Rock Paper Scissors

Rachel Pilarowski, Justin Thomson, Marwan Oro
Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI
e-mails: rpilarowski@oakland.edu, jnthomson@oakland.edu, Moro@gmail.com

***Abstract*—Rock Paper Scissors is a popular game played by people of all ages. The rules are relatively simple to follow: rock beats scissors, scissors beat paper, and paper beats rock. After 3 seconds, each player picks one of the options and whoever wins scores a point. This concept has been digitalized using an FPGA in order to implement the fast-paced game.**

I. Introduction

Rock Paper Scissors is a simple game with simple mechanics that most people are familiar with and understand. Because of this, the game makes a good candidate for the project.

The main control system of the project will be a Finite State Machine, which was learned about in class. This circuit will control the current state of the game. The idea is that whichever player gets to 3 points first wins the game. One master clock will also be used in order to synchronize each circuit and give the players a specified amount of time between each turn.

Though this game does not have any practical applications or significance, the hope is that it digitalizes a common game that many people will be able to play.

II. Methodology

Below, Figure 1 shows the block diagram for the complete game. Each player has 3 switches with each one being either rock, paper or scissors. This is how the user will select which one they want to choose. The output along with a countdown will be displayed on the different seven-segment displays. After the countdown has reached zero, the score will change depending on who won the round. The game is considered over once a player has reached the score 3.

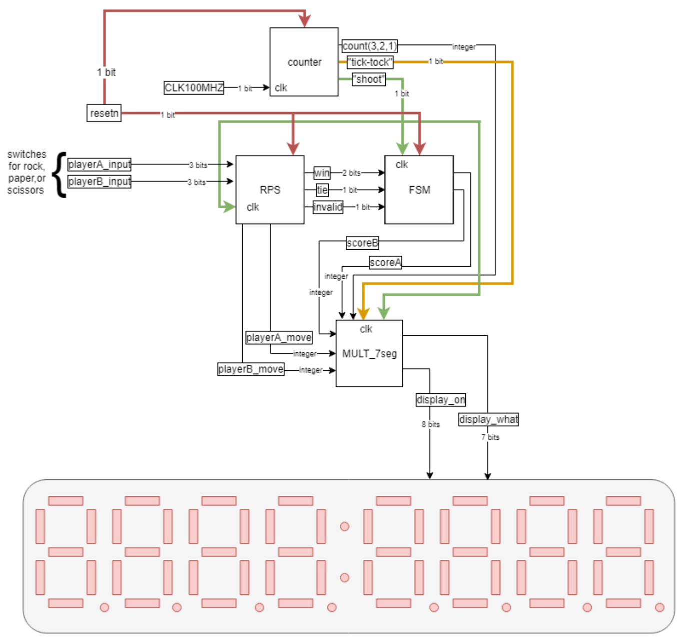

Figure 1: The complete block diagram

A. Comparator

The comparator, which is called RPS in the above diagram, takes the raw input from the switches and from that it determines which player wins. If either an invalid input was entered, such as two or more switches being high, or both players chose the same option, then the component will catch it and make either “tie” or “invalid” high. If this happens, the user will see a “-“ as their input and the score will not be changed. This circuit was done easily in VHDL by assigning Rock, Paper, and Scissors to an integer value. Then based on the user’s input, the corresponding integer will be compared to the other player’s input. The outputs of this circuit go directly to the FSM. The FSM controls the state of the game. Therefore, it takes which player won and outputs the score onto hex-to-seven-segment displays. Because of this, the players will always have access to the score. The inputs of the switches are also fed to the multiplexer which controls the display in order to display what option each player chose.

B. State Machine

The second element to this digital circuit, which controls a lot of the functionality of the game, is the Finite State Machine. This state machine takes in the output from the comparator and a signal from the counter in order to change states. Below in Figure 2 shows the complete state diagram.

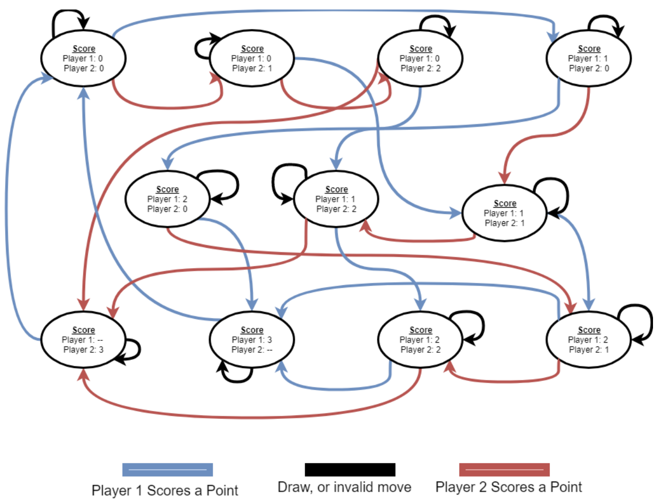

Figure 2: The complete state diagram for the FSM

As can be seen from the diagram, if player one or player two scores a point, the state machine moves to the corresponding state. There are 11 states total. One of those state represent if player one won and another state to represent if player 2 won. If either draw or invalid is high, the state machine will stay on its current state. If player 1 or player 2 has a score of 3, that means the game is over. The only way for the game to go back to the first state is if the reset button is pushed. The output of the state machine is just the score for both players. The outputs are fed into the multiplexing display to be displayed to the players throughout the game.

C. Counter

The counter module in this project is used to determine several things such as when to compare two players' moves, when to check for conditions which determine state changes, as well as the integer value of the timer. The top file receives input from the 100MHz clock in the Nexys 4 and uses this clock input to count to values which correspond to the length of time we wish to count for. This counter helped to ensure that the relevant processes were triggered when needed, and remained idle when not in use. Using a counter also allowed us to have the ability to multiplex several seven segment displays at once, which overall made the game more user friendly.

D. Multiplexing Display

The seven-segment display multiplexor is a module which was used to determine when individual seven segments displays should be enabled, and what should be displayed on them when they are enabled. This component allowed us to give the user more feedback regarding the state of the game and made the game much easier for the player to recognize what was going on. This component primarily processes what to display when the timer is counting, and then what to display after the players' moves are compared and a winner is declared.

III. Experimental Setup

In order to test the functionality of the game, the behavior simulation was used as well as the Nexys board. A test bench was created in order to test the functionality of the state machine. Different test cases were used to make sure the state machine can handle all different cases. A tie was given, an invalid input was given as well as normal plays to make sure that it increases the score correctly. As for the counting module and the display, different bitstreams were generated with changes to the code until it worked. This was mostly trial and error based. Once the final code was programmed onto the board, it was given to players to test to make sure no problems or issues came up.

IV. Results

In the end, a working game of Rock, Paper, Scissors was created. The board would display a 3-2-1 countdown at the beginning of each round and there would be a 2 second buffer time to show the changed score. All outputs were shown correctly on the seven-segment displays as well. Rock was shown with "r', "S" for scissors, "P" for paper and "-" for invalid. Once the game was over, the clock continued to count down and inputs were still displayed to the user, however, the score was not changed as a player had already won.

A majority of the architecture for this game was based on topics that were discussed in class. For example, the finite state machine was stressed in class and the methodology for creating one was followed. However, topics that were not discussed in class were also used. For example, the RPS comparator could have been done with Boolean expressions and truth tables, however, it was found to be much easier to work with integers instead. Also, it was determined that using more than one clock in the system is a poor design because of clock shift and the clock shift propagation.

All results were expected. Even as the game was developed and resulted in something that was not expected, a reason was always discovered. For example, before the multiplexing display worked correctly, the outputs would be shown to the board but not bright and they would almost blink very quickly. We found that this was due to the speed of the clock and the amount of time that the multiplexor would display each output. Therefore, these times were adjusted in order to make the display work correctly.

Conclusions

This project proved that Finite State Machines are powerful and almost essential in any digital circuit. It also proved that even the simplest concepts, like a game that you can play with your hands, needs to be carefully planned. If any improvement were to be made to this game, one would be to stop displaying the count down and the user inputs once the game is over. Then, once the reset button is clicked, continue displaying everything. Another design improvement could be to use something other than switches such as buttons. This would make it easier for the players to be more discrete with their choices.