Lab 4

CS M152A

Aaron Cheng

Aditya Padmakumar

Introduction

The goal of this lab was to use our collective experience from the past labs to design and create our own FPGA project. Instead of having a strict project outlined for us, this lab allowed us the opportunity to go through the process of designing a task to be performed on the FPGA.

We chose to implement the casino game of Baccarat. Because the rules and mechanics of the game are quite extensive, we have not included them in this report. However, they can be found in our lab proposal.

Our implementation of Baccarat on the FPGA utilizes the board's switches, LEDs, push buttons, 7-segment display, as well as an external keypad connected using a 12-pin Pmod connector. In essence, our game allows the user(s) to set the amount of players, view the amount of tokens each player has, view/change the amount of tokens each player is betting each round, bet on either the "player" or the "banker", view the cards dealt to both the "player" and the "banker", view the result of each round, and to view the ultimate winner of each game.

Here is a rundown of the complete flow of a game:

1. First, the number of players is set by pressing the Rotate button to change the number of players and Enter button to submit and enter the "bet" mode.
2. a. In the "bet" mode, players use the switches to indicate whether they are betting on the "player" (low) or the "banker" (high). The players initially start in this mode and can return by pressing the Bet button.
   b. By pressing the View button, a player can view the amount of tokens they have by setting the single switch corresponding to that player to high and all other switches low.
   c. By pressing the Amount button, a player can view/change the amount of tokens they are betting each round. The single switch corresponding to that player must be set high to view the amount. To change it, the player must input the amount using the keypad and then press the Rotate button.
3. When ready to start the round, the players can press the Enter button when in the "bet" mode. They will now be able to view the cards that are dealt to the "player" and "banker" by pressing the Rotate button.
4. After viewing the cards, pressing Enter will show the results of the round and the players' tokens will be increased or decreased according to the result.
5. Hitting Enter will return the user(s) to #2. After 10 rounds have been completed, the player with the most amount of tokens will be shown, declaring the winner.

Design Description

Similar to how we began the implementation in Lab 3, we identified the states of operation that the game would be in and what actions would cause transitions between those states. The states that we identified were Initialize, Choose, View Tokens, Bet Amount, View Cards, Final Scores, and Winner. The Initialize state allows the user to set the amount of players, between 1 and 8. The Choose state allows players to choose to bet on the "player" or the "banker". The View Tokens state allows the players to view the amount of tokens they

currently have. The Bet Amount state allows the players to view and change the amount of tokens they are betting each round. The View Cards state allows the players to see the cards that have been dealt to the “player” and “banker”. The Final Scores state allows the players to view the results of each round. Finally, the Winner state displays which player won the game.

We decided to create a module for each of these states, as we felt that each of these states were already modular, and added a few more modules to make the FPGA functional. Figure 1 shows the modules.

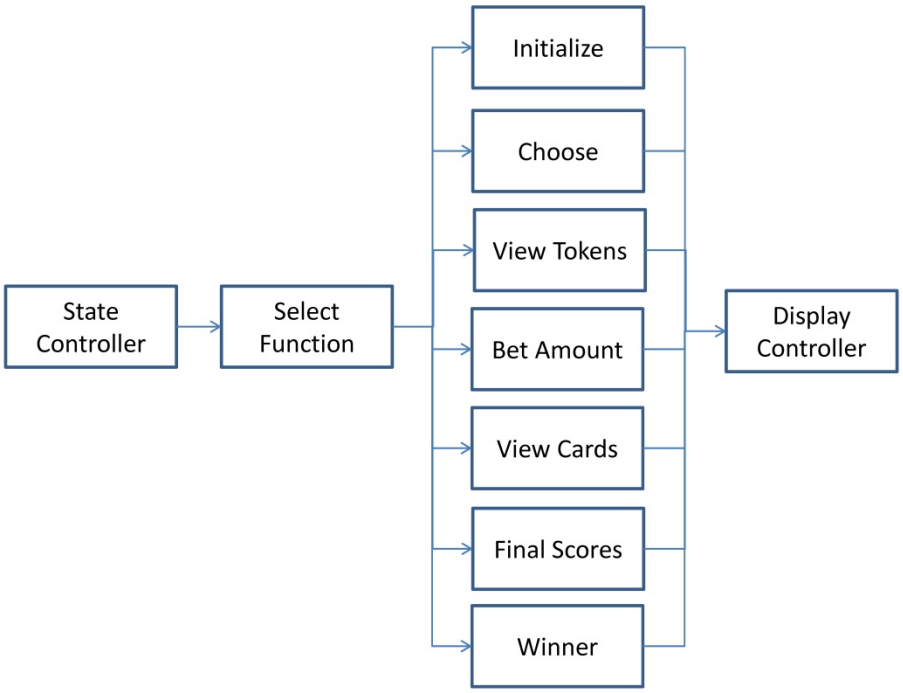

Figure 1: High Level Representation of Modules

The state controller module’s purpose is to determine the state of the game based on the current inputs and the current state. It takes as input the board’s clock, the Enter, Bet, View, and Amount buttons, and a fast clock signal, which is created by another clock module, as was in Lab 3. It then outputs a four-bit signal that signifies which of the seven states that the game will be in during the next clock cycle.

The select function module takes as input the four-bit next state signal from the state controller as well as the board’s clock. Depending on which state the game will be in, the select function module will output a different 7-bit signal to signify which of the seven state modules will be enabled and have access to the display controller.

The display controller takes as input the fast clock, the 7-bit which module signal from the selection function module, and numerous 20-bit signals from seven state modules that signify what to show on the 7-segment display. The 7-bit which module signal determines which module to take 20-bit signal to use, and ignores the rest. During each cycle of the very fast clock, the display controller outputs seg and an, together which determine which of the four seven segment displays and which of its segments to the light up.

The Initialize module takes as input the board's clock, the 1st bit of the which module signal, the Rotate button, and the fast clock. When the Rotate button is pressed, the Initialize module changes the number of players being displayed and saves the number of players, which is part of its output. It also outputs a 20-bit signal to the display controller.

The Choose module takes as input the board's clock, the 2nd bit of the which module signal, the switches, and the number of players. When the switches are switched high, this tells the module that the corresponding players are betting on the "banker", which is saved when the game leaves this module. It outputs the 20-bit signal to the display controller as well as the board's LEDs, which are also used to store who each player bet on.

The View Tokens module takes as input the 3rd bit of the which module signal, the switches, the number of players, and the eight 10-bit signals that indicate how much tokens each player has. When a single switch is high, and the corresponding player falls within the number of players, this module shows the number of tokens that player has. This is done through its 20-bit output signal to the display controller.

The Bet Amount module takes as input the board's clock, the 4th bit of the which module signal, the fast clock, the Rotate button, the number of players, eight 10-bit token signals, the switches, the new_game signal, and the input portion of the Pmod connector. When a single switch is high, and the corresponding player falls within the number of players, this module shows the number of tokens that player is currently betting. A smaller decoder module within the Bet Amount module allows the Bet Amount module to detect key presses of the keypad. This allows the player to change the token bet amount. When the Rotate button is pressed, the current value of the display is stored if it is less than or equal to the amount of tokens that player has. The Bet Amount module outputs the output portion of the Pmod connector, the eight 10-bit signals that indicate how much each player is betting, and the 20-bit signal to the display controller.

The View Cards module takes as input the board's clock, the fast clock, the 5th bit of which module signal, and the Rotate button. The View Cards module generates the random cards that the "player" and "banker" are dealt using a mod 13 counter. When the Rotate button is pressed, this module changes which card is shown. This module outputs the 20-bit signal to the display controller, and six 4-bit signals that indicated the "player's" three cards and the "banker's" three cards.

The Final Scores module takes as input the board's clock, the 6th bit of the which module signal, the new_game signal, the six 4-bit card signals, the eight 10-bit bet signals, and the LEDs. When this module is enabled, it calculates the scores using the card signals and increases or decreases the amount of the tokens each player has, which is stored as registers within this module, depending on whether each player bet correctly or not. It then outputs the eight 10-bit signals that indicate the current amount of tokens that each player has, as well as the 20-bit signal to the display controller.

Finally, the Winner module takes as input the board's clock, the 7th bit of the which module signal, the number of players, and the eight 10-bit token signals. When enabled, this module uses the token amount signals to determine who has the most amount of tokens, and uses the display to declare the winner. It outputs the 20-bit signal to the display controller and the new_game signal, which signifies to some other modules to reset certain registers to prepare for a new game.

Simulation Documentation

For the different modules in this project, we conducted tests in an incremental fashion. Starting from the Initialize module, we went added on to each module. This way, we were able to test our modules directly by observing the output on the FPGA board.

Initialize Module

Test	Expected Result	Successful?
7 seg display must initially contain value for: SET1	7 seg display shows SET in the first 3 digits and 1 in the final digit.	Y
Rotate button is pressed. Check if count increments.	7 seg display shows SET in the first 3 digits and 2 in the final digit.	Y
Rotate button is pressed 8 times. Check if count loops back to 1.	7 seg display shows SET in the first 3 digits and 1 in the final digit.	Y
Enter button is pressed. Leave this module.	7 seg is set to blank to indicate that the module is no longer being displayed.	Y

Choose Module

Test	Expected Result	Successful?
During this module the 7 seg display must contain value for: BET	7 seg display shows BET in the first 3 digits and nothing in the final digit.	Y
Setting any switch to high should turn on the respective LED.	LED above the high switch is bright. Rest are not bright.	Y
Multiple switches must light up.	LED is bright when more than 1 switch is high.	Y
Enter button is pressed. Leave this module. LED states should remain.	LED is unchanged. 7 seg is set to blank to indicate that the module is no longer being displayed.	Y

View Tokens Module

Test	Expected Result	Successful?
7 seg display must initially contain value for: ----	---- is displayed on the 7 seg display only when more than 1 switch is in the same state.	Y
When only 1 switch is high, display token value.	Number value is displayed on 7 seg display. Returns to ---- when more than 1 switch is high.	Y
Bet button is pressed. Leave this module.	7 seg is set to blank to indicate that the module is no longer being displayed.	Y
Amount button is pressed. Leave this module.	7 seg is set to blank to indicate that the module is no longer being displayed.	Y

Bet Amount Module

Test	Expected Result	Successful?
7 seg display must initially contain value for: ----	---- is displayed on the 7 seg display only when more than 1 switch is in the same state.	Y
When only 1 switch is high, display token value.	Number value is displayed on 7 seg display. Returns to ---- when more than 1 switch is high.	Y
PmodKYPD input must be displayed.	Values from 0-9 are dispayed on 7 seg display. Each digit shifts to the right when a new digit is inputted.	Y
Bet button is pressed. Leave this module.	7 seg is set to blank to indicate that the module is no longer being displayed.	Y
View button is pressed. Leave this module.	7 seg is set to blank to indicate that the module is no longer being displayed. PmodKYPD input is no longer displayed.	Y

View Cards Module

Test	Expected Result	Successful?
7 seg display must initially contain value for: P1 x. Where x is a digit.	7 seg display contains: P1 in the first two spots followed by a space and a single number.	Y
Rotate button should cycle through and display P2, P3, b1, b2, b3 values.	7 seg display contains: xx in the first two spots followed by a space and a single number. Here xx is one of P1, P2, P3, b1, b2, b3. The display cycles back to P1 after b3.	Y
LEDs from previous module must stay constant.	Changing the switches does not affect the LEDs as it did before.	Y
Enter button is pressed. Leave this module.	7 seg is set to blank to indicate that the module is no longer being displayed.	Y

Final Scores Module

Test	Expected Result	Successful?
7 seg display must contain a single digit followed by two blank spaces and another single digit.	X__Y is present on the 7 seg display. Here X and Y represent a value from 0-9.	Y
Enter button is pressed. Leave this module.	7 seg is set to blank to indicate that the module is no longer being displayed.	Y

Winner Module

Test	Expected Result	Successful?
Display must contain PX where X is the number of players set during the Initialize module.	7 seg display contains a blank space followed by 'P!' (here ! is a number between 0-9), followed by another blank space. ! was not greater than the value set during the Initialize stage.	Y
Enter button is pressed. Leave this module.	7 seg is set back to the Initialize module to indicate that the previous module is no longer being displayed and that a new game has begun.	Y

Having test all the game modules in such a manner, we were able to ensure that the game progressed according to our expectations. We thought testing each module alone would not be a good indicator of whether the module was implemented successfully or not.

Conclusion

Based on the tests we designed for this project and the demo we conducted to both our TA and classmates, we can confidently say that our design successfully implements the casino game of Baccarat on a FPGA board.

During the lab we dealt with the following problems:

- We initially thought of implementing the role of our PmodKYPD in our View Amount module with a PS2Keyboard instead. However, there were minimal resources available online for us to understand how this could be implemented. After 4 hours worth of failed attempts we decided to use the PmodKYPD. We only required numeric input for our module so the PmodKYPD met our requirement.
- Simulating the input of the PmodKYPD was challenging as we required to provide a stream of input to simulate the user entering values using the PmodKYPD. We found that actually programming the FPGA board and using the PmodKYPD directly was much more efficient.

This lab was a great opportunity to apply what we have learnt during the first 3 labs. However due to the varying experience different students have had in the past with Verilog and FPGA boards, some projects were much more advanced than others. Grading such projects can lead to unfair advantages, due to the open-ended nature of this lab.