Will Krska and Conor Naughton

EC551 Fall 2022

Lab 2

Our overall design is shown in the following block diagram

Terminal Interface Top

UART\_TX\_data

UART\_RX\_data

Key\_select\_data

Data Loader

Key\_select\_flag

Key\_select\_data

Mode\_select\_data

Mode\_select\_flag

Keyboard Interface

UART Interface

UART Transmitter

UART Receiver

PS/2 Controller

As shown, the design can be broken down into three main segments, a processing element and interfaces for handling both the keyboard and UART signals. The following is a description of these segments.

**Processing Element/Data Loader**

This block

**PS/2 Keyboard Controller**

To build our keyboard controller, we started with the demo provided in the lab manual as a base and built the additional functionality needed for this lab. In particular, within the PS/2 receiver module, we added a new block which is triggered on the positive edge of the clock and checks the second byte of the keycode variable to identify whether a key release has occurred (this happens when the second byte is F0). When this happens, we set a keystroke flag high and send the PS/2 byte value of the keystroke up to the top keyboard module. When we are coming from the initial state, the next step in this block is to check to see if the current keystroke was the enter key (this happens when the first byte is 5A). This indicates that the user has entered a mode selection value and triggers a modeflag to go high as well as sending the byte value of the mode to the top module. Once a valid mode byte (I, L, A or B) has been sent, the module continues to send keystrokes to the top module as they appear which will be accumulated for instruction formation. In the keyboard top module, there are finite state machines for both the keystroke and mode data which are entered on each clock cycle and stay in the initial state until they receive a high flag. Once they get the high flag, they hold a corresponding flag low for another clock cycle and then put it high and sent that signal up to the overall terminal interface module. The reason for this sequence is because the PS2 receive module runs with a 50MHz clock while the top module runs with the system 100MHz clock so if we only passed through the high flag, it would get sent twice before the flag going low was recognized.

**UART Interface**

Our UART interface also built upon the demo code which was linked in the lab manual. This example code simply loaded in an introductory message and then sent another message whenever a button was pressed. It accomplishes this using a case statement which controls when messages are sent to UART and adds delays in order to provide time for the message send process to complete. These states include a Wait\_Mode which is where the state machine sits until it gets a flag identifying that it has received a mode select or a keystroke. We modified the introductory message as directed in the lab manual and then added logic within the case statement which controlled the messages being sent over UART to identify which mode has been entered based on the mode\_select signal originating from the keyboard controller. Based on which mode was selected, a predefined array of strings is loaded into the UART using the same byte by byte method used for the introductory message. In the case that the mode selected is not amongst those which we intend to use for this lab, a invalid mode message is sent and the state machine returns to the loop looking waiting for the correct mode to arrive.

**Terminal Interface**

Our terminal interface