Login Registers

List of Authors (Nicholas Sajjakulnukit, Brett Shoup, David Martel)
Electrical and Computer Engineering Department
School of Engineering and Computer Science
Oakland University, Rochester, MI
e-mails: nnsajjak@oakland.edu, BShoup@oakland.edu, Djmartel@oakland.edu

***Abstract*—This program implements a series of registers to store information given by the user and displays this data on the 7-segment display based on user-inputted selection on the switches.**

I. Introduction

This project utilized elements of previous labs to implement several simultaneous registers to store information and creates state machines to determine which registers should be updated based on the inputs. It also took input from a keyboard through the USB hub, which was stored as data on the registers.

The purpose of this project was to design a system that allows users to store textual information to be recalled later, and to do so in a way that organizes the information in an intuitive manner that is easy to retrieve and read. Though the project itself uses usernames and passwords as an example, this system could theoretically work for any similar use with very little changes.

II. Methodology

A. *Keyboard and Character Parser*

The Keyboard provides the most essential input into the project, and requires a specific parser to process the data. Using sample code provided by Professor LLamocca [1], a 9-bit signal is received for all alphanumeric inputs given by a keyboard, and all characters that can be represented on a 7-segment LED display are processed as valid and assigned as a 7-segment equivalent signal. All other characters are processed as invalid and given no signal. Processing the outputs accordingly required knowledge of how the data is passed by the keyboard, in particular how the keyboard clock tells the board how to interpret the bitstream. [2]

B. *Address and Value Selector*

Based on the address input, a 8-bit enable signal is generated to selectively enable one register at a time. Additionally, a state machine produces an output that is used to update each value in the enabled register sequentially as valid inputs are passed by the Keyboard input. This state machine proceeds only while the read/write input is high, as this is the only situation where the state machine should be working. In addition, these processes output an LED signal that allows the user to see what state the process is in through the LEDs on the board.

C. *Registers*

8 registers are instantiated to cover the 8 possible addresses. Each holds 16 values for this project, designated as 8 username and 8 password values. Each register is enabled only if their particular bit in the enable signal is high, and each value is updated only if the code from the value selector state machine matches the address of the value within the register. The registers themselves output all stored values as signals to the control circuit, and are reset to 0 if the reset switch is low.

D. *Display*

The display component reads the status of the read/write input and decides whether to prompt the user on the 7-segment displays or output data to the 7-segment displays. When outputting data, the display component cycles through a mixture of stock outputs and the stored register values every few seconds. This is done by using a state machine to sequentially enable each 7-segment

display in turn and display the stored LED representation of the data. The cycling of messages is managed by a state machine that switches between outputs to the display based on a slowed clock process, which allows each message to be read.

III. Experimental Setup

This project is assembled using a keyboard, the Nexys 4 FPGA board and its integrated components, and the Xilinx ISE 14.7 program for VHDL. The physical setup of the project only requires that the keyboard by plugged into the USB Host, and that switches 0-5 are available for use. Switch 5 is the reset switch, switch 4 enables the display process, and switch 3 is the read/write input, while the other switches are the address input. Based on the state of the inputs from both the switches and the keyboard, the 7-segment display should be producing an output provided the reset switch is high, and the 16 LEDs should be lit only while an input is being written to the registers. If the reset switch is low, no outputs should be seen on the 7-segment display or the LEDs.

IV. Results

The project functions as expected, though there are a few unforeseen complications in its use. Users cannot input valid inputs in sequence and must instead input an invalid input in between valid inputs. This is caused by the way that keyboard inputs are classified as valid and invalid, and the fact that the process that detects a change in this signal prevents multiple valid inputs to be sent sequentially. Additionally, the initial value added is occasionally dropped while the last input is often duplicated, which shifts values one digit to the side. This is most likely due to a synchronous error with the modified clock that causes changes to the system to occur in the wrong clock cycle, causing problems with the control state machine. There is also a slight issue with the value selector where a valid input from a previous cycle may cause the state machine to prematurely progress to the second state before an input is given.

Conclusions

From this project, it can be clearly seen that values within a register can be modified selectively instead of all together as displayed in the labs, and that state machines provide an efficient way of managing changes in signals based on specific constraints. In addition, some processes that require serialized input are often easier to process if the clock used is slowed down to a manageable extent for human input, else erroneous inputs are given.

A few improvements that can be made to this project are an intermediary signal to allow for sequential inputs, redundancy checking in the control state machine to ensure that the proper values are modified, and an expanded list of alternate characters to cover restricted characters. In addition, the project would benefit from a more precise period of message cycling in order to allow for better readability for the user.

References

[1] Daniel LLamocca, Unit 7 code . *2013*

[2] Chu, Pong P. *FPGA Prototyping by VHDL Examples Xilinx Spartan-3 Version*. Hoboken, N.J.: Wiley-Interscience, 2008. 183-196. Print.