Final Project [Tic-Tac-Toe]

(In 2 dimension)

ECE 249

Session: 3-6pm

TA: Jill Cannon

Joseph S Kim

Ghazy Mahub

Introduction

As a final project for ECE 249, we will develop a multi-player tic-tac-toe game with the knowledge we learned from the previous laboratory exercises. For our project, we will need a user interface, a game controller, a display for game and the winning status of the game.

Design

We tried to think a project, which can be developed with half for TTL implementation and half assembly code of HC12. We thought of creating a simple game and as a result, we came out an idea for making a tic-tac-toe game. Tic-Tac-Toe is played by two players and on a board with 9 spaces that are arranged in 3 rows and 3 columns. The players are assigned a marker that is either an X or O. The players then take turns placing their mark in an empty space until all the spaces are full, or until one player gets 3 of their marks in a horizontal, vertical, or diagonal row. If the later happens, that player is the winner of the game. However, if all the spaces on the board are filled and neither player has 3 of their marks in a row, then that game is over, and there is no winner. We will use TTL for showing the game status, and use HC12 micro-controller to control the game.

Design Issues

There are lots of question arise with our design. We separated our project into different parts, thought about every problems, and made a decision for each problem. The following are the design issues that we encountered.

1) What type of game we want to make?

We have different options for our tic-tac-toe game. We can create either two human players game or AI (micro-controller) Vs. human player. We thought of creating both options for our project. We realized that creating AI control part in our project will be a challenging one, and we kept in mind AI part as the goal of our project.

2) What will we use for the player inputs?

We decided to use the I/O box switches as our input device at first, and if time permits, we will change the keypad as our input device. We thought using I/O box switches was easier but later we found out that using keypad is more efficient and easier to write the assembly code.

3) How do we hold the player's inputs?

We will use TTL for holding the data because in earlier laboratory exercises we used D and JK flip-flops to hold the data. LEDs are best suited for the data that we stored for our game and we will need two colors for two players. The design of our tic-tac-toe data storing device is shown in the TTL implementation part.

4) Where will the win/lose/tie output display and which device will we use?

This is another important issue for showing the game's result. There are eight different ways to win the game and we decided to use VHDL instead of TTL because we just need to compute the input combinations. Using TTL is possible but we avoided for saving our debugging time.

5) How do we control the game for the computer move?

We decided to use MC68HC12 for AI control part since we knew that we could write assembly code and achieve the desired outputs. We expected that it would be the hardest part of our project.

Hardware/software distribution

As we planned before, we will create our project with half for hardware part and half for software part. We took hardware part and the other team took software part writing both assembly and VHDL codes for our project. And also we developed the TTL implementation.

Implementation

1. Required Hardware Components

1) MC68HC12 micro-controller
2) Vantis Mach 211 CPLD
3) Keypad
4) Keypad decoder (74C992)
5) 9 Bi-color LEDs
6) 3 single color LED
7) 5 D flip-flops (74LS74)
8) 5 XOR (7486)
9) 1 inverter (7404).

2. Decisions for Hardware Choice

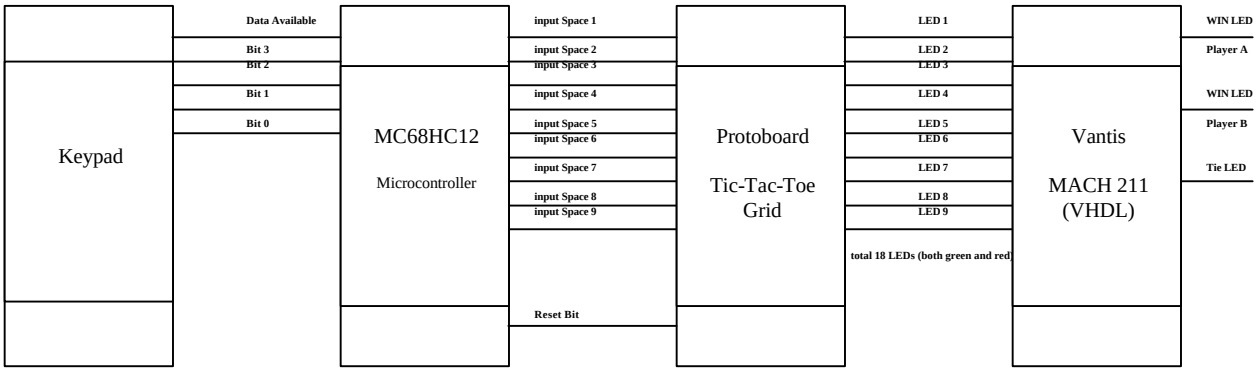
Experiences in earlier laboratory exercises gave us a lot of knowledge for choosing devices. We knew that MC68HC12 micro-controller is capable of generating AI output and we chose it as our control device. As we stated earlier, we chose Vantis CPLD for checking wining output combination and to get the simple TTL implementation. Before choosing the keypad, we used I/O box switches for the user interface but later while developing the assembly code, we found out that using keypad and keypad decoder is much more easier and it simplifies the assembly code a lot. The problem in implementation of assembly code is discussed in detailed later.

We could use either JK flip-flops or D flip-flops as our data-storing device but we chose D flip-flop to get the simple logic. Bi-colored LEDs are necessary for our project because we need to show the input of different players. XOR chips are also necessary for giving the player turn output. The detailed design for TTL implementation will discuss later.

3. TTL implementation

Block Diagram

Block Diagram

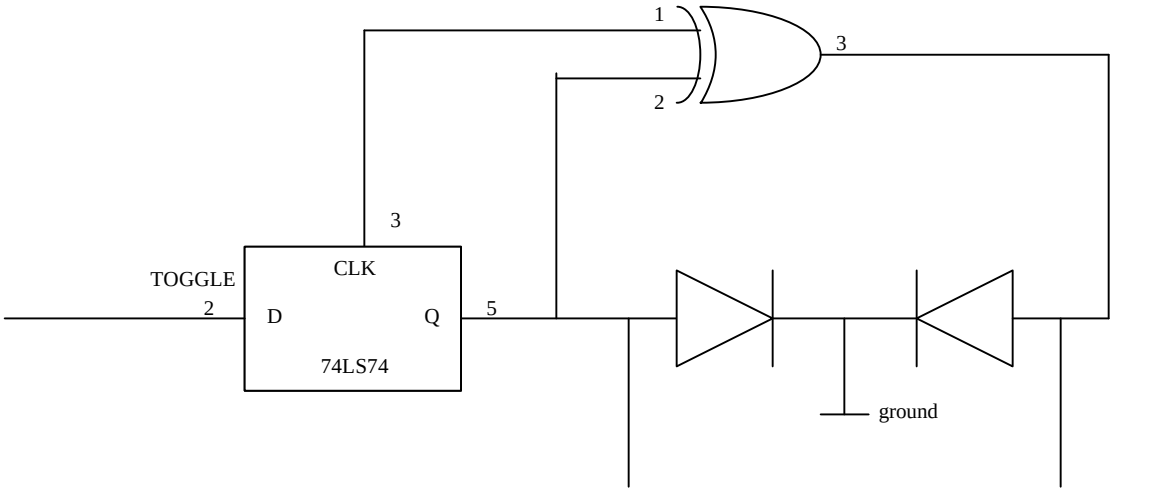


Tic-Tac-Toe Grid

Tic-Tac-Toe Grid

1	2	3
4	5	6
7	8	9

We used D flip-flop as our data-storing device as shown below.



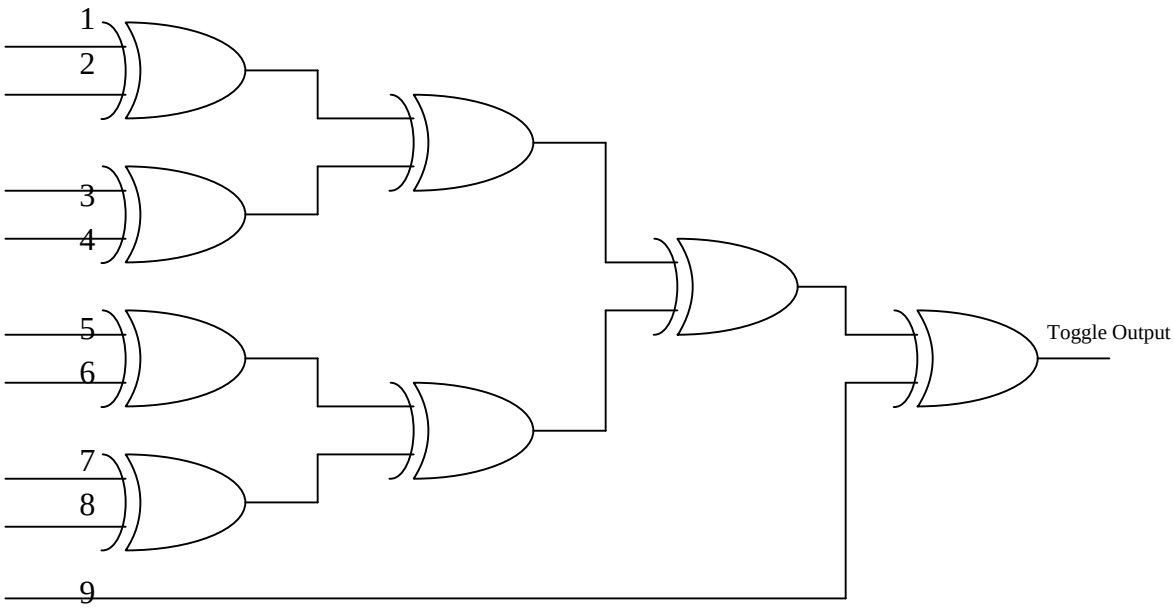
LED 1a (green) LED 1b (red)

The data-storing device will store the data such that: if the clock input of D flip-flop becomes Hi, D flip-flop will store the TOGGLE input in D latch. As a result of storing data, one of the LED will light up. If the TOGGLE input is Hi, the left LED will light up, and if the TOGGLE input is Low and Clock is Hi, the right LED will light up. Therefore, we will have two different outputs for show the result. This design has one error and it is as follows. If the clock input is Hi and goes back to Low, Q output of D flip-flop will be Hi and as a result, the left LED will light up. At the same time, the output of XOR will be Hi, and therefore, the right LED will also light up and we will get the yellow colored LEDs instead of red or green. We can ignore that error because in our tic-tac-toe game, if one player selected a space, the clock input cannot be Low anymore. The complete diagram for all nine data-storing device is shown on the attached sheet.

Player Turn Controller

The TTL implementation for player turn controlling signal is simple. We cascade the XOR chips so that it will give the toggle output if one of the space is selected. This part is the simplest TTL implementation that we designed for giving the player turn. Each input is connected to the clock input of D flip-flop so that it will generate a toggle output if one player selects the empty space.

Player Turn Controller



Input from Microcontroller
Output to Tic-Tac-Toe Grid

Testing and Debugging

For the first 3 weeks of our project period, we developed the tic-tac-toe game like below.

> [1]Our circuit will be responsible for lighting LEDs to mark where each player has played as well as tracking the past plays of the game in order to be able to declare a winner. The microcontroller will be used for decision making function. A feature that we might add to our Tic-Tac-Toe game would be a LED display to prompt players. Input will come from switches. Output will be a LED display. These LEDs will be able of displaying red to indicate which player has used the space. If the space has not been taken, the LED will pulse between 'on' and 'off'.

But, as we were testing this circuit for 3 weeks, we confronted to the problems, and also our circuit logic didn't match with software part either. First, logic behind the circuit was too much complicated, so that it was easier to make mistakes when we circuit them. In real, we couldn't figure out what the problems are and how to debug. Second, we realized that using I/O switches are more difficult because the inputs have nine bits and we have to use two ports of micro-controller as our input ports. Moreover, in the assembly code, we have to load two accumulators whenever we want to check the game situation. Therefore, we considered encoding the nine bit inputs into six bits but in the mean time, we found out that using keypad will only give the five bits outputs and we changed our plan for using the keypad.

The keypad will be worked as we expected and other team developed the assembly code based on the output of keypad. When we test our code with circuit, we faced another problem because the keypad sometimes gives two inputs and sometimes only one input. Later, we found that it is because we are not connecting the complement of "Data available" bit into its input, and not adding capacitors to the required inputs.

Wiring of eighteen inputs to CPLD chip also consumed some amount time. After getting all the parts working, we successfully demonstrated the two-player version of our game. We were about to finish the AI Vs. player version but due to time constraint, we did not succeed in debugging the code. The problem we thought was the memory location that we chose to store the data was incorrect. Writing a good AI control code with the limitation of HC12 micro-controller is a challenging one, and we found out that the micro-controller can handle only a certain amount of instructions. We tried to fit our code between memory locations of 0800 and 09FF our code barely fit in between those locations. Our code can make the micro-controller move for one input

[1] *Details of previous circuit and description of hardware is attached in Appendix A*

but unfortunately, the AI part stops execution its output after that. We debugged the AI part of our code up to the last minute but we were not that lucky.

Conclusion

We were very pleased to say that our project went smoothly except without being able to develop a successful AI Vs. human player version. We incorporated all the experiments that we learned throughout the semester: choosing suitable device, combining hardware and software together to achieve the project goal, and writing VHDL and assembly codes. Our TA also helped us a lot to finish this final project. While we were stuck with problems with circuit, Jill helped us for debugging and using the keypad and its decoder. Without Jill's help, we will not able to finish the two-player version of this game. Throughout the whole ECE 249 laboratory, Jill helped us in debugging circuits, explaining all the devices that we did not understand, and on the top of all this, he is one of the best TAs that we ever had.

Appendix A.

Details of previous circuit and description of hardware part with diagram

In order to describe the circuit, I will start at the left of the drawing and work my way across, explaining the components as I go.

At the left most portion of the circuit layout, there are 4 switches. These switches are the inputs for the two 4 to 16 decoders (74154) that will be used to select the space on the tic-tac-toe board that the player would like to play in. Since there are only 9 spaces that the player can play in, only outputs 1 through 9 from the decoders will be used. Once the player has selected the spot that they want to play in, they will flip the PLAY switch (right side of drawing) in order for their move to be recorded and displayed. (The players must select a space with a number of 1 through 9 and that space must not have already been played in, otherwise the circuit will cause the player's turn to be forfeited.)

The decoder's outputs then are routed to a 9-bit storage register consisting of nine JK flip-flops (74112). The inputs of the flip-flops are NOR gated (7402), so that the players cannot play over a previously played spot. All of the JK flip-flops are clocked by a single pulse clock that will only allow them to be updated after a player flips the PLAY switch.

The outputs of all 18 JK flip-flops are then run into three quadruple 2-line to 1-line MUXs (74157). Finally the MUX outputs will be wired to the tic-tac-toe board. The select input of each of the MUXs will be wired to a slow clock. As the clock oscillates, the MUXs will oscillate between outputting the contents of the player 1's register and player 2's register. When the RESET switch (right side of drawing) is flipped, player 1's register is cleared and player 2's register is preset. As the clock pulses, the output to the inputs to the tic-tac-toe board will oscillate between 1 and 0. This will cause the LEDs on the board to flash on and off. The flashing LED's are to show that a spot has not been played in. When player 1 plays, the JK flip-flop corresponding to the spot they played in will be updated with a 1 when the PLAY switch is flipped.

Once player 1 plays in a spot, the gating on player 2's register will block player 2 from being able to play in the same spot. The registers for player 1 and player 2 will now contain the same logic value for the same spot on the tic-tac-toe board. When the clock causes the MUXs to pulse back and forth between registers, the LED in that spot will stay on. When player 2 plays in a spot, the same type of logic will be performed, except the spot that player 2 plays in will be a logic 0 and the spot will be marked by the LED staying off as the slow clock pulses.

In the bottom right corner of the drawing, the one pulse clock is shown. The clock will be triggered by the PLAY switch. The one pulse clock will also clock a JK flip-flop that is hard wired to have its output change whenever it receives a clock pulse. This JK flip-flop is wired to the enables of each decoder as will as an LED. This LED will show which player's turn it is as the game is played.

* The logic diagram for this hardware is on the next page.