2025

Final Project Report

EECS 443 – DIGITAL LOGIC DESIGN (UNIVERSITY OF KANSAS)

RILEY MEYERKORTH, NICHOLAS HOLMES

Project Name

Gameboy: FPGA

Team Members

- Riley Meyerkorth
- Nicholas Holmes

Terminology

- RPC: Rock, Paper, Scissors
- HL: Higher or Lower

Work Distribution

Riley Meyerkorth

Gameplay logic, randomness logic, game switching logic

Nicholas Holmes

7-segment display logic, LED logic

Demo

We demonstrated the project on May 7th, 2025 at 1:15PM. The TA we demoed for was Md Alvir I Nobel.

Project Description

Create a system to play games on the Nexys FPGA board with only the components on-board. This will be done by utilizing the LEDs and 7-segment display for game selection/data display, and switches/buttons for player input.

There will be (at minimum) 1 game that will be programmed on the board:

- **Rock Paper Scissors**: simple rock paper scissors. The player will select rock, paper, or scissors using one of the inputs. The board will randomly select one of the options, and then a winner will be decided.

If there is time, other games can be programmed as well and switched to using switches/buttons. These include:

- **Higher or Lower**: the board will randomly generate a number between a specific range of values. The user will then input numbers in binary through the switches and press a button to submit their guess. The board will then either display “HIGHER” or “LOWER”.

Hardware Architecture

We are utilizing the Nexys A7 FPGA board to implement the project. The main components we are currently utilizing on the board are as follows:

- 15 switches
- 15 LEDs
- 5 buttons (up, down, left, right, center)
- 7-segment display

Block Diagram

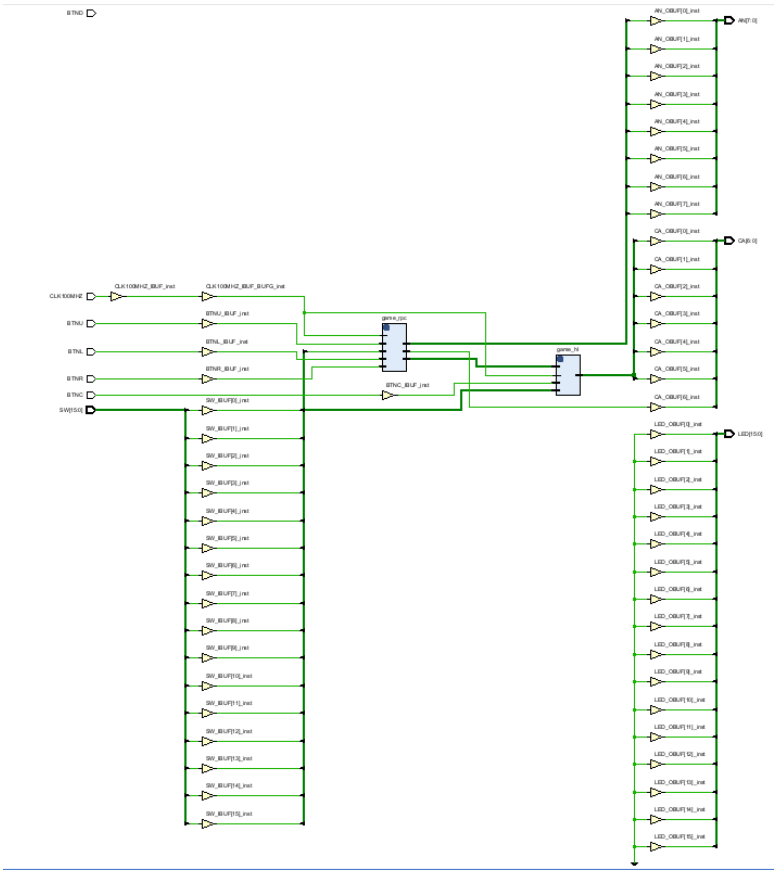

Figure 1: The Vivado schematic diagram of our top-level file (somewhat difficult to read)

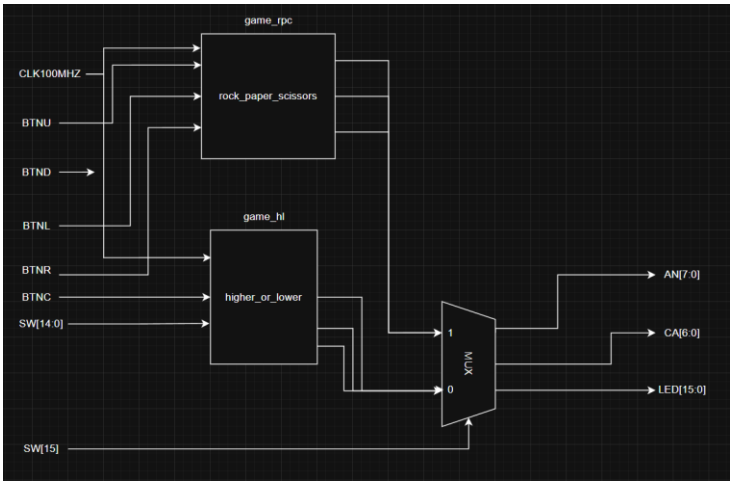

Figure 2: Our custom and compact block diagram (much more readable)

Component(s) Behavior

rock_paper_scissors

The overall behavior of the rock_paper_scissors component takes in the inputs of 3 buttons (for rock, paper, and scissors), an input clock signal (for timing the display), and outputs to the 7-segment display.

Once the user has made their selection, the circuit randomly selects its choice using an LFSR. It then changes the output of the 7-segment display to display its selection, the result of the round, and the updated scores. After this is complete, the game repeats.

In a step-by-step format, this is the overall gameplay loop:

1. The user selects their choice using the buttons (rock, paper, or scissors)
2. The user's choice is displayed on the 7-segment display ('r', 'P', or 'S')
3. The computer selects it's choice (rock, paper, or scissors)
4. The computer's choice is displayed on the 7-segment display ('r', 'P', or 'S')
5. The choices are compared and the winner of the round is determined
6. The result for the user is displayed on the 7-segment display ('w', 'L', 't')
7. The appropriate statistics are updated on the 7-segment display
8. Return to Step 1

The controls:

- Rock: BTNL
- Paper: BTNU
- Scissors: BTNR

Overall, it is a complex circuit that can accurately simulate a Rock Paper Scissors game.

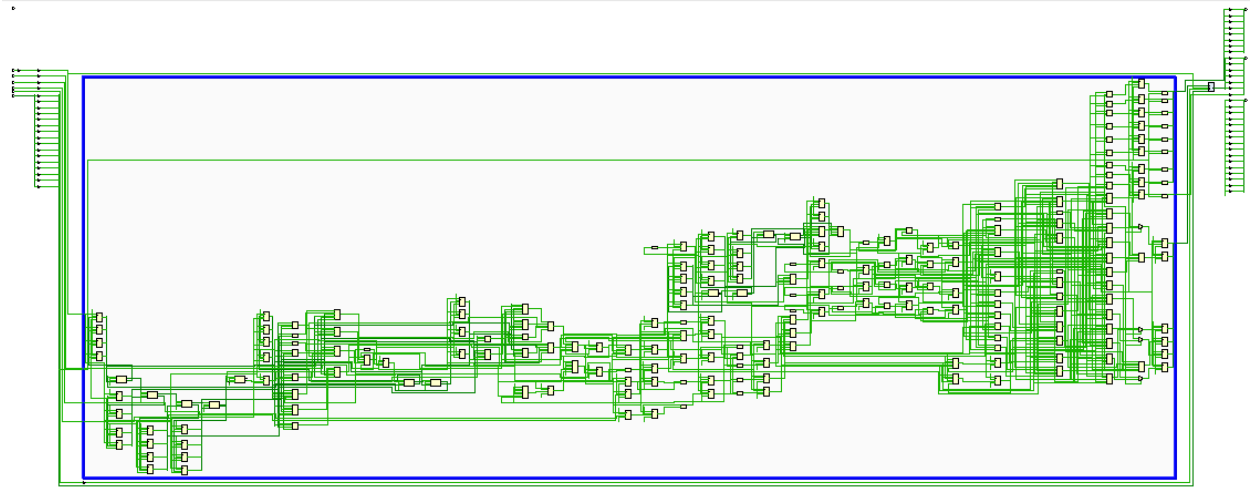

Figure 3: The generated schematic from Vivado of the rock_paper_scissors component

higher_or_lower

The overall *correct* behavior of the higher_or_lower component takes in the inputs of 15 switches (for the number selection/guess), an input clock signal (for timing the display), and a 7-segment display output.

When the game is selected, the computer randomly selects a number using a LFSR. This is the number that the user must guess. The user can input guesses using the 15 switches, which represent a 15-bit number. Once the guess has been inputted, the user presses the “guess” button. The computer compares the guess to the selection, and outputs to the 7-segment display whether the selection is higher or lower than the guess.

This game of slowly getting closer to the selection continues until the selection and guess match. At this point, the computer displays a win message and re-selects a random number. The game loop then repeats.

In a step-by-step format, this is the overall gameplay loop:

1. The computer selects a random number to be a selection
2. The user enters a sub-loop for inputting guesses:
    a. The user inputs their guess in binary using the switches
    b. The user “submits” their guess by pressing the “guess” button
    c. The computer compares the selection to the guess
        i. If the selection is higher, the 7-segment display shows the “Higher” message (‘H’) and the sub-loop repeats
        ii. If the selection is lower, the 7-segment display shows the “Lower” message (‘L’) and the sub-loop repeats

iii. Otherwise, the selection matches the guess and the sub-loop breaks out
3. The 7-segment display shows a win message (‘w’)
4. Return to Step 1

The controls:

- Submit guess button: BTNC
- Guess input: SW[14:0]

In its current state, there is a bug where the computer’s selection can randomly change. We believe this is due to a miswiring of the LSFR to update on every time a user inputs a guess instead of only when the win condition is met.

Regardless, since the Higher or Lower game was marked as an optional game, and the main point was to create the system/groundwork for modular games, we believe it is perfectly fine to submit this as-is; especially considering that the main game logic is there.

In a way, the game “works”, it’s just a harder version of it.

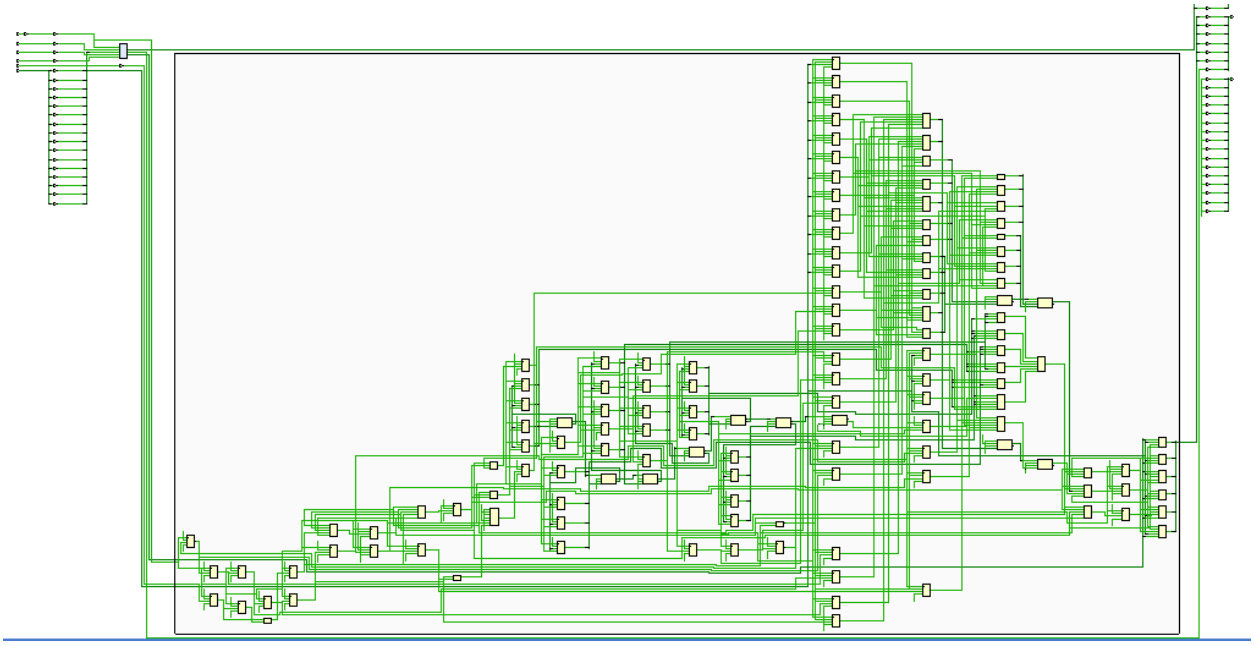

Figure 4: The generated schematic from Vivado of the higher_or_lower component

Simulation/Implementation Results

Simulation

Due to the random nature of our project, we found it much easier to test our program on-board rather than simulating. Due to our personal technology, we lost very little time doing this and overall *saved* time during testing.

However, this does not prevent us from creating *some* estimations for very specific game outputs (which are somewhat obvious if you know the games with prior knowledge).

SW[15]	Game
0	Rock, Paper, Scissors
1	Higher or Lower

Table 1: The overall expected game-selection inputs and outputs

User Selection	Computer Selection	Game Result (User)
Rock	Rock	Tie
Rock	Paper	Loss
Rock	Scissors	Win
Paper	Rock	Win
Paper	Paper	Tie
Paper	Scissors	Loss
Scissors	Rock	Loss
Scissors	Paper	Win
Scissors	Scissors	Tie

Table 2: The expected results from RPC for the user

User Guess	Computer Selection	Game Result
583	6002	Higher
8525	46	Lower
1000	1000	Win

Table 3: The expected results from random guesses/selections in the HL game

Implementation

Game Selection

In a blank state, the board automatically runs the RPC game. The game can be selected using the left-most switch (SW[15]).

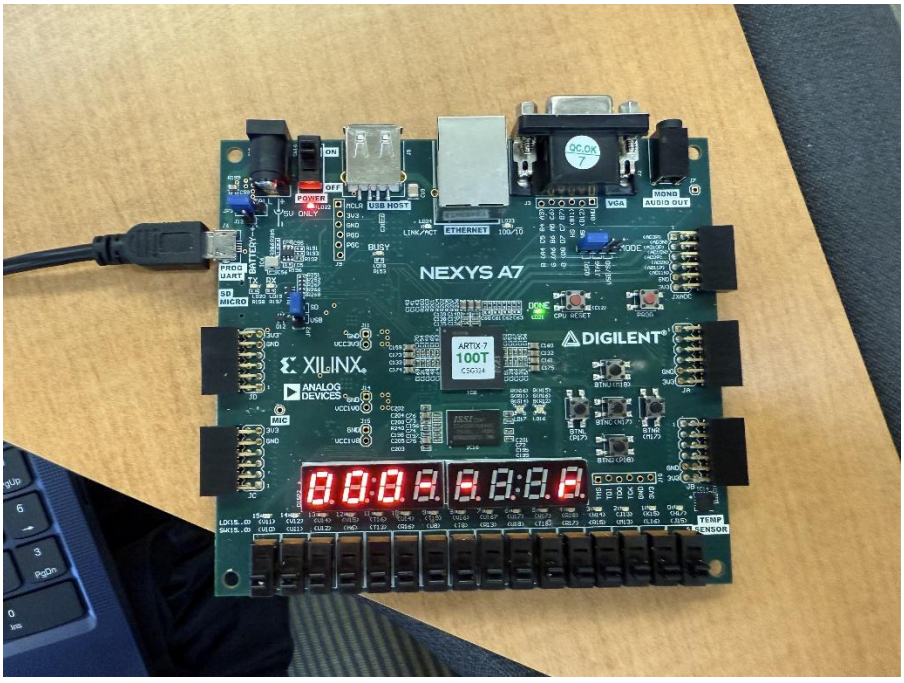

Figure 5: The board set to the RPC game

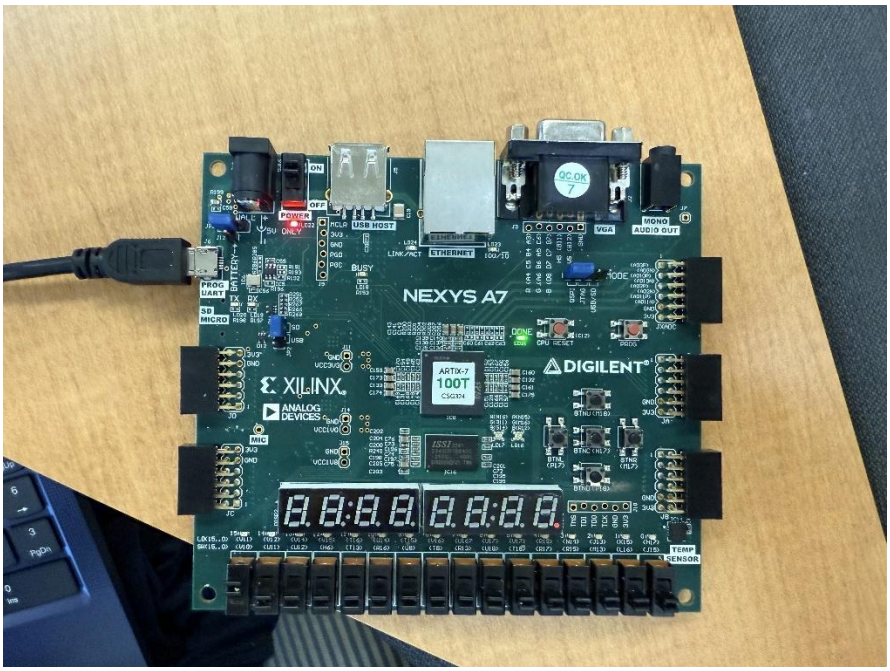

Figure 6: The board set to the HL game

Rock Paper Scissors

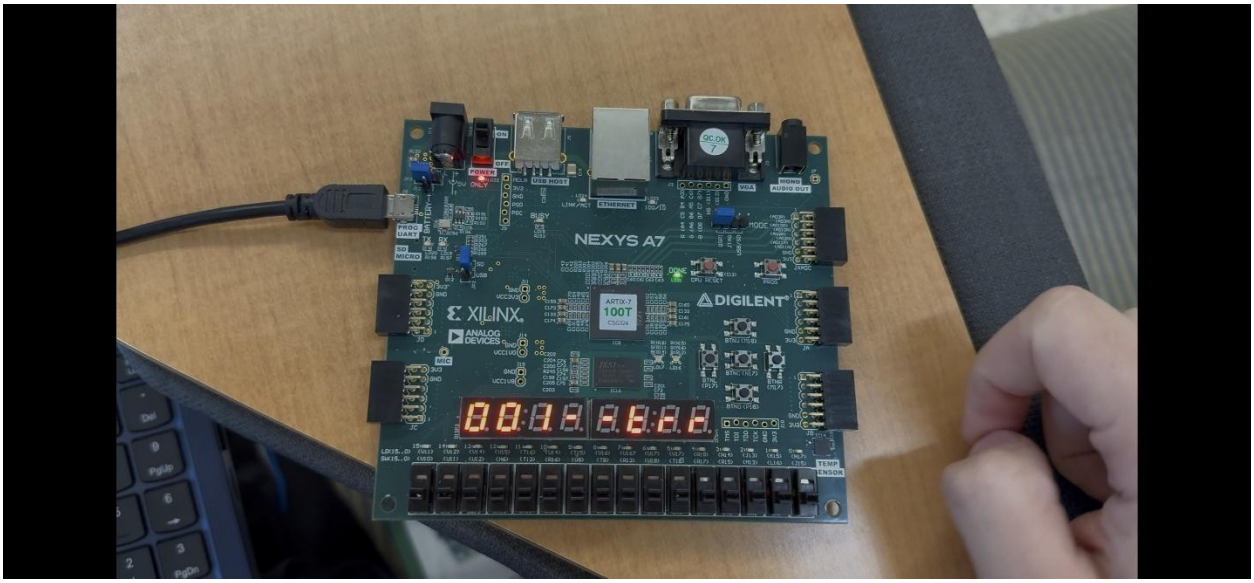

Figure 7: The user chose rock, the computer chose rock, so the game is a tie

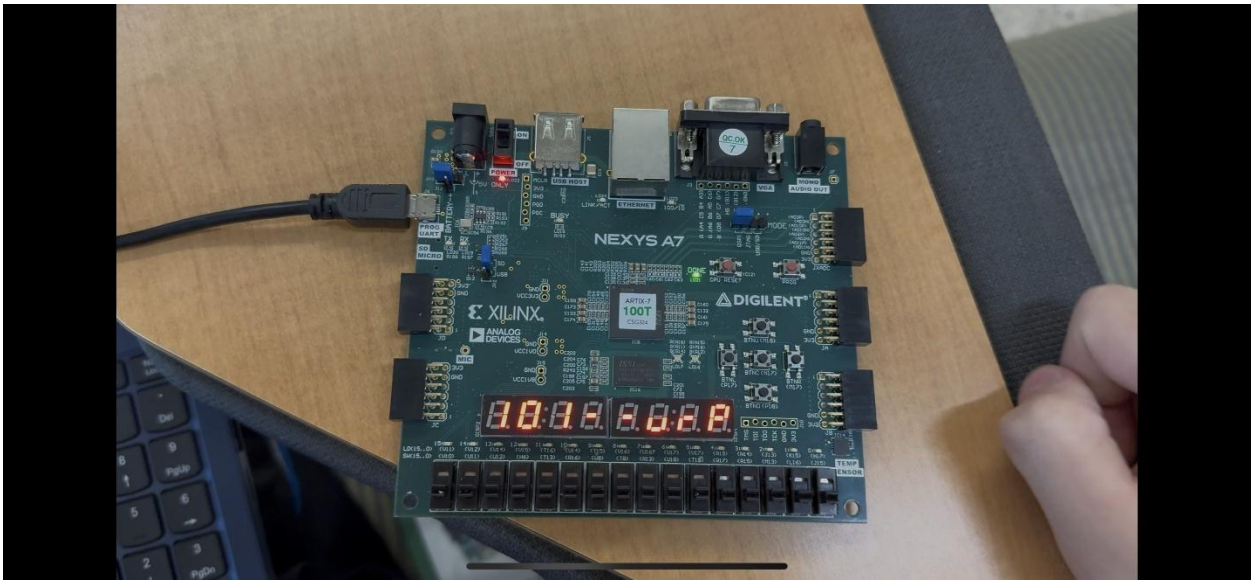

Figure 8: The user chose paper, the computer chose rock, so the user wins

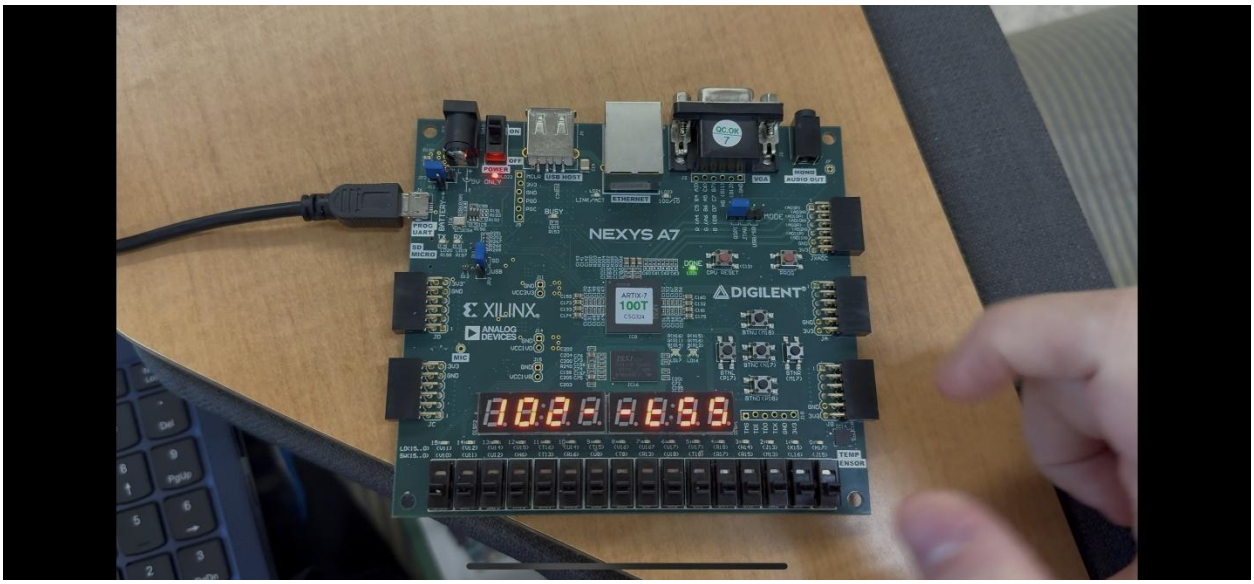

Figure 9: The user chose scissors, the computer chose scissors, so the game is a tie

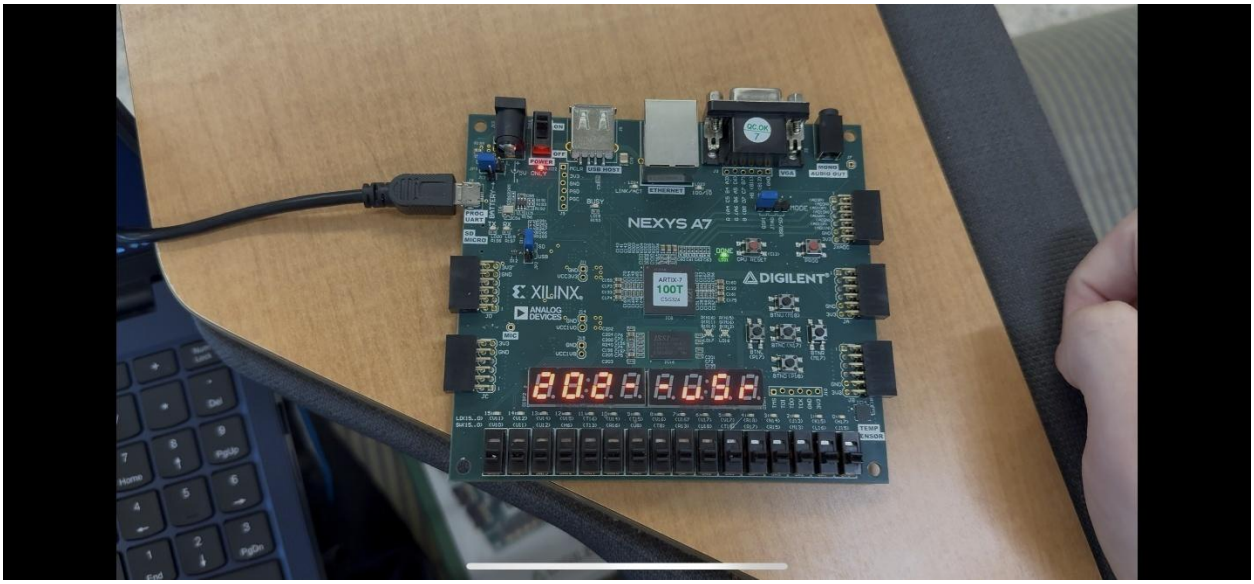

Figure 10: The user chose rock, the computer chose scissors, so the user wins

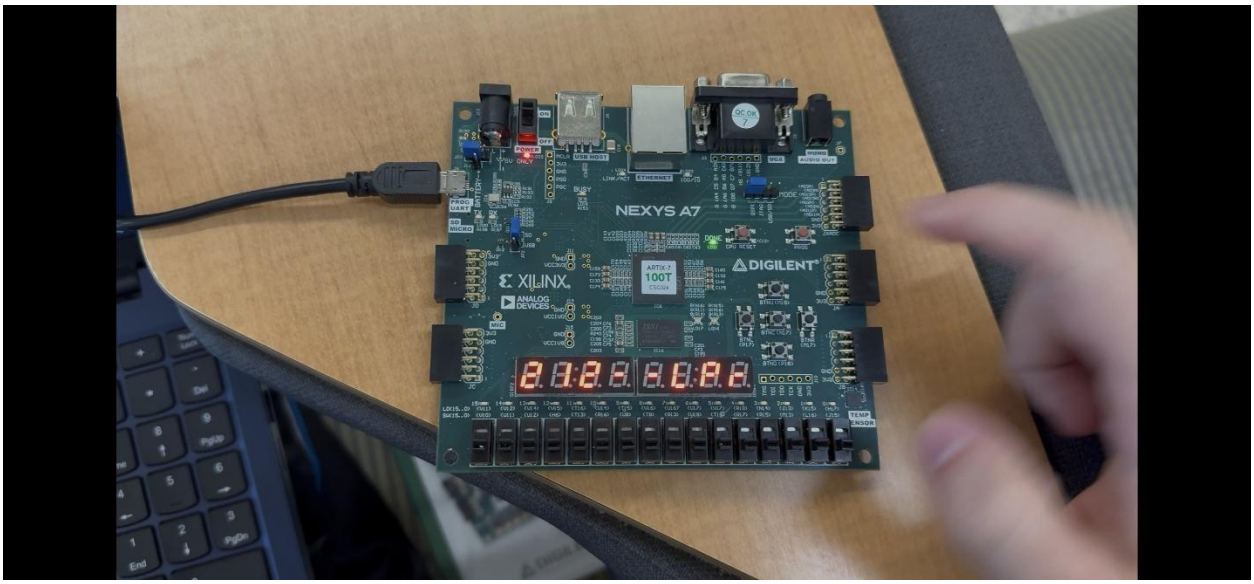

Figure 11: The user chose rock, the computer chose paper, so the user loses

Higher or Lower

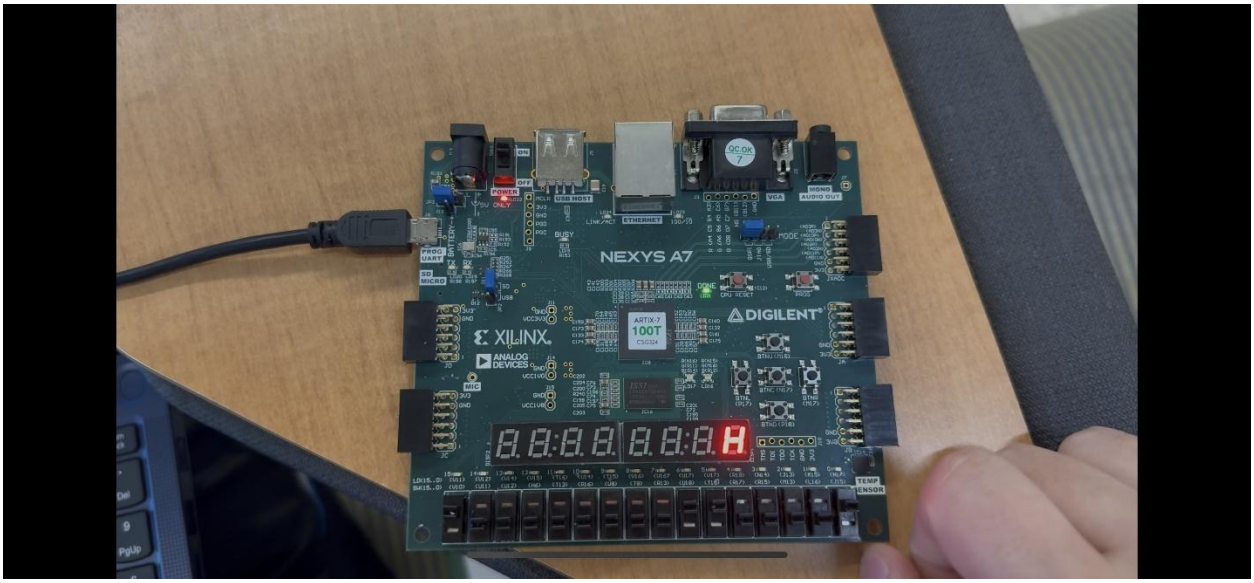

Figure 12: The user guesses 0000000011100001, and the computer responds that they must guess a higher number

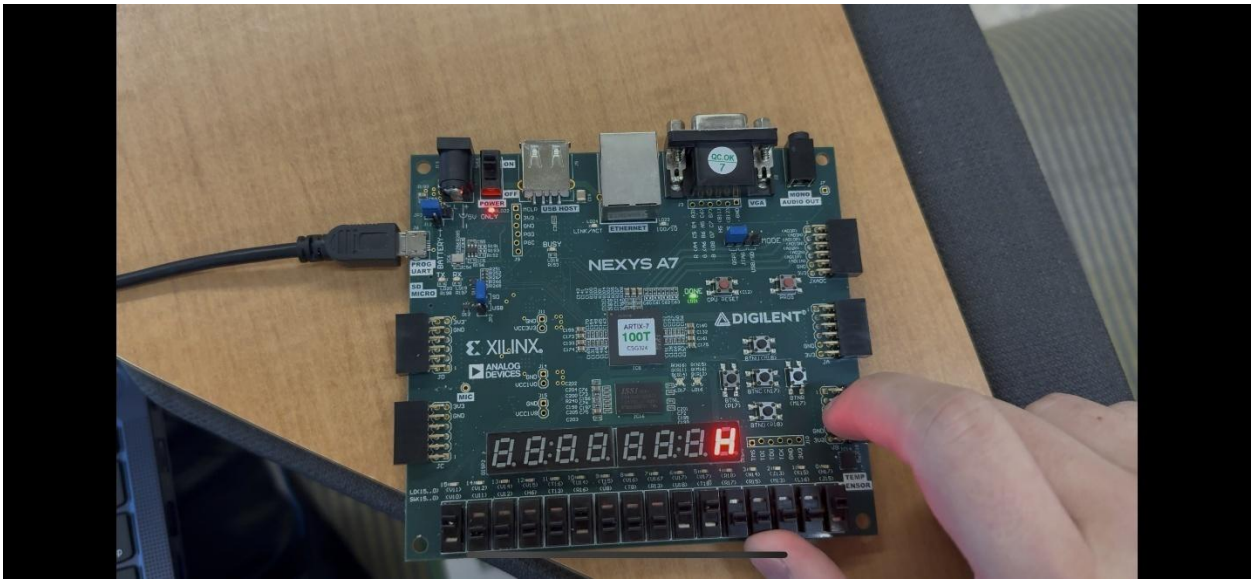

Figure 13: The user guesses 0000100011100001, and the computer responds that they must guess an even higher number

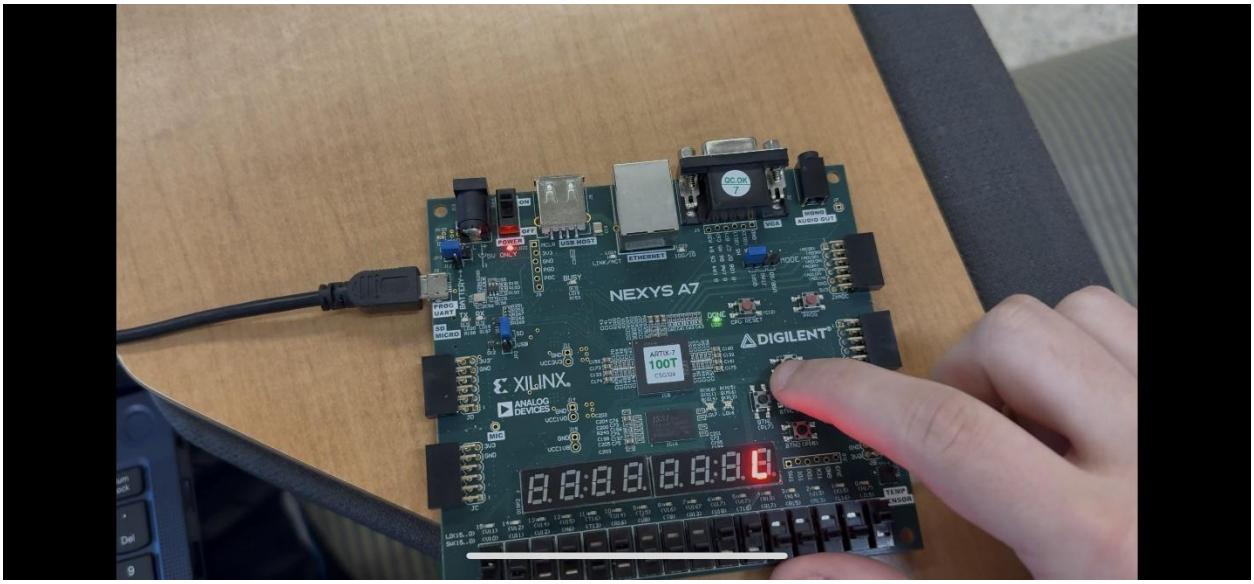

Figure 14: The user guesses 100010001100001, and the computer responds that they must guess a lower number

Discussion

Overall, we believe that our project was a success. We were able to create a system that allows for modular games to be programmed onto the Nexys board.

For RPC, we believe that the implementation on-board accurately reflects what we planned on accomplishing for the project. We have a very nice display, easy-to-use input methods, and accurate game logic. Additionally, the board displays the expected behavior that we calculated.

For HL, we believe that we were able to accomplish much more than we planned on. While there are a few bugs in the timing of the random number generation, the overall gameplay logic is working and functional. It aligns with our expectations for the behavior of the HL game.