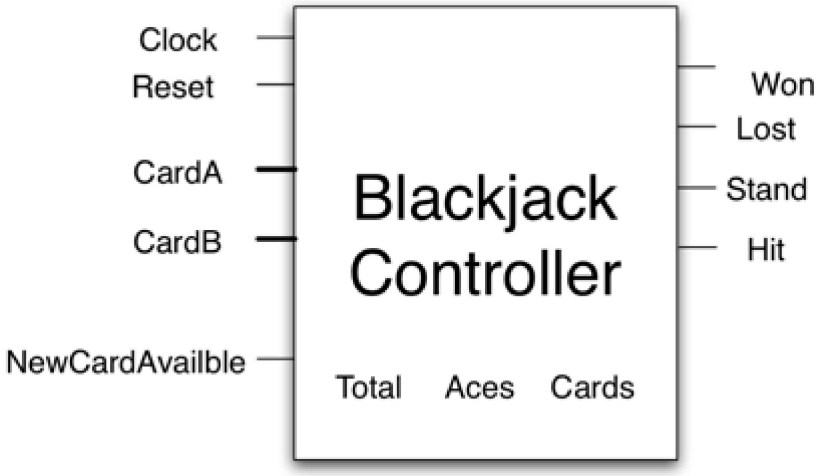
Lab 14 - Finite State Machines on FPGA

In this lab you will design and build a simple blackjack game controller on the FPGA using VHDL. You will begin by defining a finite state machine that will satisfy the design requirements, design the VHDL implementation, simulate the design in Quartus II, and finally implement it on the FPGA board. You will also use the Logic Analyzer to display the input and output signals of your circuit.



Problem Statement: The program is played with a deck of cards. Cards 2 to 10 have vales equal to their face value, and an ace has a value of either 1 or 11. The object of the game is to accept a number of random cards such that the total score (sum of values of all cards) is as close to 21 without exceeding 21.

Your automated dealer should be able to perform the following functions:

- Start with two cards and determine the working value.
- If the initial value is 21 you have blackjack, and assert a won signal.
- If at any time your value is less than 16 you hit (take another card).
- If you have exceeded 21 and you are counting an ACE in your hand as 11, count it as 1.
- If your hand totals greater than 17 but less than or equal to 21, then assert the stand signal.
- If you hand totals greater than 21 and you have no aces counting as 11, assert the bust signal.
- If you have received a total of 5 cards and your total does not exceed 21, assert the won signal (5 card Charlie)

Design Considerations: This finite state machine will require interaction with the user in order to transition between states. Card values should be specified using the dipswitches on the board and there are 13 unique card values, (2-10, J, Q, K, A). Thus,

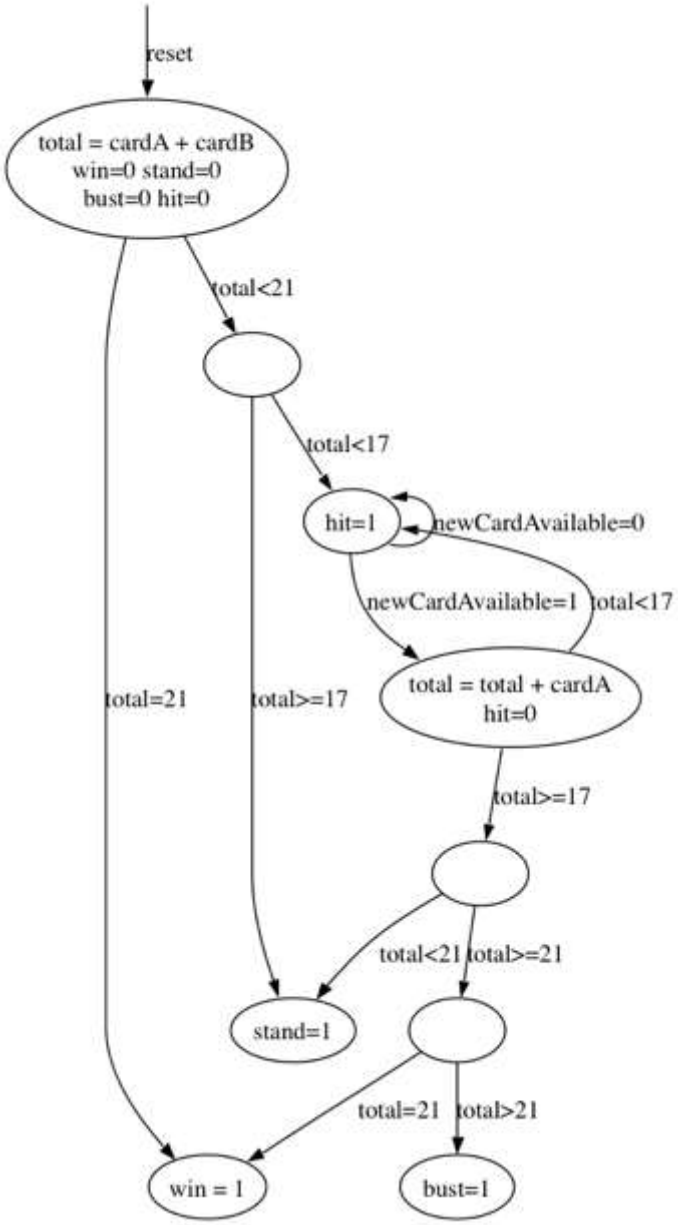
one bank of 8 switches should be able to handle two card values simultaneously (required for the deal of hand). However, 10, J, Q, K all represent the same value. So our encoding can be as follows:

2 = 0010, 3 = 0011, 4 = 0100, 5 = 0101, 6 = 0110, 7 = 0111, 8 = 1000, 9 = 1001, 10/J/Q/K = 1010, A = 1011

The FSM will be able to make some decisions immediately, but any time a “hit” is required it must allow the user to change the value of the switches before accepting the card and continuing.

We assume the clock is the internal FPGA clock. The reset signal is what indicates to start a new game and should assume that two cards are ready for dealing a new hand. CardA and CardB are the two new cards for a new hand. When the controller requests a hit, the CardA value should be changed and the NewCardAvailable signal should be asserted. This should be a debounced switch. The won, lost, stand, and hit signals should be relatively obvious. The hit is really a hit request so that the user can assert the new card and tell the controller to proceed.

You should need to keep track of three main values, the total value of your cards, the total number of cards, and the total number of aces that you are currently counting as 11. When you hit, you should adjust your card total and the number of cards. If you move an ace from being counted as 11 to 1, you need to reduce the number of aces.



The figure above is an example a FSM state transition diagram for a blackjack dealer that does not handle aces as a value of 1 nor does it detect 5 card charlies. This FSM diagram is not optimal and actually does not properly handle the case of being dealt two aces and assumes that only valid card values are supplied as input. The process is to add the first two card values into the total. It then checks if you have blackjack and goes straight to the win state. Otherwise it checks to see if the total is <17 then hits otherwise stands. If hitting, it waits for a new card, if still < 17 gets another hit. If >=17 it checks if its <21 and stands, or if >=21 it then then checks to see if it is 21 and wins or >21 and busts.

Part I: Finite State Machine Design

A. Draw a state transition diagram for your design, showing the next state for each combination of the present state and the input signals. Use the example above to direct your design techniques.
B. Simplify your state machine using techniques learned in 132 or from lab 10.

You will not need to determine some low level details such as flip flop type and excitations as you will specify the behavior of the FSM in VHDL and allow Quartus (the synthesis tool) to generate the logic for your automatically.

PartII: VHDL and Quartus II Simulation

A. Design the VHDL for your blackjack controller based on the Behavioral VHDL lecture
   You are recommended to use a process style format with a clock and reset.
B. Save and compile your design for simulation
C. Simulate your design. Generate a timing diagram in the Waveform editor that shows that your blackjack controller works properly for the various conditions (requiring a hit, bust, wining on blackjack, winning on 5 card Charlie, etc.)
D. Print out your VHDL and timing diagram. Make sure your name and appropriate labels are on the printouts.

PartIII: FPGA Board Realization and Logic Analyzer

A. Instantiate your circuit on the FPGA board. You will need to map to dipswitches for CardA and CardB inputs. Use a debounced switch for NewCardAvailable. You may map the output signals to LEDs.
B. When your design is working correctly show it to the instructor or TA.
C. Generate a timing diagram that matches the test cases in your simulation using the logic analyzer. Make sure you label your signals in the logic analyzer to match the names from the simulation.
D. Print out your Logic Analyzer output.

Part IV: Write a lab report

1. Purpose
2. Method
3. Results
4. Conclusions - Acknowledgments