Matrix Keypad Decoder

Introduction

In this lab you will design a module named **keypad** that scans the matrix keypad and (1) asserts a **pressed** output when any key is pressed, and (2) outputs a (binary) number between 4'd0 and 4'hD corresponding to the key that is being pressed. Pressing the * and # keys should output the number corresponding to the last two digits of your BCIT ID.

You will test your module by building the supplied **lab4** project. This project contains a **lab4** top-level module that instantiates your **keypad** module. The **lab4** module will display the value output by your **keypad** module on 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,
    output logic [3:0] row,
    input logic [3:0] col,
    output logic pressed,
    output logic [3:0] digit ) ;
```

Whenever a key is pressed your module should assert **pressed** and set **digit** to the binary value of the key if one of the number keys or the keys A-D 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.

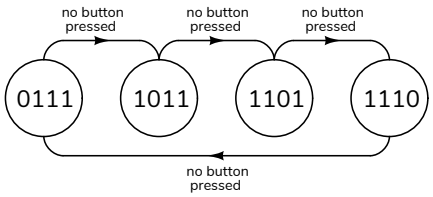
row and **col** will be connected to the matrix keypad as in a previous lab. **clk** will be connected to a 200 Hz clock.

Design

Your design should set successive row outputs low and check if any of the column inputs is low. If a key along the row that was set low is pressed, then the column connected to that key will go low. Scanning of the rows should pause if any column inputs are low. If no column is low, then no key along that row is being pressed and your design should continue scanning.

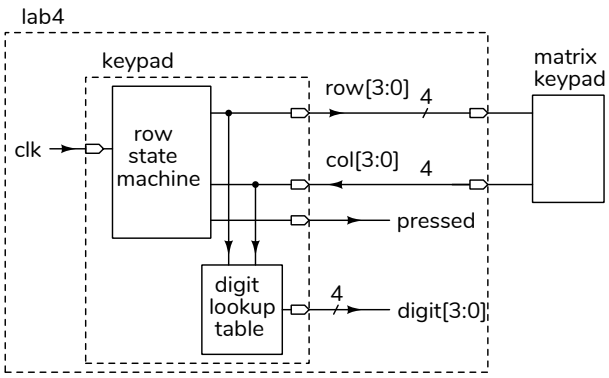
Your circuit should continuously test each of the rows in order from top to bottom and stop scanning when it detects that a key is being pressed. It should resume scanning when no key is being pressed.

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 **row[3:0]** output[1].

A 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.

2. Combinational logic 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.

3. Combinational logic that sets **pressed** when any key is pressed (i.e. any column is low).

The **row** output state machine can be implemented in a single **always_ff** statement. **digit** and **pressed** can be implemented with **assign** statements.

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`.

You state machine should be able to start 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.

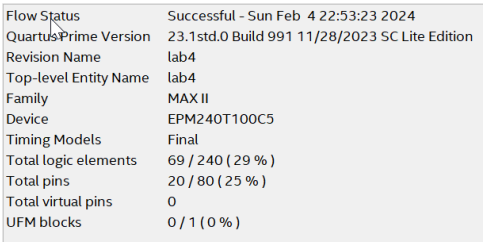
Submissions

Lab Report

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

1. A PDF document containing:

   - A block diagram of your design. Make sure that it conforms to the course requirements.
   - A listing of your Verilog code that meets the requirements above.
   - A screen capture of the compilation report. 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 %)

2. If you did not demonstrate your lab in person, a video showing the keypad and display as you press each key on the keypad, ending with the * and # keys.

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

[2] It should open with Quartus