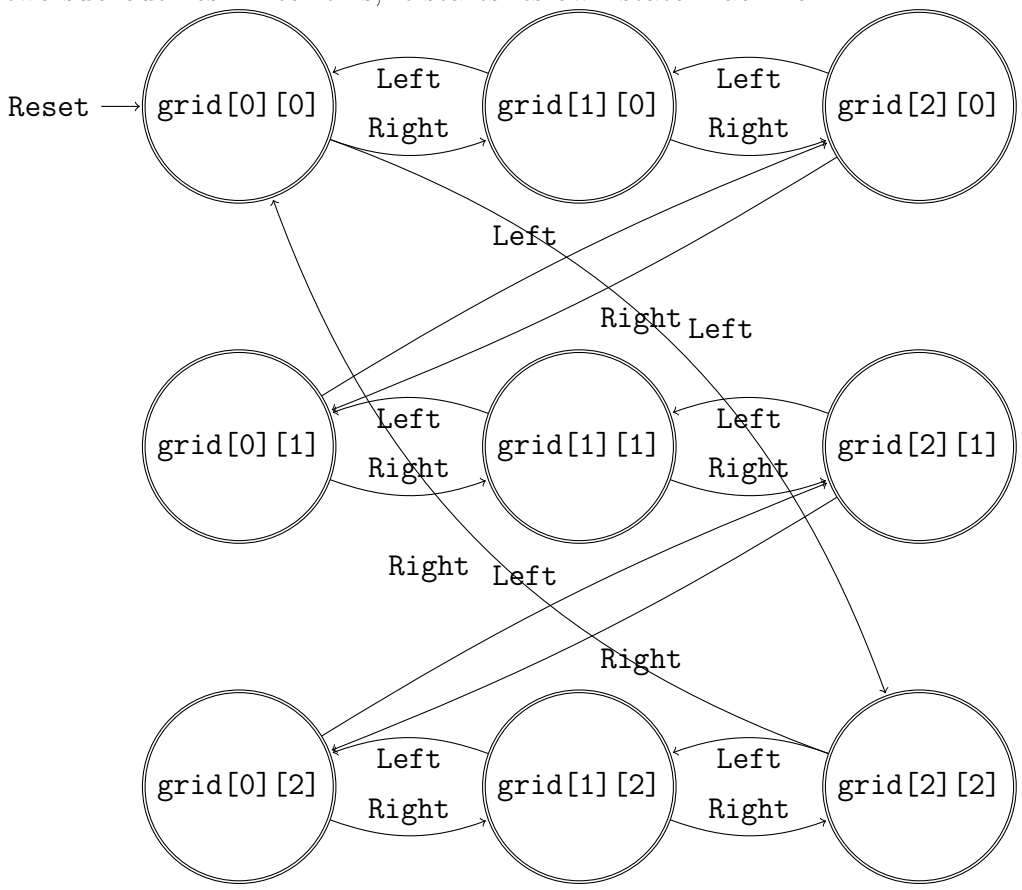
As shown in the diagram, we implemented a six-state state machine in the core design, with initial state being `initial` upon an asynchronous `Reset` signal from `Button_Up`. Push button `Button_Down` produces signal `Start`, which brings the system into `clear` state. Inside `clear` state, the system initializes all the local registers for game bookkeeping. These registers include `gameOver` true when a winner has been found, `location[3:0]` keeping track of the cursor, `p1Win` true when player 1 wins, `p2Win` true when player 2 wins, `draw` true when the board full but no one wins, and `[1:0]mapBoard[8:0]` recording all moves on board in an array. After initialization, the system moves into `check` unconditionally.

In `check` state, two players play in turns and the system processes input signals. The player can move the cursor around the board using `Button_Left` and `Button_Right`, and the value of `location` changes accordingly. After placing the cursor at a desired location, the player can confirm the selection by pressing `Button_Center`, which generates `Enter` signal. Upon this signal, the system in check state first updates `mapBoard[location]`, where `location` represents the cursor's location. The value of `mapBoard[location]` is updated to 1 for player one, and 2 for player two. Players take turn to play this game manually, as signal `player` is toggled by `Switch0`. After updating `mapBoard` array, the system goes through

each row, each column, and each diagonal, and checks if the values are the same. If they are the same, the `gameOver` flag is set on, and the system moves into accepting states `p1win` or `p2win`, depending on whether it finds `1, 1, 1` or `2, 2, 2`. At the end of each round's check, the system also detects whether the board has been full. If the board has been full but `gameOver` flag has not been set on, the system moves into `draw` state since no further moves can be taken and neither player has won. If none of the aforementioned jump conditions are satisfied, the system stays in `check` state and waits for next input. The three accepting states `p1win`, `p2win`, and `draw` simply update the bookkeeping registers, and wait for `Start` signal to bring the system back to `clear`.

3. Front End

The front-end is the top design of this project, and it calls the backend and `hvsync_generator` as two subroutines. After this, it starts its own state machine.



This nine-state state machine corresponds to the 3×3 grid in the graphical front-end, and it shares the same set of control signals `Reset`, `Start`, `Left`, `Right`, and `Enter`. To ensure proper synchronization with the backend, it also reads values of `p1Win`, `p2Win`, `draw`, and `player` from the backend. The nine states represents the nine grids in the game as well as the nine-element `mapBoard[]` array in the backend. The system can jump back and forth among the nine states using `Left` and `Right` since the states are circular doubly-linked. Upon `Reset`, the system resets all local registers for colored area enabling, and goes into `G00` state. In fact, the system also waits for the core to get into `check` state, represented by `p1Win || p2Win || draw` being false. When the game is in session, the system updates the

cursor location based on current state, and reacts to `Enter`, `Left`, and `Right` signals. Signal `Enter` enables green or blue color being displayed at current cursor location depending on `player` value. Signals `Left` and `Right` simply moves the state back and forth, with the effect of cursor moving.

we have also used seven-segment-display LEDs to display the game's status. The first display `SSD0` displays the player's number (0 or 1). The next three displays `SSD1`, `SSD2`, and `SSD3` correspond to game's results: `draw`, `p2Win`, and `p1Win`.