ECSE-323
Digital System Design

Lab #4 – ***VHDL for Sequential Circuit Design*** ***Fall*** 2017

Introduction

In this lab you will learn how to use VHDL to describe sequential logic circuits, including finite state machines, using process blocks.

Learning Outcomes .

After completing this lab you should know how to:

- Use process blocks in VHDL descriptions.
- Write VHDL descriptions of sequential circuits.
- Design Finite State Machines using VHDL.

Table of Contents

This lab consists of the following stages:

1. Design of a combinational Rules module.
2. Simulation of the Rules module.
3. Design of a card dealer circuit.
4. Simulation of the dealer circuit.
5. Design of a testbed for the dealer circuit.
6. Testing of the dealer circuit on the Altera board.
7. Writeup of the lab reports

TIME CHECK

You should be this far at the end of your *first* 2-hour lab period!

1. Design of the Rules Module .

Your card game system should have a mechanism for determining whether a given card can be legally played. Different card games will have different rules.

In this lab you will make a rules module for the "***21***" card game. In this game there is a "dealer" and a non-dealer (roles will switch within the game" which contains cards that have been played. The top card in this pile will determine what card can be legally played next, according to the following rules:

- The rules form the description for the controller of the system. The controller will activate the different modules or hardware based on the activities the system is supposed to be performing. There will be a requirement to keep a running total of games one and also provision for randomization of the deck between games.

You and the computer take turns in being the dealer – 5 hands total

· Aim of the 21 Card Game is to get 21 or as close to as possible.

· Number cards have their face value, jacks, kings and queens are worth 10. Ace can be either 1 or 11 and the player who holds the ace gets to choose the value of the card.

· The dealer and all other players have two cards (in this case there is only one player). With the exception of the dealer the players have their cards face up. The dealer has one card up and one card face down.

· The dealer goes to each player one at a time (in this case there will be only one player, you). The player needs to decide if they want another card (hit) or will sit on what they have. You can have as many cards as you like as long as you don't go over 21.

- The dealer does this with every player. Players are not competing against each other, but against the dealer.
- The dealer then turns over their other card and needs to decide what to do. If the dealer has 16 or under then they must take another card.
- If the dealer has 21 (Ace and a ten value card) the dealer wins.
- If the dealer goes bust then everyone else wins.
- We reshuffle the deck of cards after every game.

There should be a rule on who is the dealer first. Highest card draw?

The winner is the best three out of five games.

This is taken from the link (there many websites with these same rules):

https://planningwithkids.com/2012/06/20/21-card-game/

In a standard deck of cards there are 4 suits (Spades (code=0), Clubs (1), Hearts (2), Diamonds (3)) with 13 different face values ranging from 1 (Ace), 2 through 10, Jack (value=11), Queen (12) and King (13).

We can encode the face value and suit of cards into a 6-bit binary number, where the encoded value is:

V = (face_value – 1) + (suit * 13).

For example, the ***9 of Hearts*** would be encoded as ***V=8+2*13=34***

To convert to encoded value of a card back into its suit and face values, we can use the modulo13 function developed in lab #1:

face_value = V modulo 13 + 1
suit = floor(V/13)

For example a code of ***45*** would correspond to a face value of ***7*** and a suit of ***3*** (diamonds).

Write a VHDL description for a circuit that implements the Rules module. This circuit will take in two 6-bit signals that represent cards (face value and suit). There should be a single output which indicates whether the proposed play is success (sum less than or equal to 21) or fail (sum exceeded 21).

In this case it is whether the card would create a sum greater than 21. If yes, then the player has lost the hand if no then the player can ask for another card or hold.

Use a process block with ***case*** statements and/or ***if*** statements to describe the circuit's functionality.

Use the following entity declaration:

```
entity gNN_rules is
 port ( play_pile_top_card : in std_logic_vector(5 downto 0);
        card_to_play       : in std_logic_vector(5 downto 0);
        legal_play         : out std_logic);
end gNN_rules;
```

2. Simulation of the Rules Module .

Once you have finished your VHDL description of the Rules module, show it to the TA and explain its design.

Next, compile the circuit and create a symbol for it, then insert the symbol into a new empty schematic. Then, following the process learned in the previous lab, perform a *timing* simulation of the circuit.

There are 2^12 different input patterns (4196) which is too many to manually check. So just test a representative set of input cases. This test set should include at least one instance of a violation and an agreement with each rule. Note the worst-case maximum propagation delay.

Show your simulations to the TA.

TIME CHECK

You should be this far at the end of your *second* 2-hour lab period!

3. Design of the Card Dealer Circuit.

A card game would be no fun if the sequence of card was the same every time you played. So we need some way of randomizing the cards that are dealt out. Fortunately you have already designed much of what is needed to do this.

There are many algorithms for shuffling a deck of cards. Some are more efficient (faster) than others, and some give better randomization (less predictable). Decks will be implemented in our system with the stack circuit you designed in lab #3. There will be a total of 3 "decks" used in the complete system – the ***deal deck***, from which cards are dealt, and the two ***players' decks***.

We will not actually shuffle the deck. Instead, we will initialize the deck to have cards ordered from 0 through 51. Then, when a card is to be dealt, a slot location (between 0 and NUM-1) will be chosen at random. This card will then be removed (popped) from the deal deck and transferred to the player's deck.

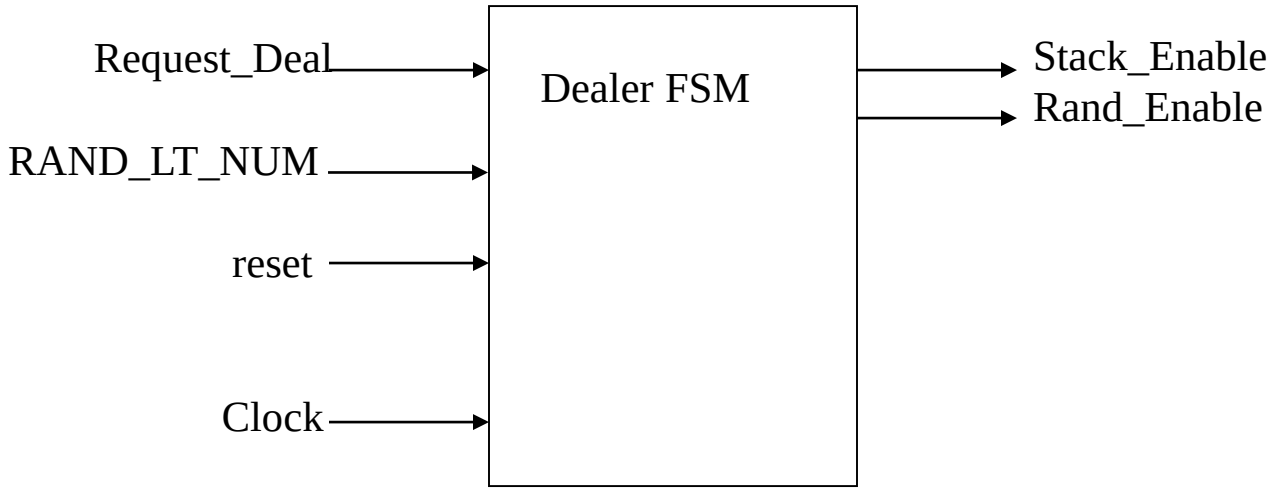
Design of the Card Dealer Circuit (continued).

The random number generation can be done with the RANDU circuit that was designed in lab #2. This circuit generates a 32-bit value, whereas we need only 6-bits! So, we can just extract out 6 bits from the 32-bit value and use those. It is better to use the higher order bits (MSBs) as these are more random than the lower order bits.

Using 6-bits gives us a number between 0-63, whereas we need a number between 0 and ***NUM-1***, where ***NUM*** is the current number of elements in the deck (stack). Thus, some of the random numbers that we generate will not be valid. We must therefore develop a system that can repeatedly generate random 6-bit numbers until we get one that is in the range of ***0-NUM-1***. One difficulty with this approach is that it will take a long time when ***NUM*** is small. We can speed this up by just considering ***M*** bits where ***M*** is the smallest integer such that ***2^M > NUM-1***.

You can implement the needed operations using a Finite State Machine.

Using the VHDL approach outlined in class, describe a ***Moore***-style finite state machine with input and output signals as given in the diagram shown below:



The FSM should implement the following sequence:

1. Wait for the Request_Deal input to go low
2. Wait for the Request_Deal input to go high
3. Assert the Rand_Enable output (which causes a new random number to be loaded into a register). This is asserted only in this this state.
4. Compare the random number to the value of NUM (done by an external module, not by the FSM)
5. If RAND_LT_NUM is low (i.e. the output of RANDU is greater or equal to NUM) go to step 3, otherwise go to step 6
6. Assert the Stack_Enable output (which enables a POP operation on the stack). This should be asserted only in this state.
7. Go to step 1

Draw the FSM's state transition diagram and show it to the TA.

4. Simulation of the Dealer FSM.

When the VHDL description for the Dealer FSM is complete, create a symbol for it, and insert an instance into a new Quartus schematic window.

Show the completed VHDL description of the FSM to the TA and ex to him or her how it works.

Then perform a *timing* simulation of the FSM. You will need to provide suitable input signals (e.g. the RAND_LT_NUM signal) by hand, since you are only simulating the FSM and not the rest of the system (such as the RANDU circuit or the comparison to NUM). Adding in the rest of the parts will be done in the next section of the lab.

Show the results of the simulation to the TA.

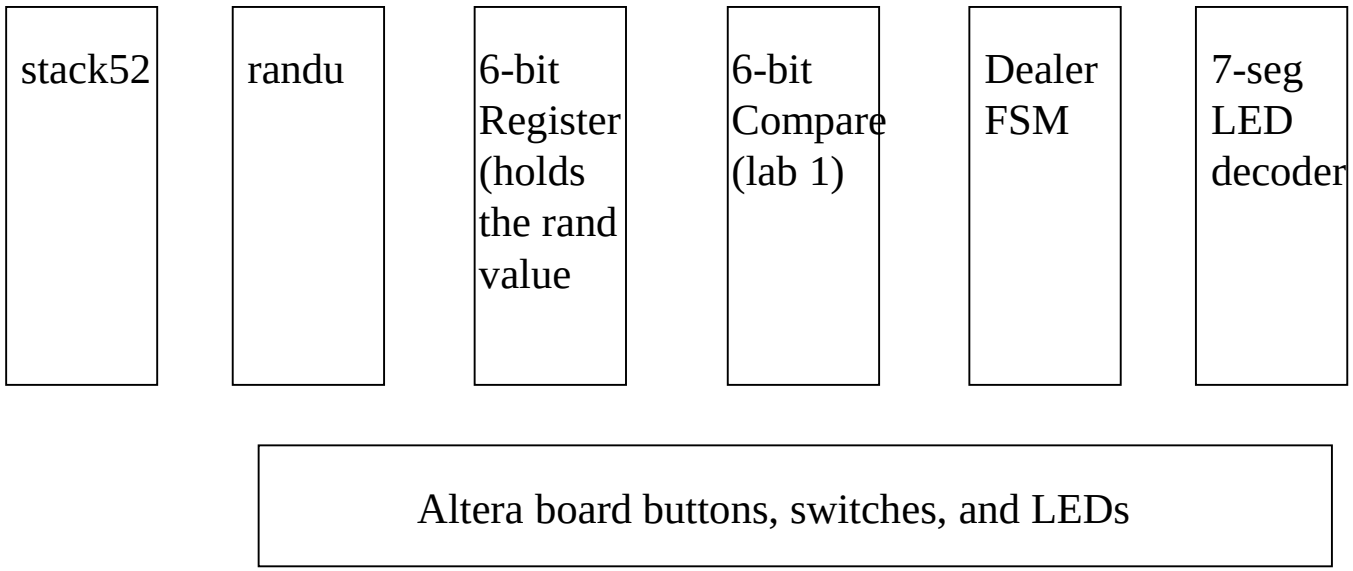
TIME CHECK

You should be this far at the end of the *third* 2-hour lab period!

5. Design of the Dealer Testbed Circuit.

The Dealer FSM is just part of the system. In lab 5 you will put everything together, but for now, you will just construct part of the system, enough to give the Dealer FSM a real test.

Using schematic capture, use the following types of modules (and any others you see fit to use) and put together a testbed circuit that will permit testing of the Dealing function:



6. Testing of the Dealer Testbed Circuit.

Show the completed schematic diagram of the Dealer testbed to the TA and explain to him or her how it works.

Compile the design, and look at the compilation report. Pay attention to the resource usage. How many logic cells does the testbed use? What percentage of the total number is this?
Download your design to the Altera board hardware.

Show the operation of the testbed to the TA, being sure to demonstrate the proper dealing of cards. Only one card at a time should be dealt (keep track of this by displaying the NUM signal). The cards should be randomly selected.

TIME CHECK

You should be this far at the end of the *fourth* 2-hour lab period!

7. Writeup of the Lab Report .

Write up two reports, one for the Rules module and the other for the Dealer FSM circuit.

The report must include the following items:

- A header listing the group number (and company name if you gave it one), the names and student numbers of each group member.
- A title, giving the name (e.g. ***gNN_Rules***) and function of the circuit.
- A description of the circuit's function, listing the inputs and outputs. Provide a pinout or symbol diagram.
- The VHDL description of the circuit.
- A discussion of how the circuit was tested, giving details of the testbed and showing representative simulation plots.
- A summary of the timing performance of the circuit, giving the timing analysis and the simulated propagation delays.
- A summary of the FPGA resource utilization (from the Compilation Report's Flow Summary).

The reports should be done in html or pdf (preferred), or in Microsoft Word, and submitted to the WebCT site .

Make sure that you have uploaded ***all*** of the design files (e.g. .bdf and .vhd files) used in your project. These should be included with your reports in a single zip file.

Grade Sheet for Lab #4

Fall 2017.

Group Number: _____.

Group Member Name: _____. Student Number: _____.

Group Member Name: _____. Student Number: _____.

Marks			TA Signatures
	1.	VHDL description of the Rules circuit	.
	2.	Simulation of the Rules circuit	.
	3.	Dealer FSM state transition diagram	.
	4.	VHDL description of the Dealer FSM	.
	5.	Simulation of the Dealer FSM	.
	6.	Schematic of the Dealer Testbed	.
	7.	Testing of the Dealer Testbed on the Altera board	.

Each part should be demonstrated to one of the TAs who will then give a grade and sign the grade sheet. Grades for each part will be either 0, 1, or 2. A mark of 2 will be given if everything is done correctly. A grade of 1 will be given if there are significant problems, but an attempt was made. A grade of 0 will be given for parts that were not done at all, or for which there is no TA signature.