EE201 Final Project: Connect Four

Francisco Romero
Electrical Engineering, B.S.
Physics Minor

Akash Desai
Computer Engineering Computer Science, B.S.

December 5, 2013
EE 201 Fall 2013

Abstract

Connect Four is a two-player game in which players take turns dropping their respective pieces onto a 6-by-7, 42-slot game board. The object of the game is to be the first player to connect four pieces in a straight line horizontally, vertically, or diagonally. By choosing to design Connect Four, we aimed to apply our knowledge of various concepts learned in the EE 201 course, as described throughout the rest of the project report. The game design was divided into two parts: the state machine design that dictates how each player decides a column and plays their piece, and the VGA output design to create the game's visual display. While Connect Four is a simple, yet addicting game with few rules, designing and implementing the game proved to be a challenging, yet rewarding experience.

Introduction and Background

The design of Connect Four is based on the various labs, homework assignments, and test questions we worked on throughout the course of the semester. Understanding key concepts including single-stepping, data path design, and RAM functionality were crucial to the successful implementation of the game. These concepts can be found in labs such as the divider single-step, finding the greatest common divisor (GCD), and merging arrays, which we constantly referred back to during the game's design process. In addition, the Verilog Register-Transfer Level (RTL) homework assignments and test questions prepared us to effectively implement blocking and non-blocking assignments, mutually-exclusivity and all-inclusivity, along other important Verilog syntax when coding the game. As evidenced by the complexity of the design, creating Connect Four would not have been possible without the knowledge and hands-on experience we obtained in EE 201.

The Design

Our Connect Four project is a composition of six different modules, each of which provides an important functionality to the game's overall design. What follows is a description of each module and how it is contributes to the Connect Four gameplay.

Horizontal and Vertical Sync (HVSync) Generator Module

The HVSync Generator module used in this design is the unmodified version created by Da Cheng and is in charge of managing the row (CounterY) and column (CounterX) counters corresponding to the different pixels on the VGA display. This module is also in charge of determining whether the current pixel counter is in display area parameters (i.e. if CounterX is less than 640 and CounterY is less than 480). The outputs of HVSync Generator are connected to the Connect Four VGA module, which then uses the counters and display area parameters to color the different pixels on the VGA display.

Connect Four VGA Module

As suggested by the name, this module's task is to create the display of the game board and player pieces. This module is also based on the VGA template created by Da Cheng. The two most important inputs into the module are the two pixel counters, ConnectX and ConnectY from the HVSync Generator module. Using a series of "define" statements and conditional checks for ConnectX and ConnectY, we were able to draw the game board with different-colored circles for each player's piece. In order to draw the circles on the VGA display, we created rectangular parameters based on the position of ConnectX and ConnectY to fill in all of the pixels within the circle's boundaries. Accessing the game's memory determines the color of each playable slot, as well as the color of the current player's piece. Using the current row and column position, we pinpoint which circle each game piece is dropped into, and then used this current position as the memory read-address. The content of that memory address is used to determine which player's piece occupies the slot. Finally, the color of the board space is outputted back to the top module to display the corresponding board colors.

Connect Four Memory Module

To save the current state of the game board, we created a Random-Access Memory (RAM) that holds the values corresponding to each player's move. The RAM has 42, 2-bit wide addresses (one address for each playable slot) and has the capability to either read data or write data. The two modules that access the RAM are the Connect Four State Machine and the Connect Four VGA. The Connect Four State Machine uses both the read-and-write functionality of the RAM to check if an address is available before modifying it with the respective player's piece. If this module wishes to write to the RAM, it must first pass the RAM a memory address and enable its ability to write. Subsequently, the state machine can then pass the data it wishes to write before disabling the write enable. Both the Connect Four State Machine and Connect Four RAM access the RAM's stored content by passing an address and taking in the outputted value from the specified memory space. Additionally, the Connect Four Memory module checks for all possible win combinations in the horizontal, vertical, and diagonal directions and the tie situation.

Debouncer Module

Taken from the EE 201 divider lab, this module serves to interface with the buttons of the Field Programmable Gate Array (FPGA) to ensure a single pulse for each button push. Running at the full 100-MHz clock of the FPGA, this module takes each button as a separate input and returns a Single Clock Enable (SCEN) output to the top module. Since Connect Four is a strategy-based game, any wrong moves can prove costly to a player's chances of winning the game. Thus, it is important that each move be made as precisely as possible, which is the primary task of this module's SCEN output.

Connect Four Top Module

The purpose of the Connect Four Top module is to instantiate and connect the ports between all of the modules. The top module also interfaces with the FPGA's Seven-Segment Displays (SSD), LEDs, and buttons, which are all used in the Connect Four game. As a result, the inputs and outputs in the top module's port list not only connect to the other modules, but also to the NEXYS-board's .ucf file. The .ucf file serves to interface between the top module and the board's various human interface options (SSDs, LEDs, switches, VGA pixel colors, etc.).

Connect Four State Machine Module

The Connect Four State Machine module dictates how the game will transition between different states and describes what RTL stimulus takes place in each state. Each state has two components: the RTL stimulus and the Next State Logic (NSL). This module outputs the current state to the other modules so they can update their respective interfaces. As an example, the VGA uses the states to output the proper pixel colors while the top module checks the current state to update the SSDs or LEDs.

For Connect Four, the state machine is essentially two identical halves corresponding to player one and player two. From the initial state, the state machine will transition to player one's turn and will stay in this state until player one selects an available, non-full column to make a move. The next three states serve to a) determine the closest available game board slot to drop the piece into, b) animate the drop, and c) modify the respective slot address in the memory with player one's piece. Player one's check state then determines whether a winning combination was played, and will either transition to the player one win state, or to player two's turn. The same combination of states is then played out for player two. Players alternate turns until one plays a piece that gives them four-in-a-row either horizontally, vertically, or diagonally. A draw state can also be reached in the rare case that the players fill the board without a winning combination (see Figure 1). A detailed state machine diagram can be seen in Figure 2.

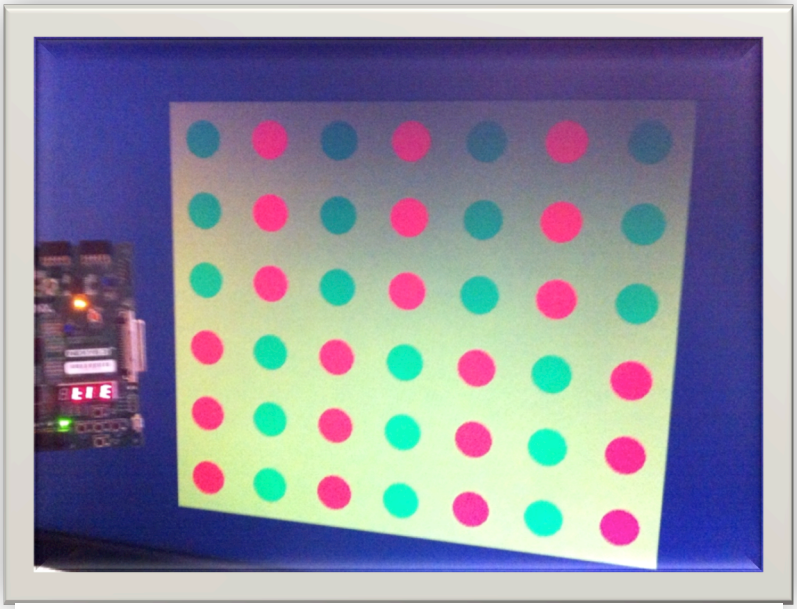

Figure 1

Test Methodology

In order to successfully implement our Connect Four project, we needed the two main parts of the design to function properly: the visual aspect and the gameplay aspect. As a result, our testing methodology was simple: continuously synthesize the code and view the outputted .bit file on the VGA display to determine functionality. While this testing methodology is somewhat tedious and impractical, it allowed us to verify proper implementation of both the aspects of the game design at once.

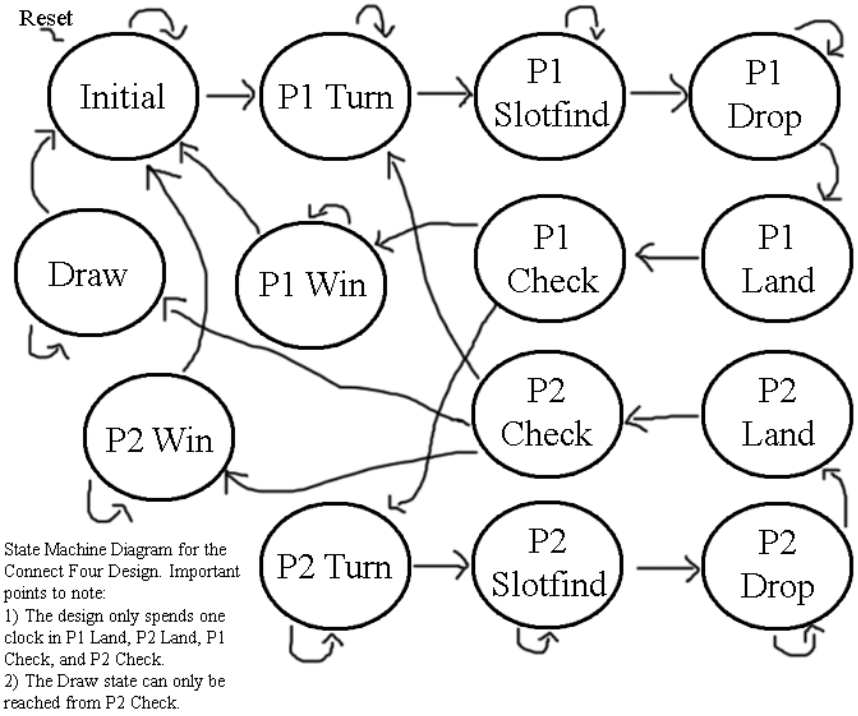

Figure 2

Verifying functionality of the visual portion of the design was done by carefully drawing and calculating each dimension of the board, along with its playable slots. The respective board and piece colors were coded by understanding in which state each pixel should take on a certain color. To test whether each piece interacted with the display correctly, we moved the game pieces around various times without selecting a slot, dropped pieces into random spaces, and carefully observed how the piece-drop animations were carried out. If a slot was mistakenly colored or if a game piece disappeared midway through the animation, we would return to the pertinent piece of code and reevaluate its functionality.

The gameplay aspect was an equally lengthy process, as this involved checking whether the board buttons functioned properly, whether the RAM was properly saving each player's move, and whether the memory was capable of determining a win or a tie. Some of the pivotal tests to ensure flawless functionality included pushing the buttons left and right aimlessly and then dropping the piece, ensuring each player was capable of winning the game, ensuring the tie condition was met, and ensuring the state transition conditions were working properly. As a result, we are confident that our game is capable of working with any input pattern presented.

Conclusion

Over the course of four weeks, we successfully designed and implemented Connect Four using our knowledge of the concepts learned in EE 201. Nevertheless, our version of Connect Four still leaves room for further enhancement and improvement that future EE 201 students can consider for their final projects. Some of these ideas include:

- An option to allow players to select their piece color at the beginning of the game.
- Find a way to draw a line through the winning combination.
- Design an option for single-player vs. computer.
- Create a test bench to test different gameplay scenarios. Not only will this provide confidence in the design, but it will also speed up the debugging process.
- Create a menu screen for gameplay options such as player color or difficulty for single-player (Challenging!).

We both agree EE 201 was a rewarding, well-organized course that taught us a variety of different concepts related to designing and understanding digital systems. The effective coordination between the lectures, labs and homework assignments helped to facilitate understanding of the course material, especially when the material was difficult to grasp.

Course Positives:

- Effective lecture, lab, and homework coordination.
- Webcasts were very helpful for introducing homework and lab material.
- Lab structure. Having students first implement a lab using the Xilinx schematic tools, and then designing the same lab using Verilog or writing a test bench for the given lab is a great way to understand the design structure.
- Lab reports and exam questions pertaining to the lab were helpful in reassuring student's knowledge of the pertinent course material.
- Final projects are a great way for students to demonstrate their understanding of the course material in a creative manner.

Suggestions for Course Improvements:

- Have final project reports due the day of the final exam.
- Space the final three weeks of material more evenly. This way, students are not overwhelmed with new, challenging material alongside the final project.

We would like to thank the EE 201 teaching team for their time and dedication in making the course a meaningful and rewarding experience!