Retro Connect Four on FPGA with VGA Display and SNES Controllers

University of Southern California – Viterbi School of Engineering

Arthur Krut – B.S. Computer Engineering and Computer Science, 21’

Martin Romo – B.S. Computer Engineering and Computer Science, 21’

Electrical Engineering 354 – Introduction to Digital Circuits

Spring 2019

Abstract

The purpose of this research is to build a functioning representation of the popular multiplayer children's game Connect Four on a Xilinx Nexys 3 FPGA. The user interface component of the game will be supplied through two Super Nintendo controllers that have been augmented to operate with the FPGA. The graphical component can be viewed on a monitor being supplied a VGA signal from the FPGA. The main logic for the game is written in a state machine that is cognizant of the multiplayer nature of the game and takes advantage of combinational logic to check if a player has won the game with a certain sequence. Possible applications of this research include building libraries that simplify the process of interfacing between an FGPA and VGA monitor, as well as building other games that have simple graphical user interfaces.

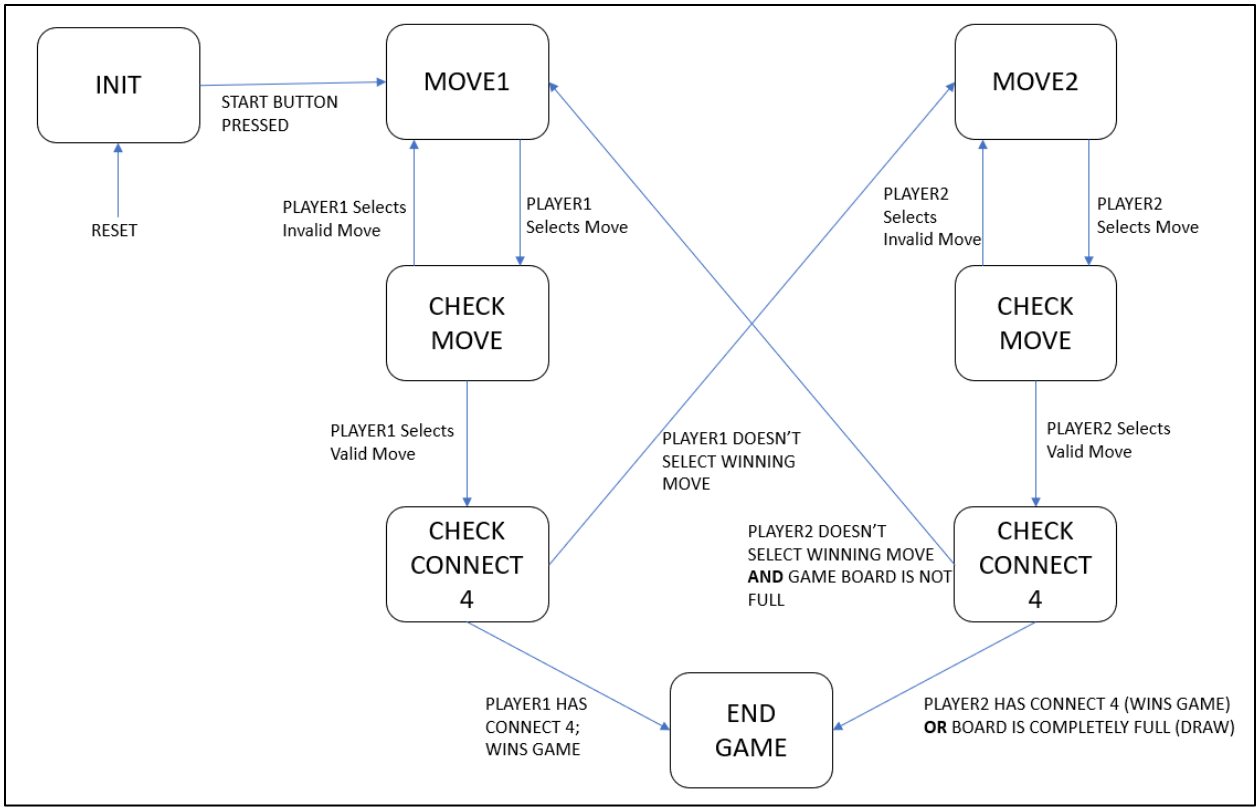
Introduction & Background

The prior research and experiments that helped us complete this project include the master VGA Verilog file and the Keypad Lab files. The master VGA file helped us create a cohesive user interface between the FPGA and the monitor that allowed us to display a grid to simulate gameplay. The Keypad Lab helped explain how to use the PMOD connectors on the Nexys 3 to create a persistent connection between the SNES controllers. Additionally, Fabio Andres's article, *SNES Controller Module - DE0-NANO-SOC,* was extremely helpful in understanding the pinouts of the controllers and detecting the button presses.

Design

Our goal with this project is to construct a Connect Four game on the FPGA that displays gameplay to a monitor via VGA and takes user input from Super Nintendo controllers. To accomplish this, it is necessary to rewire the SNES controllers so that the data they produce on button presses can be detected by the FPGA.

In order to make the source code modular and readable, multiple Verilog modules must be constructed for differing functions of the program. The ***connect_four*** module is responsible for maintaining the state machine logic for the game, in addition for tracking the board the game is played on. The board is represented by a register of size 42x1 that mimics a 6x7 grid that the game is played on. An additional register of size 42x1 is declared to track what color a certain board position is filled with if a piece occupies that space. Given a coordinate pair (i,j) that represents a board position, the formula 7*i +j produces the desired index of the pair in both the color and board registers. For the primary game logic, a state machine was implemented with six states: INIT, START, MOVE1, MOVE2, CHECK_MOVE, CHECK_C4, and END. Essentially, INIT and START serve as initialization code for beginning a new game. MOVE1 and MOVE2 process each of the players chosen moves. Next, CHECK_MOVE processes the validity of a move following either the MOVE1 or MOVE2 states. Last, CHECK_C4 makes extensive use of combinational logic to check whether there is a sequence of four positions that have the same colors following a player's move. It does this by using persistent for-loops to check for each horizontal, vertical, and diagonal combination that the grid permits. Once a player wins the game, the logic enters the END state. The logic for the state machine is shown below.



** Logic for state machine within **connect_four** module*

The **vga_display** module is responsible for displaying the graphical user interface to the monitor via a VGA signal from the FPGA. This served as the most daunting task throughout the development experience because of how sensitive the display was to extremely small changes. Additionally, the original intent for the game was to be played on a GUI that used circles rather than squares, however, this was extremely difficult to accomplish. On reset, the module will display an empty grid with the text "PRESS START" in order to begin the game. Following this, gameplay can commence until a winner is determined by the **connect_four** module. Once this occurs, depending on which player wins the game, either the text "RED WINS" or "BLACK WINS" will appear at the top of the display. In order to reset the game, the controller is programmed to allow many buttons to clear the display and reset the game logic.

The final module that the design makes use of is the **snes_controllers** module which is responsible for processing the user input from the SNES controllers and storing the result of any potential button presses in a register. Given the primitive nature of the Super Nintendo controllers, some investigation into the inner circuitry of the controllers shows that there are only five wires that must be interfaced with the FPGA: a line for +3.3 V, a line for GROUND, a CLOCK pin, a LATCH pin, and a DATA pin. Using two of the PMOD controllers on the Nexys 3 board, a small amount of rewiring of the controllers allows for easy integration with the FPGA. In terms of the logic for detecting button presses, the board sends a 12-microsecond signal on the LATCH pin to indicate to the controller that a series of 16 clock impulses will be sent from the board so that the state of each button can be sent on its own clock period along the DATA pin.

The state machine for the button detection logic makes use of 6 states: IDLE, STATE1, STATE2, STATE3, STATE4, and FINISH. THE IDLE state detects the beginning of any high logic coming from the LATCH pin and transitions to STATE1. STATE1 times the length that the LATCH pin is high and transitions to STATE2 if the time is greater than or equal to 12-microseconds. STATE2 prepares the state machine to begin processing button presses, which is completed in STATE3 and STATE4. The Seven Segment Displays (SSD) on the board are used to ensure that button presses are detected on either of the controllers. Finally, when 16 clock periods are completed and each button has been queried for a press, the controller moves to the FINISH state.

In terms of the integration of each of the modules, a singular instance of the **connect_four** and **vga_display** is sufficient, while two instances of the **snes_controllers** are needed to operate the multiplayer nature of the game. The **connect_four** module shares the board and color registers with the **vga_display**, while the **connect_four** is reliant on the output register of the **snes_controllers** to detect if the LEFT, RIGHT, SELECT, START, or A buttons are pressed.

** ***Button Legend*** **
LEFT – moves current square left
RIGHT – moves current square right
A – drops current square in column
SELECT – restarts game
START – begins game
****

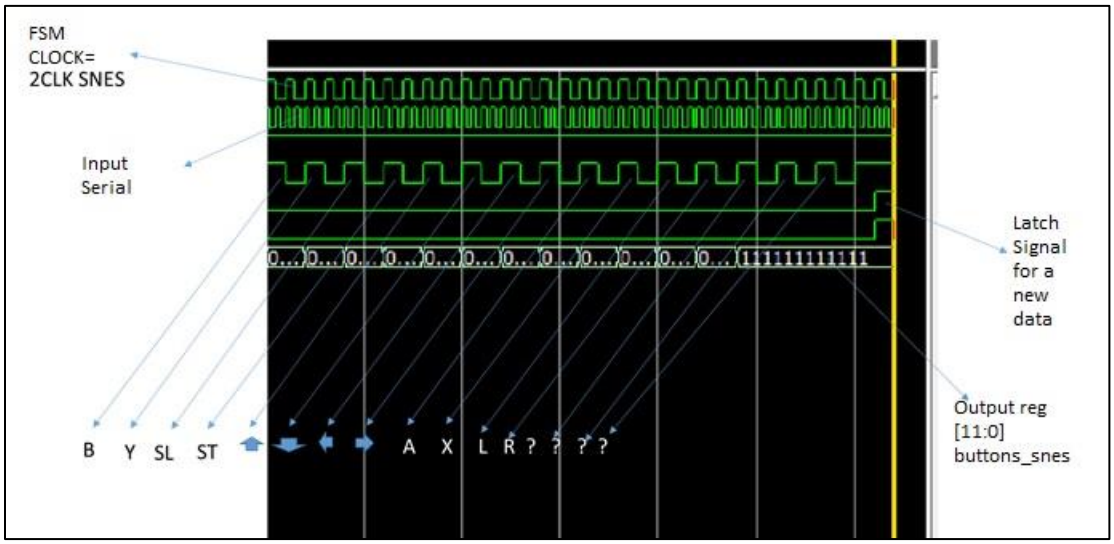
** ***Wire Legend*** **
RED – 5 V
BLACK – Ground
BLUE – Clock output signal
Yellow – Latch output signal
GREEN – Data input signal
****

Test Methodology

Tests were performed through the Questasim testbench in order to ensure that the controller state machine and the game logic state machines are operating correctly. Unfortunately, it was extremely difficult to perform any conclusive tests on VGA display without simply generating a program file and seeing the output on the screen, although later it was discovered that libraries are available that will simulate testbench files specific for testing VGA displays.

For the game logic testbench, button presses were tested such that it was confirmed that moving a square left or right and dropping a square all work properly in the context of the state machine. This testbench was deployed before integrating the multiplayer controllers and was created with the assumption that the on-board buttons control the movement of squares.

Additionally, a testbench was constructed to test the state machine that governs the SNES controllers. This testbench simply moves through the state machine logic by inducing a high signal on the LATCH pin and ensuring that the controller state machine moves through every subsequent state in the correct amount of clocks. The graphic below (courtesy of FGPA Lover) is representative of the result of this testbench.



** Courtesy of Fabio Andres and FGPA Lover*

*** The output register of this testbench considers whether each of the buttons on the controller have been pressed. Though there are only 12 buttons on the board, 16 clocks are necessary before the next set of buttons can be detected because the controller sends 4 negligible signals that can be disregarded.*

Conclusion & Future Work

Though the results of this research are promising, further enhancements can be made to create a better gameplay experience. First and foremost, the grid that is displayed to the user should have circles rather squares in order to make the gameplay more authentic, though this is a tall task considering the tedious nature of VGA programing. In addition, to allow multiple users to play on different screens, it is possible to integrate I2C or RS232 serial communication protocols that will allow multiple FPGA's to communicate with one another.

Overall, this research posed a series of issues ranging from the endless number of errors produced by the VGA signal to logic errors in the controller state machine. Despite the challenging nature of the project, it was a rewarding experience that helped create an understanding of how to interface multiple dynamic systems on the FPGA.

We would like to thank our teaching assistant Jinglei Chang for his encouragement and support with this project, and each lab all semester long. Lastly, we would like to thank Professor Gandhi Puvvada for immersing us in the world of digital design throughout the semester.