Bittar, Catherine (260687735)
Sorto-Ventura, Kevin-Rafael (260692767)

24 October 2017
McGill University

ECSE 323: Digital System Design
Lab #4: g21_rules, g21_dealer_FSM

Table of Contents

Introduction

The purpose of this lab was to design two modules which can be used in the final lab which will consist of a modified game of Blackjack. The first module called *g21_rules* is used to evaluate whether a player or dealer has gone bust, i.e: sum of cards in hand is greater than 21. The second module consists of a dealer state machine: *g21_dealer_FSM*. It allows us to deal random cards from the stack. In addition, a test-bed was created in order to test the dealer module on the Altera Board.

The design was completed using the FPGA design software Quartus. It was also used to compute simulations of the designed circuit. This design was part of the project file *g21_lab3.QPF* (rather than *g21_lab4.QPF*) due to issues importing files in Quartus.

g21_rules

Circuit Function

The g21_rules circuit has 2 inputs and 2 outputs.

Type	Name	Length (bit)
Input	play_pile_top_card	6
Input	card_to_play	6
Output	legal_play	1
Output	total_value*	6

*Note that the *total_value* output were not required in the lab instructions[1] but they were added as the instructions did not consider needing to maintain a sum as the input.

As per the lab instructions [1], both inputs are 6 bits, but it should be noted that the card_to_play input has two components: the MSB and the trailing 5 bits. The MSB (card_to_play[5]) represents the presence of an ace with value 11 in the current hand. This is present as it allows the player to change the value of this ace to 1 to avoid going bust. The trailing 5 bits, card_to_play[4..0], represent the sum.

It should be noted that total_value output follows the same encoding as the card_to_play input.

The rule which is implemented in this circuit is that the sum of the values of cards in the hand of a player cannot exceed 21.

Circuit Design

The circuit is described in the VHDL design file *g21_rules.vhd*. The design is sequential and consequently uses a process. Several “if” statements are used to establish the logic as specified below, see Figure 1 Decision tree for g21_rules module.

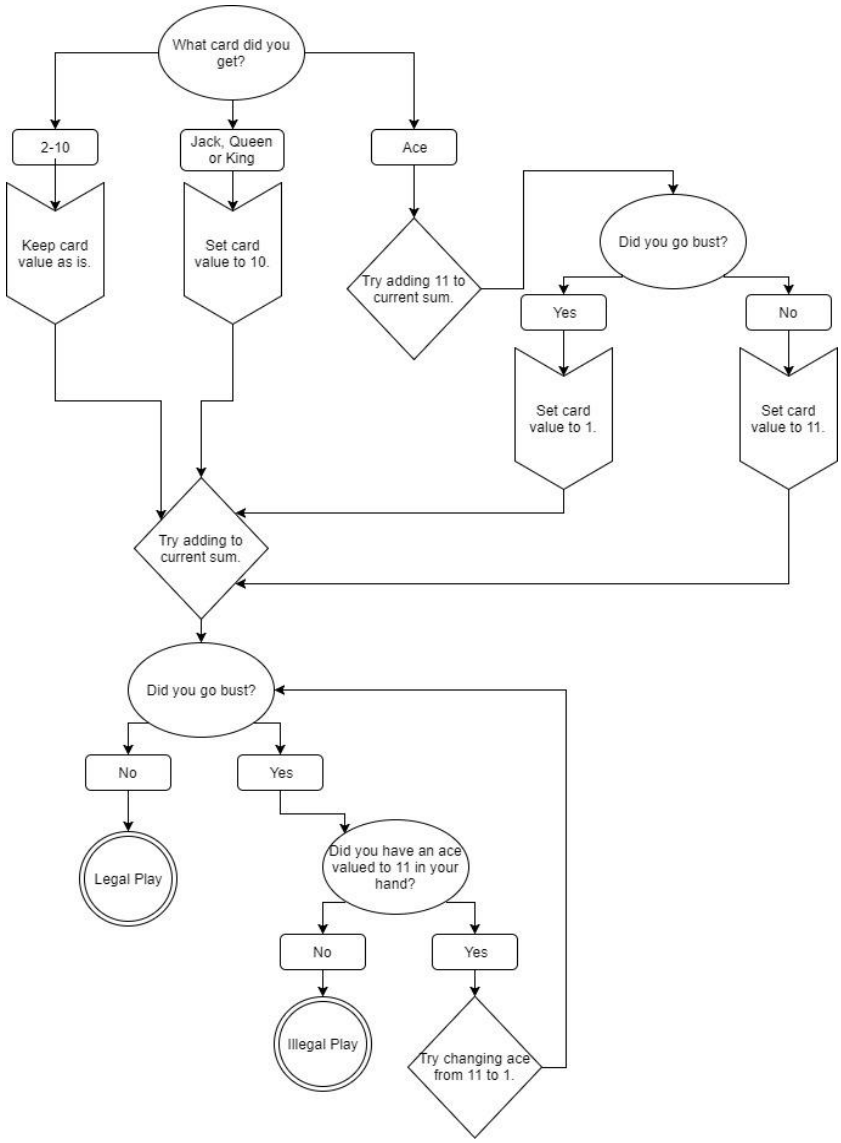

Figure 1 Decision tree for g21_rules module.

Schematic

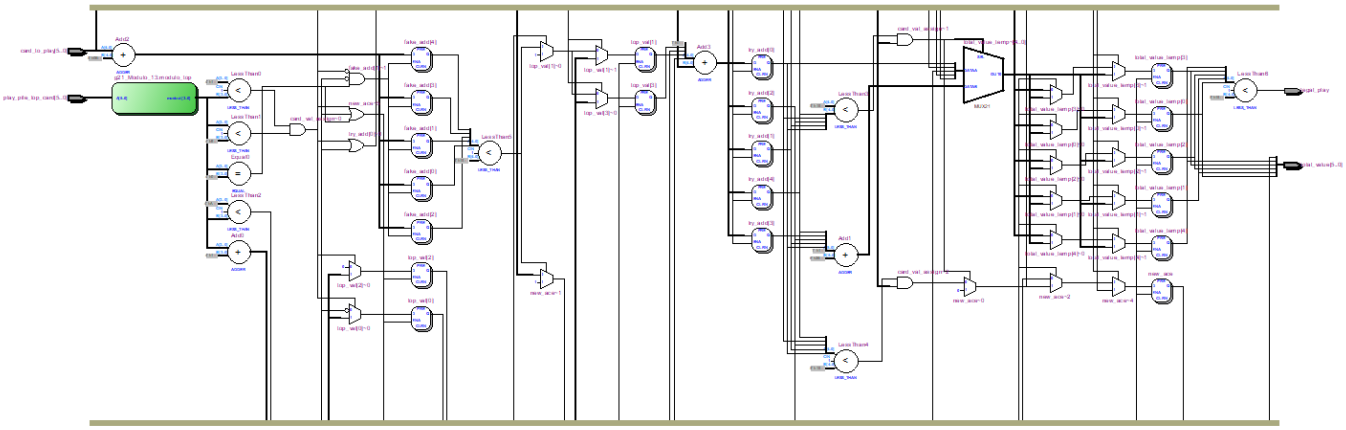

Figure 2 RTL schematic generated from VHDL code for g21_rules module.

Simulation and Discussion

A timing simulation was performed on the circuit to ensure it was functioning correctly. Two examples of results of the simulation are shown below. As shown in the schematic above, there are three sets of flip-flops, this indicates there will be a delay between the input and output. This is reflected in the simulations, and should be considered when implementing this circuit as a part of a larger system.

In the case of Figure 3 g21_rules timing simulation with current sum of 15 without an ace. the value in the hand is 15 and there is no ace with value 11. It can be observed that the play is legal while the card to add to the hand is less than 7. The total value matches this behavior. It should be noted that there is noise present during the transition between card values. This is a limitation of the circuit which should be considered when implementing it in a larger system. In the current context, however, it should be fine as the outputs will be evaluated given one
set of inputs at a time.

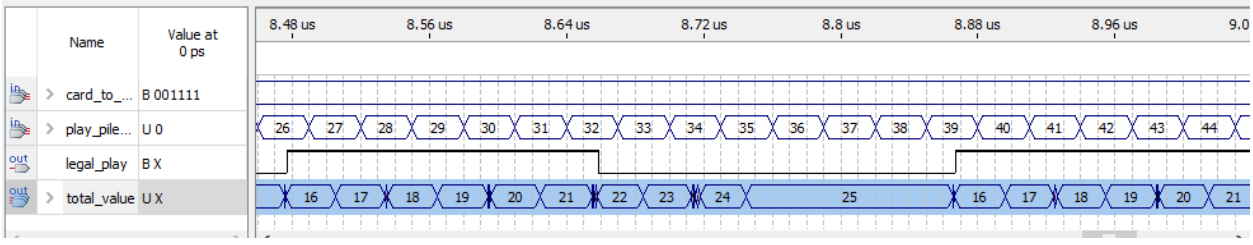

Figure 3 g21_rules timing simulation with current sum of 15 without an ace.

The following simulation was completed using the same current total of 15, but in this case an ace was present. It should be noted that the total_value is encoded with the leading bit representing the presence of an ace with value 11. For example, an output of

49 is 110001 in binary. Removing the leading bit, the output was 10001 which is 17 in decimal. This is appropriate as the previous value was 15 and the new card was 2.

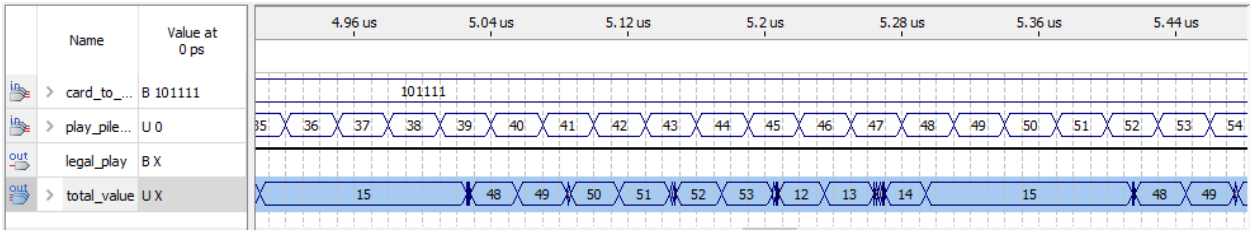

Figure 4 g21_rules timing simulation with current sum of 15 with an ace.

g21_dealer_FSM

Circuit Function

The g21_dealer_FSM circuit has 3 inputs and 1 output.

Type	Name	Length (bit)
Input	Request_deal	1
Input	reset	1
Input	Clock	1
Output	Rand_enable	1

As with the rules module, the implemented input/outputs were selected by the team and do not match exactly those given in the instructions [1]. The function of this module is to deal a card from a deck. This is a state machine which toggles between two states as seen below. When Rand_enable is on, a card is dealt.

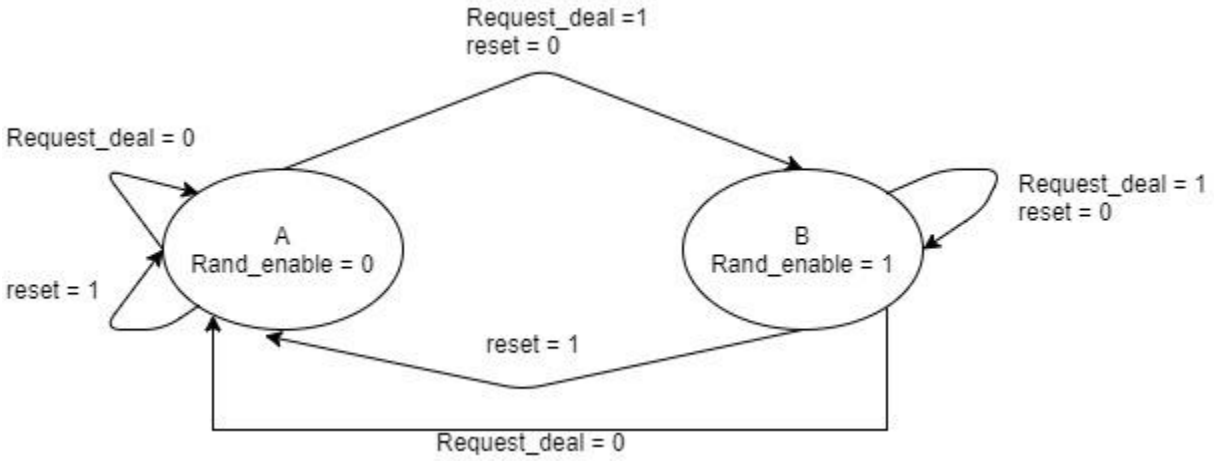

Figure 5 g21_dealer_FSM state diagram.

Circuit Design

The circuit is described in the VHDL design file *g21_dealer_FSM.vhd*. Instead of shuffling the cards of a deck as it would be done in real life, a random card from the deck is dealt every time a card is requested. There are three sections to the design: the random address generator, the state machine, and the deck of cards.

A. Random address generator
   The *g21_RANDU* circuit developed in Lab #2 is used to generate 32-bit random numbers. The 6 most significant bits from this number are extracted to form the basis of the random address. However, there is no guarantee that the number will correspond to an address. The lab instructions have a provision which checks that the random number is smaller than the number of cards in the deck in the state machine. Instead of this, the 6 extracted bits are divided by the number of cards in the deck (NUM) and the remainder of this division is used as the random address. This ensures the random address generated always corresponds to a non-empty address in the stack.
B. State machine
   The state machine determines if a card has been requested and should be dealt. When the correct inputs are given, it enables the request function.
C. Deck of cards
   The deck cards consists of the *g21_stack52* module designed in Lab #3. The INIT function is used to initialize values in all the registers. When the request function is enabled, it selects the ADDR of the random address generator and switches to POP mode for one clock cycle to output a single card.

Schematic

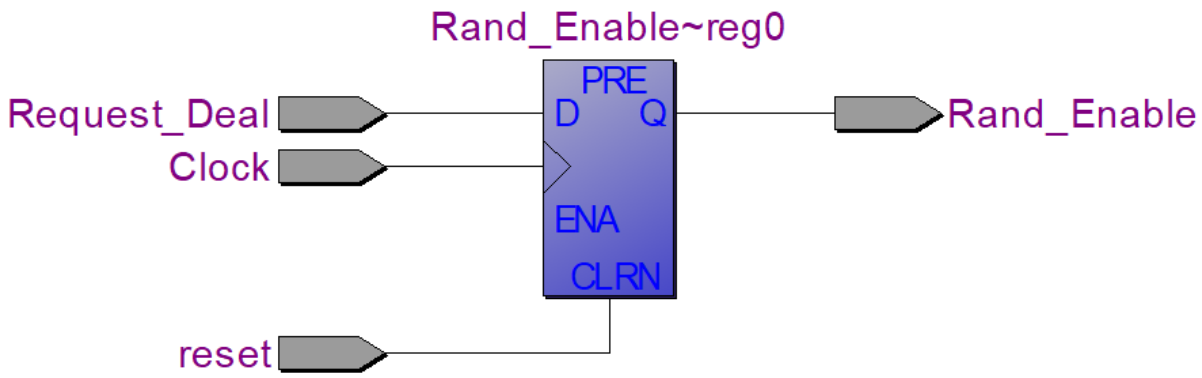

Figure 6 RTL schematic generated from VHDL code for g21_dealer_FSM.

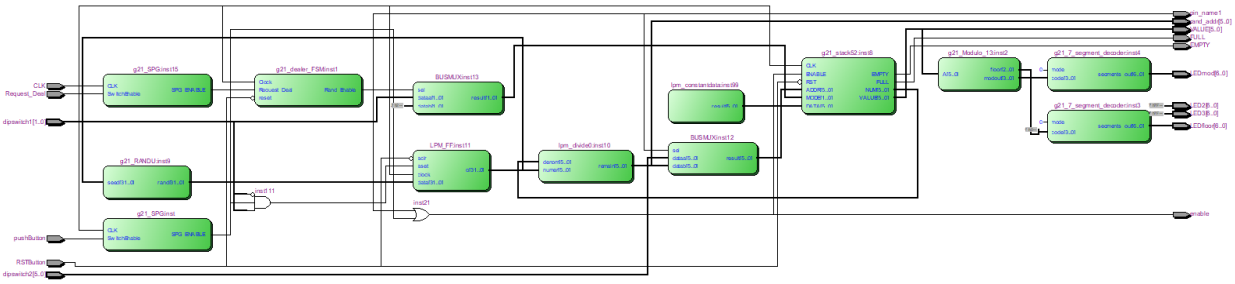

Figure 7 RTL schematic generated from block diagram for dealer test-bed.

Simulation and Discussion

A timing simulation was done on the circuit to ensure it was functioning correctly. As seen in Figure 6 RTL schematic generated from VHDL code for g21_dealer_FSM.Figure 8 g21_dealer_FSM timing simulation.the state machine is simplified to a D flip-flop. The results of the simulation can be found in Figure 8 g21_dealer_FSM timing simulation. All possible cases were tested. It can be observed that a delay is incurred as the D flip flop waits for the rising clock edge to change values. This should be considered when implementing it in another application.

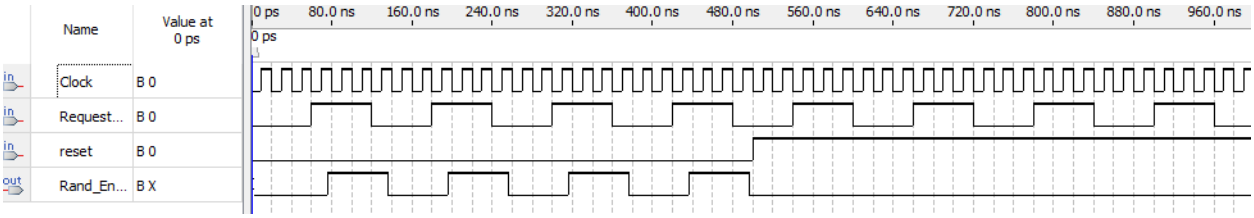

Figure 8 g21_dealer_FSM timing simulation.

The testbed circuit, which was developed to implement all three mechanisms described in the circuit function, was used to further test the state machine. It was successfully implemented on the physical Altera DE board.

Conclusion

The purpose of this lab was to design two modules which would be later used to create the final product: a game of Blackjack. The first module implemented the rules of the game, allowing the program to know whether or not a play was legal. The second module consisted of creating a module that dealt out cards to the players. Although the lab instructions were followed in spirit, several decisions were taken to implement the functions differently and more efficiently. In the case of the rules module, simulations showed it is vulnerable to noise. This should be considered when implementing it in the final product.

References

[1] Prof. J. Clark, Lab Instructions, “ECSE-323 Digital Systems Design: Lab #4 - VHDL for Sequential Circuit Design Fall 2017”, Department of Electrical and Computer Engineering, McGill University, Oct. 2017.