Matrix Keypad Decoder

Note: You must attend your lab session and demonstrate your solution to get marks for completing this lab. See the Lab Demonstration *section below.*

Introduction

In this lab you will design a module named **keypad** that scans the matrix keypad and outputs a binary number between 4'd0 and 4'd9 corresponding to the key that is being pressed, or 4'hf if no key is being pressed. Pressing the * and # keys should output the binary number corresponding to the last two digits of your BCIT ID.

To test your module you will include it in a supplied project that displays the digit that is being pressed (0 through 9) on the rightmost digit of the LED display.

You will use the same components as in a previous lab, connected the same way.

Matrix Keypad

Review the description of the matrix keypad in a previous lab.

Requirements

Your module must be declared as follows:

```
module keypad
  ( input logic clk,
    input logic [3:0] row,
    output logic [3:0] col,
    output logic [3:0] digit ) ;
```

Your module should set **digit** to the binary value of the key if one of the number keys is pressed. Pressing * and # should output the last two non-zero digits of your BCIT ID. For example, if your ID were A01234506 then pressing * should set **digit** to 4'd5 and pressing # should set **digit** to 4'd6.

Your module should set **digit** to 4'hf if no key is pressed or if a letter is pressed.

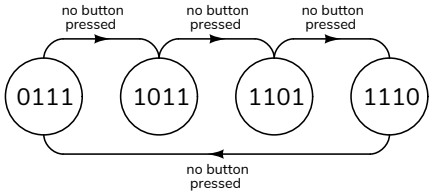
row and **col** will be connected to the matrix keypad as in a previous lab. **clk** will be connected to a 200 Hz clock as in a previous lab.

Design

Your design must set successive column outputs low and check if any of the row inputs is low. If a button along the column that was set low is pressed, then the corresponding row will go low. Scanning of the columns should stop if any row inputs are low. If no rows are low, then no button along that column is being pressed and your circuit should continue scanning.

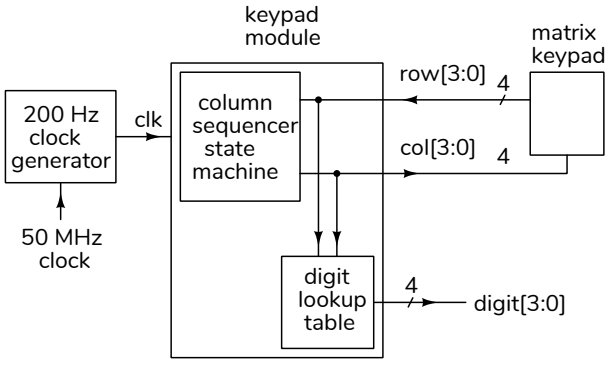
Your circuit should continuously test each of the columns in order from the left column to the right column and stop scanning when it detects that a button is being pressed. It should resume scanning when no button is being pressed.

The state transition diagram for a row output state machine that implements the above functionality is:



where the states are labelled using the binary value of the **col[3:0]** output[1].

A possible block diagram for your design is:



[1]These state encodings are "one-cold" – the logical complement of "one-hot" encoding with the same advantages.

Component Connections

The CPLD board, keypad and LED display should be connected as in a previous lab.

Procedure

Download and open the **lab4.qar** Quartus Archive file containing the project files[2], create a folder in an appropriate location and restore the project to this folder.

Use File / New... / System Verilog HDL File to create a new file and save it as **keypad.sv** to your project folder. The "Add file to current project" box should be checked so the file is added to the project. Write your **keypad** module in this file.

The downloaded project includes the other required source files (**lab4.sv** and **clkdiv.sv**) and the default pin assignments as described in a previous lab. Change the pin assignments if necessary.

Connect the keypad and LED to the CPLD board. If you're using the default pin assignments you can download the **lab4.pof** file and program the CPLD to check your hardware.

Compile your design and program the CPLD. Test your design and fix any errors.

Hints

Your **keypad** module will consist of:

1. A state machine that sequences through four states in order from left to right as shown in the state transition diagram above.

   You can implement this state machine in a single **always_ff** statement.

2. A combinational logic circuit that sets the value of **digit** corresponding to the row and column that are low. For example if the second column from the left (2) is low when the bottom row (0) is set low then **digit** should be set to 0.

   You can implement this logic in a single **assign** statement.

You can concatenate the row and column values to create an 8-bit value. For example, you could use the expression `{row,col} == 8'b1110_1011`.

[2]It should open with Quartus

Design your state machine so that it will recover from *any* state. This is important because the CPLD registers may not power up with any of the states in your state machine.

You can use the **lab4.pof** file on the course website to test your hardware. You can view the lab4demo.mp4 video for an example of the required behaviour.

Lab Demonstration

You must demonstrate your lab during your scheduled lab period. I recommend that you have working design before coming to your lab session.

After successfully demonstrating the behaviour described above, the lab instructor will ask you to make a simple change to the behaviour of your design to demonstrate your understanding. Successfully making the change will count for half of this lab's completion mark.

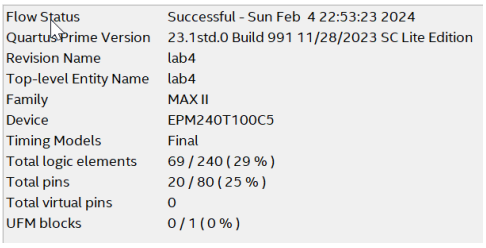
Submissions

Lab Report

Submit the following to the appropriate Assignment folder on the course website:

1. A PDF document containing:

   - A listing of your Verilog code that meets the requirements above.
   - A listing of your Verilog code that includes the change requested by the lab instructor.
   - A screen capture of one of the compilation reports. For example:



Flow Status	Successful - Sun Feb 4 22:53:23 2024
Quartus Prime Version	23.1std.0 Build 991 11/28/2023 SC Lite Edition
Revision Name	lab4
Top-level Entity Name	lab4
Family	MAX II
Device	EPM240T100C5
Timing Models	Final
Total logic elements	69 / 240 (29 %)
Total pins	20 / 80 (25 %)
Total virtual pins	0
UFM blocks	0 / 1 (0 %)

Follow the *Report and Video Guidelines* and *Coding Guidelines* documents on the course website.