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

8 December 2017
McGill University

ECSE 323: Digital System Design
Lab #5: Card Game System Integration

Table of Contents

List of Figures

List of Tables

Introduction

The purpose of this lab was to complete the design of a Blackjack card game by integrating various components which were designed during the last four labs. This includes the stack, the rules modules, as well as smaller components like the 7-segment decoder. This left two components to design: the controller FSM, and the data path which includes the user interface. Both components were successfully tested directly on the Altera DE1 board.

The design was completed using the FPGA design software Quartus. This design was part of the project file *g21_lab5.QPF*.

Deal_FSM

Circuit Function

The Deal_FSM circuit is a finite state machine with 14 inputs and 12 outputs defined below. The inputs are used to determine the next state, while the outputs are used as control signals for the rest of the circuit.

Table 1 Deal_FSM inputs and outputs

Type	Name	Length (bit)	Purpose
Input	Request_Deal	1	Button press – indicates player wants another card.
Input	Stop	1	Button press – indicates player has finished playing.
Input	INIT	1	Button press – indicates start of a new hand.
Input	reset	1	Button press – indicates player want to reset game.
Input	Clock	1	Clock signal
Input	Dealer_legal	1	True if current dealer total is less than 21.
Input	Player_legal	1	True if current player total is less than 21.
Input	Dealer_total	5	Current dealer sum.
Input	Player_total	5	Current player sum.
Input	Dealer_high	1	True when dealer sum is greater than player sum.
Input	Player_high	1	True when player sum is greater than dealer sum.
Input	Game_over	1	True if game is over (one player has won 3 out of 5 games).

Input	score_dealer	2	Number of hands won by dealer in current game.
Input	score_player	2	Number of hands won by player in current game.
Output	rand_seed	1	Starts the random address generator.
Output	Clear_count	1	Clears the counters for the number of hands won.
Output	Dealer_Enable	1	Enables card to be dealt to the dealer.
Output	Player_Enable	1	Enables card to be dealt to the player.
Output	player_turn	1	Lights an LED to indicate player turn.
Output	Compare_enable	1	Enables the comparison of the dealer and player sums.
Output	Count_dealer	1	Increment the hands won by dealer in current game.
Output	Count_player	1	Increment the hands won by player in current game.
Output	Count_gd	1	Increment games won by dealer.
Output	Count_gp	1	Increment games won by player.

The circuit has twenty states which enable and disable different output signals that control the data path circuit. Please see the design file *Deal_FSM.vhd* for the details of the state transitions conditions.

Circuit Design

The circuit is described in VHDL design file *Deal_FSM.vhd*. Although the instructions [1] recommended the use of a second FSM to control the computer player, a design decision was made to integrate both in one FSM. This had the advantage of simplifying its implementation. This FSM can be broken up into five distinct operations, although each of these is composed of several states: start of hand, initial card deal, player play, dealer play, and winner determination.

Start

The start operation initializes the circuit and waits for the user to start the game. During this operation, the random address generator is started with a seed, and the deck of cards is initialized. This operation also handles the total game count. Three states make up the start operation.

Table 2 Start operation states and signal assigments

State	Function	Signal Assignments
`rand_seed`	Initialize the random address generator	`Rand_seed <= ‘1’;`

`waiting`	Wait for user to press start button. Reset signals to ‘0’.	`Dealer_Enable <= '0';`
		`Player_Enable <= '0';`
		`count_dealer <= '0';`
		`count_player <= '0';`
		`Compare_enable <= '0';`
		`player_turn <= '0';`
		`count_gp <= '0';`
		`count_gd <= '0';`
		`rand_seed <= '0';`

`ready`	Check if game is over and increment dealer or player game totals.	`if(game_over = ‘1’ and player_score = ‘11’) then count_gp <= ‘1’;`
		`if(game_over = ‘1’ and dealer_score = ‘11’) then count_gd <= ‘1’;`

Initial Card Deal

This operation consists of dealing out two cards to both the dealer and the player and checking if either has already reached 21. If either player reaches 21, the hand terminates. Note that the dealer’s first comparison occurs before the players, as the dealer wins if there is ever a tie. Six states make up the initial card deal operation.

Table 3 Initial card deal operation states and signal assigments

State	Function	Signal Assignments

`dealer_card1`	Deal first card to dealer. Clears hand totals if game was over. Disables game count signals.	`Dealer_enable <= ‘1’;`
		`count_gp <= ‘0’;`
		`count_gd <= ‘0’;`

		`if(game_over = ‘1’) then Clear_count <= ‘1’;`
`player_card1`	Deal first card to player. Disables clear hand signal.	`Player_enable <= ‘1’; Dealer_enable <= ‘0’; Clear_count <= ‘0’;`
`dealer_card2`	Deal second card to dealer.	`Dealer_enable <= ‘1’; Player_enable <= ‘0’;`
`player_card2`	Deal second card to player.	`Player_enable <= ‘1’; Dealer_enable <= ‘0’;`
`dealer_fc`	Verify if dealer has reached 21.	`Player_enable <= ‘0’; Dealer_enable <= ‘0’;`
`player_fc`	Verify if player has reached 21.	`Player_enable <= ‘0’; Dealer_enable <= ‘0’;`

Player Play

This operation allows the player to request additional cards. If the player reaches 21, he wins. If the player exceeds 21, the dealer wins. Four states make up the player play operation.

Table 4 Player play operation states and signal assigments

State	Function	Signal Assignments
`player_wait`	Wait for player to ask for a card or indicate he is finished.	`player_turn <= ‘1’;`
`player_play`	Enable card deal to the player.	`Player_enable <= ‘1’; player_turn <= ‘0’;`
`player_en`	Disable the card deal so only one card is dealt.	`Player_enable <= ‘0’;`
`player_test`	Verify if player has reached or exceeded 21.	`Player_enable <= ‘0’;`

Dealer Play

The play of the dealer replaces the Computer_FSM specified by the instructions [1]. The operation requests cards until the dealer total has exceeded 16. Four states make up the dealer play operation.

Table 5 Dealer play operation states and signal assigments

State	Function	Signal Assignments
`dealer_wait`	Check if dealer has exceeded 16 or reached 21.	`player_turn <= ‘0’;`
`dealer_play`	Enable card deal to the dealer.	`Dealer_enable <= ‘1’;`

`dealer_en`	Disable the card deal so only one card is dealt.	`Dealer_enable <= ‘0’;`
`dealer_test`	Verify is dealer has exceeded 21.	`Dealer_enable <= ‘0’;`

Winner Determination

The winner is determined by comparing the totals of the dealer and player if neither has gone bust or reached 21. Note that the dealer breaks any ties. This operation handles the hands won count. Three states make up the winner determination operation.

Table 6 Winner determination operation states and signal assigments

State	**Function**	**Signal Assignments**
`compare`	Enable external comparator module.	`Compare_enable <= ‘1’;`
`dealer_win`	Increment dealer hands won count.	`count_dealer   <= ‘1’;`
`player_win`	Increment player hands won count.	`count_player   <= ‘1’;`

Schematic

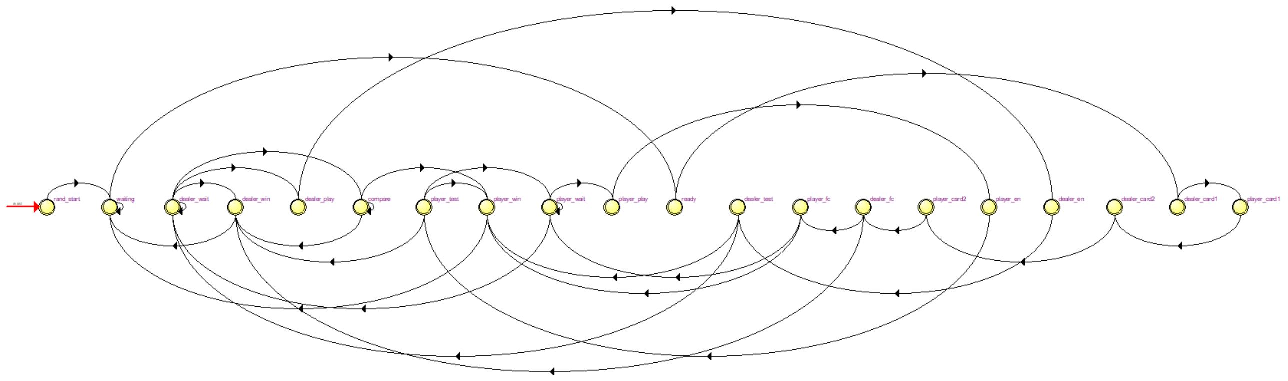


Figure 1 Deal_FSM generated State Machine diagram

Discussion

No simulations were completed for the FSM because of the sheer quantity of states. It was found to be simpler to integrate the FSM into the datapath and test it directly on the Altera DE1 board.

The most important design decision made here was to integrate the Computer_FSM as a part of this controller. This had the advantage of simplifying the implementation of FSMs in the overall circuit. With a single controller, the control signals would all come from the same module as opposed to two different ones, as would be the case with a separate FSM for the computer player. The limitation lies in further implementations of the FSM. As the dealer’s play is effectively hard-coded to be controlled by the computer, it would be far more complex to exchange the role of the computer. This would useful if the game were expanded to more players and the computer would play for a different player. However, because this was the final lab of the current project, it was decided that this simplification of the controllers was acceptable.

Data Path Circuit

Circuit Function

The data path circuit has 5 inputs and 7 outputs. It is the final circuit which incorporates modules created over the course of the semester to play Blackjack on the physical Altera DE1 board.

Table 7 Data path circuit input and outputs

Type	Name	Length (bit)	Purpose
Input	CLK	1	Clock Signal
Input	butt_INIT	1	Input mapped to a button on the board and triggers the start of a hand.
Input	butt_reset	1	Input mapped to a button on the board and triggers the reset of the games.
Input	butt_Stop	1	Input mapped to a button on the board that indicates player has finished playing.
Input	butt_Play	1	Input mapped to a button on the board that indicates player has requested a card.
Input	sel_0	1	Input mapped to 1 switches on the board and designates the display of the 7-segment LEDs.
Output	7_dealer_dec	7	Output mapped to 7-segment LED and displays the dealer’s total decimal position, or the previous card played.
Output	7_dealer_un	7	Output mapped to 7-segment LED and displays the dealer’s total unit position.
Output	7_player_dec	7	Output mapped to 7-segment LED and displays the player’s total decimal position, or the dealer’s games won.

Output	7_dealer_un	7	Output mapped to 7-segment LED and displays the player’s total unit position, or the players games won.
Output	player_turn_led	1	Output mapped to a green LED on the board and triggers when it is the player’s turn to play.
Output	score_player	3	Output mapped to 3 green LEDs on the board which indicate the number of hands won by the player.
Output	score_dealer	3	Output mapped to 3 red LEDs on the board which indicate the number of hands won by the dealer.

The data path circuit incorporates the user interface with the circuit functionality. User inputs are control signals for the rest of the circuit. The outputs of the circuit are in turned translated to the user so that the user can continue to communicate with the board.

Circuit Design

The data path circuit is defined in the block diagram *g21_lab5_testbed.bdf*. To facilitate its design, several components were written to interface the output signals of the functional circuitry into LED patterns the user would be able to interpret. Within the functional circuitry, circuits which perform tasks which are related to the function of the circuit, not the UI, additional modules were also created to simplify the assembly of the data path. The following provides descriptions of the design of these components.

<ins>User Interface</ins>

The user interface is composed of push buttons, dipswitches, 7-segments LEDs and red/green LEDs. It was decided to use the red/green LEDs to display the hand score of the player and dealer as opposed to displaying them in decimal on the 7-segment LEDs so that a single dipswitch so that there would only be 2 different 7-segment LED displays to toggle between. Although it has the advantage of being simpler visually for the user, it is limiting as if the rules were to change, if a game were to become best out of 7, significant changes would need to be made to the circuit.

Single Pulse Generators

There are four single pulse generators present in the data path circuit, one for each of the push buttons which are used as inputs. The circuit was defined in Lab #3 in the design file *g21_SPG.bdf*. It ensures a single pulse is generated on the signal when the button is pressed by setting a minimum number of clock cycles between when the signal can be high. This can also be called a “debouncer” circuit as it mitigates the effect of the mechanical switch bouncing when it is pressed.

Show Totals

The show_totals circuit is used to translate the two binary values of the player and dealer totals into decimal values that could be displayed on the 7-segment LEDs. This circuit is defined in Lab #5 as *show_totals.vhd*. It separates the binary value of each total into its decimal and unit position value using the divide and mod functions. It incorporates the *g21_7_segment_decoder* defined in Lab #2 to translate each of the 4 values into the correct segments to be properly displayed to the user.

Show score

The show_scores circuit is used to translate the hand score which is a 2-bit value into a 3-bit vector corresponding to each LED to light up depending on the score. It is defined in Lab #5 as *show_score.vhd*. Two instances of the circuit are used to display the hand score of the dealer and the player. Its output is wired to the LEDs directly.

g21_7_segment_decoder

Defined in the Lab #2 as *g21_7_segment_decoder.vhd*, this circuit is used whenever a value needs to be displayed on the 7-segment LEDs. On top of being used in the show_totals circuit, it also used on its own to display the game scores of the dealer and player, as well as the previous card played from the stack.

Functional Circuitry

The functional circuit is composed of modules which complete tasks in conjunction with several registers and counters.

Controller

The Deal_FSM circuit functions as the controller for the data path circuit. By enabling and disabling various control signals, it ensures all the operations are occurring at the right time in the right order.

Card Deck

The deck of cards is an instance of the g21_stack52 circuit which was developed in Lab #3 (*g21_stack52.bdf*). It is initialized at the start of the of each hand. Cards are popped from random address to simulate shuffling of the deck. The module can be enabled by either the INIT button being pressed or by the FSM when it requests a card.

Random Address Generator

The random address generator was developed during Lab #4, but the actual design entity was only created in Lab #5 under *random_address_gen.bdf*. It uses the *g21_RANDU.bdf* circuit from Lab #2 and a mod function to generate random addresses when the module is enabled. It is initialized after a reset by asynchronously setting a seed value.

Determine Winner

The g21_determine_win circuit was developed in Lab #5 and is described in *g21_determine_win.vhd*. It is enabled by the compare signal from the FSM. It compares the totals values of the dealer and player to determine which wins. Note that the dealer breaks any ties.

Game Over

The g21_game_over circuit determines if a game is over. This occurs when either the dealer or player gets to 3 hands won. It is described in *g21_game_over.vhd* as a part of Lab #5.

Registers

Three LPM_FF registers are used in the data path to hold the values of the totals of the dealer and player, as well as the last card taken from the stack. The registers are useful for the totals as the values are used in a feedback loop. For the last card dealt, the registers are necessary to ensure only cards dealt are shown to the user. Since the random number generator is continually outputting random addresses, the stack output value will continually change unless the value is stored in a register at the right moment.

Counters

Four lpm_counters are used in the circuit to keep track of the game and hand scores of the player and the dealer. These components are useful as they are very easy to increment scores, and reset when a game finishes or a reset occurs.

Schematic

The following are some of the more important parts of the datapath schematic. Please refer to the block diagram *g21_lab5_testbed.bdf* for the full schematic.

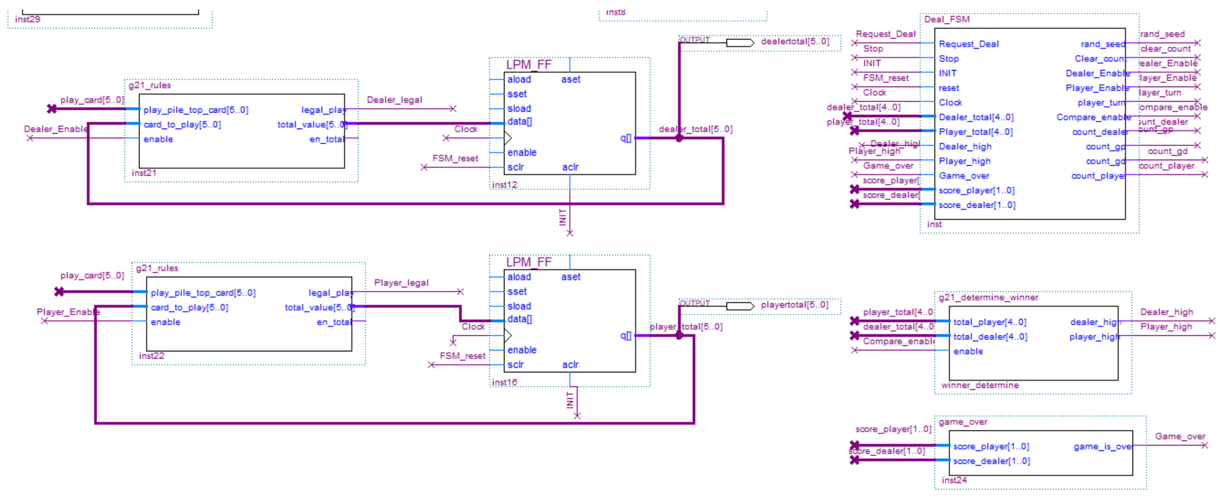


Figure 2 Schematic of Deal_FSM, rules, determine_winner and game_over

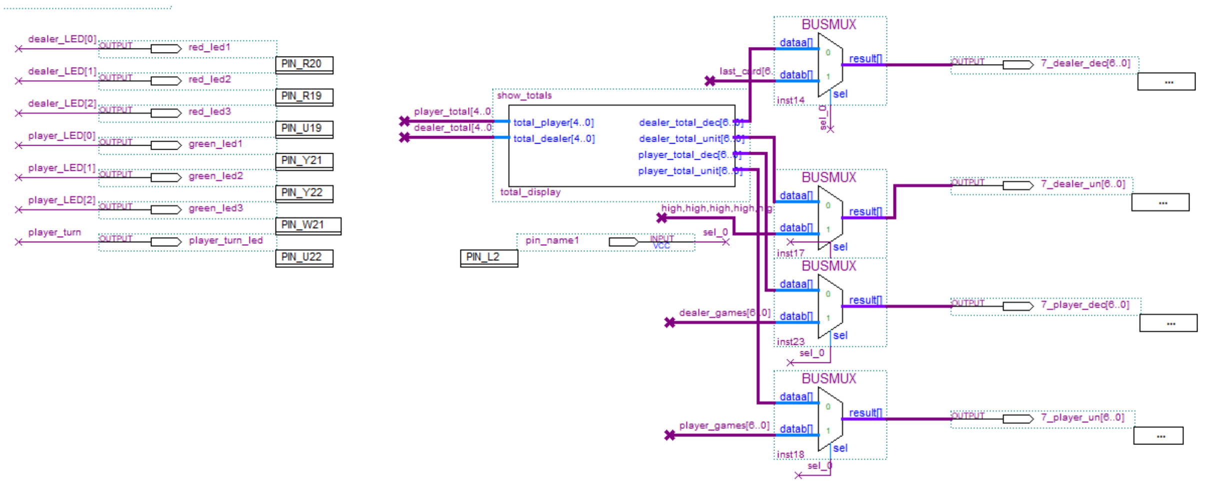


Figure 3 Schematic of LED and 7-segment display output assignments

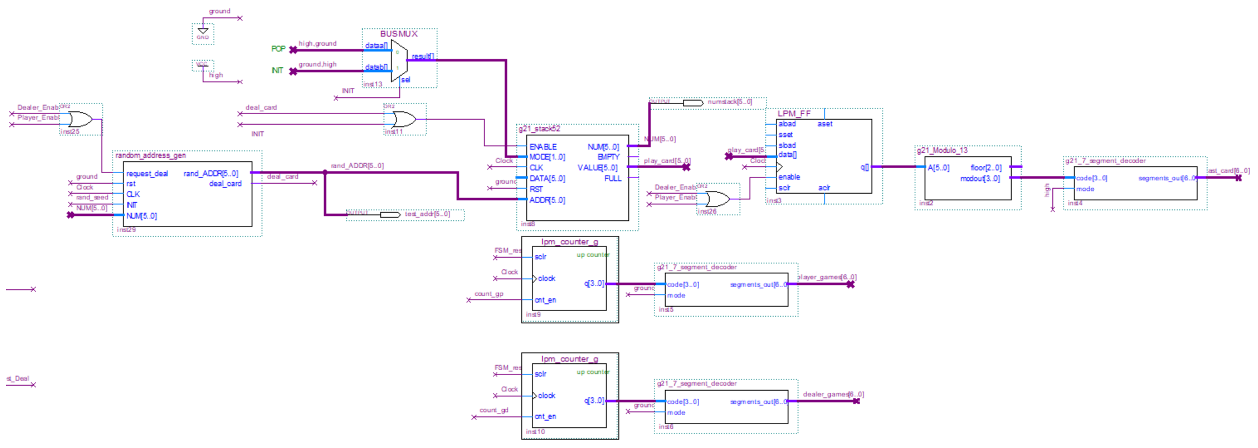


Figure 4 Schematic of random_address_gen, stack52, game counters and last card

Discussion

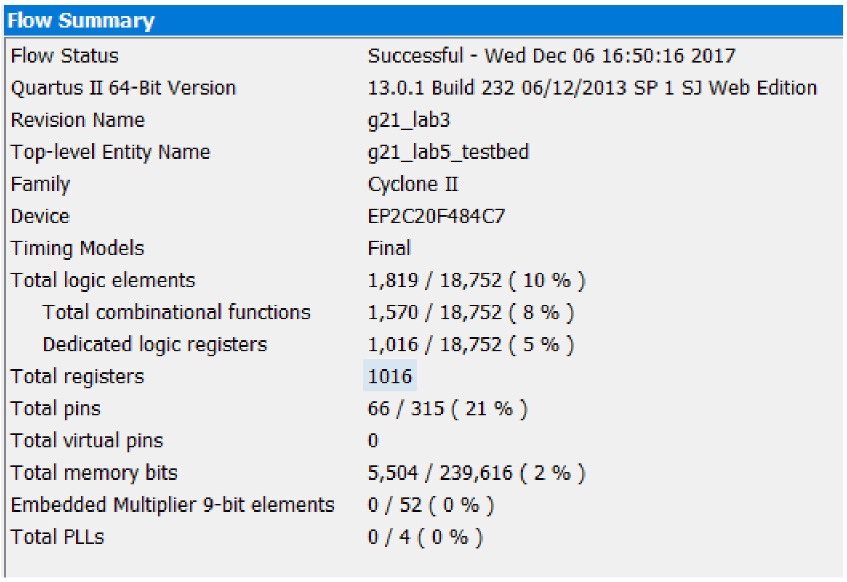
As with the Deal_FSM circuit, no simulations were conducted on the data path circuit. To be debugged, the circuit was loaded to the Altera DE1 board after every modification was made to the code or design file. The circuit was ultimately successfully functional.

Although the circuit functions for this specific application, several components are “hard coded” and thus to extend the functionality of this circuit, significant modifications would need to be applied. This is a definite limitation of the circuit.

Furthermore, the data path consists of several interconnected modules which increases the possibility of errors. For every extra module, more control signals are needed, but this makes debugging more tedious and challenging. The circuit is highly sensitive to timing issues as several operations need to occur in synchronized fashion.

In such situations, race conditions are likely to develop. It was attempted to limit such conditions by using more control signals and slowing down the operation. For example, the final comparison between the totals of the player and dealer is enabled, even though it is only combination logic. This was put in place to avoid that the comparison between the totals would occur before one of the totals had time to propagate to the comparator circuit. Therefore, upon completion of play, an entire clock cycle is dedicated to enabling this circuit. This ensures the comparison only occurs after both totals have been updated. In this case, efficiency was sacrificed for protection against race conditions.

It should be noted that a small percentage of the board is in use as per the Figure 5 g21_lab5_testbed Compilation Flow summary.



Flow Summary

Flow Status	Successful - Wed Dec 06 16:50:16 2017
Quartus II 64-Bit Version	13.0.1 Build 232 06/12/2013 SP 1 SJ Web Edition
Revision Name	g21_lab3
Top-level Entity Name	g21_lab5_testbed
Family	Cyclone II
Device	EP2C20F484C7
Timing Models	Final
Total logic elements	1,819 / 18,752 (10 %)
Total combinational functions	1,570 / 18,752 (8 %)
Dedicated logic registers	1,016 / 18,752 (5 %)
Total registers	1016
Total pins	66 / 315 (21 %)
Total virtual pins	0
Total memory bits	5,504 / 239,616 (2 %)
Embedded Multiplier 9-bit elements	0 / 52 (0 %)
Total PLLs	0 / 4 (0 %)

Figure 5 g21_lab5_testbed Compilation Flow summary

Conclusion

The game of Blackjack was successfully implemented on the Altera DE1 Cyclone II boards. The circuit used various components which were developed in the labs which occurred over the course of the semester. Although the final product is functional and meets all requirements, some components were designed with this specific implementation and with the specific instructions and thus are not easily extendable to other applications. This should be considered if the circuit, or its associated modules were to be used in other contexts.

References

[1] Prof. J. Clark, Lab Instructions, “ECSE-323 Digital Systems Design: Lab #5 – System Integration for the Card Game”, Department of Electrical and Computer Engineering, McGill University, Nov. 2017.