COMP3222/9222 Digital Circuits & Systems
Session 2, 2018

Blackjack Player - Weeks 11 — 13

This page was last updated: 08/19/2019 15:34:05

Overview

This final (design challenge) lab involves the design and implementation of a datapath and controller for a Blackjack card player. You are to design a machine that plays the game of Blackjack with a particular strategy. An initial, simple version of the player is to be developed and implemented first. The initial version is then to be enhanced to obtain the final implementation.

Important Notice re Lab marking

This lab must be marked off by Week 13 at the latest.

Please confirm your lab marks for all labs with your tutor by the end of Week 13.

Reminder

Please remember to copy the files you create on a lab machine to your own secure storage and to remove these from the lab machine before you leave the lab!

Specifications

You are to design a machine that plays the game of Blackjack with a particular strategy. The game is to be played as follows:

Goal:

- Get as close to 21 points as possible without going over.

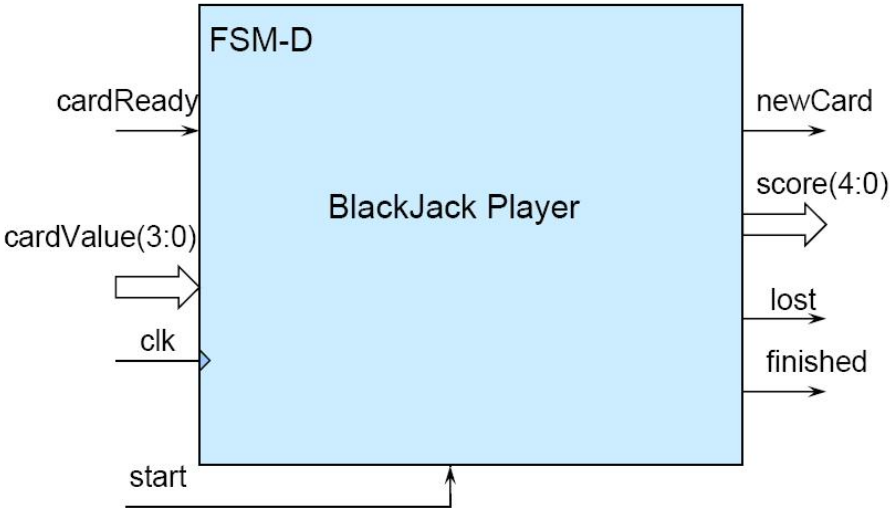
Rules:

- The game uses a standard 52-card deck, or multiple 52-card decks.
- The value of a number card (2-10) is equal to its number.
- The value of all face cards (Jack, Queen and King) is 10.
- The value of an Ace is 11, but it can count for 1 instead.
- You can keep asking for cards as long as you like until you stop or go over 21.

The strategy your player must demonstrate is:

- Ask for cards while the current total value of cards is 16 or less.
- Initially treat all Aces as worth 11 points
- If you exceed 21 points and you have previously received an Ace, count it as 1 point instead.
- Stop when your score is either between 17 and 21 (inclusive), or your score is above 21 and you don't have an Ace in reserve.

The following diagram illustrates the inputs and outputs of the BlackJack Player:



The inputs and outputs work as follows:

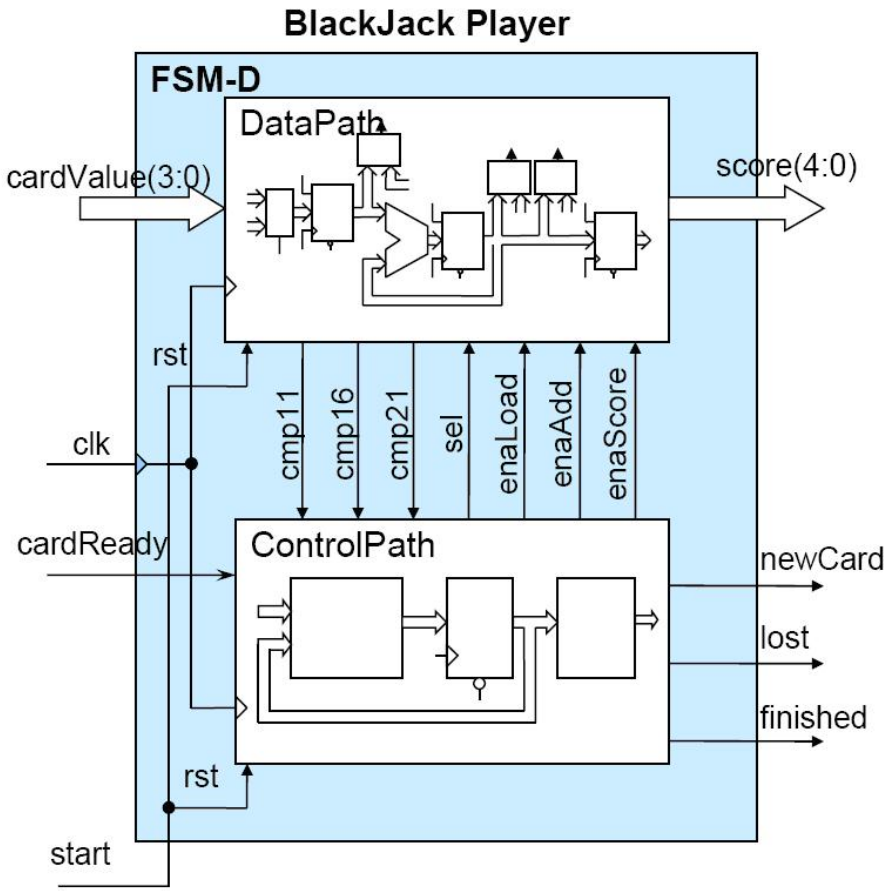
Port	**Description**
clk	Clock for the machine.
start	Asynchronous signal that restarts the player when needed.
cardReady	Signal is asserted to indicate a new card is ready. It should be de-asserted after newCard goes low.
cardValue	Value of the new card (legal values are 2 – 11). Assume the player is only presented with valid input values.
newCard	The Player asserts this signal when it wants a new card and de-asserts it when it receives a new card.
score	Signal which outputs the Players current or final score.
lost	This indicates if the Player has lost (i.e. it is busted: score > 21).
finished	This indicates the current hand is finished (i.e. 16 < score < 22).

Carefully note the handshake between the cardReady and newCard signals. The system works via the following steps:

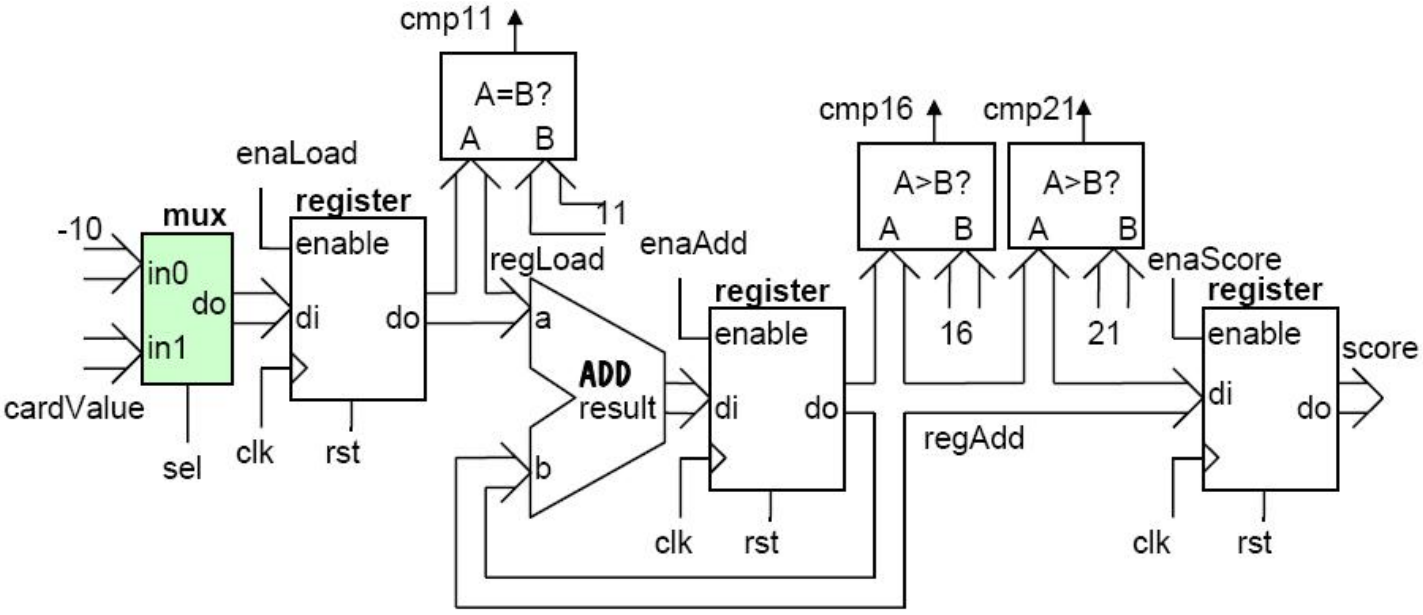
- The machine asserts newCard to indicate it wants a new card
- The operator (i.e. YOU) asserts cardReady after you have setup cardValue
- The machine de-asserts newCard to show it has received the card
- The operator de-asserts cardReady to indicate it has seen that the machine has taken the new card

This handshake ensures both the machine and controller will not get out of sync even if they are running at different clock speeds or are asynchronous. This is known as a request-ack handshake and is seen in many designs when communication is needed.

You can design the machine any way you like as long as it uses the above signal definitions. A state machine with a datapath will likely be the easiest approach. An example implementation showing the split between control and datapath which you may wish to use or modify is shown below:



While the control path implements your FSM, the datapath could be structured as illustrated in the following diagram:



Some template code that defines the top level of the Blackjack player using the shown signals is provided for download. Descriptions of the files follow:

File	Description
Blackjack.vhd	Top-level file for project

Blackjack_DataPath.vhd	Entity declaration only for the datapath shown above
Blackjack_fsm.vhd	Entity declaration only for the control FSM above
Blackjack_7Seg.vhd	Entity declaration for OPTIONAL use of 7-segment displays to report score
Blackjack.sof	Sample implementation with the desired functionality

Note the synchronisation process provided in the Blackjack.vhd code. This eliminates switch bounce caused when the switches change their value (the input signal can oscillate between 0 and 1 for several milliseconds when you flip a switch — see p.724 of the text for a discussion of this topic).

Your implementation is required to make the following connections with the Cyclone II Starter Board:

Signal name	**Connected component**
clk	On-board Oscillator (Typically 50MHz)
sw_clk	Switch 9. Use this switch to control the clock during testing if you want. See the comments in Blackjack.vhd to use it.
start	Key 0.
cardValue	Switches 3 downto 0 (3 is MSB)
cardReady	Switch 4
newCard	Green LED 5
lost	Green LED 6
finished	Green LED 7
score	Red LEDs 4 downto 0 (4 is MSB)

The handshake between the newCard and cardReady signals allows the system to be interfaced with an asynchronous system (i.e. YOU). You can make use of a switch to clock your circuit during debugging by making the indicated changes in the Blackjack.vhd file. Also make the changes if you want to simulate your design (to avoid the slow switch de-bouncing). However, your final design must operate using the on-board clock.

Your main tasks are:

- Fill in the architecture description of the datapath. You are free to use either structural (components you define in other files) or behavioural code here.
- Fill in the behavioural architecture description of the control FSM. Implement this as a Mealy machine.

Exercises

1. (2 marks) Develop an initial version of the player that does not convert Aces to 1 point when the score exceeds 21. That is, simply finish when the player scores above 16 points, indictaing whether it has lost or not — do not adjust the score while playing.

   1 mark will be given for your paper design, 1 mark for your FSM implementation and 1 mark for your datapath implementation.

   In order to be in a good position to complete the lab on time, Exercise 1 should be signed off by the end of Week 12.

   Demonstrate your implementation to your demonstrator. Discuss the modifications you will make to the code to complete Exercise 2.

2. (2 marks) When the initial version works correctly, enhance the design to arrive at a final version which adjusts the score for Aces when the player would lose without doing so.

   1 mark will be given for your paper design, 1 mark for your FSM implementation and 1 mark for your datapath implementation.

   Demonstrate the player to your lab demonstrator. Be prepared to discuss your implementation.

Hints

- Derive the state transition diagram for the initial and final FSM and confirm their correctness with your demonstrator
- After you have implemented each FSM, simulate it to ensure you have captured your design correctly
- The functionality of the player when two or more Aces are received should be noted. The input sequence 11, 11, 10, 10 results in the score being sequentially updated to 11, 12, 12, 22 and the lost flag being asserted when the score reaches 22. The player is thus required to remember that it has already seen an Ace when it is dealt a second Ace.
- Please ask the demonstrator for help if you are stuck or confused.

Credits

The initial concept and diagrams for the Blackjack player are due to Marcel Jacomet of MicroLab-I3S. This lab was initially created by Jorgen Peddersen of UNSW.

Lab completed:

Mark: _____ / 4

Signed: _____

Dated: _____
